Proceedings of Mining Minds Release Version 3.0

Research & Technical Specifications

Principal Investigator:

Professor Sungyoung Lee

April 19, 2017

Ubiquitous Computing Laboratory
Kyung Hee University, Korea

Table Of Contents

<u>Section 1</u>

Mining Minds Publications

Section 1.1

Mining Minds Paper

Banos *et al. BioMed Eng OnLine* 2016, 15(Suppl 1):S76
DOI 10.1186/s12938-016-0179-9

BioMedical Engineering OnLine

RESEARCH

Open Access

The Mining Minds digital health and wellness framework

Oresti Banos[1], Muhammad Bilal Amin[1], Wajahat Ali Khan[1], Muhammad Afzal[1], Maqbool Hussain[1], Byeong Ho Kang[2] and Sungyong Lee[1*]

From 3rd International Work-Conference on Bioinformatics and Biomedical Engineering-IWBBIO 2015
Grenada, Spain. 15–17 April 2015

*Correspondence: sylee@oslab.khu.ac.kr
[1] Department of Computer Engineering, Kyung Hee University, 1732 Deokyoungdae-ro, Giheung-ug, Yongin-si 446-701, Korea
Full list of author information is available at the end of the article

Abstract

Background: The provision of health and wellness care is undergoing an enormous transformation. A key element of this revolution consists in prioritizing prevention and proactivity based on the analysis of people's conducts and the empowerment of individuals in their self-management. Digital technologies are unquestionably destined to be the main engine of this change, with an increasing number of domain-specific applications and devices commercialized every year; however, there is an apparent lack of frameworks capable of orchestrating and intelligently leveraging, all the data, information and knowledge generated through these systems.

Methods: This work presents Mining Minds, a novel framework that builds on the core ideas of the digital health and wellness paradigms to enable the provision of personalized support. Mining Minds embraces some of the most prominent digital technologies, ranging from Big Data and Cloud Computing to Wearables and Internet of Things, as well as modern concepts and methods, such as context-awareness, knowledge bases or analytics, to holistically and continuously investigate on people's lifestyles and provide a variety of smart coaching and support services.

Results: This paper comprehensively describes the efficient and rational combination and interoperation of these technologies and methods through Mining Minds, while meeting the essential requirements posed by a framework for personalized health and wellness support. Moreover, this work presents a realization of the key architectural components of Mining Minds, as well as various exemplary user applications and expert tools to illustrate some of the potential services supported by the proposed framework.

Conclusions: Mining Minds constitutes an innovative holistic means to inspect human behavior and provide personalized health and wellness support. The principles behind this framework uncover new research ideas and may serve as a reference for similar initiatives.

Keywords: Human behavior, Digital health, dHealth framework, Quantified self, Wearable sensors, Big data, Cloud computing, Context-awareness, Knowledge bases, User experience

Background

Healthcare systems are facing unprecedented financial limitations at a time of rising demand for their services [1]. The magnitude of these constrains makes utterly necessary to change current care models in a bold manner, from late disease management to preventive personalized health, involving a major shift in when, where and how care and support is delivered to each particular patient and service user [2]. In fact, it is generally recognized that most prevalent diseases are partly caused or aggravated by lifestyle choices that people make in their everyday life. Unwholesome diets, tobacco use and sedentary conducts, among other unhealthy habits, potentially contribute to develop severe illnesses [3, 4] and also limit the effectiveness of medical treatments [5]. Thus, enabling people to make healthier choices, to be more resilient, and to deal more effectively with illness and disability when it arises, turns to be a fundamental part of this necessary new health perspective.

Information and communication technology is called upon to be a cornerstone of the new health era, playing a crucial role in empowering people to take charge of their own health and wellness, by providing them timely and ubiquitously with personalized information, support and control [6]. In fact, an extraordinary interest has been lately shown by the industry in the development of specific applications and systems for health and wellness management, particularly boomed by the growth of wearable and mobile technology [7]. The immediate targets of these solutions are healthy lifestyle services, especially oriented to the fitness domain, which primarily allow to track primitive user routines and provide simple motivational instructions. For example, mainstream commercial systems such as Withings Activite [8], Garmin Vivofit [9], Fitbit Surge [10] or Misfit Shine [11], which consist of sensorized bracelets and gadgets normally accompanied by mobile apps, provide some basic healthy recommendations based on the measured taken steps or slept hours. More prominent health and wellness systems have been shown at the research level, for example, to alert on physical conditions [12] or detect chronic illnesses [13], yet most of them are prototypes or work-in-progress. Some of these systems also provide educational modules and personal coaching for promoting healthier lifestyles and managing health conditions [14]. Despite their interest, main limitations of these solutions refer to misperformance, limited scope and lack of interoperability with other similar systems and applications.

To overcome the shortcomings of application-specific solutions and leverage the potential of health information systems in a wide sense, general frameworks capable of managing these resources are required. A few attempts are found in this respect in the literature, for example, in [15] a middleware framework integrating multiple interfaces and multiparameter monitoring of physiological measurement is presented. In [16], distributed signal processing algorithms for the analysis and classification of sensor data are provided as part of a framework for rapid prototyping of body sensor networks. A mobile platform to collect users' psychological, physiological and activity information for mental health research is presented in [17]. The authors of [18] propose a healthcare platform particularly devised for interfacing and processing data from body-worn physiological sensors and home appliances, with a proven utility in daily medication management. A novel framework that provides advanced functionalities for resource and communication abstraction, wearable health data acquisition and knowledge extraction is introduced in [19]. Most visible initiatives are especially being underpinned in the mobile health domain. That is the case of [20], an open mobile health project to help developers produce digital health data as useful and actionable as possible. Google Fit [21] by Google, SAMI [22] by Samsung or HealthKit [23] by Apple are examples of new commercial platforms also devised to integrate and share users health data among diverse health and wellness applications.

Despite important contributions have been made through these platforms, there is still much room for improvement. For example, most mobile health frameworks are bound to the computational capabilities of the smartphone, require continuous maintenance and updates of end-user applications and normally trap data into their devices. Moreover, multiple systems and applications can generate similar health data and outcomes leading to unnecessary redundancy and overcomputation. These systems mostly operate on-demand, thus determinants of health and wellness states can also be lost if not registered in a continuous manner. Platforms devised to share and integrate health and wellness data underutilize cloud resources while simply using them for storage. In the light of these limitations we present Mining Minds [24, 25], an innovative distributed framework that builds on some of the most prominent digital technologies to enable the provision of personalized healthcare and wellness support. This framework is particularly devised to seamlessly investigate on people's behavior and lifestyles in an holistic manner through mining human's daily living data generated through heterogeneous resources. Mining Minds aims to innovatively exploit the potential of cloud computing not only for storage but also for high performance computation supporting the discovery of personal and public health and wellness patterns, of primal necessity to facilitate proactive and preventive support.

Requirements of a digital health and wellness framework

Diverse types of data are normally required to neatly describe a person's health and wellness state, ranging from physical-sensory- and logical-personal profile and interests-, to social-human relations- and clinical-medical-data. Many technologies are increasingly available for the collection of these data, such as wearable devices, ambient sensors, social networks or advanced clinical systems. Thus, an important requirement of a digital health and wellness framework is to provide a certain level of abstraction from heterogeneous resources to make their utilization transparent to the user. Health and wellness data go beyond standardized structured formats such as "traditional" electronic health records, particularly including other multimedia and unstructured data. Therefore, another primal requirement is to be capable of dealing with this dimension of heterogeneous data, as well as the underlying implications of the management of structured, semi-structured and unstructured data.

Not only data variety constitutes a key factor, but also data volume. Massive amounts of data are generated over time on and around the subject with the advent of new sensing and multimedia technologies. Accumulating and digesting these amounts of data are not trivial tasks and need to involve sophisticated processing and storage mechanisms to enable the persistence and availability of the data. Similarly, the rapid pace of data generation makes it necessary to also take into account data velocity as a reference factor. This proves to be especially challenging when referred to data that represents real-time regular monitoring, such as continuous electrocardiogram measurements or body motion data. Another important concept that applies to health and wellness data is veracity. Different data types may represent similar concepts or contradict each other, or even be of little interest. Therefore, digital health and wellness frameworks should count on governance mechanisms to determine the consistency of the data, ensuring it is certain, meaningful, clean and precise.

Extracting the determinants of health and wellness is a very challenging task that requires more than simply collecting and persisting personal data. Accordingly, digital health and wellness frameworks must include automatic intelligent mechanisms to process person-centric data and extract interpretable information and insights for ensuring a personalized health and wellness support. Moreover, insights should not only be gained from individual users but from the collectivity. Thus, another important requirement consists in the application of advanced techniques to process information in "de-identified" form to enable population management and deeper insights into cause and effect. These insights can be particularly leveraged by health and wellness care systems to extend, adapt and evolve the knowledge provided by human domain experts.

Health and wellness information and knowledge are principally devoted to support advanced care services. Mechanisms such as alerts, recommendations or guidelines are particularly used as services to catalyze both information and knowledge to be delivered in a human-understandable fashion to users and stakeholders in general. However, most digital health and wellness systems only support general services that do not differentiate among people's particular needs or interests. Therefore, an important requirement is to provide services that operate on a person-centric manner. To do so, expert systems are required, for example, to precisely map user needs to the best possible recommendations, personalize the recommendations explanation or customize the mechanisms for the communication of these recommendations.

Users of health and wellness systems may be of a very diverse nature and play different roles. For example, busy patients may require to get a quick glimpse of their health conditions, fitness enthusiasts wish to observe a detailed description of their vitals and clinical experts be interested in an "in-depth" description of both health and wellness outcomes of multiple people. Accordingly, user interfaces need to be customized to the needs of each particular subject. Similarly, the user experience is of worth consideration. Users perceptions of system aspects such as utility, ease of use and efficiency should be taken into account to provide the most personalized experience. In fact, the user experience is dynamic as it is constantly modified over time due to the person changing circumstances. Thus, user responses and behavior need to be continuously tracked to support a sufficient level of personalization that helps guarantee adoption and engagement.

Finally, as it may be obvious, but unfortunately not often considered, all the aforementioned requirements need to be neatly accommodated to user security and privacy principles. The necessity of privacy and security is crucial for systems that build over sensitive information, and further augmented when data and services are shared by multiple entities in a distributed way. Data ownership, malicious data usage, as well as regulatory and legal policies are important hindrances in the widespread use and acceptance of health and wellness care systems. Therefore, it is of utmost importance to neatly adequate privacy, security, protection and risk management measures to all the processes concerned in a digital health and wellness framework.

Mining Minds architecture

In the light of the aforementioned requirements we present here "Mining Minds", a novel framework aimed at comprehensively mining human's daily life data generated from heterogeneous resources for producing personalized health and wellness support.

Mining Minds philosophy revolves around the concepts of data, information, knowledge and service curation, which refer to the discovery, processing, adaptation and evolution of both contents and mechanisms for the provision of high quality support services. Motivated by these concepts, a multilayer architecture is particularly devised for Mining Minds—Fig. 1. In a nutshell, the data curation layer (DCL) is in charge of processing and persisting the data obtained from the multimodal data sources (MDS), which abstractly defines the possible sources of user health and wellness data. This includes, but is not limited to, data from social networks, questionnaires, wearable biomedical devices or ambient intelligence systems. The data processed by DCL is primarily used by the information curation layer (ICL) to infer low-level and high-level person-centric information. This information mainly describes the user context and behavior, and, to some extent, their physical, mental and social state. The information extracted by ICL is leveraged by the knowledge curation layer (KCL) to nurture and evolve the health and wellness knowledge primarily created by human experts. Data, information and knowledge are

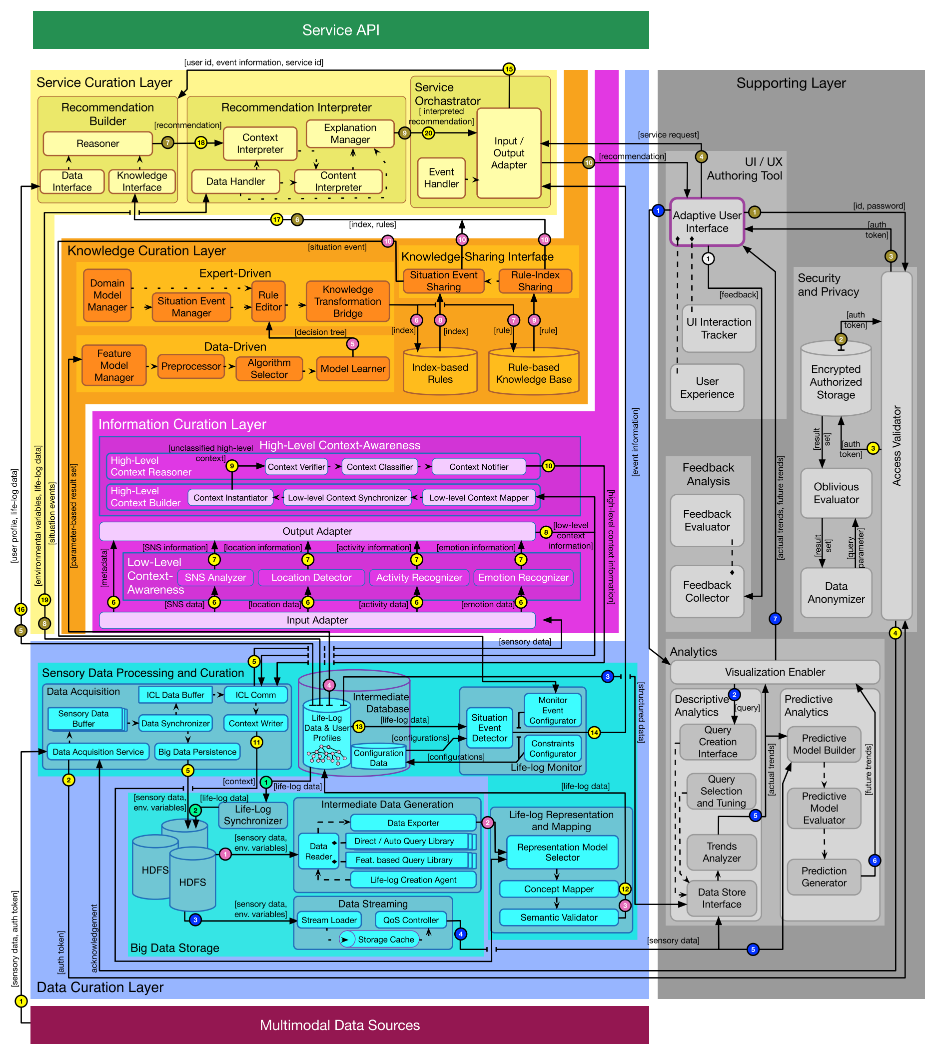


Fig. 1 Mining Minds framework architecture and operational diagram

used by the service curation layer (SCL) to create intelligent health and wellness support services, mostly in the form of smart coaching and support recommendations. All the contents and processes are accommodated in terms of security and privacy by the supporting layer (SL), which also provides analysis of user experience, feedback and trends to guarantee the highest personalization.

Data curation layer

Data curation layer is responsible for acquiring, curating and persisting the data obtained from MDS so it can be processed for higher level understanding. To that end DCL relies on two main modules, Sensory Data Processing and Curation and Big Data Storage. Within the former, Data Acquisition supports the acquisition and synchronization of raw sensory data obtained from diverse sources, both in real-time and offline manner, as generic data streams. Due to the heterogeneous nature of the data, it is acquired asynchronously in real-time and temporarily cached in data buffers. These data buffers are initialized depending upon the number of data sources, i.e., each data source has a data buffer in the Data Acquisition component. All the data buffers are synchronized and communicated to ICL for the determination of the associated low and high-level contexts. In parallel, this synchronized data is stored in Big Data Storage for non-volatile persistence.

Upon receiving the context information determined by ICL, the context instances are curated by the Representation and Mapping component as a time-based log registering the detected human behaviors. This time-based log is termed as user Life-Log or simply Life-Log and persisted in the Intermediate Database for shareability with other layers and applications. The stream of life-log instances is analyzed by a monitoring component called life-log monitor (LLM). The responsibility of the LLM is to perform time-based monitoring of the different attributes and variables hosted in the Life-Log, and support trigger-based mechanisms to notify SCL for the occurrence of an abnormal or special event related to a given user. These abnormal events normally represent risky or unhealthy behaviors and are here defined as "situation events" or "situations" in general. These situations are described through diverse constraints, e.g., age, gender or medical conditions and monitorable variables, e.g., intensity of a particular activity or its duration. Situation events can be generated both statically at design-time and dynamically at run-time upon request from KCL.

The life-log data persisted in the Intermediate Database is regularly synchronized with the Big Data Storage. Big Data Storage also provides read access to raw sensory and life-log data. In case of historic data required by SL for analytics or KCL for data-driven rule generation, Big Data Storage provides queries for data streaming and intermediate data generation. These queries can be customized on request to return specific data based on the attributes selected by KCL and SL. Security and privacy components from SL are further involved in these processes to request authentication and data stream encryption before its persistence or sharing.

Information curation layer

Information curation layer represents the Mining Minds core for the inference and modeling of the user context [26]. ICL is composed by two main modules, namely, low level context awareness (LLCA) and high level context awareness (HLCA). LLCA is in charge of converting the wide-spectrum of data obtained from the user interaction with the real and cyber-world, into abstract concepts or categories, such as physical activities, emotional states, locations and social patterns. These categories are intelligently combined and processed at HLCA in order to identify more meaningful semantic representations of the user context.

Low level context awareness is composed by four key components, respectively, Activity Recognizer, Emotion Recognizer, Location Detector and SNS Analyzer. The identification of the user physical actions is performed through the Activity Recognizer. This component may build on several sensing modalities as they happen to be available to the user, such as wearable inertial sensors, video and audio. The output of this component corresponds to elementary activity categories such as "sitting" or "walking". The Emotion Recognizer is defined to infer user emotional states, such as "surprise" or "sadness", by using video and audio data as well as more sophisticated sources exploring human physiological variations and responses. The user situation is determined by the Location Detector, which essentially builds on the data collected through indoor and outdoor positioning sensors, such as video and GPS, to specify the exact location of the user. The SNS Analyzer is in charge of processing the information generated by the user during their interactions in regular social networks, including posts, mentions, traces and even global social trends, in the form of both text and multimedia data. From here, personal and general interests, conducts and sentiments may be determined. All these components require compatible multimodal sensory data to operate. The provisioning of the necessary data is performed through the Input Adapter, which receives and routes the data curated by DCL to each LLCA component depending on its nature. Once new low-level context categories are identified after the analysis of this data, the Output Adapter serves them to DCL for persistence and to HLCA for further processing.

High level context awareness makes use of two components, namely, High-Level Context Builder and High-Level Context Reasoner, to represent, verify, classify and categorize the user high-level context. The context representation and verification is performed through ontologies, adopted in the past as a unified conceptual backbone for modeling context, while its classification and categorization is done through ontological inference and reasoning. Whenever new information is received from LLCA, a new ontological instance is created by the High-Level Context Builder and categorized into one of the considered high-level contexts by the High-Level Context Reasoner. Thus for example, based on the actual time—e.g., midday; location—e.g., restaurant; and inferred activities—e.g., sitting; this component can determine the precise user context—e.g., lunch.

Knowledge curation layer

Knowledge curation layer is devised to enable the creation and evolution of both health and wellness knowledge. The knowledge is created either by the domain expert or knowledge engineer by using expert-driven or data-driven approaches. The Expert-Driven module provides a set of rule authoring components to allow specialists to describe in a logical form causes or premises and effects or conclusions, e.g., "if gender is male and age lower than 65 then activity level should be moderate". The authoring process is further supported through evidence materials and domain vocabularies to confirm the viability of the rules and facilitate their elaboration. The Data-Driven module leverages the contents of the life-log for the automatic generation of rules. To that end, a data broker interface is defined to glean the contents of interest from the data persisted in DCL based on the features or attributes chosen by the expert, e.g., gender, emotional state and activity level. The process is automated by selecting and learning diverse mining models to discover and represent the underlying relationship among the considered health and wellness factors.

In both expert-driven and data-driven cases the generated rules are verified in terms of consistency and validated to avoid potential violations or redundancy with existing rules prior to be stored into the Knowledge Bases. KCL rules are not only persisted in traditional knowledge bases but also indexed according to salient conditions of these rules, also called "causes" or "situations". These situations refer to particular attributes of the rules than can be monitored by the platform and used for triggering the execution of specific rules. Accordingly, during the rule creation process the expert can select these condition attributes for their particular monitoring at DCL. The categorization of the knowledge bases through these indexes is particularly considered to enhance the performance of the reasoning processes hosted in SCL. In fact, once a situation is detected only its associated rules are shared with SCL through the Knowledge-Sharing Interface upon request of this layer.

The evolution of the knowledge is procured through two main mechanisms. On the one hand, the expert creation process can be considered as a sort of maintenance per se. In that view, rules may be dynamically updated or replaced based on new health and wellness findings from experts. On the other hand, rules can be added, replaced or modified through the data-driven approach while using new life-log contents collected from different users.

Service curation layer

Service curation layer provides the means to transform the data, information and knowledge curated by DCL, ICL and KCL into actual health and wellness support services. The services are managed by the Service Orchestrator, in charge of attending the potential requests, invoking the necessary services and coordinating the processes involved in the curation of the services. The requests may be of various types, i.e., scheduled on time—e.g, "every day at 8 am"; triggered by direct user queries—e.g., "suggest me an exercise plan for today's workout"; or based on events—e.g., "user arrives at home". The last type of request particularly relates to the concept of situation, already described in previous sections. The idea is that the LLM component from DCL triggers SCL whenever a situation event is identified in order to generate a new recommendation for the user.

The services needed to satisfy a given request are invoked from an extensible catalog containing reference and auxiliary services. A major reference service is devised for this architecture for the generation of personalized health and wellness recommendations. This service consists of two parts. First, generalized recommendations are developed by the Recommendation Builder component through reasoning on the user profile and life-log data provided by DCL and the knowledge facilitated by KCL for the specific domain of the service. In the case of handling a request derived from a situation detection the indexed rules hosted by KCL are particularly employed. Second, the recommendations undergo a personalization process through the Recommendation Interpreter in order to deliver the one that best fits the user interests and demands. Through this component all the potential recommendations are filtered based on the user preferences, conditions and possessions, as well as their actual context. Thus, for example, when the objective of the recommendation is to encourage the user to exercise, cycling would be avoided if the user does not own a bike, or a visit to the regular gym omitted in case the person is on a business trip. Similarly, this component can delay the delivery of a given recommendation when it is considered not to be a convenient moment for the user, e.g., if the person is in the middle of a meeting. Prior to be communicated to the user, the recommendation is refined to be easily interpreted by including multimedia contents to increase the interpretability and also incorporating motivational and engagement strategies to foster the user interest and attention.

Supporting layer

The role of SL is to enrich the overall Mining Minds functionalities through advanced analytics, interactive and personalized UI/UX, implicit and explicit feedback analysis, and adequate privacy and security mechanisms.

The Analytics module is in charge of mining in a multi-dimensional and retrospective manner the data sets collected and curated from multiple users to reveal population health and wellness associations, patterns and trends. These trends may refer to current facts as well as expected or future tendencies. The exploration of present trends is performed through the Descriptive Analytics, which employs statistical techniques to relate explanatory variables of the persisted data. Thus for example, based on the analysis of the inferred people lifestyles, it can be found that there is a growing use of hot beverages among adolescents, which further relates to a dramatic increase of stress patterns. The discovery of potential future facts is carried out by the Predictive Analytics, which develops on the outcomes of the Descriptive Analytics to make forecasts by using regression and machine learning models. Descriptive and predictive analytics contents are organized by the Visualization Enabler, which adjusts the style of the information to be communicated to the users based on their expertise and role.

Evaluating the services supported by Mining Minds requires feedback from the users, which is here powered by the Feedback Analysis component. The sources of feedback may be of a diverse nature, ranging from explicit feedback provided by the user, for example, through questionnaires, to implicit feedback obtained from the user behavioral responses. Analyzing implicit and explicit feedback from the users is motivated by the aspects of functionality, content, and presentation. Functionality-based feedback refers to the findings obtained while comparing, for example, the system recommendations and the behavioral reaction of the user to those recommendations. Content-based feedback measures the user satisfaction with respect to the specific information provided as part of the delivered services. Finally, presentation-based feedback measures the human-computer interaction with respect to the user interface (UI), which is of particular utility to understand the user experience (UX). All these types of feedback are devised to help assessing the level of interest and adherence of users to the services provided through Mining Minds as well as to evolve and maintain the internal contents and processes handled by the platform.

Considering user preferences, habits or mood, the UI/UX module enables the end-user applications interface to be adapted accordingly. This adaptation is needed to adjust the human-computer interaction experience with respect to font size, theme, or audio levels, among other characteristics. Two main components are involved in this process. First, the UI Interaction Tracker collects the data from the interaction between the person and the application to analyze the user's ability to understand and use the system, e.g., the readability of the contents or the perceptibility of the controls. Then, the UX component measures the satisfaction level based on the analysis of the collected data. The immediate result is a dynamic adaptation of the UI based on the measurements extracted from the evaluation of the UX.

Given the sensitivity of the collected user data, privacy and security need to be assured and exhibited, not only for storage, but also during the processing and delivery of services. To that end, state-of-the-art cryptographic primitives along with indigenous protocols are considered. For secure storage, the AES standard is particularly used, whereas for oblivious processing, homomorphic encryption and private matching is used. Considering the intensive data flow between end-user applications and Mining Minds, data randomization techniques are used to ensure a high entropy for minimal leakage of information. An authorized model ensures the legitimate disclosure of personal data and services with users. Slow processing of information is a common byproduct of the encryption; thus, to assist partial swiftness to Mining Minds, sensitive and non-sensitive information is decoupled where required. Anonymization procedures are also considered to enable the use of the collected and mined users data by third party agents, e.g., for research purposes.

Mining Minds implementation

An initial implementation of the proposed framework particularly oriented to promote healthy lifestyles and physical activity management is described here. Mining Minds is a distributed platform where the cloud environment plays a key role for supporting both persistence and limitless computational power. The Mining Minds implementation has been deployed over a hybrid cloud combining Microsoft Azure public cloud environment [27] and a Xen private cloud [28] for the big data storage, which runs over Hadoop File System with MapReduce [29]. For better scalability and performance each layer is deployed over a separate virtual instance on Microsoft Azure. DCL, ICL, KCL and SCL are hosted on standard Microsoft Azure instances with Windows Server 2012 R2 as guest operating system [30], while SL functionalities partake of the others. The cloud-based deployment of layers allows the encapsulation of their responsibilities as well as the re-usability of their features through an inter-layer communication. This communication is implemented by establishing service contracts among the layers, which communicate by means of RESTful web services [31] and high performance sockets [32]. The communication between MDS and DCL is implemented through sockets given its real-time nature. Similarly, a high performance socket-based implementation is particularly used for DCL-ICL communication for the transference of sensory data and context determination in real-time, and communication between DCL and the big data storage on private cloud. The most important service contracts with remaining layers are supported by DCL RESTful web services, which serve a data model with the structure of the Intermediate Database, here hosted by Microsoft SQL Server [33]. This data model is shared among the layers as an object model of service contract. The required data and information is populated by DCL and provided as responses to the upper layers.

To support active lifestyle services in this version ICL only implements the Activity Recognizer. This component consists of various steps that mainly combine signal processing and machine learning techniques to define a specific human activity recognition model capable of distinguishing among various commonplace activities [34]. The main input of this model is body motion data, namely, acceleration, which can be broadly obtained from smartphones and wearable inertial sensors. Acceleration is preferentially used here since it is the most prevalent sensor modality in standard activity recognition approaches [35]. A non-overlapping sliding window of three seconds is used for the data segmentation [36], and time and frequency features extracted for their discrimination potential [37]. The implemented model combines Support Vector Machines and Gaussian Mixture Models for the classification process, which have been demonstrated of particular utility in this domain [38, 39]. The developed Activity Recognizer further supports three operation modes depending on the available data registered from the user. Specifically, a hierarchical approach is developed so that the model can determine the user activity based only on the inertial data collected through the smartphone, smartwatch or a combination of smartphone and smartwatch data if available.

Health and wellness knowledge is defined by medical experts and hosted in the Knowledge Bases of KCL. To that end, a simple rule authoring tool [40] is considered for the rule creation. Evidences and domain vocabularies are particularized to the definition of physical management and activity promotion plans [41]. SCL processes the contents generated by DCL, ICL and KCL for the generation of personalized physical activity recommendations. After a request is processed by the service orchestrator, generalized recommendations are produced by applying rule-based reasoning [42] on the existing knowledge and user data. User health and wellness data is transformed into a proper input query by using auxiliary services hosted in the service catalog. Similarly, auxiliary services are implemented for user goal discovery, e.g., ideal weight and calories to be burned per day [43]. During the reasoning, the interpreter analyzes each rule in the knowledge bases and fires the appropriate rules using a forward chaining procedure [44]. Recommendations are personalized by using content-based filtration techniques [45] employing user-centric information such as activity level and preferred physical activities.

Security and privacy components of SL are distributed among the different layers. Encryption techniques are employed to withstand any compromise on data storage facility or its unauthorized access, as well as to make health-related data processing and evaluation HIPAA compliant. Concretely, AES [46], private matching [47] and multidimensional anonymization [48] have been chosen to support the encryption and control access. Moreover, since the systems are deployed on public clouds, processing over direct encryption without losing accuracy is required. The indigenously proposed system of oblivious term matching [49] is considered here to that end.

Health and wellness promotion services

Various exemplary applications and tools have been developed to showcase some of the potential health and wellness services supported by Mining Minds—Fig. 2. Personalized weight management is procured through an application that promotes activity routines customized to the user characteristics and preferences in order to attain a healthy weight. The app further provides the person with valuable information regarding their physical behavior, energy expenditure and weight loss patterns. Behavior change and healthy lifestyle promotion is intended through a personal coaching application which delivers action recommendations and educational facts upon detection of unhealthy physical conducts. Conversely to other digital health and wellness systems and platforms, Mining Minds is not only devised to support regular users or patients but also specialists. Medical experts are facilitated with a comprehensive tool to inspect users behavior, engagement and satisfaction in a continuous and retrospective manner. Apart from diverse statistics reporting personal goals, achievements and physical activity patterns, the tool allows the specialist to check the specific information and recommendations delivered by the platform to each particular user. Finally, an intuitive rule authoring tool has also been developed to enable the creation and management of the health and wellness knowledge exploited by Mining Minds. The main features and utilities of these applications and tools are described next.

Personalized weight management app

A poor estimation of calories and activities as well as an unrealistic definition of milestones represent two of the most common reasons for failure in most weight loss programs. Accordingly, the main objective of this service is to empower people in the control of their weight through a continuous track of exercise and energy consumption and a personalized physical routine promotion to achieve the daily expenditure goals. Users are initially requested to sign up into the application by entering their personal

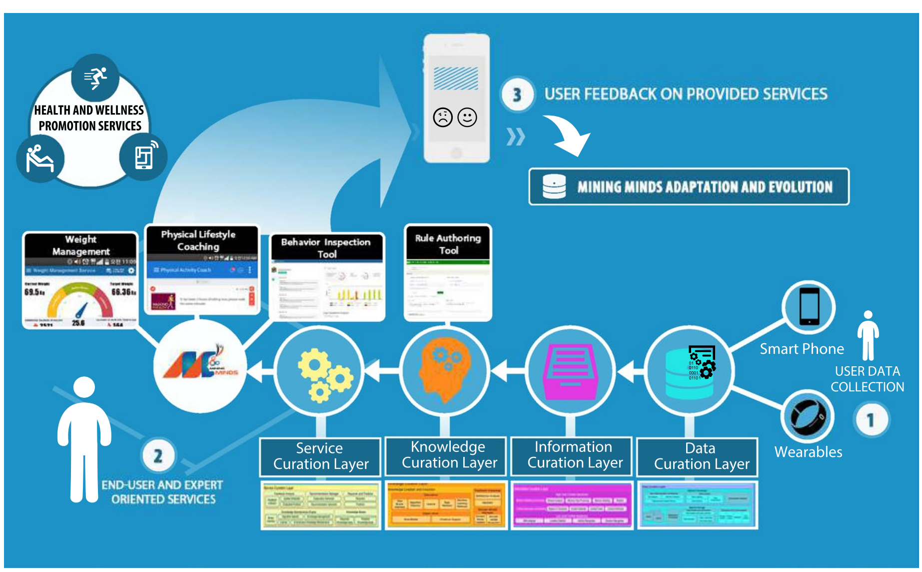


Fig. 2 Mining Minds health and wellness service scenario

information such as demographics—age, gender, weight and height, preferences in terms of activities and exercise level—sedentary, moderate or intense. All this information is securely stored and processed by the Mining Minds platform to calculate the user physical state, ideal weight, as well as the calories to be burned every day, all displayed for simple access on the app main dashboard—Fig. 3, top-left. The amount of calories burned by the user in the present day is also displayed on this view. This value is estimated by the platform by analyzing the user activity patterns. To determine these patterns, Mining

Fig. 3 Personalized weight management app

Minds elaborates on the acceleration data measured by the user smartphone, which is timely streamed through WiFi or 4G to the platform. To promote the user activity to achieve the daily calorie goal, exercise recommendations are given in an easy-to-understand manner. The recommendations contain precise indications on the duration of the activity and its execution style as well as motivational statements for encouraging the user. The recommended activities, their duration and intensity are personalized to each individual based on their profile. The evolution of the user's actual weight with respect to the planned one is presented in a different frame—Fig. 3, top-right. Here the user can easily self-report their current weight upon timely request of the platform. Other supportive features of the application provide the user with statistical analysis of burned calories and activity patterns —Fig. 3, bottom-left, and a calendar view of the user comportment—Fig. 3, bottom-right, specifically devised to support users in their self-monitoring and control.

Physical lifestyle coaching app

Behavior change and healthy lifestyle promotion constitute central objectives in public health interventions. The service defined here explores sophisticated coaching mechanisms to raise people's health awareness while inducing wholesome activity habits, changing unhealthy routines and educating on healthier physical lifestyles. To that end the developed application continuously captures the user's body motion data registered through the inertial sensors of the smartwatch and smartphone. The data is then streamed to Mining Minds which processes it to infer the user's behavior and determine potential risk or unhealthy situations. After an unhealthy behavior is detected, e.g., "one hour of continuous sitting", the platform automatically generates a personalized physical recommendation or healthy educational fact, e.g., "stretch your legs, arms and back". Recommendations and facts are conveniently delivered according to the user context and availability, and displayed on the application main screen in a timeline view—Fig. 4. Both recommendations and facts are also accompanied by multimedia contents—video, images and audio—to instruct the user on how to follow them as well as to attract and increase their interest and understanding. Moreover, users can value the delivered recommendations and facts according to their experience—"likes"/"dislikes"— and also provide comments on them—e.g., "I cannot carry out the recommended stretching exercises" or "My back hurts when I bend my waist". This information constitutes a key source of feedback for experts and Mining Minds itself to realize the comprehensiveness, applicability and impact of the services delivered by the platform.

Behavior inspection tool

Intelligent monitoring and smart coaching mechanisms are not planned to replace the role of specialists but rather complement it. In fact, the idea is that not only patients but also medical experts can benefit from the data, information, knowledge and services handled by Mining Minds. The expert inspection tool developed here is particularly devised to facilitate and expedite the task of health and wellness counseling specialists. The tool— Fig. 5—presents in an intuitive yet comprehensive fashion some of the most prominent user-centric information managed by the platform. On the left side of this expert view the specialist can check the recommendations and facts delivered by Mining

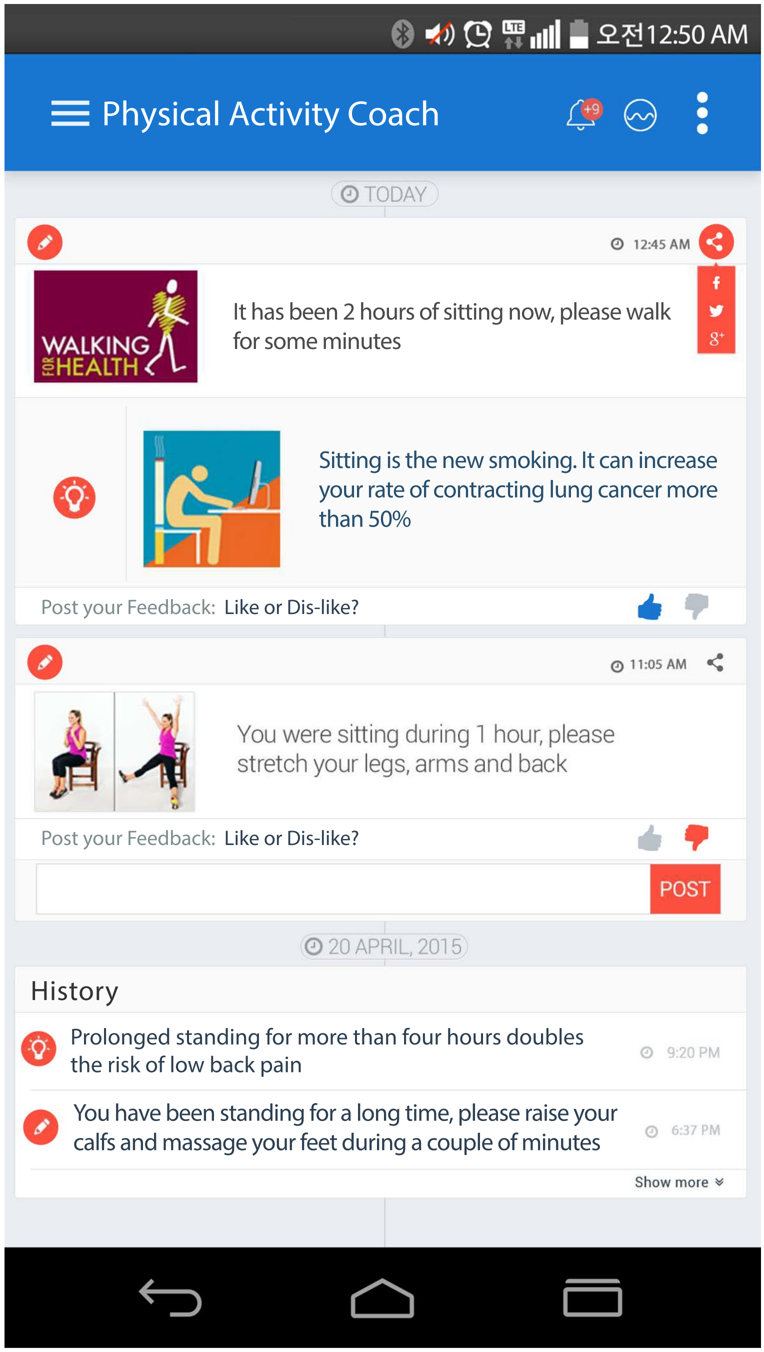


Fig. 4 Physical lifestyle coaching app

Minds to the user, the reason behind these suggestions as well as the feedback provided on them. On the right side, diverse sort of analytics describing the physical achievements of the user, their behavioral patterns and their rating of recommendations and facts are shown. Energy expenditure achievements and physical activity patterns are displayed in a daily, weekly and monthly basis, thus providing the expert with a detailed view of the user past and present status. The user feedback analytics is of particular interest to help

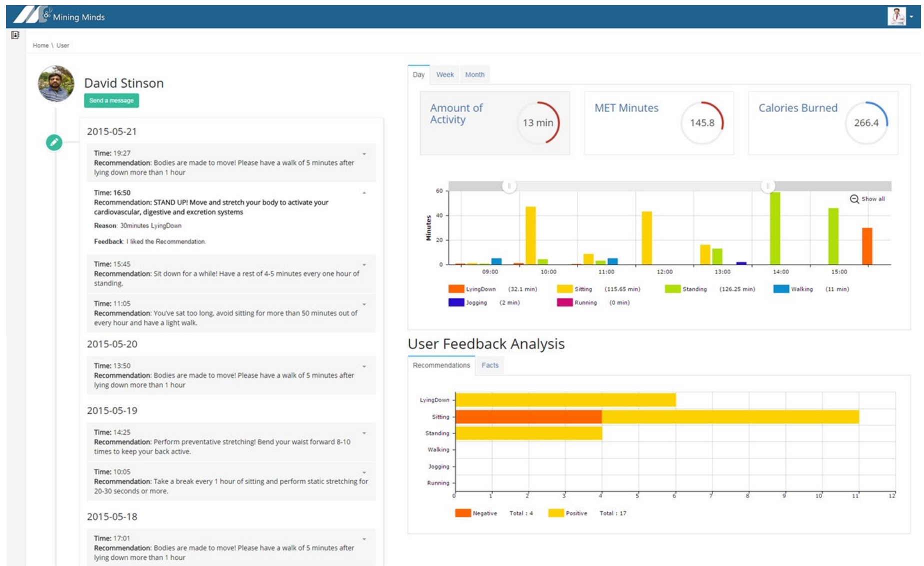


Fig. 5 Behavior inspection tool

experts identify what kind of recommendations and facts are more positively valued and which ones may not be accepted. The tool is also incorporated with a feature that allows the specialist to directly communicate with the user through the developed apps by sending comments in the form of notifications. Through this tool experts can in principle deal with more users while reducing the time required for the assessment of their progresses and evolution.

Rule authoring tool

Health and wellness experts are not only consumers of the services supported by Mining Minds but also content producers. The creation and management of Mining Minds health and wellness knowledge is handled by the specialists through an advanced rule authoring tool—Fig. 6. This rule authoring tool is an adapted version of a prior one first introduced in [40]. The rule authoring tool provides domain experts with an easy to use dashboard to manage the existing rules, thus making possible their addition, update or deletion. An intuitive environment is provided for the creation of new rules and associated meta-information. The rule authoring tool incorporates a sophisticated physical activity wellness model which incorporates multiple domain concepts and vocabularies facilitating the rule creation task. The tool is also equipped with intelligent code completion technology to expedite the rule creation process and reduce the chance of errors. After the rule is created, the expert can simply save it, thus making it available for its use in Mining Minds.

Evaluation and discussion

A preliminary evaluation of the implemented version of the platform and services is performed here. An important asset of the platform refers to the curation and persistence of sensory data by DCL. Most health applications delete sensory data after processing it;

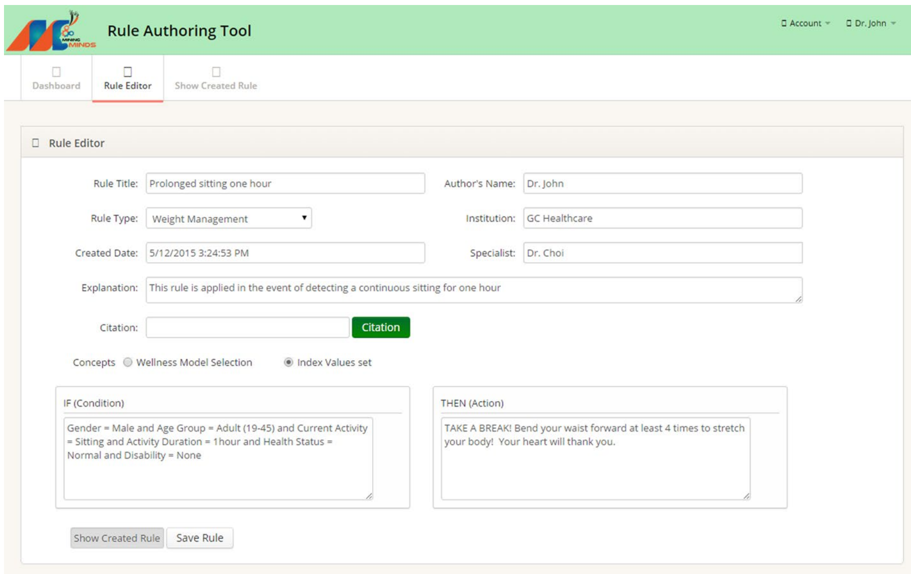


Fig. 6 Rule authoring tool

however, persisting this information is of worth for generating datasets that can be used to evolve the knowledge models or learn new ones. To benchmark DCL capabilities, the accuracy and performance of the platform in the collection, processing and storage of the sensory data is measured. To that end, continuous data service calls over the period of 24 h are generated and evaluated. The accuracy is measured by the rate of missing data packets, here summarized in Table 1. The results show a very low error, 0.06 % in average, which means that practically all the sensory data sent to the platform is safely processed. The performance, presented in Table 2, measures the capacity of the system to store the data packets into the platform storage. The stress test shows a high consistency with the increasing usage of the system, which is capable of writing 2.2 requests or packets per second in average, each one composed by 7800 records of sensory data.

The capability of ICL for inferring user activities presents important advantages with respect to other wellness systems, which frequently rely on simple step counting for activity tracking. For example, it permits to derive more precisely the user energy expenditure based on the cost of each performed activity, specially for those that do not entail any ambulation. To evaluate the potential of the implemented activity recognition

Table 1 Accuracy of the data curation process

No. of service calls	No. of missed data packets	Packet loss error (%)
30,000	6	0.02
60,000	22	0.04
90,000	39	0.04
120,000	55	0.05
150,000	96	0.06
180,000	308	0.17
Average		0.06

Table 2 Performance of the data persistence process for different operation runs

Run duration (h)	2	4	6	8	10	12	14	16	18	20	22	24
Performance (avg data writes/s)	2.20	2.21	2.20	2.20	2.10	2.21	2.21	2.21	2.20	2.20	2.20	2.20

model ten volunteers aged from 23 to 37 years were requested to perform the supported activities, namely, "walking", "jogging", "running", "stretching", "sweeping", "eating", "sitting", "standing" and "lying down". The performance of the model is evaluated by comparing both actual and detected activities for three different scenarios defined upon the available sensing technology, respectively, smartphone—Table 3, smartwatch—Table 4 and both devices—Table 5. The results prove notable recognition capabilities in general, yielding an overall F-score of 0.93 for the case in which only the smartphone is used for registering the user's body motion, 0.92 when the smartwatch is solely employed, and 0.95 when both devices are used.

An initial user-centric analysis is also performed in terms of adherence to the provided recommendations. Ten independent volunteers between the ages of 26 and 38 years were asked to use the developed applications during a couple of weeks to measure the response time to the recommendations generated by the platform. This time accounts for the period elapsed since the user receives a recommendation and follows it. The

Table 3 Activity recognition performance when operating on the smartphone data

Activity	SE	SP	PPV	NPV	F-score
Eating	0.87	0.99	0.86	0.99	0.86
Running	0.97	1.00	1.00	1.00	0.99
Sitting	0.94	0.98	0.93	0.98	0.94
Standing	0.88	0.99	0.95	0.98	0.91
Walking	0.99	0.99	0.98	1.00	0.99
Jogging	0.99	1.00	0.98	1.00	0.99
Stretching	0.96	0.99	0.91	1.00	0.93
Sweeping	0.93	0.99	0.90	0.99	0.91
Lying down	0.84	1.00	0.92	0.99	0.88

Each metric correspond to *SE* sensitivity, *SP* specificity, *PPV* positive predictive value, *NPV*, negative predictive value and F-score

Table 4 Activity recognition performance when operating on the smartwatch data

Activity	SE	SP	PPV	NPV	F-score
Eating	0.82	0.99	0.86	0.99	0.84
Running	0.96	1.00	0.94	1.00	0.95
Sitting	0.93	0.94	0.85	0.97	0.89
Standing	0.85	0.99	0.94	0.97	0.90
Walking	0.96	0.98	0.95	0.98	0.96
Jogging	0.92	1.00	0.97	1.00	0.95
Stretching	0.93	1.00	0.97	0.99	0.95
Sweeping	0.95	1.00	0.99	1.00	0.97
Lying down	0.85	1.00	0.93	0.99	0.89

Each metric correspond to *SE* sensitivity, *SP* specificity, *PPV* positive predictive value, *NPV* negative predictive value and F-score

Table 5 Activity recognition performance when operating on both smartphone and smartwatch data

Activity	SE	SP	PPV	NPV	F-score
Eating	0.89	1.00	0.88	1.00	0.88
Running	0.97	1.00	0.99	1.00	0.98
Sitting	0.95	0.98	0.94	0.98	0.95
Standing	0.91	0.99	0.95	0.98	0.93
Walking	0.99	0.99	0.98	1.00	0.99
Jogging	0.98	1.00	0.98	1.00	0.98
Stretching	0.97	0.99	0.92	1.00	0.94
Sweeping	0.94	1.00	0.94	1.00	0.94
Lying down	0.90	1.00	0.93	1.00	0.92

Each metric correspond to *SE* sensitivity, *SP* specificity, *PPV* positive predictive value, *NPV* negative predictive value and F-score

average number of recommendations per day were 9, ranging from 5 to 14. The subjects response time varied from 1 min to 1 h, with average values shown in Table 6. These results may give some hints on the interest shown in the use of these services, although further analysis, including more subjects and longer time spans, is required to obtain solid conclusions.

Finally, the effectiveness and usability of the developed expert tools is also assessed. To that end, different aspects of the tools were evaluated by six medical experts—two nutritionists, two fitness instructors and two nurses—from an independent health and wellness counseling company. The experts were instructed on how to use the tools and then provided with a set of questionnaires to evaluate their look and feel, interface layout complexity, time required to access a given resource or create a new rule and the understandability and correctness of the concepts and contents facilitated by these tools. The results of the evaluation prove a satisfaction level of 7.5 out of 10 in average. The aspects that were more highly rated correspond to the usefulness, organization and simple access to the displayed health and wellness information. For the behavior inspection tool the experts particularly valued having a user-centric description of the behavioral patterns plus the possibility of identifying the acceptability of the delivered recommendations through the feedback analysis report. For the rule authoring tool the specialists especially considered the benefits provided by the health and wellness models during the rule creation process, although they showed some concerns regarding the amount of time required to write a given rule. All the positive and negative feedback obtained through all these evaluations is being considered at the moment for evolution and improvement of the developed services.

It is worth noting that all the developed apps and tools have been designed as end-user interfaces to the contents and services curated by Mining Minds, thus presenting important advantages for the customers, such as an effective reduction of the resources

Table 6 Average user response time (in minutes) to recommendations

User	1	2	3	4	5	6	7	8	9	10
Avg response time	24.47	34.44	3.42	5.38	40.44	7.21	28.29	13.99	8.56	36.84

consumption—mainly in terms of storage, computation and battery, no need of regular updates of the client application, shareability of contents among diverse systems and applications, as well as a more dynamic and interactive experience. Despite these important advantages, there are some limitations that need especial consideration. Mining Minds builds on the assumption of having most mobile devices and systems of the Internet of Things fully and seamlessly connected in the near future. However, this condition is currently not always satisfied; therefore, the applications may require to support temporary local storage and offline data transmission to overcome potential network disconnections. Another open issue refers to the cost of the communication between the apps and the platform. Applications such as the ones presented here operate over WiFi and 4G interfaces. While the use of WiFi presents no economic burden, some users could be concerned about using their data plans when huge amounts of data need to be transferred. For example, the developed user applications transmit around 500kB/min to communicate the sensory data to the platform, which translates into approximately 30GB/month when used non-stop. With the advent of 5G communications, flat-rate data plans are expected to become commonplace, thus helping to reduce the possible burden for the end-user. In either case, the utilization of compressed sensing techniques [50] is particularly envisioned for the future to make the data transmission more efficient. These mechanisms and other sophisticated strategies are also worth considering to reduce battery consumption, for example, by interrupting the transmission of sensory data during periods of user inactivity.

Conclusions

This work has presented Mining Minds, a novel digital framework for personalized healthcare and wellness support. The framework has been neatly designed taking into account crucial requirements of digital health and wellness systems. This work has also described a unique architecture defined to provide the necessary functionality to enable curation and mining of data, information, knowledge and services for personalized health and wellness support. An initial realization of the key architectural components as well as various exemplary applications and tools showcasing some of the benefits provided by Mining Minds have also been presented and evaluated. The work is ongoing to complete the implementation of the devised architecture with new additional components, as well as to evaluate its services on a large-scale testbed.

Consent

Written informed consent was obtained from the participants for publication of this case report and any accompanying images.

Authors' contributions

OB is the Technical and Scientific Coordinator of the Mining Minds project and lead researcher of the Information Curation Layer. MBA is the Architect Lead of the Mining Minds project and lead researcher of the Data Curation Layer. WAK is the lead researcher of the Supporting Layer. MA is the lead researcher of the Service Curation Layer. MH is the lead reseacher of the Knowledge Curation Layer. BHK is a Senior Consultant of the Mining Minds project. SL is the Principal Investigator of the Mining Minds project. All authors read and approved the final manuscript.

Author details

[1] Department of Computer Engineering, Kyung Hee University, 1732 Deokyoungdae-ro, Giheung-ug, Yongin-si 446-701, Korea. [2] School of Computing and Information Systems, University of Tasmania, Churchill Avenue Hobart, Tasmania 7005, Australia.

Acknowledgements
This work was supported by the Industrial Core Technology Development Program (10049079, Development of mining core technology exploiting personal big data) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea).

Competing interests
The authors declare that they have no competing interests.

Declarations
Publication costs for this article were funded by the Industrial Core Technology Development Program under grant agreement 10049079.

Published: 15 July 2016

References

1. Hafner T, Shiffman J. The emergence of global attention to health systems strengthening. Health Policy Plan. 2013;28(1):41–50.
2. Hood L, Flores M. A personal view on systems medicine and the emergence of proactive P4 medicine: predictive, preventive, personalized and participatory. N Biotechnol. 2012;29(6):613–24.
3. Matheson EM, King DE, Everett CJ. Healthy lifestyle habits and mortality in overweight and obese individuals. J Am Board of Fam Med. 2012;25(1):9–15.
4. Gillen MM, Markey CN, Markey PM. An examination of dieting behaviors among adults: links with depression. Eat Behav. 2012;13(2):88–93.
5. Demark-Wahnefried W, Jones LW. Promoting a healthy lifestyle among cancer survivors. Hematol Oncol Clin North Am. 2008;22(2):319–42.
6. Swan M. Health 2050: the realization of personalized medicine through crowdsourcing, the quantified self, and the participatory biocitizen. J Pers Med. 2012;2(3):93–118.
7. Powell AC, Landman AB, Bates DW. In search of a few good apps. JAMA. 2014;311(18):1851–2.
8. Withings Activité. http://www.withings.com/eu/en/products/activite. Accessed 29 Aug 2015.
9. Garmin Vivofit. http://sites.garmin.com/en-US/vivo/vivofit/. Accessed 29 Aug 2015.
10. Fitbit Surge. https://www.fitbit.com/surge. Accessed 29 Aug 2015.
11. Misfit Shine. http://www.misfitwearables.com/products/shine. Accessed 29 Aug 2015.
12. Banos O, Villalonga C, Damas M, Gloesekoetter P, Pomares H, Rojas I. Physiodroid: combining wearable health sensors and mobile devices for a ubiquitous, continuous, and personal monitoring. ScientificWorldJournal. 2014;2014(490824):1–11.
13. Oresko JJ, Jin Z, Cheng J, Huang S, Sun Y, Duschl H, Cheng AC. A wearable smartphone-based platform for real-time cardiovascular disease detection via electrocardiogram processing. IEEE Trans Inf Technol Biomed. 2010;14(3):734–40.
14. Hermens H, op den Akker H, Tabak M, Wijsman J, Vollenbroek M. Personalized coaching systems to support healthy behavior in people with chronic conditions. J Electromyogr Kinesiol. 2014;24(6):815–26.
15. Chen P-H, Chen H-M. Framework design-integrating an android open platform with multi-interface biomedical modules for physiological measurement. J Converg Inf Technol. 2012;7(12):310–9.
16. Fortino G, Giannantonio R, Gravina R, Kuryloski P, Jafari R. Enabling effective programming and flexible management of efficient body sensor network applications. IEEE Trans Hum Mach Syst. 2013;43(1):115–33.
17. Gaggioli A, Pioggia G, Tartarisco G, Baldus G, Corda D, Cipresso P, Riva G. A mobile data collection platform for mental health research. Personal Ubiquitous Comput. 2013;17(2):241–51.
18. Yang G, Xie L, Mantysalo M, Zhou X, Pang Z, Xu LD, Kao-Walter S, Chen Q, Zheng L-R. A health-iot platform based on the integration of intelligent packaging, unobtrusive bio-sensor, and intelligent medicine box. IEEE Trans Indus Inf. 2014;10(4):2180–91.
19. Banos O, Villalonga C, Garcia R, Saez A, Damas M, Holgado JA, Lee S, Pomares H, Rojas I. Design, implementation and validation of a novel open framework for agile development of mobile health applications. Biomed Eng Online 2015;14(S2:S6),1–20.
20. Estrin D, Sim I. Open mhealth architecture: an engine for health care innovation. Science. 2010;330(6005):759–60.
21. Google Fit. https://developers.google.com/fit/overview. Accessed 29 Aug 2015.
22. Sami S. https://developer.samsungsami.io/sami/sami-documentation/. Accessed 29 Aug 2015.
23. Apple HealthKit. https://developer.apple.com/healthkit/. Accessed 29 Aug 2015.
24. Banos O, Bilal-Amin M, Ali-Khan W, Afzel M, Ahmad M, Ali M, Ali T, Ali R, Bilal M, Han M, Hussain J, Hussain M, Hussain S, Hur TH, Bang JH, Huynh-The T, Idris M, Kang DW, Park SB, Siddiqui M, Vui L-B, Fahim M, Khattak A-M, Kang B-H, Lee S. An innovative platform for person-centric health and wellness support. In: Proceedings of the international work-conference on bioinformatics and biomedical engineering; 2015.
25. Banos O, Bilal-Amin M, Ali-Khan W, Afzel M, Ali T, Kang B-H, Lee S. Mining minds: an innovative framework for personalized health and wellness support. In: Proceedings of the 9th international conference on pervasive computing technologies for healthcare (PervasiveHealth 2015); 2015.
26. Banos O, Bang JH, Hur TH, Siddiqui M, Huynh-The T, Vui L-B, Ali-Khan W, Ali T, Villalonga C, Lee S. Mining human behavior for health promotion. In: Proceedings of the 37th annual international conference of the IEEE engineering in medicine and biology society (EMBC 2015); 2015.
27. Microsoft Azure. http://azure.microsoft.com. Accessed 29 August 2015.

28. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A. Xen and the art of virtualization. ACM SIGOPS Oper Sys Rev. 2003;37(5):164–77.
29. Padhy RP. Big data processing with hadoop-mapreduce in cloud systems. Int J Cloud Comput Serv Sci. 2012;2(1):16–27.
30. Microsoft Windows Server. http://www.microsoft.com/en-us/server-cloud/products/windows-server-2012-r2/. Accessed 29 August 2015.
31. Richardson L, Ruby S. RESTful web services. Sebastopol: O'Reilly Media Inc; 2008.
32. Kim K, Kim C, Jung S-I, Shin H-S, Kim J-S. Inter-domain socket communications supporting high performance and full binary compatibility on xen. In: Proceedings of the Fourth ACM SIGPLAN/SIGOPS international conference on virtual execution environments; 2008. p. 11–20.
33. Microsoft SQL Server. http://www.microsoft.com/en-us/server-cloud/products/sql-server/. Accessed on 29 Aug 2015.
34. Banos O, Damas M, Pomares H, Rojas F, Delgado-Marquez B, Valenzuela O. Human activity recognition based on a sensor weighting hierarchical classifier. Soft Comput. 2013;17:333–43.
35. Lara O, Labrador M. A survey on human activity recognition using wearable sensors. IEEE Commun Surv Tutor. 2012;99,1–18.
36. Banos O, Galvez J-M, Damas M, Pomares H, Rojas I. Window size impact in human activity recognition. Sensors. 2014;14(4):6474–99.
37. Preece SJ, Goulermas JY, Kenney LPJ, Howard D. A comparison of feature extraction methods for the classification of dynamic activities from accelerometer data. IEEE Trans Biomed Eng. 2009;56(3):871–9.
38. Han M, Vinh LT, Lee Y-K, Lee S. Comprehensive context recognizer based on multimodal sensors in a smartphone. Sensors. 2012;12(9):12588–605.
39. Banos O, Damas M, Pomares H, Prieto A, Rojas I. Daily living activity recognition based on statistical feature quality group selection. Expert Syst Appl. 2012;39(9):8013–21.
40. Ali T, Hussain M, Ali Khan W, Afzal M, Lee S. Authoring tool: acquiring sharable knowledge for Smart CDSS. In: International conference of the IEEE engineering in medicine and biology society; 2013, p. 1278–81.
41. Ainsworth BE, Haskell WL, Herrmann SD, Meckes N, Bassett DR, Tudor-Locke C, Greer JL, Vezina J, Whitt-Glover MC, Leon AS. 2011 compendium of physical activities: a second update of codes and met values. Med Sci Sports Exerc. 2011;43(8):1575–81.
42. Forgy CL. Rete: A fast algorithm for the many pattern/many object pattern match problem. Artif Intell. 1982;19(1):17–37.
43. Schofield W. Predicting basal metabolic rate, new standards and review of previous work. Hum Nutr Clin Nutr. 1984;39:5–41.
44. Baget J-F. Improving the forward chaining algorithm for conceptual graphs rules. In: International conference on principles of knowledge representation and reasoning; 2004, p. 407–14.
45. Gong S. Learning user interest model for content-based filtering in personalized recommendation system. Int J Digit Content Technol Appl. 2012;6(11):155–62.
46. Daemen J, Rijmen V. The design of rijndael: AES—the advanced encryption standard. Heidelberg: Springer; 2002.
47. Zhong S, Chen T. An efficient identity-based protocol for private matching. Int J Commun Syst. 2011;24(4):543–52.
48. Ghinita G, Karras P, Kalnis P, Mamoulis N. A framework for efficient data anonymization under privacy and accuracy constraints. ACM Trans Database Syst. 2009;34(2):9.
49. Pervez Z, Awan AA, Khattak AM, Lee S, Huh E-N. Privacy-aware searching with oblivious term matching for cloud storage. J Supercomput. 2013;63(2):538–60.
50. Li S, Xu LD, Wang X. A continuous biomedical signal acquisition system based on compressed sensing in body sensor networks. IEEE Trans Indus Inf. 2013;9(3):1764–71.

Section 1.2

Data Curation Layer (DCL) Paper

sensors

MDPI

Article

On Curating Multimodal Sensory Data for Health and Wellness Platforms

Muhammad Bilal Amin [1], Oresti Banos [2], Wajahat Ali Khan [1], Hafiz Syed Muhammad Bilal [1], Jinhyuk Gong [1], Dinh-Mao Bui [1], Soung Ho Cho [1], Shujaat Hussain [1], Taqdir Ali [1], Usman Akhtar [1], Tae Choong Chung [3,*] and Sungyoung Lee [1]

1 Ubiquitous Computing Lab, Department of Computing Engineering, Kyung Hee University, Giheung-gu, Yongin-si, Gyeonggi-do, Seoul 446-701, Korea; mbilalamin@oslab.khu.ac.kr (M.B.A.); wajahat.alikhan@oslab.khu.ac.kr (W.A.K.); bilalrizvi@oslab.khu.ac.kr (H.S.M.B.); kjh@oslab.khu.ac.kr (J.G.); mao@oslab.khu.ac.kr (D.-M.B.); chosoungho@oslab.khu.ac.kr (S.H.C.); shujaat.hussain@oslab.khu.ac.kr (S.H.); taqdir.ali@oslab.khu.ac.kr (T.A.); usman@oslab.khu.ac.kr (U.A.); sylee@oslab.khu.ac.kr (S.Y.L.)

2 Telemedicine Group, Center for Telematics and Information Technology, University of Twente, Enschede 7500AE, The Netherlands; o.banoslegran@utwente.nl

3 Intelligent System Laboratory, Department of Computer Engineering, Kyung Hee University, Giheung-gu, Yongin-si, Gyeonggi-do, Seoul 446-701, Korea

* Correspondence: tcchung@khu.ac.kr; Tel.: +82-31-201-2569

Academic Editor: Panicos Kyriacou
Received: 4 February 2016; Accepted: 21 June 2016; Published: 27 June 2016

Abstract: In recent years, the focus of healthcare and wellness technologies has shown a significant shift towards personal vital signs devices. The technology has evolved from smartphone-based wellness applications to fitness bands and smartwatches. The novelty of these devices is the accumulation of activity data as their users go about their daily life routine. However, these implementations are device specific and lack the ability to incorporate multimodal data sources. Data accumulated in their usage does not offer rich contextual information that is adequate for providing a holistic view of a user's lifelog. As a result, making decisions and generating recommendations based on this data are single dimensional. In this paper, we present our Data Curation Framework (DCF) which is device independent and accumulates a user's sensory data from multimodal data sources in real time. DCF curates the context of this accumulated data over the user's lifelog. DCF provides rule-based anomaly detection over this context-rich lifelog in real time. To provide computation and persistence over the large volume of sensory data, DCF utilizes the distributed and ubiquitous environment of the cloud platform. DCF has been evaluated for its performance, correctness, ability to detect complex anomalies, and management support for a large volume of sensory data.

Keywords: data curation; multimodal sensory data; data acquisition; lifelog; healthcare; wellness platform

1. Introduction

In recent years, there has been a shift in the way healthcare is handled and its supporting systems. This change has made a drastic impact on the design of conventional healthcare models. Instead of late disease management and cure, these models are focusing on preventative-personalized health. Consequently, an opportunity is provided for healthcare providers to focus on when, where, and how; care and support are delivered to the particular patient and service consumer [1,2]. The reason for this shift is the rising financial stress that healthcare systems have to face to support the growing demand for their services [3]. Therefore, service providers are pushing forward for wellness-based models and conducting research to investigate their effectiveness. The latest studies in biomedical healthcare have shown that the most prevalent diseases are partly caused or aggravated by poor lifestyle choices that people make in their daily routine. Unhealthy and fast-food diets, use of tobacco, and sedentary routines with a lack of exercise are among the potential contributors to developing illnesses and also limit the effectiveness of medical treatments [4–6].

With the advent of smart and personal devices, an opportunity has emerged for healthcare providers and biomedical researchers to empower people to take care of their health and wellness by providing them with timely, ubiquitous, and personalized support [7]. Consequently, the influx of fitness wearables with smartphone applications and systems supporting health and wellness has taken the market by storm [8]. For example, commercial systems and platforms such as Withings Activite [9], Garmin Vivofit [10], Fitbit Surge [11], Misfit Shine [12], Apple Watch [13] and HealthKit [14], Samsung Gear [15], LG smartwatches [16], Google Fit [17], and Microsoft Band [18] with Microsoft Health [19], all primarily consist of sensor-based bracelets accompanied by mobile apps, and provide some basic health recommendations based on the measured steps, calories, or hours of sleep. In parallel, research groups are also working on health and wellness systems that can alert of physical conditions [20] or detect chronic illnesses [21]. Despite this enormous effort by the industry and research, most current solutions are single-device focused and have a limited scope [1]. Therefore, they are unable to generate a context-rich user lifelog which can provide a holistic view of user activity and behavior [22]. Such lifelogs are a necessity for the evolutionary wellness systems that support the self-quantification of its users [23]. Moreover, a context-rich lifelog is also a low-cost way to acquire valuable user information on which effective interventions from healthcare professionals can be based [24]. Considering the limitations of existing efforts as an opportunity, we have proposed and implemented a comprehensive cloud-based sensory data acquisition framework called Data Curation Framework or DCF.

The contribution of Data Curation Framework (DCF) is aligned with the definition of lifelog, meaning it is a black box of user life events [22]. Multidimensional insights into user activities and behaviors require a context-rich lifelog that can be developed by the accumulation of data from a larger set of multimodal data sources [22,25]. Keeping this perspective as the primary motivation, DCF implements five core concepts in its design philosophy (illustrated in Figure 1). (i) The ability to continuously sense for raw sensory data from multimodal data sources in real time; (ii) The device-independent acquisition of raw sensory data. This concept contributes to the compatibility of DCF for IoT-based environments [26]. This property increases the ability of DCF to integrate a larger set of sensory devices; (iii) The induction of a larger set of sensory devices results in a richer context. DCF provides comprehensive curation of this context over a user lifelog. This context-rich lifelog can be used in multidimensional ways, e.g., data accumulated from a smartphone, a smartwatch, and a depth camera can accurately identify the context of a user posture in an environment. Therefore, a health and wellness platform using DCF can make recommendations not only pertaining to his activity but also the later on effect of that activity on his muscular health; (iv) DCF is equipped with a lifelog monitoring tool called LLM. In comparison with device-based activity recognition, lifelog monitoring looks for anomalies over richer context occurring over time. For reliability, expert-driven rules [27] provide intelligence to this monitoring (Rule creation is not part of the DCF scope); (v) DCF provides persistence to support the large volume of heterogeneous and multimodal raw sensory data associated with the lifelog. This property enables DCF to support the forthcoming concepts of data-driven knowledge generation [28], descriptive [29] and predictive analytics [30], and visualization [29].

DCF is a cloud-based implementation, as the cloud supports the infrastructure level DCF requirements. For continuous sensing, the ubiquitous nature of the cloud provides DCF with the ability to acquire sensory data in different contexts and environments. Similarly for device independence, the cloud provides a central yet scalable computational resource that can accumulate sensory data from clients without being concerned with their computational abilities. For lifelog maintenance and monitoring, the cloud provides a hub for context curation and monitoring for anomalies detection. Lastly, to support the volume of data accumulated by DCF, the cloud provides a big data platform.

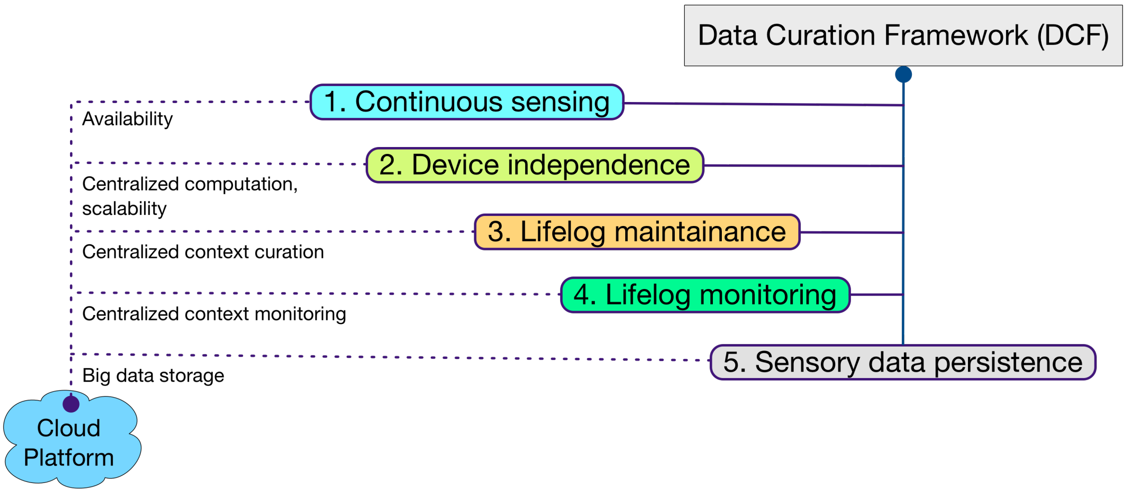


Figure 1. Data curation framework (DCF) philosophy.

DCF is currently part of the implementation of our ongoing research project called the Mining Minds platform [31]. DCF executes as a separate layer (Data Curation Layer or DCL) at the infrastructure level of the Mining Minds architecture [1]. It accumulates raw sensory data in real time from multimodal data sources and provides curation to the context identified on that data via the Mining Minds Information Curation Layer (ICL). DCL as the implementation of DCF monitors the context for anomalies based on the rules derived by the Mining Minds Knowledge Curation Layer (KCL). Furthermore, it preserves all of the sensory data in big data storage to support analytics and visualization provided by the Mining Minds Supporting Layer (SL). However, in this paper, we only discuss data curation as a framework independent of the Mining Minds platform.

The contents of this paper are as follows. Section 2 describes the related work with respect to DCF. Section 3 provides an overview of the methodology of DCF. Section 4 presents the details of the components participating in the implementation of DCF. Section 5 describes the execution scenarios of DCF in collaboration with a health and wellness platform. Section 6 evaluates various aspects of DCF including its accuracy, performance, scalability, its ability to perform reliable user lifelog monitoring and persistence. Section 7 concludes this paper.

2. Related Work

Following the boom of health- and wellness-based systems; lots of systems for sensory data accumulation and analysis have been proposed for the people's healthcare and wellbeing. For analysis of these systems we have classified them into three different categories: (i) Mobile Health, where implementation is focused on the evolution of smartphones and their embedded sensors; (ii) Wearable Health, where implementation is incorporating wearable bands and smartwatches in combination with smartphone or gateways; and (iii) Data Accumulation and Curation, where the implementation focuses on the accumulated data from variety of sensors for analysis. The categories are discussed in the following subsections.

2.1. Mobile Health

In the last decade, the influx of smartphone usage in our daily lives has been enormous. These smartphones are equipment with various multimodal sensors like accelerometer, PPG, and GPS, which in fact used intelligently, can accumulate user activity data in real-time for further utilization. Therefore, the combination of smartphone devices and web services provides more opportunities for healthcare systems to evolve with reduced cost. Researchers have indicated that new requirements and trends are embedded SIM and mobile health platforms (mHealth) [32]. In [33], authors presented the capability of mobile devices (mobility, low-cost, remote, and equipped with sensors) in conjunction with web-service based approach, providing services relating to healthcare. Furthermore, the highlights of the proposed approach are listed and compared with existing remote health monitoring systems. Moreover, in the case of elderly, mobile phones are already participating in providing healthcare services [34]. In [35], the authors presented a mobile phone-based platform to collect the psychological, physiological, and activity information of the user for mental health research. In [36], a mobile version of a data processing toolbox originally devised for computer-based architectures and principally used for human behavior modeling was provided. In [37], a middleware integrating multiple interfaces for multi-parameter monitoring of physiological measurement is proposed. In [38], tools have been suggested for the analysis of the provenance of mobile health data. In [39], authors present a client-server life logging platform. It enables context capturing by the mobile phone and data processing at the server to extract meaningful high-level context.

In [40], the authors present a lightweight lifelogging framework called UbiqLog. This framework provides a configurable interface to enable and disable sensors within the smartphone. Although more sensors can be added due to the compatibility of the framework's data model; however, the sensors have to be embedded within the form factor of smartphone device. Authors in [40] acknowledge the fact that lifelog data tends to grow rapidly; therefore, an external server like storage is required for permanent persistence. Lifelog generated by UbiqLog is neither shareable or reusable by external systems; furthermore, it does not support any monitoring on the data. Similarly in [41], authors present an open-source toolkit for sensory data collection over smartphones called AWARE. For conducting small-scale studies, AWARE stores all the accumulated data locally on the smartphone; however for larger scale, data is uploaded to the cloud. The toolkit provides a context plugin that can passively perform analysis on the accumulated data. In similarity with DCF, AWARE also implements a publish-subscribe model; however, it is only utilized for context sharing.

In [42], the DigMem system is proposed, which utilizes distributed mobile services, linked data, and machine learning to create rich and interactive HDMs (Human Digital Memory). An HDM, created by the data from the pervasive devices available in user's environment, produces a more dynamic and data-rich memory. Information such as how the user felt, where the user was, and the context of the environment can be established [42]. In contrast with DCF, the DigMem is a tri-component system with no service-based support for sharing its core asset data, i.e., memory boxes with health and wellness platforms. Although it collects data from multimodal sources, DigMem requires a compatible environment where its smartphone application can broadcast and look for information. Collected data is linked using semantic web technologies and presented as a memory box to its web application for visualization. DigMem's concept of linked data is conceptually similar to DCF's curation of user context; however, the focus of DigMem is on the visualization aspect of this data. Whereas in DCF, curated data is monitored for anomalies and its services provide lifelog as a user timeline for visualization. In [42], authors acknowledge the inclusion of big data in lifelogging systems and discusses the challenges involved in its incorporation in [43]. However, the implementation of big data in DigMem is not addressed.

2.2. Wearable Health

In the last decade, the usage of wearables has evolved from medical needs to a personalized accessory. Projects like SenseWare(SWA) [44] armband that collects physiological data using a bi-axial accelerometer, galvanic skin resistance (sweat rates), heat flux (heat dissipated from the body), and skin and near body temperature, have been utilized. In [45], the authors used SWA for the estimation of energy expenditure and step count during treadmill walking in cystic fibrosis patients, compared to healthy adults. In [46], the SWA has also been utilized for the monitoring of adherence in women with rheumatoid arthritis.

In the wearable health research Microsoft's SenseCam [47] is considered a revolutionary pervasive device [48]. It is equipped with a digital camera and multiple sensors, including sensors to detect changes in light levels, an accelerometer, a thermometer for detecting ambient temperature, and a passive infrared sensor for detecting the presence of people. SenseCam has been used in various studies, for example, in [49], authors use the technology as a memory aid for capturing the user's daily routine. The images recorded are mapped as a lifelog and presented in a timeline format. In [50], the authors use SenseCam for the tracking of sedentary behavior. However, the focus of this utilization is about activity data accumulation and tracking.

In [51], authors present the InSense system. It utilizes acceleration, audio, and visual sensing, to perform real time context recognition. Conceptually InSense and DCF can be aligned and compared as they both work for real time data collection with a larger set of data sources to generate a context-rich lifelog. However in InSense, data accumulation and its understanding are two separate activities. After the data is collected, it is manually annotated and rated by the user to create an interest operator. This process is completely offline and quite cumbersome for the user who has recorded an activity for few hours. Furthermore, the lifelog becomes rich as a consequence to annotation; thus, an effective monitoring in real time cannot be performed for the InSense users.

In more recent years, the utilization of multimodal sensor-based technologies has evolved to become personalized accessories (e.g., wearable bands and smartwatches such as Garmin Vivofit, Fitbit Surge, Misfit Shine, and Apple Watch). These technologies process independently of smartphones. The wearable wrist sensor stores and transfers the sensory data using a dongle device or a smartphone to a health management system. However, these systems are single data source-oriented, i.e., they only consider a wristband sensor; therefore they cannot infer correct context about what a user is presently doing.

Regardless of the considerable research and development of healthcare and wellness systems, only a few implementations exist that execute as independent platforms for tackling complex and realistic scenarios. Highly funded commercial initiatives like Apple's HealthKit, Google Fit, and Microsoft Health have evolved the approach from an application to an ecosystem; however, these implementations are still single device or data integration-centric.

2.3. Data Accumulation and Curation

Considering lifelog data as an asset, some of the latest works have been focused on data accumulation and its extended utilization. In [52], the authors present the challenges of data accumulation in the Internet of Nano-Things [53] (IoNT) perspective. The challenges are discussed at communication protocol level, hardware level and software (middleware) level. Possible solutions and applications have been discussed as a proposed design.

In [54], authors present an energy efficient continuous sensing framework called Insight. The implementation of this framework focuses on small and wearable devices, and has been demonstrated using a smartwatch. The data accumulation process is energy efficient, and this data is further used for prediction. Comparing the implementation of Insight with DCF, it is evident that the volume of data both the frameworks are concentration on is very different. Insight only uses wearable devices with small data footprint; furthermore, it does not use raw sensory data. If required, it leverages secondary application like Google Fit for the user activity data. Opportunistic (Event and interval driven) sensing [55] of Insight with stated sensing policy also contributes to limited size. This data is persisted in a folder like structure on the device's storage. On the other hand, DCF implements continuous accumulation in a device-independent mode; resulting in a much more scalable framework for a larger set of multimodal data sources. Consequently, the data generated in DCF is significant in volume which requires big data storage and the curated lifelog is context rich. DCF provides services on sharability of this data for knowledge generation, analytics, and visualization. As acknowledged by the authors in [54], computationally complex processes should be handled by devices with advanced computing abilities. Therefore, DCF utilizes the computational ability of the cloud to execute compute-intensive processes on the accumulated raw sensory data.

Some of the systems have a particular perspective on the accumulation and utilization of healthcare data originated at the clinical level. Instead of more real-time accumulation, these implementations are focused on evolutionary repositories with aggregation of historical and more recent clinical data available for utilization. Among these clinical data-oriented systems, NetBio [56] is a prominent implementation. It assembles vast amounts of curated and annotated, clinical and molecular data. This method enables NetBio clients to make unique discoveries that otherwise would be impossible with their own private datasets. NetBio uses big data technology for permanent persistence and core logic layers to make correlations between the billions of data points from the public domain with private genomic and clinical data sets. At a higher-level, NetBio provides a rich set of APIs that enable clients to integrate NetBio within their workflows and scenarios. Their current clients include pharmaceutical R&D and academic medical centers. The initial system was implemented for oncology; nevertheless, it is now expanding into metabolic and autoimmune diseases.

In contrast to all of the implementations discussed above, DCF is a novel attempt to implement a raw sensory data acquisition, curation, and monitoring framework. The sensory data acquisition services of DCF are independent of data sources. It is equipped to handle multimodal data sources directly communicating with the framework or via smartphone-like gateways. The acquired multimodal data is synchronized to represent an event of time for each user. With scalability in mind, numerous multimodal data sources can communicate in parallel with the DCF, making it a more IoT-oriented implementation. Furthermore, instead of being dependent on the computational ability of smartphones, DCF considers all of the communicating devices as a source of raw sensory data; thus, generating a context-rich user lifelog. The computation over the accumulated data and lifelog is performed over a cloud platform, keeping the framework compatible with data source with low computational abilities. From an evolutionary perspective, complex computational algorithms for context identification, data fusion, and mining can be implemented without disturbing client implementations. Moreover, the accumulated data can be used for concepts of data-driven knowledge generation, descriptive and predictive analytics, and visualization. Samsung SAMI [57] is also moving in a similar direction with an independent API and cloud support; however, their implementation is more data exchange-centric, and utilization for healthcare and wellness requires custom implementation and monitoring. Moreover, DCF curates the identified low- and high-level context on user accumulated data as a comprehensive lifelog. The idea of lifelog has already been presented in [22,24,25]; however, DCF implements anomaly detection on lifelog instances based upon expert-driven rules in correlation with user profiles, keeping the monitoring vigilant as well as personalized.

3. Proposed Methodology

This section provides an overview of the proposed methodology of DCF. Participating components in the implementation of DCF are introduced in this section; however, technical details of these components are described in the following section.

DCF is intended to be incorporated as a foundation layer for health and wellness platforms where real-time multimodal sensory data acquisition, its curation as a lifelog, and monitoring for anomalies is essential. Furthermore, it provides permanent storage to acquired sensory data for extended usage (e.g., visualization, analytics, data-driven knowledge generation). For the independent execution of DCF, its implementation is encapsulated with interfaces for sensory data acquisition and its persistence. The curation of a lifelog and its monitoring is implemented as an implicit property of DCF with a separate interface to incorporate anomaly detection rules and publish-subscribe based response.

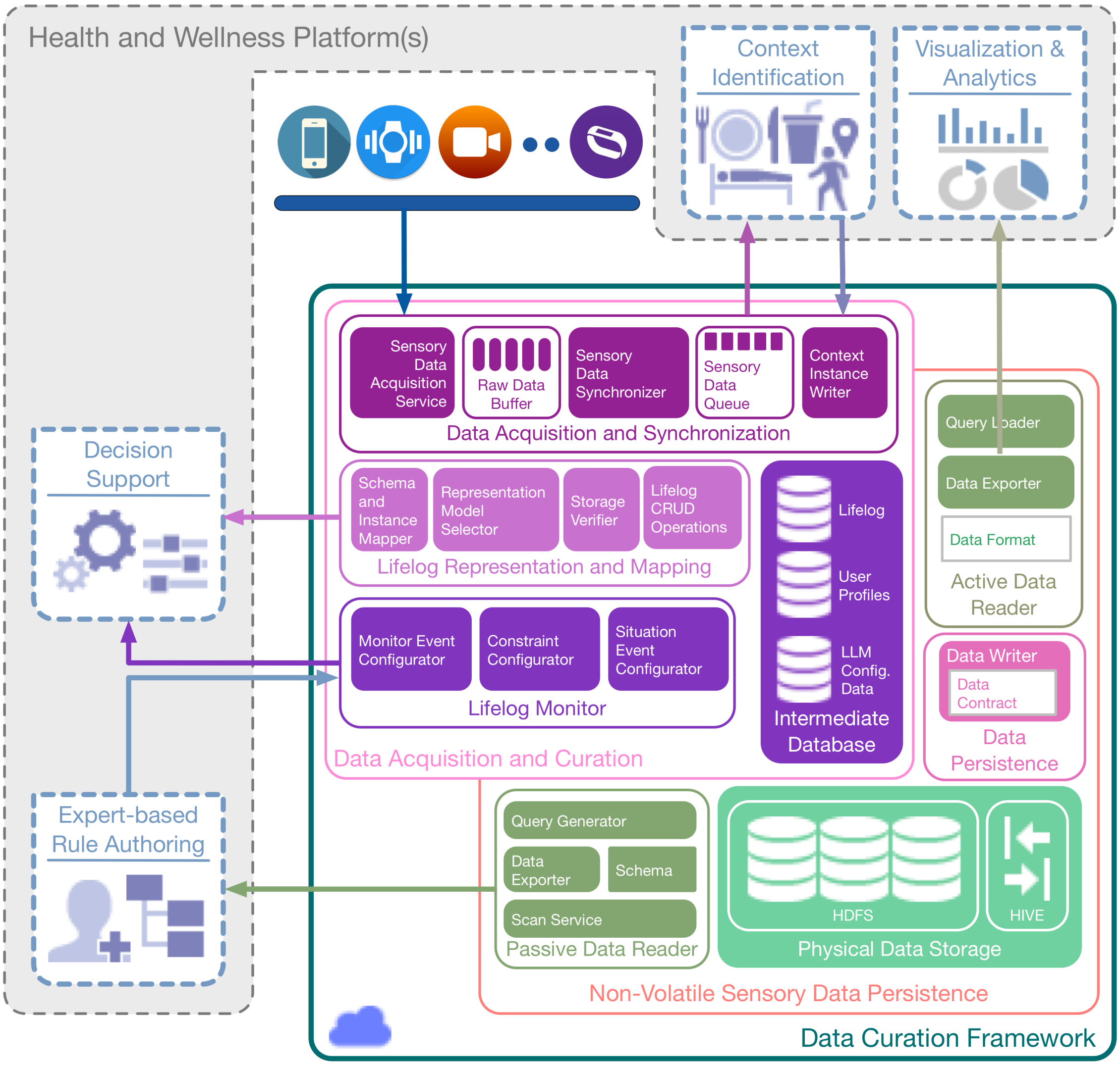


Figure 2. Data curation framework (DCF) architecture.

As illustrated in Figure 2, DCF consists of two primary components, i.e., *Sensory Data Processing and Curation*, and *Non-volatile Sensory Data Persistence*. Within the former, the *Sensory Data Acquisition and Synchronization* subcomponent obtains the raw sensory data from multimodal data sources, both in a real-time (active) and offline (passive) manner. This data is synchronized based upon the user identification and the time stamp of the data generation, and subsequently, it is queued for the context determination. The definition and methodology of the identification of context vary in health and wellness platforms [58]; therefore, an interface to the synchronized sensory data residing in the queue is provided by DCF. This feature facilitates health and wellness platforms to plug-n-play their context identification engine without worrying about the real-time data acquisition from multimodal sources in a distributed environment. In response, these engines can feedback the DCF with identified context. The subcomponent of *Lifelog Mapping and Representation* receives the identified context and curates it by mapping the context instances to a time-based log registering the detected human activities and behaviors. The lifelog persists in the Intermediate Database for reuse. The stream of lifelog instances is analyzed by monitoring the subcomponent known as the *Lifelog Monitor (LLM)*; it is responsible for performing customized unit-based (e.g., time-based, physical activity-based, nutrition-based) monitoring of user context available in the lifelog, cross-linked with the user profiles. *LLM* draws the association between the context available in the lifelog and implements anomaly detection based on expert-driven rules. Anomalies detected by *LLM* represent risky or unhealthy behavior and are described by various constraints (e.g., age, gender, medical conditions) and monitor-able variables (e.g., the intensity of a particular activity and its duration). *LLM* is equipped to provide a notification-based response to its client (a health and wellness platform or a user) with the help of a publish-subscribe mechanism.

The *Non-volatile Sensory Data Persistence* component is responsible for providing permanent and distributed big data persistence to the raw sensory data. It is termed non-volatile, as no update or delete operations are performed on the raw sensory data storage. The subcomponent of *Data Persistence* provides a data contract that is implemented by the clients of *Physical Data Storage* for permanent persistence of the data. *Non-volatile Sensory Data Persistence* also provides mechanisms to access this persisting data as a response to health and wellness platforms. This response can be of an online or offline type. For an online response, the *Active Data Reader* subcomponent is used. It provides a continuous stream of sensory data for extended data operations, including visualization and predictive analytics. For an offline response, the *Passive Data Reader* subcomponent is used. It provides a batch response that can be effectively used for training machine learning-based models and can provide data insights to experts for rule generation. To create periodic backups of lifelog data, the subcomponents of *Data Persistence* are again utilized. This method provides permanent storage of the lifelog data, which can be utilized in the future for user behavior analysis.

4. Implementation Details

DCF is a hybrid cloud implementation with components distributed between a private and a public cloud. Therefore, privacy and security of user lifelog data has been given high importance. For its implementation, we have used custom built Health Fog framework [59], which provides Fog computing [60] as an intermediary layer between the client and DCF. Furthermore, it implements Cloud Access Security Broker (CASB) for data access policy. The modular design of Health Fog is capable of engaging data from multimodal data sources together with the adequate level of security and privacy using existing cryptographic primitives.

In the staging and production environments of DCF, the component of Sensory Data Processing and Curation is deployed over a public cloud (Microsoft Azure [61]) with compute-intensive instances to support scalability and high performance. However, the component of Non-volatile Sensory Data Persistence is deployed over a private cloud environment with distributed commodity machines, customized security [62], and large-space hard drives. Implementation details of these components are described in the following subsections.

4.1. Raw Sensory Data Acquisition and Synchronization

Implementation of Raw Sensory Data Acquisition and Synchronization (DAS) consists of a REST [63] service that collects raw sensory data from multimodal data sources. The key in this acquisition is the association of accumulated data with their time of origination. All data sources subsist independently along with independent clocks; therefore, a logical clock is required for identifying the data origination at the same time from multiple sources. Consequently, DAS implements the time frame-based synchronization methods called Complete- and Incomplete-sync.

Complete-sync is the sunny day scenario in which all the data pertaining to an instance of time is accumulated within the specified time frame. It waits for the data accumulation from all the data sources at the DCF server within the duration of a three second time frame. As soon as all the data is received, complete-sync encapsulates it as one message. Subsequently, the message is timestamped and enqueued in Sensory Data Queue for context determination by a health and wellness platform.

In contrast to Complete-sync, Incomplete-sync executes when data from any of the sources is not received in the three-second time frame. This execution is further classified in two different modes, i.e., Eager and Rendezvous. In the Eager mode, the message is created with only the available data and buffered for context determination. In the Rendezvous mode, for the missing data, it associates the last sent data from the raw sensory buffer. Subsequently, a message is created and buffered for context determination. Implementation details of Complete-sync and Incomplete-sync with its dual modes are explained in Algorithm 1.

Due to the non-blocking (asynchronous) and scalable nature of data acquisition from multimodal data sources, DAS uses Node.js [64] for server-side implementation. Furthermore, it implements in-memory buffers for temporary storage of data that has arrived. Each buffer represents a particular type of multimodal data source. A buffer synchronizer is executed so that the accumulated data can be synchronized with the data that originated at the same time from all other data sources.

For communication from a multimodal data source to DAS, a JSON [65] object is defined. The contents of this object consist of four fields, i.e., a user id, a data source id, a timestamp of the data origination at a data source, and the payload. Due to service-based implementation, DAS is independent of the number of multimodal data sources.

Algorithm 1 Time-based synchronization for raw-sensory data acquisition.

Require: $buffer_{src}[1,...,n]$: n is the total number of data sources

Ensure: $buffer_{dst}$: queue for time-synchronized data packets

```
procedure SYNC(buffer_dst)
    msg ← create_msg(NULL)
    while i ≤ No_of_datasources do
        buffer_src[i] ← Recv(data)                          ▷ Complete-sync execution
        msg.add(create_msg(buffer_src[i]))
        if time_sec > time_window then
            if send_only = TRUE then                        ▷ Incomplete-sync: Eager execution
                break
            end if
            while j ≤ No_of_datasources do                  ▷ Incomplete-sync: Rendezvous execution
                j ← i + 1
                if buffer_src[j].has_contents then
                    msg.add(create_msg(buffer_src[j])
                end if
            end while
            break
        end if
    end while
    msg.timestamp ← buffer_src[i].timestamp
    buffer_dst.enqueue(msg)
end procedure
```

4.2. Data Persistence

The asset of DCF is its persistence of a user's raw sensory data (from multimodal data sources) with the associated context as a lifelog. DCF defines two levels of abstraction on the data acquired. The first level is a higher-level abstraction referred to as an Intermediate Database hosted within a relational database (RDBMS). This database hosts three types of data, i.e., the user lifelog that represents user context over a period of time, the user profiles, and the knowledge base consisting of rules for anomaly detection required by LLM. The intermediate database is deployed over a public cloud instance.

The second level is a lower-level abstraction referred to as non-volatile distributed storage; it is hosted over Hadoop-based big data storage [66] on a private cloud instance. This storage also hosts three types of data; however, the granularity of data is at a finer scale. For example, this storage provides permanent persistence to all the raw sensory data acquired from the multimodal data sources; it maintains user-invoked backup of large-sized multimedia content, such as video data captured from a 3D camera and periodic backups of user lifelogs with associated user profiles.

The philosophy behind this two-level abstraction architecture of data storage is due to four main factors. (i) Performance, the user lifelog data is frequently accessed in soft-real time. Therefore, due to the inherent speed of the RDBMS, its storage is leveraged for lifelog persistence and access; (ii) Data relationships, the user activities and behaviors are encapsulated as context by a comprehensive object-based relationship. For recommendation generation or high-level context determination by health and wellness platforms, relationships among entities of user profiles and their context are necessary. Thus, a tightly coupled structured data formation is held. Therefore, the schema-based approach of the RDBMS is leveraged for persistence instead of the unstructured data storage of big data; (iii) Data volume, the magnitude of data generated by multimodal data sources, grows exponentially in a short period of time. Therefore, big data storage with its distributed non-volatile persistence is utilized for storing the raw sensory data as well as periodic backups of user profiles and lifelogs; (iv) The frequency of data updates, the user lifelog changes and evolves rapidly due to the continuous change in user context in real time. Due to the higher cost of data updates at the big data level, RDBMS is leveraged.

4.3. Lifelog Representation and Mapping

In DCF, the identified context is mapped over a timeline as a lifelog, providing a black box to user activities and behaviors over time. The lifelog is maintained in a relational database instance at the intermediate database of DCF. The maintenance of this lifelog including CRUD operations is the responsibility of the subcomponent of Lifelog Representation and Mapping (LLRM). The lifelog is represented as an object-oriented abstraction called a lifelog model in this mapping subcomponent. The lifelog model encapsulates attributes such as performed activities, preferred and visited locations, health and wellness goals to achieve, and recognized emotions of the users. For agility and extensibility, this object model is an implementation of Facade and Adapter design patterns [67]. The conceptual model of LLRM is illustrated in Figure 3.

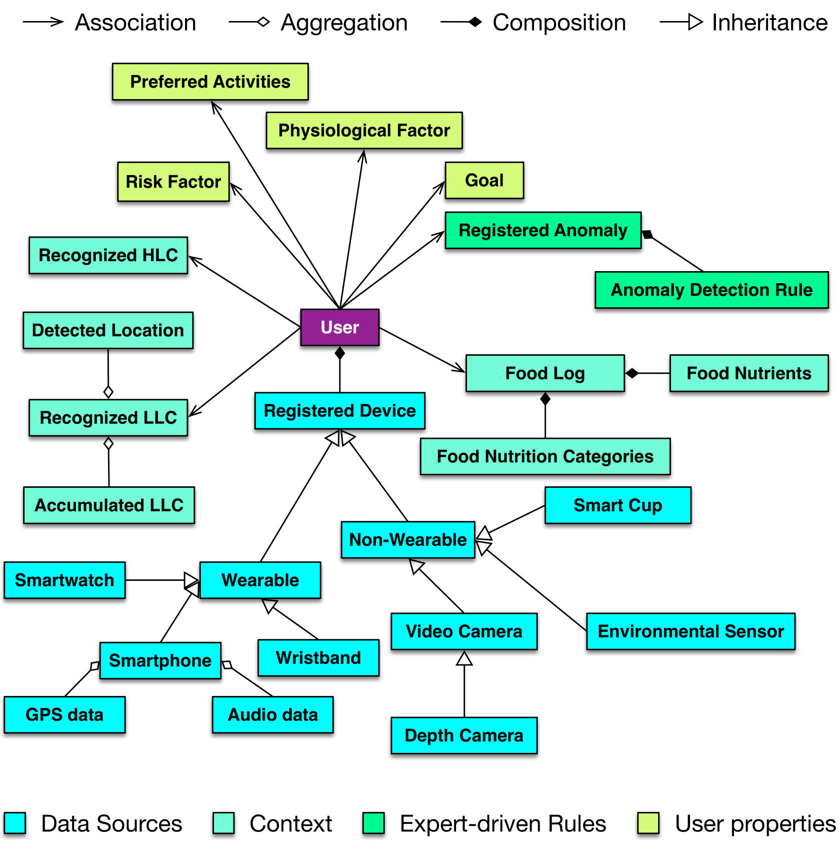


Figure 3. Lifelog model.

The implementation of LLRM contains a mapper module that extracts objects, attributes, and relationships from input XML/JSON data as resources. It identifies and maps the classes for the extracted resources and places the instances accordingly using a deep copy method [68].

The Storage Verifier module validates the data according to the schema of the data tables in the intermediate database. Furthermore, it also validates the compatibility of data with the defined constraints (keys, values, primary keys, and foreign keys) on the data tables.

Depending upon the utilization, the lifelog model has higher-level abstractions, called representations, to customize the lifelog data access for health and wellness platforms. The current implementation provides four different representations: (i) The first representation provides an interface to internal DCF components for storing and retrieving lifelog data; (ii) the second representation provides an interface to external clients of DCF for retrieving lifelog data with user profiles; (iii) the third representation provides an interface to external clients of DCF for retrieving lifelog data and its associated raw sensory data from non-volatile storage; (iv) the fourth representation provides an interface to external clients of DCF for incorporating decisions made by health and wellness platforms as part of lifelog data. These representations are compatible with XML and JSON document formats.

4.4. Lifelog Monitoring

Lifelog Monitoring (LLM) is designed to perform continuous monitoring of the user context representing his activities and behavior in his lifelog. LLM itself does not recognize activities; however, it leverages the external activity recognition abilities of the health and wellness platforms to provide associated context, and it executes at a higher-level abstraction and draws associations between a user's low- and high-level context [69], then maps them to the anomaly detection rules built by an expert. As soon as this mapping is established, LLM registers them over the target user's lifelog and starts the monitoring process.

To generate notification-based responses, LLM implements an observer pattern [67]. This mechanism facilitates LLM to host a subscriber list of clients for a particular user's lifelog. If an anomaly is detected, all of the subscribed clients are notified. This implementation is helpful for scenarios in which not only the user but also his care provider and a third party wellness platform need to be alerted to a particular anomaly-related event.

LLM is designed to incorporate anomaly detection rules that are generated by the expert dynamically. This nature of the LLM requires a robust model to handle all of the anomalies to be detected in a particular domain. Therefore, LLM divides domains in terms of measuring quantities. The current implementation of DCF supports time as the measuring quantity for physical activities, as well as macronutrients (fat, carbohydrates, and proteins) and calories for food intake.

The overall complexity of the implementation of LLM is divided into three modules, i.e., the monitored event configurator, the constraint configurator, and the anomaly event detector. Configurator modules are built to incorporate the anomaly detection rules. These rules are submitted to LLM using a JSON-based communication contract. This contract is described in the following listing:

```
{
// Common communication format
 "AnomalyConditions":[
 {
     "ConditionKey" : "Age",
     "ConditionType" : "String",
     "ConditionValue" : "Adult",
     "ConditionValueOperator" : "="
 },
 {
     "ConditionKey" : "Duration",
     "ConditionType" : "String",
     "ConditionValue" : "1h",
     "ConditionValueOperator" : "="
 },
 {
     "ConditionKey" : "CurrentContext",
     "ConditionType" : "String",
     "ConditionValue" : "Sitting",
     "ConditionValueOperator" : "="
 }, "AnomalyID" : "1"
 ]
}
```

To uniquely identify dynamic anomalies in multi-situation scenarios, they are managed as key-value pairs. The output of the configurator modules is well-defined anomaly detection information, consisting of its monitored event and its target measuring quantity.

The anomaly detector module is responsible for monitoring the user lifelog for monitored events. For monitoring to occur, at least one respective anomaly associated with a particular monitored event must be registered for a user. As soon as the monitored event is detected, the anomaly detector registers the respective well-defined anomaly detection rule for that particular user. Lifelog data pertaining to these monitored events are filtered out from the stream of user context and managed in a monitorable log. This log has information related to the target measuring quantities. Different monitored events have different target quantities; therefore, the monitorable log facilitates the LLM to manage multiple monitored events for independent monitoring by persisting unique targets. The anomaly detector compares the difference of the measured quantities in regular intervals. Whenever the target is achieved, the anomaly detector publishes a notification to all the subscribers of the user's lifelog. The execution process of LLM is illustrated in the sequence diagrams of Figures 4 and 5.

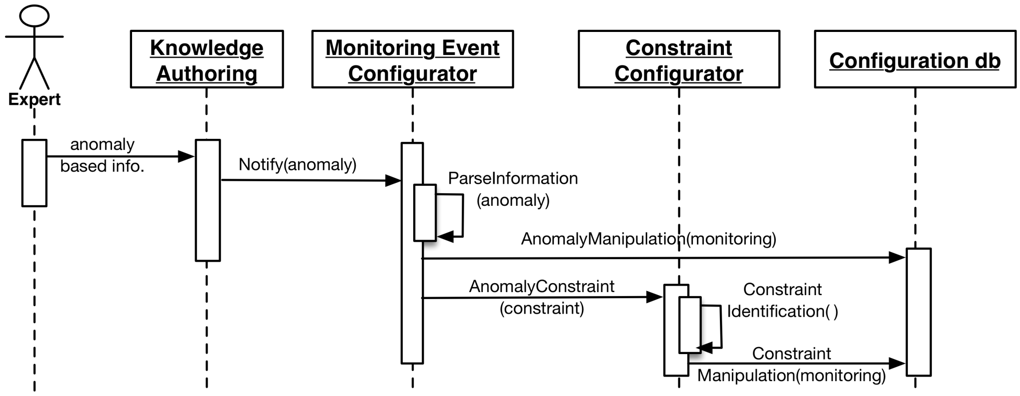


Figure 4. Sequence diagram of anomaly registration by the expert.

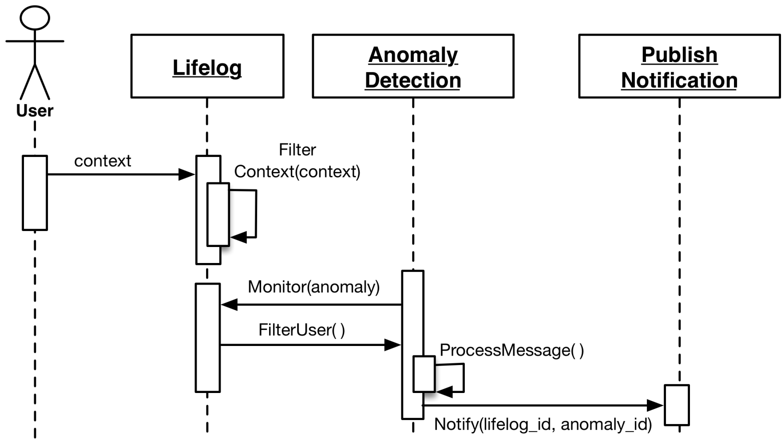


Figure 5. Sequence diagram of message flow in the lifelog monitoring.

4.5. Non-Volatile Sensory Data Persistence

The component of non-volatile sensory data persistence is responsible for providing permanent and distributed big data storage to raw sensory and lifelog data. Its subcomponents are designed to execute create and read operations as per the needs of the health and wellness platforms. These components primarily use Hive [70] queries over the Hadoop Distributed File System (HDFS).

The subcomponent of Big Data Persistence is responsible for storing the raw sensory data communicated from the data acquisition and synchronization component every three seconds. This subcomponent is also used for user invoked periodic persistence of multimedia contents such as 3D depth videos.

There are two types of data reading subcomponents for access over distributed storage, i.e., Active Data Reader and Passive Data Reader. Active data reader (ADR) is primarily used to provide a continuous response to a real-time request for data visualization and analytics by a health and wellness platform. ADR hosts a repository of pre-defined Hive queries for a data read. It is an evolutionary repository that grows with more usage and the addition of more clients. Upon receiving the request for data, ADR matches the request parameters with the available set of queries. In the case of a parameter match, the selected query is executed with a continuous stream-based response back to the client.

A passive data reader (PDR) is primarily designed from a machine learning perspective, as historical data persistence in the HDFS is useful for the training of models for data-driven knowledge generation. These models can support the expert to generate rules, which can be further utilized by LLM for anomaly detection. Execution of PDR is a two-step process: first, the schema of the data is sent to the client for parameter selection and request generation; second, after the selection is performed and sent to PDR with the request, it generates a query dynamically upon receiving the parameters. This query is executed over HDFS and the response is created and provided in a JSON format. The execution of PDR is an offline process and may take hours to complete.

5. Execution Scenarios

To understand the workings of DCF, we have incorporated its execution in the Mining Minds project as our client health and wellness platform. Depending on the needs of the platform and its participating layers (ICL, KCL, Service Curation Layer or SCL, and SL), DCF concurrently performs the online and offline execution. These executions are termed as active and passive execution flows. These execution flows are illustrated in Figure 6 and described in the following subsections.

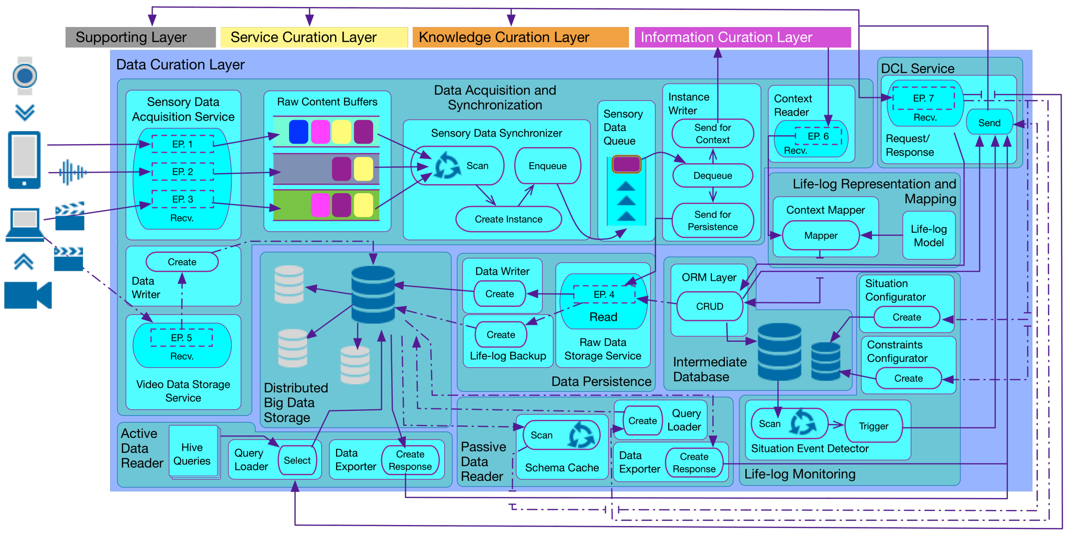


Figure 6. DCF execution flow.

5.1. Active Execution Scenario

The latest version of the Mining Minds platform supports the recognition of ten activities and three emotions (described in Table 1). They are recognized by ICL and returned as identified high- and low-level context. To facilitate ICL, DCF acquires four different types of raw sensory data (activity, location, audio, and 3D video) per user in an online manner. DCF communicates with three multimodal data sources for raw sensory data acquisition, i.e., a smartwatch, a smartphone, and a depth camera. The smartwatch accumulates accelerometer, gyroscope, magnetometer and PPG data, and synchronizes itself with the Mining Minds smartphone app. The smartphone itself collects accelerometer, gyroscope, magnetometer, GPS, speed, and audio bytes of three seconds for determining user activities, locations and emotions. The Mining Minds smartphone app communicates all this raw sensory data to the DCF. For soft-real-time data acquisition, this non-blocking communication occurs every three seconds. In parallel, a depth camera (Kinect [71]) attached to a PC also communicates with the DCF for sending the semantics of the captured 3D video. These semantics provide activity and behavior-related information of the user to ICL. Inside DCF, the sensory data acquisition service receives this accumulated information asynchronously and buffers the data depending on its data source. These raw data buffers are initialized based on the number of data sources, i.e., each data source has a raw data buffer in the sensory data acquisition and synchronization subcomponent. All the buffer contents are synchronized and enqueued for determination of the associated low- and high-level context. From this point forward, the synchronized data is dequeued and then sent in parallel to ICL and distributed big data storage for context identification and non-volatile persistence respectively.

Upon receiving this synchronized data, ICL identifies the context and communicates it back to DCF. To reduce the ICL to DCF traffic, ICL only communicates if there is a change in context. For example, if a user has been sitting for more than one hour at work, ICL will identify the sitting activity at the office with its low- and high-level context identification. This context will not be communicated to DCF unless the user changes his activity to another. The received context from ICL is converted into a user lifelog instance and mapped to the object-oriented representation of the lifelog by the LLRM subcomponent. Furthermore, this subcomponent is also mandated for executing Create, Read, Update, and Delete (CRUD) operations regularly on lifelog data; therefore, it maintains the mapped data in the intermediate database.

The stream of incoming context is continuously monitored by LLM for anomalies described by well-defined rules. In the case of detection, LLM publishes a notification to SCL of the Mining Minds platform for a possible recommendation generation. Consequently, SCL gathers the required data, including the related context and user profile, and generates and pushes a recommendation to the user's smartphone. For a user request-based recommendation generation by SCL, DCF operates in a pull-based request response. LLRM receives the query generated by SCL via service. Required data, i.e., user profile and lifelog data, are extracted via a read operation and returned as a response to SCL for recommendation generation.

Table 1. Multimodal raw sensory data sources.

Data Source	Sensor(s)	Context Type	Context Description
Smartwatch	accelerometer, gyroscope, PPG.	activity	climbing stairs, running, walking, cycling.
Smartphone	accelerometer, gyroscope, longitude, latitude.	location	home, office, gym, school, restaurant, cafe.
Smartphone	audio	emotion	anger, sadness, happiness.
Depth camera	video	activity	eating, sitting, standing, stretching, sweeping, lying down.

As mentioned earlier, in parallel to context determination, raw sensory data is also sent to non-volatile storage for persistence. This data is required by SL of the Mining Minds platform for visualization and descriptive analytics. Following a request based on registered parameters, static queries are invoked by the active data reader subcomponent. It generates a stream-based response, as SL requires a continuous stream of data for visualization and analytics.

5.2. Passive Execution Scenario

The passive execution flow of DCF pertains to the processes required for evolutionary support of not only DCF but also the overall Mining Minds platform. One of the primary processes that requires the passive execution of DCF is the provisioning of raw sensory data from non-volatile persistence for data-driven knowledge generation by KCL of the Mining Minds platform. This execution is based on a two-step process. First, KCL requests DCF for the schema of data maintained in the distributed big data storage. The passive data reader subcomponent processes this request by replying to KCL with the most recent version of the schema describing the data maintained in big data storage. Consequently, a domain expert at KCL's authoring tool selects the parameters based on the schema and generates the conditions for which data should be extracted from the big data storage. These parameters with conditions are returned to the passive data reader, which generates a dynamic query based upon the KCL provided parameters and executes it over big data storage. A response to this execution is returned to KCL in a pre-defined service contract. Extracted data from DCF assists KCL in the generation of rules that are used by LLM in the continuous monitoring of lifelog data during the active execution flow.

Another process that invokes the passive execution of DCF is the periodic backup and synchronization of the user's lifelog data persisted in the intermediate database with the associated raw sensory data maintained in the big data storage. The subcomponent of the lifelog sync performs this operation. SL of the Mining Minds platform uses this historic data for descriptive analytics.

Inside DCF, an instance of the data acquisition service also executes passively to copy the actual video contents from the user's PC to big data storage. This is also a periodic execution and exclusively authorized by the Mining Minds user.

6. Evaluation and Discussion

DCF has been evaluated from various aspects, e.g., from raw sensory data acquisition perspective, its accuracy of synchronization for sensory buffers, performance, and scalability has been evaluated. From context monitoring perspective, the ability of LLM to monitor upon user lifelog for existing anomalies, for dynamically added anomaly detection rules at runtime, and monitoring multiple users with various activities at runtime. From raw sensory data persistence perspective, its ability to read and write sensory data, and execution time of different queries with varying complexities. The results and discussion of these evaluations have been discussed in following subsections.

6.1. Accuracy of the Synchronization Process during Raw-Sensory Data Acquisition

To evaluate the accuracy of the synchronization process, a smartphone (Samsung Galaxy S5 running Android 5.0 Lollipop, Samsung, Seoul, South Korea) and second-generation Kinect (rel. 2014) connected to a PC communicate with the DCF server running on a cloud instance with the 64-bit Windows 8.1 operating system, 16 GB of RAM, and a 3.10 GHz AMD A8-7600 Radeon R7, AMD, Sunnyvale, CA USA with 10 computing cores of 4C + 6G. Multiple requests from a single user from both clients (smartphone and PC), generated and synchronized at the server, are validated for accuracy. A time window of three seconds is allocated to temporarily hold the incoming data in memory buffers. For a conclusive evaluation, 100 data packets containing activity, location, voice, and 100 data packets containing video data are sent from the smartphone and the PC respectively. These communications occur randomly at different points in time during the defined three second window. The results from this evaluation reflect that the raw sensory data from both clients has been successfully synchronized at the server. The evaluation in Figure 7 illustrates the exact time in milliseconds of raw sensory data packets sent from both the smartphone and PC during ten different time windows. For example, for the first time window, the smartphone communicated at 521 ms; however, the PC communicated at 2545 ms. The response synchronizer at DCF was able to identify both of these data packets as part of a single window of communication. As illustrated, all of the data packets have been successfully synchronized to the server, proving the correctness of our synchronization process over data acquisition of raw sensory data.

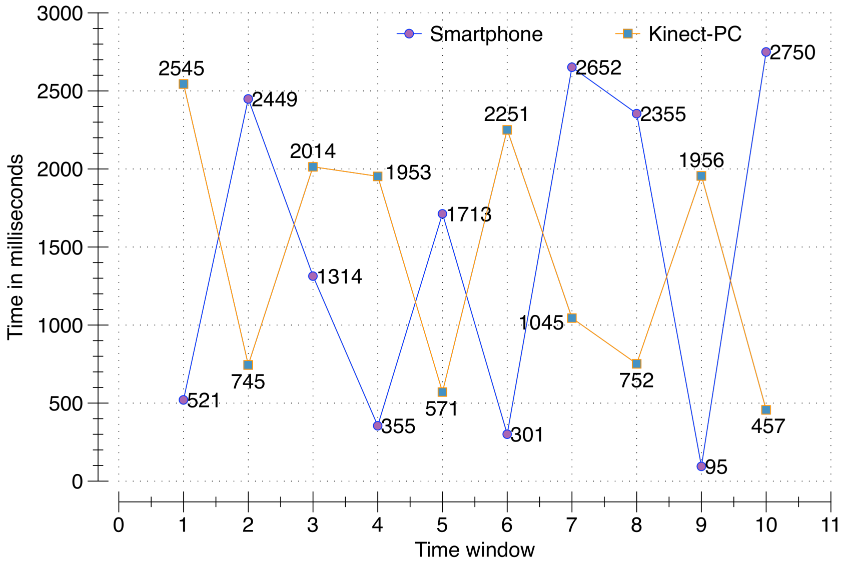


Figure 7. Data synchronization testing per time-window.

6.2. Performance Testing during Raw-Sensory Data Acquisition and Synchronization

To evaluate the performance during raw sensory data acquisition and its synchronization we setup an environment of five multimodal data sources: (i) LG Watch R(TM), smartwatch connected with a Samsung Galaxy S5 smartphone running Android 5.0; (ii) Shimmer3 sensor connected with a Samsung Galaxy S5 smartphone running Android 5.0; (iii) Samsung Galaxy S6 Edge running Android 5.0 for audio data; (iv) Second-generation Kinect (rel. 2014) connected with a Samsung Notebook 9 running MS Windows 8.1; and (v) a PC emulating an environmental sensor. These sensors communicate with a DCF server (64-bit Windows 8.1 operating system, 16 GB of RAM, and a 3.10 GHz AMD A8-7600 Radeon R7 with 10 computing cores of 4C + 6G) in an asynchronous manner by sending sensory data packets with an increment from 10 to 10,000. The target timeout deadline given to the data acquisition and synchronization subcomponent is 5 seconds to accumulate, synchronize and queue the synchronized data in a memory buffer. The results of the performance testing are presented in Figure 8. From the results, it is quite evident that our implementation of the data acquisition and synchronization subcomponent is able to accumulate and synchronize 10,000 packets from the five sensors within the provided timeout deadline. Due to server warmup, our implementation initially took 3.5 s for 10 packets; however, the performance improved with subsequent executions. The fastest time was recorded for 800 packets with a total time of 3.15 s. The time to accumulate, synchronize and enqueue increased after 1800 packets and reached at 10,000 packets with a total time of 3.31 s, which is still 51% faster than the provided deadline of 5 s.

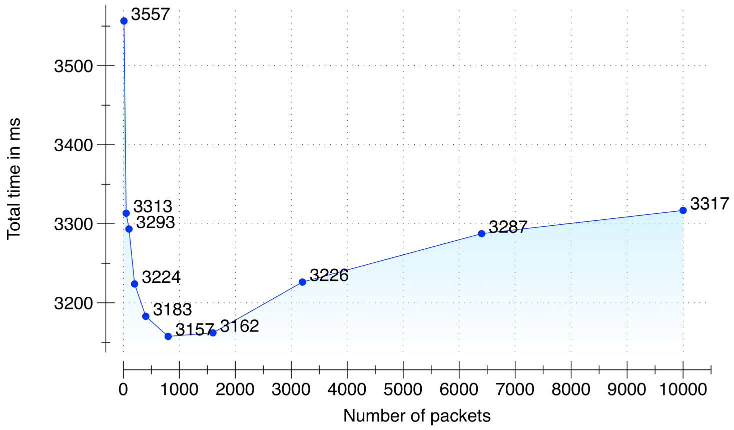


Figure 8. Performance testing.

6.3. Scalability Testing during Raw Sensory Data Acquisition and Synchornization

To evaluate the ability of DCF to scale appropriately with the increase in number of multimodal data sources we setup an environment that simulates an incremental number of data sources (from 5 to 320). This test case stresses the data acquisition and synchronization subcomponent with a fixed packet size of 10,000 packets per data source with each packet containing a pre-accumulated sensory data of user activity for three seconds (30 kb) by using Samsung Galaxy S5 smartphone running Android 5.0. DCF server is running on a cloud instance with NodeJS v5.8, 64-bit Windows 7 Operating System, 4 GB of RAM, and 3.8 GHz AMD A10-5800K APU. Results of scalability testing are presented in Figure 9. From the results, it can be seen that our implementation was able to scale successfully from 5 multimodal data sources to 160. For 160 data sources with the maximum of 10,000 packets per data source, DCF was able to accumulate, synchronize and enqueue within 1.2 min (=72,258.2 ms). Therefore, our presented implementation can contribute significantly in environments where larger set of data sources will be added over time. As a response, DCF can accumulate, synchronize, and buffer the raw sensory data for these devices without any distinctive time delay. Although this evaluation depends on the size of the data packets; however, from our calculation user activity data accumulated over a duration of three seconds using a smartwatch and a smartphone is 24 kb, which is smaller than the size of the packet used in this evaluation. To check the threshold of our implementation, we stress tested this evaluation with 320 data sources (illustrated in Figure 10). Consequently, the total time increased exponentially to 41 min (=2,470,064 ms).

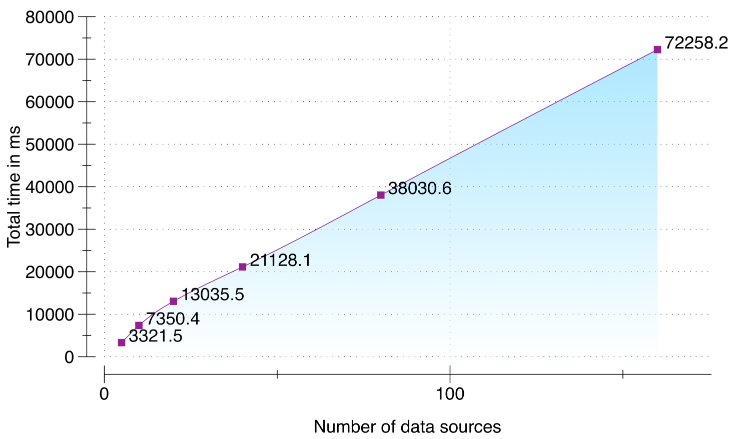


Figure 9. Scalability testing.

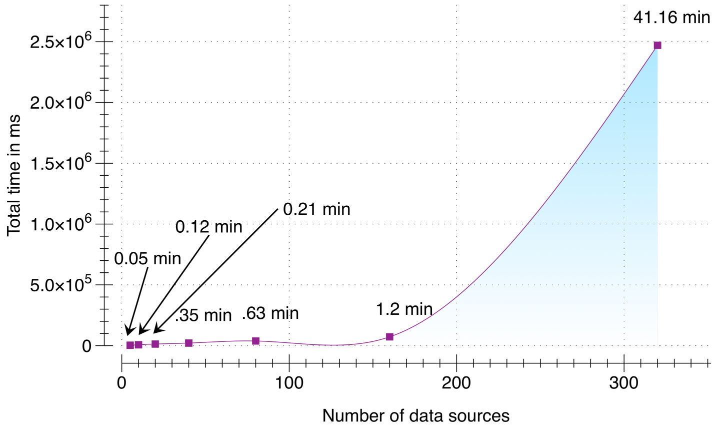


Figure 10. Stress testing of scalability.

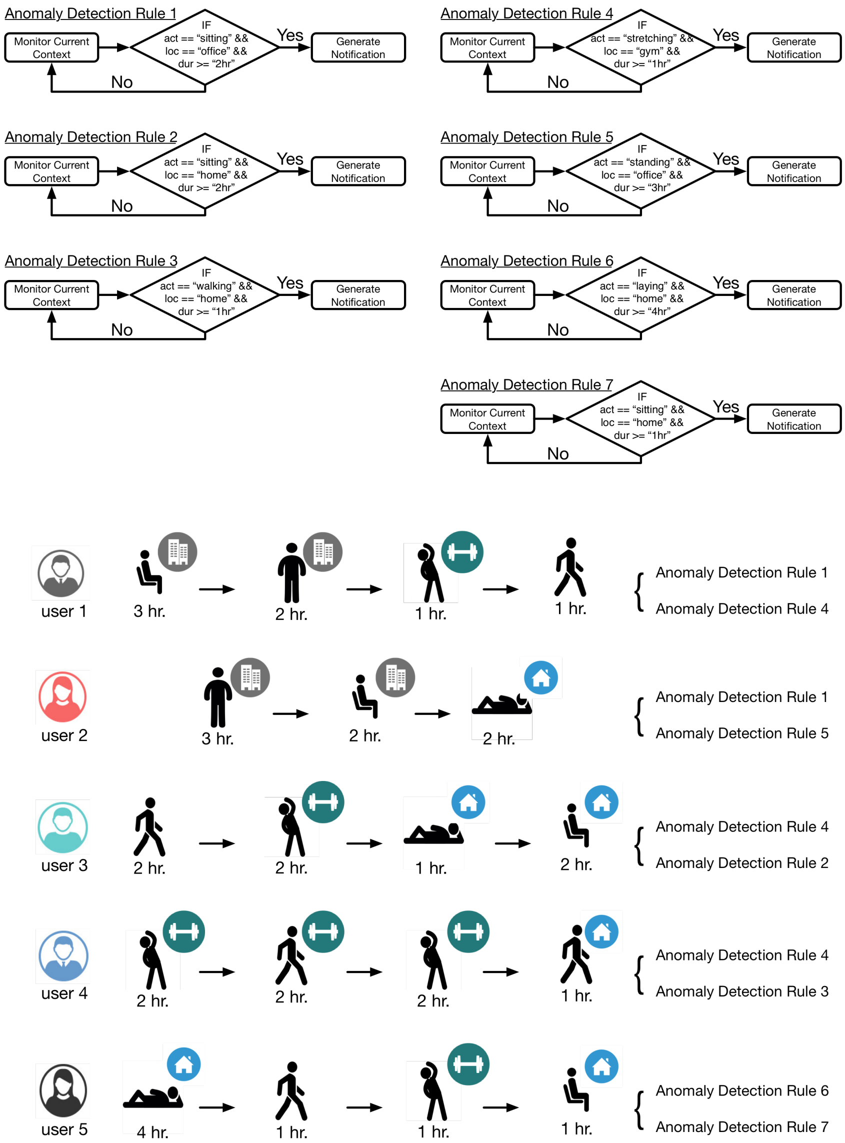


Figure 11. Anomaly detection rules and scenarios.

6.4. User Lifelog Monitoring

The presence of an anomaly is a key-driving concept of lifelog monitoring. In the presence of registered anomaly detection rules, LLM provides monitoring for single or multiple users over their lifelog.

For the evaluation of LLM, five users were selected to provide context information by performing various activities at user specified locations. Each user provided context using an LG Watch R(TM) for activity data (accelerometer, gyroscope, and PPG sensors were used), smartwatch connected with a Samsung Galaxy S5 smartphone running Android 5.0 for location data (GPS sensor was used) and a Second-generation Kinect (rel. 2014) connected with a Samsung Notebook 9 running MS Windows 8.1 for indoor activity data using depth camera. Using the LLM interface, seven rules were dynamically incorporated in the execution of LLM. Definition of these rules and their correspondence with the users is illustrated in Figure 11. The monitoring was performed over a seven hour snapshot of user lifelog; however, to save the time of the participants, the hour was scaled down to minutes. DCF accumulated 840 records of entries per user (= 4200 records in total); however, as lifelog only records when the context change and marks the enteries with starting and ending time, the number of records vary from user to user.

The monitored lifelog of the user 1 is shown in Figure 12. The context of the user is sitting at his workplace. After continuously monitoring for 2 h, LLM publishes a notification as he has been in a sedentary state according to the applied anomaly detection rule 1. The user continues his context for another hour then changes it to standing for 2 h and later on doing some exercise at the gym. According to another registered rule for his lifelog, as the user continues the exercise for 1 h, LLM publishes another notification.

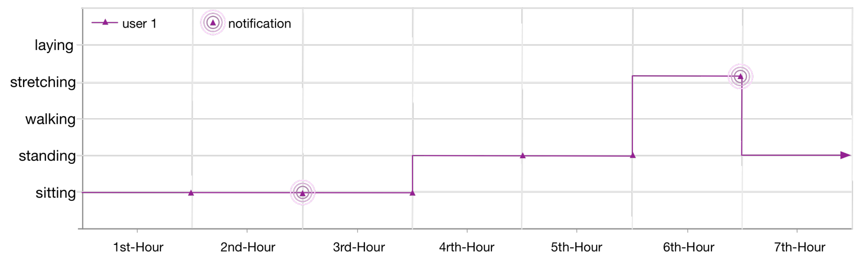


Figure 12. User 1, 7 h lifelog.

Similar to the first user, the lifelog of user 2 is illustrated in Figure 13. As the lifelog of this user is registered with anomaly detection rules 1 and 5; therefore, the user has been notified twice in 7 h. Firstly, for standing at her workplace for 3 h; nevertheless, she continues her context for another hour. Secondly, for being sedentary and sitting at her workplace for continuous 2 h.

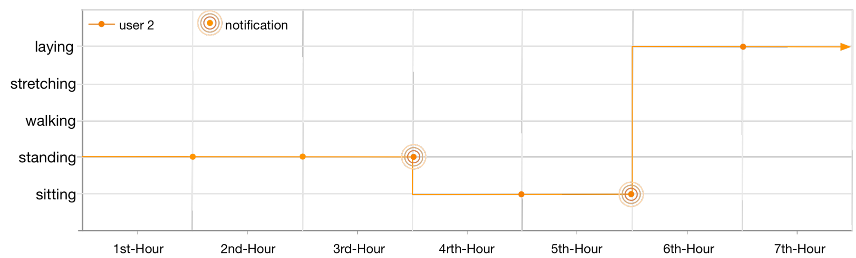


Figure 13. User 2, 7 h lifelog.

The lifelog of user 3 is illustrated in Figure 14. As the lifelog of this user is registered with anomaly detection rules 4 and 2; therefore, the user has been notified twice in 7 h. Firstly, for doing exercise at the gym for continuous 1 h; nevertheless, the user continues his context for another hour. Secondly, for being sedentary and sitting at his home for continuous 2 h.

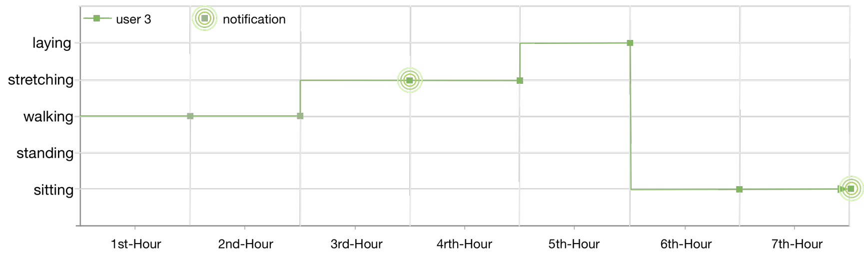


Figure 14. User 3, 7 h lifelog.

User 4 gets notified thrice (illustrated in Figure 15) as his lifelog is registered for anomaly detection rules 3 and 4. These rules are invoked by LLM when the user performs exercise at the gym for continuously 1 h twice and walking at home for 1 h.

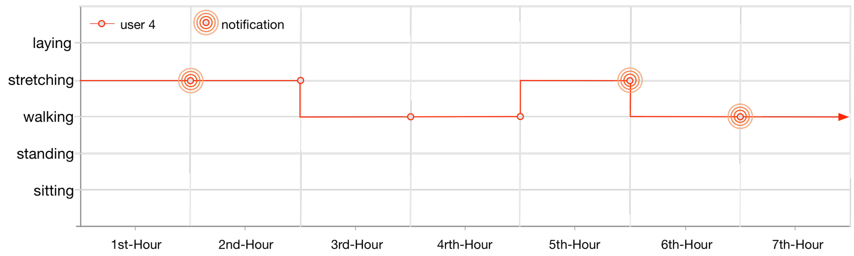


Figure 15. User 4, 7 h lifelog.

User 5 gets notified twice (illustrated in Figure 16) as her lifelog is registered for anomaly detection rules 6 and 7. These rules are invoked by LLM when the user is laying at home for continuously 4 h and sitting at home for 1 h.

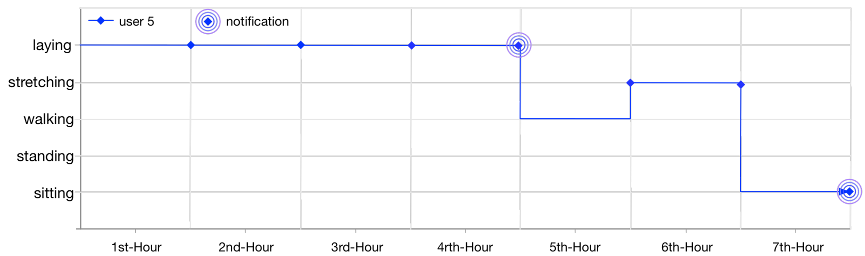
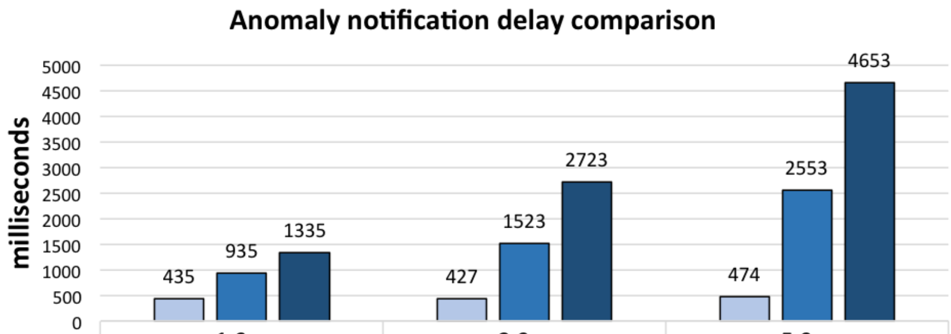


Figure 16. User 5, 7 h lifelog.

The performance of the LLM is based on its timely generation of notifications for the health and wellness services of client frameworks. For its evaluation, the delay between the occurrence of anomaly and publication of notification is evaluated. The results are crosschecked with the help of anomaly log. LLM monitors the lifelog based on a cyclic monitoring interval defined in seconds. It publishes the subscribers with the maximum delay less than the interval of monitoring cycle.

For this evaluation 1000 requests were generated for each monitoring interval and DCF server hosting LLM is running on a cloud instance with 64-bit Windows 8.1 Operating System, 16GB of RAM, and 3.10 GHz AMD A8-7600 Radeon R7 with 10 computing cores of 4C + 6G. Results from this evaluation are shown in Figure 17. From these results it can be observed that the interval duration greater than 1 s is appropriate for the evaluation environment, i.e., integrated execution with mining minds platform. Due to the distributed nature of mining minds platform, client communication overhead is introduced; therefore, leading a delay of little over the monitoring cycle interval in some cases. However, keeping the monitoring cycle interval at 3 s, LLM publishes notifications with 96.97% efficiency. In the case of 5 s, notification publication efficiency is 95.84%.



	1-Sec	3-Sec	5-Sec
Min	435	427	474
Avg	935	1523	2553
Max	1335	2723	4653

Figure 17. LLM performance evaluation.

6.5. Non-Volatile Data Persistence

The big data storage is deployed over a private cloud instance at the Ubiquitous Computing Laboratory of Kyung Hee University, South Korea [72].

To evaluate the non-volatile data persistence components over HDFS, we have performed experimentation on read and write operations. These operations have been executed for three different raw sensory data sizes, i.e., 1 Gb, 5 Gb, and 9 Gb. This data is distributed over a private cloud infrastructure having four nodes with the following configurations: (i) Name-node, equipped with Intel Core i-5 3.3 GHz, 4 Gb of RAM; (ii) First data node, equipped with an AMD 2.7 GHz, 2 Gb of RAM; and (iii) 3rd and 4th data nodes are equipped with Intel Core i-5 3.3 GHz, 2 Gb of RAM.

In this evaluation (illustrated in Figure 18), the subcomponents of data persistence and the passive data reader are evaluated. As expected, the write operation is substantially faster than the read operation. The time difference for both read and write is proportional to the volume of raw sensory data in big data storage.

To evaluate the execution and response time of the active data reader over HDFS, eight different SL-based queries are executed. These queries (described in Table 2) with varying complexities are performed over 1.7 Gb of lifelog data maintained over big data storage. The structure of this data consists of user detected locations, user recognized low- and high-level context, and record of published recommendations as a response to the anomaly detection. For the user recognized low-level context the data structure consists upon the duration (start and end time) associated with recognized high-level context from identified locations, i.e., Home, Office, Yard, Mall, Restaurant, Outdoors, and Transportation. To measure the response and execution time accurately, each query has been executed 50 times. The evaluation is illustrated in Figure 19. This execution is performed with a single mapper and its associated reducer; however, with an increase in mappers the execution time improves.

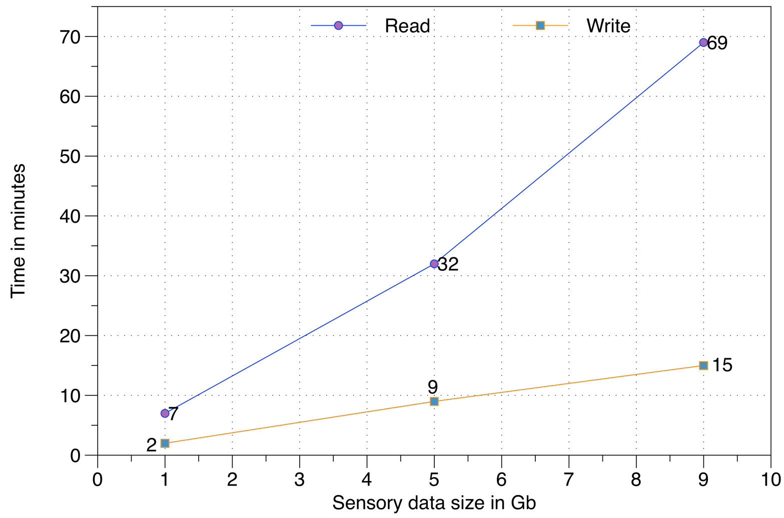


Figure 18. Read and write time over big data storage.

Table 2. Query execution scenarios.

Scenario	Query
A	SELECT count(userid) FROM userrecognizedactivity WHERE userid = 39;
B	SELECT count(userid) FROM userrecognizedactivity;
C	SELECT count(userid) FROM detected location WHERE StartTime BETWEEN '1/29/2014' AND '12/10/2014';
D	SELECT count(userid) FROM detected location WHERE LocationLabel = 'Home';
E	SELECT count(userid) FROM tblUserRecognizedHLC WHERE StartTime BETWEEN '5/10/2015,2:45:01 AM' AND '12/9/2015,11:58:31 PM';
F	SELECT count(RecommendationID) FROM tfblrecommendation WHERE RecommendationDate BETWEEN '5/18/2015,10:01:59 AM' AND '12/15/2015,2:32:09 PM';
G	SELECT count(RecommendationFeedbackID) FROM ShetblRecommendationFeedbacket3 WHERE FeedbackDate BETWEEN '5/18/2015,10:01:59 AM' AND '7/29/2015,3:20:28 PM';
H	SELECT count(UserRecognizedEmotionID) FROM tblUserRecognizedEmotion WHERE StartTime BETWEEN '5/10/2015,2:30:01 AM' AND '12/14/2015,2:00:00 PM';

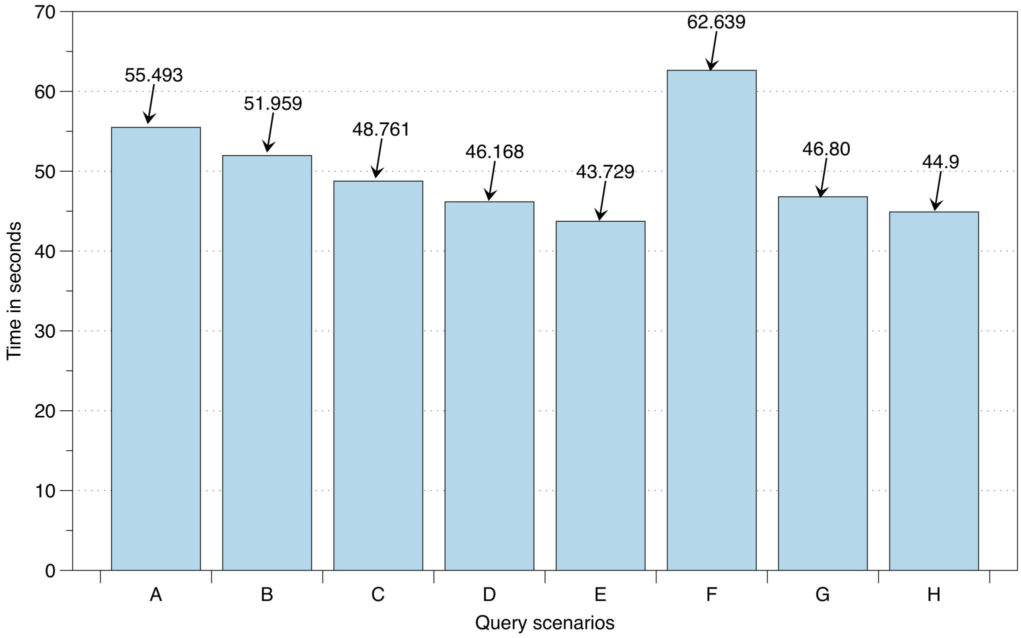


Figure 19. Query execution and response time over big data storage.

From this thorough evaluation, it is evident that DCF can accurately perform synchronization of raw sensory data from multimodal data sources. Its performance-oriented implementation has been validated by performance and load-testing. Its ability to monitor a lifelog has been tested and validated with simple to complex situations for multiple users. Furthermore, the execution-time over data maintained in big data storage has also been evaluated over real raw sensory data sets of large volumes. These evaluations are designed keeping the requirements and contributions of DCF in mind. Furthermore, the results reflect positively on the claimed novelty stated in earlier sections of this paper.

7. Conclusions

In this paper, we presented the Data Curation Framework (DCF). This framework focuses on curation of accumulated data from multimodal data sources in real time such that a context-rich user lifelog can be generated. This lifelog offers a holistic view on user activity and behavior which can further be utilized in multidimensional ways including effective interventions from healthcare professionals. The data source-independent implementation of DCF makes it more scalable and IoT compatible. Furthermore, it monitors this lifelog of registered users for the detection of anomalies. This monitoring is able to integrate static, dynamic, and complex created by the expert. DCF incorporates multi-level abstraction on the data, depending upon its usage and persistence. Frequently required user lifelog and profile data is maintained in an intermediate database; whereas, the historic and raw sensory data is maintained in non-volatile storage provided by big data technologies. This property enables DCF to support the forthcoming concepts of data-driven knowledge generation, descriptive and predictive analytics, and visualization.

Keeping the requirements of a data accumulation framework for health and wellness platforms, we have evaluated DCF for its performance, scalability, accuracy of the synchronization process of raw sensory data from multimodal data sources, monitoring of user life log, and data persistence. From the results, it is evident that DCF's implementation performs efficiently and effectively in realistic situations and scenarios while integrating with a health and wellness platform as the client.

Acknowledgments: This work was supported by a grant from the Kyung Hee University in 2013 [KHU-20130439].

Author Contributions: Muhammad Bilal Amin conceived of the original idea for the framework. He drafted the manuscript, designed the architecture and algorithms. Oresti Banos and Wajahat Ali Khan contributed in framing the idea and the design of the architecture. Hafiz Syed Muhammad Bilal implemented and evaluated the LLM component. Dinh-Mao Bui implemented and evaluated the DAS component. Taqdir Ali designed and implemented the lifelog model and LLRM component. Soung Ho Cho, Usman Akhar, Shujaat Hussain, and Jinhyuk Gong implemented and evaluated the Non-volatile data persistence components. Tae Choong Chung and Sungyoung Lee led the research in the integration of DCF as DCL for Mining Minds platform.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Banos, O.; Amin, M.B.; Khan, W.A.; Afzal, M.; Hussain, M.; Kang, B.H.; Lee, S.Y. The Mining Minds Digital Health and Wellness Framework. *BioMed. Eng. OnLine* **2016**, in press.
2. Hood, L.; Flores, M. A personal view on systems medicine and the emergence of proactive P4 medicine: Predictive, preventive, personalized and participatory. *New Biotechnol.* **2012**, *29*, 613–624.
3. Hafner, T.; Shiffman, J. The emergence of global attention to health systems strengthening. *Health Policy Plan.* **2013**, *28*, 41–50.
4. Matheson, E.; King, D.; Everett, C. Healthy lifestyle habits and mortality in overweight and obese individuals. *J. Am. Board Family Med.* **2012**, *25*, 9–15.
5. Gillen, M.; Markey, C.; Markey, P. An examination of dieting behaviors among adults: Links with depression. *Eat. Behav.* **2012**, *13*, 88–93.
6. Demark-Wahnefried, W.; Jones, L. Promoting a healthy lifestyle among cancer survivors. *Hematol./Oncol. Clin. N. Am.* **2008**, *22*, 319–342.
7. Swan, M. Health 2050: The realization of personalized medicine through crowdsourcing, the quantified self, and the participatory biocitizen. *J. Personal. Med.* **2012**, *2*, 93–118.
8. Powell, A.C.; Landman, A.B.; Bates, D.W. In search of a few good apps. *J. Am. Med. Assoc.* **2014**, *311*, 1851–1852.
9. Withings Activite. Available online: http://www.withings.com/eu/en/products/activite (accessed on 24 June 2016).
10. Garmin Vivofit. Available online: http://sites.garmin.com/en-US/vivo/vivofit/ (accessed on 24 June 2016).
11. Fitbit Surge. Available online: https://www.fitbit.com/surge (accessed on 24 June 2016).
12. Misfit Shine. Available online: http://www.misfitwearables.com/products/shine (accessed on 24 June 2016).
13. Apple Watch. Available online: http://www.apple.com/watch/ (accessed on 24 June 2016).
14. Health Kit. Available online: http://www.apple.com/ios/health/ (accessed on 24 June 2016).
15. Samsung Gear. Available online: http://www.samsung.com/us/mobile/wearable-tech (accessed on 24 June 2016).
16. LG Smartwatches. Available online: http://www.lg.com/us/smart-watches (accessed on 24 June 2016).
17. Google Fit. Available online: https://fit.google.com/fit/u/0/ (accessed on 24 June 2016).
18. Microsoft Band. Available online: https://www.microsoft.com/microsoft-band/en-us (accessed on 24 June 2016).
19. Microsoft Health. Available online: https://www.microsoft.com/microsoft-health/en-us (accessed on 24 June 2016).
20. Banos, O.; Villalonga, C.; Damas, M.; Gloesekoetter, P.; Pomares, H.; Rojas, I. Physiodroid: Combining wearable health sensors and mobile devices for a ubiquitous, continuous, and personal monitoring. *Sci. World J.* **2014**, *2014*, 1–11.
21. Oresko, J.J.; Jin, Z.; Cheng, J.; Huang, S.; Sun, Y.; Duschl, H.; Cheng, A.C. A Wearable Smartphone-Based Platform for Real-Time Cardiovascular Disease Detection Via Electrocardiogram Processing. *IEEE Trans. Inf. Technol. Biomed.* **2010**, *14*, 734–740.
22. Gurrin, C.; Smeaton, A.F.; Doherty, A.R. LifeLogging: Personal Big Data. *Found. Trends Inf. Retr.* **2014**, *8*, 1–125.
23. The Future of Lifelogging. Available online: http://blog.autographer.com/2013/05/the-future-of-lifelogging-interview-with-gordon-bell/ (accessed on 24 June 2016).
24. Zini, F.; Reinstadler, M.; Ricci, F. Life-logs Aggregation for Quality of Life Monitoring. In Proceedings of the 5th International Conference on Digital Health 2015, Florence, Italy, 18–20 May 2015.
25. Zhou, L.M.; Gurrin, C. A survey on life logging data capturing. SenseCam Symposium: Oxford, UK, 2012.
26. Want, R.; Schilit, B.N.; Jenson, S. Enabling the Internet of Things. *IEEE Comput.* **2015**, *48*, 28–35.

27. Ling, T.; Kang, B.H.; Johns, D.P.; Walls, J.; Bindoff, I. Expert-Driven Knowledge Discovery. In Proceedings of the Fifth International Conference on Information Technology: New Generations, 2008, ITNG 2008, Las Vegas, NV, USA, 7–9 April 2008; pp. 174–178.
28. Hussain, M.; Afzal, M.; Ali, T.; Ali, R.; Khan, W.A.; Jamshed, A.; Lee, S.; Kang, B.H.; Latif, K. Data-driven knowledge acquisition, validation, and transformation into HL7 Arden Syntax. *Artif. Intell. Med.* **2015**, in press.
29. Hussain, S.; Lee, S. Visualization and descriptive analytics of wellness data through Big Data. In Proceedings of the 2015 Tenth International Conference on Digital Information Management (ICDIM), Jeju Island, South Korea, 21–23 October 2015; pp. 69–71.
30. Budgaga, W.; Malensek, M.; Pallickara, S.; Harvey, N.; Breidt, F.J.; Pallickara, S. Predictive Analytics Using Statistical, Learning, and Ensemble Methods to Support Real-time Exploration of Discrete Event Simulations. *Future Gener. Comput. Syst.* **2016**, *56*, 360–374.
31. Mining Minds Overview. Available online: http://www.miningminds.re.kr/english/ (accessed on 24 June 2016).
32. Chen, H.; Jia, X. New requirements and trends of mHealth. In Proceedings of IEEE 14th International Conference on e-Health Networking, Applications and Services (Healthcom), Beijing, China, 10–13 October 2012; pp. 27–31.
33. Elgazzar, K.; Aboelfotoh, M.; Martin, P.; Hassanein, H. Ubiquitous health monitoring using mobile web services. *Procedia Comput. Sci.* **2012**, *10*, 332–339.
34. Kim, H.; Lee, K.; Kim, H.; Kim, J. Using mobile phones in healthcare management for the elderly. *Maturitas* **2014**, *79*, 381–388.
35. Gaggioli, A.; Pioggia, G.; Tartarisco, G.; Baldus, G.; Corda, D.; Cipresso, P.; Riva, G. A Mobile Data Collection Platform for Mental Health Research. *Pers. Ubiquitous Comput.* **2013**, *2*, 241–251.
36. Spina, G.; Roberts, F.; Weppner, J.; Lukowicz, P.; Amft, O. CRNTC+: A smartphone-based sensor processing framework for prototyping personal healthcare applications. In Proceedings of the 7th International Conference on Pervasive Computing Technologies for Healthcare, Venice, Italy, 5–8 May 2013.
37. Chen, P.H.; Chen, H.M. Framework design-integrating an android open platform with multi-interface biomedical modules for physiological measurement. *J. Converg. Inf. Technol.* **2012**, *7*, 310–319.
38. Prasad, A.; Peterson, R.; Mare, S.; Sorber, J.; Paul, K.; Kotz., D. Provenance framework for mHealth. In Proceedings of the 5th International Conference on Communication Systems and Networks, Bangalore, India, 7–10 January 2013; pp. 1–6.
39. Belimpasakis, P.; Roimela, K.; You, Y. Experience Explorer: A Life-Logging Platform Based on Mobile Context Collection. In Proceedings of the 2009 Third International Conference on Next Generation Mobile Applications, Services and Technologies, Cardiff, UK, 15–18 September 2009; pp. 77–82.
40. Rawassizadeh, R.; Tomitsch, M.; Wac, K.; Tjoa, A.M. UbiqLog: A generic mobile phone-based life-log framework. *Pers. Ubiquitous Comput.* **2013**, *17*, 621–637.
41. Ferreira, D.; Kostakos, V.; Dey, A.K. AWARE: Mobile context instrumentation framework. *Front. ICT* **2015**, *2*, 1–9.
42. Dobbins, C.; Merabti, M.; Fergus, P.; Jones, D. Creating human digital memories with the aid of pervasive mobile devices. *Pervasive Mob. Comput.* **2014**, *12*, 160–178.
43. Dobbins, C.; Merabti, M.; Fergus, P.; Llewellyn-Jones, D. The Big Data Obstacle of Lifelogging. In Proceedings of the 28th International Conference on Advanced Information Networking and Applications Workshops, AINA 2014 Workshops, Victoria, BC, Canada, 13–16 May 2014; pp. 922–926.
44. The SenseWear Armband. Available online: https://templehealthcare.wordpress.com/the-sensewear-armband (accessed on 24 June 2016).
45. Dwyer, T.J.; Alison, J.A.; McKeough, Z.J.; Elkins, M.R.; Bye, P.T.P. Evaluation of the SenseWear activity monitor during exercise in cystic fibrosis and in health. *Respir. Med.* **2009**, *103*, 1511–1517.
46. Almeida, G.J.; Wasko, M.C.M.; Jeong, K.; Moore, C.G.; Piva, S.R. Physical Activity Measured by the SenseWear Armband in Women With Rheumatoid Arthritis. *Phys. Ther.* **2011**, *91*, 1367–1376.
47. Hodges, S.; Williams, L.; Berry, E.; Izadi, S.; Srinivasan, J.; Butler, A.; Smyth, G.; Kapur, N.; Wood, K. Chapter SenseCam: A Retrospective Memory Aid. Available online: https://www.microsoft.com/en-us/research/publication/sensecam-a-retrospective-memory-aid/ (accessed on 24 June 2016).

48. McCarthy, S.; McKevitt, P.; McTear, M.F.; Sayers, H.M. MemoryLane: An intelligent mobile companion for elderly users. In Proceedings of the Seventh Information Technology and Telecommunication Conference IT&T, Dublin, Ireland, 25–26 October 2007; pp. 72–82.
49. Lee, H.; Smeaton, A.F.; Jones, G.; Blighe, M.; Byrne, D.; Doherty, A.; Gurrin, C. Constructing a SenseCam visual diary as a media process multimedia systems. *Multimedia Syst.* **2008**, *14*, 341–349.
50. Kelly, P.; Doherty, A.; Berry, E.; Hodges, S.; Batterham, A.M.; Foster, C. Can we use digital life-log images to investigate active and sedentary travel behaviour? Results from a pilot study. *Int. J. Behav. Nutr. Phys. Act.* **2011**, *8*, 1–9.
51. Blum, M.; Pentland, A.; Troster, G. InSense: Interest-Based Life Logging. *IEEE Multimed.* **2006**, *13*, 40–48.
52. Balasubramaniam, S.; Kangasharju, J. Realizing the Internet of Nano Things: Challenges, Solutions, and Applications. *Computer* **2013**, *46*, 62–68.
53. Akyildiz, I.F.; Jornet, J.M. The Internet of nano-things. *IEEE Wirel. Commun.* **2010**, *17*, 58–63.
54. Rawassizadeh, R.; Tomitsch, M.; Nourizadeh, M.; Momeni, E.; Peery, A.; Ulanova, L.; Pazzani, M. Energy-Efficient Integration of Continuous Context Sensing and Prediction into Smartwatches. *Sensors* **2015**, *15*, 22616–22645.
55. Kapadia, A.; Kotz, D.; Triandopoulos, N. Opportunistic sensing: Security challenges for the new paradigm. In Proceedings of the 2009 First International Communication Systems and Networks and Workshops, Athens, Greece, 5–10 January 2009; pp. 1–10.
56. NetBio. Available online: http://netbio.com/ (accessed on 24 June 2016).
57. Samsung SAMI IO. Available online: https://www.samsungsami.io (accessed on 24 June 2016).
58. Bricon-Souf, N.; Newman, C.R. Context awareness in health care: A review. *I. J. Med. Inform.* **2007**, *76*, 2–12.
59. Ahmad, M.; Amin, M.B.; Hussain, S.; Kang, B.; Cheong, T.; Lee, S. Health Fog: A novel framework for health and wellness applications. *J. Supercomput.* **2016**, 1–19, doi:10.1007/s11227-016-1634-x.
60. Fog Computing. Available online: http://internetofthingsagenda.techtarget.com/definition/fog-computing-fogging (accessed on 24 June 2016).
61. Microsoft Azure. Available online: https://azure.microsoft.com (accessed on 24 June 2016).
62. Pervez, Z.; Khattak, A.M.; Lee, S.; Lee, Y.K.; Huh, E.N. Oblivious access control policies for cloud based data sharing systems. *Computing* **2012**, *94*, 915–938.
63. REST. Available online: https://en.wikipedia.org/wiki/Representational_state_transfer (accessed on 24 June 2016).
64. Node.js. Available online: https://nodejs.org/en/ (accessed on 24 June 2016).
65. JSON. Available online: http://www.json.org (accessed on 24 June 2016).
66. Welcome to Apache Hadoop. Available online: https://hadoop.apache.org (accessed on 24 June 2016).
67. Gamma, E.; Helm, R.; Johnson, R.; Vlissides, J. *Design Patternsl: Elements of Reusable Object-Oriented Software*; Addison-Wesley Professional: Boston, MA, USA, 1994.
68. Object Copying. Available online: https://en.wikipedia.org/wiki/Object_copying (accessed on 24 June 2016).
69. Villalonga, C.; Banos, O.; Khan, W.A.; Ali, T.; Razzaq, M.A.; Lee, S.; Pomares, H.; Rojas, I. Ambient Assisted Living. In *3rd Symposium of Ubiquitous Computing and Ambient Intelligence 2008 (pp. 86–94)*; Springer: Berlin, Germany, 2008; pp. 86–94.
70. Apache Hive. Available online: https://hive.apache.org (accessed on 24 June 2016).
71. Kinect. Available online: https://en.wikipedia.org/wiki/Kinect (accessed on 24 June 2016).
72. Ubiquitous Computing Lab. Available online: http://uclab.khu.ac.kr/index.php (accessed on 24 June 2016).

Section 1.3

Information Curation Layer (ICL) Low level Context Awareness Paper

Information Sciences 369 (2016) 317–333

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Interactive activity recognition using pose-based spatio–temporal relation features and four-level Pachinko Allocation Model ☆

Thien Huynh-The [a], Ba-Vui Le [a], Sungyoung Lee [a,*], Yongik Yoon [b]

[a] *Department of Computer Science & Engineering, Kyung Hee University (Global Campus), 1732 Deogyeong-daero, Giheung-gu, Yongin-si, Gyeonggi-do, 446–701, Korea*
[b] *Department of Multimedia Science, Sookmyung Women's University, Cheongpa-ro 47-gil 100 (Cheongpa-dong 2ga), Yongsan-gu, Seoul, 04310, Korea*

A R T I C L E   I N F O

Article history:
Received 18 March 2016
Revised 8 June 2016
Accepted 14 June 2016
Available online 25 June 2016

Keywords:
Human interaction
Activity recognition
Four-level PAM

A B S T R A C T

In this paper, we go beyond the problem of recognizing video-based human interactive activities. We propose a novel approach that permits to deeply understand complex person-person activities based on the knowledge coming from human pose analysis. The joint coordinates of interactive objects are first located by an efficient human pose estimation algorithm. The relation features consisting of the intra and inter-person features of joint distance and angle, are suggested to use for describing the relationships between body components of the individual persons and the interacting two participants in the spatio-temporal dimension. These features are then provided to the codebook construction process, in which two types of codeword are generated corresponding to distance and angle features. In order to explain the relationships between poses, a flexible hierarchical topic model constructed by four layers is proposed using the Pachinko Allocation Model. The model is able to represent the full correlation between the relation features of body components as codewords, the interactive poselets as subtopics, and the interactive activities as super topics. Discrimination of complex activities presenting similar postures is further obtained by the proposed model. We validate our interaction recognition method on two practical data sets, the BIT-Interaction data set and the UT-Interaction data set. The experimental results demonstrate that the proposed approach outperforms recent interaction recognition approaches in terms of recognition accuracy.

1. Introduction

In recent decades, human activity recognition has been an active research area in computer vision and artificial intelligence due to its wide range of potential applications, such as indoor-outdoor surveillance, human robot interaction, and

☆ This work was supported by Institute for Information & communications Technology Promotion (IITP) grant funded by the Korea government (MSIP) (B0101-15-1282-00010002, Suspicious pedestrian tracking using multiple fixed cameras). This research was supported by the MSIP, Korea, under the G-ITRC support program (IITP-2015-R6812-15-0001) supervised by the IITP. This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP) NRF-2014R1A2A2A01003914.

* Corresponding author. Fax: +82312012514.
E-mail addresses: thienht@oslab.khu.ac.kr (T. Huynh-The), lebavui@oslab.khu.ac.kr (B.-V. Le), sylee@oslab.khu.ac.kr (S. Lee), yiyoon@sookmyung.ac.kr (Y. Yoon).

http://dx.doi.org/10.1016/j.ins.2016.06.016

human-computer interaction [19,36]. Although it has received attention from the scientist community, an effective method for recognizing activities in the real environment still remains a challenge because of variations of appearance, mutual occlusion, and object interactions. Most existing approaches have concentrated on the low-level features, known as local spatial-temporal features [20,39], instead of the human body representation, known as skeleton, due to limitations in the pose estimation performance. The appearance of limbs strongly varies due to variations in clothing and body shape. Besides, human objects need to be pre-localized and scaled in size by a detector as an initial assumption. However, some notable results in recent years on the human pose estimation have motivated research in human activity recognition [62].

Human activities considered in computer vision can be categorized into two classes: single action and group action. Some approaches were proposed to recognize the activities of one actor as walking, jogging, running, hand waving [56]. The interactive activities between one actor and an object were considered in many recent studies [58]. Some daily life activities in indoor environments can be listed as eating, drinking, typing, and answering phone [33]. Group action, generally performed by visual separable people with complicated interactions, such as walking together, approaching, gathering, has been investigated using human-based features and tracking information for detection and recognition [8,10,45]. Few works handled complex activities of two or more human objects such as hand shaking, hugging, punching, and patting [21,22,60], in which the spatio-temporal relations between two objects are described for activity representation by an interaction model.

In this work, we propose an effective method for human interaction recognition based on a flexible topic model. As a preprocess, locating human articulation is performed by an effective pose estimation algorithm [57]. For representing interactions, the spatio-temporal relation features, calculated from the articulated-pose coordinates, are suggested to use, which include the intra and inter-person features of joint distance and angle. These features describe the relationships between body components of single persons and also interactive participants. To overcome the problem of similar posture interaction representation, we further propose a hierarchical model based on the Pachinko Allocation Model (PAM) to exhibit the relations between features, interactive poselets, and interactions. Concretely, relative features are mapped to visual words by *k*-means clustering with a constructed codebook. In the topic modeling process, a four-level model which is flexible to connect the upper and lower layers captures the correlations between poselets through codewords and the correlations between interactions through codewords and poselets. Finally, Support Vector Machine (SVM) method is then applied for solving the multi-class classification problem.

The rest of the paper is organized as follows: Section 2 provides discussion on related works. Section 3 describes the proposed method for interaction recognition. The experimental setup, results, and discussion are then presented Section 4. Finally, the conclusions and suggestion for future works are summarized in Section 5.

2. Related work

2.1. Human pose estimation

Human pose estimation, one of the most important stages in the human activity recognition, has received attention in recent years, in which the articulated-pose coordinates or the body part areas in the still images are given. A mostly used technique is the spatial structure coding, often described by the probabilistic graphical model. Although structural-based graphical models allow exact and efficient part inference, they sometimes incorrectly localize body parts in complicated situations. Motivated by the pictorial structure model introduced by Fischler et al. [17], Huttenlocher et al. [16] modeled a human object by a collection of parts arranged in a deformable configuration. By learning latent relationships between different body parts from annotated images, Eichner et al. [13] improved estimation accuracy for unusual poses. From learned appearance models, body parts are ably plugged into any pictorial structure engines. A cascaded model [42] enhances estimation accuracy at different resolutions, where coarse models filter the pose space via max-marginals. Andriluka et al. [2] calculated dense appearance representations using shape context descriptors and then used AdaBoost to train discriminative part classifiers. To obtain the tractability and flexibility, Sapp et al. [41] combined a pictorial structure inference with a non-parametric approach using a subset of model parameters. Furthermore, a shape-based kernel for upper-body pose similarity and a leave-one-out loss function were developed in the learning phase. Building on a successful pictorial structures model, Tian et al. [47] developed an image conditioned model that integrated higher order dependent variables. In recent years, a general and flexible mixture model introduced by Yang et al. [57] based on the standard pictorial structure model captures spatial relations between part positions and co-occurrence relations between part mixtures. Moreover, two novel criteria, the percentage of correct key points (PCK) and the average precision of key points (APK), were proposed to evaluate pose estimation and articulation location, separately and jointly. Two criteria addressed the current shortcomings that are incorrect matching and matching multiple poses to the ground truth. The algorithm has shown notable results of pose estimation with state-of-the-art approaches on practical datasets.

2.2. Features for activity representation

Existing methods for recognizing actions can be categorized into two groups of feature type: local spatio-temporal interest point (STIP) [7,9,12,28,40,43,48,52], and body-pose feature [29,31,49,51,54,56,61]. Activity recognition methods using local features usually develop an effective feature descriptor which provides a pattern or distinct structure found in an image, such as point, edge, and small image path. Local features extracted in the time domain are usually learned by classification techniques, such as Decision Tree, *K*-Nearest Neighbor, Support Vector Machine, and so on. Human pose based approaches, meanwhile, recognize activities based on the features extracted from articulated joint coordinates, contained by an estimation algorithm, such as distance, angle, velocity, and plane. Local feature based approaches are practically restricted by non-robustness in dynamic scenes, low accuracy with complex actions, and poor relation in interactions. Wu et al. [52] exploited the interest point detector, proposed by Dollar et al.[12], including two separated filters, 2D Gaussian filter and 1D Gabor filter, to produce the high response at each point with significant spatio-temporal intensity variations. Some well-known feature descriptors were usually applied for feature extraction, such as Histogram of Oriented Gradients (HOG) [7,30], Histogram of Optical Flow (HOF) [48], and Scale-Invariant Feature Transform (SIFT) [43], Features From Accelerated Segment Test (FAST) [59], and Motion-Constrained SIFT (MoSIFT) [28]. Based on non-negative matrix factorization (NMF), Eweiwi et al. [14] formulated an informed sampling for action specific regions from interest points to obtain basic flows. In order to avoid outliers from the feature extraction process, Samanta et al. [40] proposed a 3D space-time interest point descriptor using Haar wavelet transform. Motion trajectory providing plentiful spatio-temporal information was potentially utilized for activity recognition [3,37]. A probabilistic trajectory analysis framework [34] was developed for understanding scene activity, in which the trajectory information was clustered into spatial routes and modeled by a hidden Markov model. Compared with STIP-based approaches, pose based methods prove the advantage in complex action and interaction recognition, however, the classification accuracy is sometimes hanged by pose estimation performance. In [31], each action was encoded as a series of synthetic 2D human poses depicted from a wide range of viewpoints. The best matched sequences of actions are then tracked by Viterbi algorithm. Instead of separating pose estimation and action recognition as two individual systems, Yang et al. [56] designed an integrated fashion system that jointly considered poses and actions to directly obtain the pose information. Wang et al. [49] proposed an efficient pose based recognition system, in which the final human articulation was constructed from local parts by a tree structural graphical model. In [51], initial skeletons were collected from a key pose dictionary and particle filters then tracked human upper body parts for activity recognition. A novel feature in [61], namely Poselet Activation Vector, was combined with contextual information, obtained by sparse coding on foreground and background, for action explanation. Extensive pyramidal feature (EPF) constructed from the Gabor filter, Gaussian pyramid, and wavelet transform, was proposed by Liu et al. [29] for pose presentation. The orientation, intensity, and contour information were also encoded by EPFs. A pose dictionary established by shape of contour points from the human silhouette was formulated by Cai et al. [54] to recognize single activities.

2.3. Interactive activity modeling

Another issue, widely considered in the human activity recognition, is interaction modeling from sparse features. This issue is much more importance in the interaction recognition because the inter-relation between objects should be further modeled besides the intra-relation within each object. In [52], a Latent Structure SVM model was introduced for formulating the relationship of action classes – scene classes and the compatibility of multilevel features – action classes. To capture the semantic meaning of body-parts between two interactive objects, Alazrai et al. [1] proposed a motion-pose geometric descriptor (MPGD) based on the concept of anatomical planes. Moreover, a hierarchical framework, consisted of one representation layer and three classification layers, was designed in the recognition phase. Kong et al. [24,25] modeled the actions by large-scale global features and local body part features using a discriminative model to recognize potential interactions. Ryoo et al. [39] introduced a novel spatio-temporal relation matching model to understand human activities captured in their UT-Interaction dataset. A unified-discriminative model was considered for interaction recognition by Meng et al. [32] using inter-person relation features. In [18], modeling activities and matching them in the spatio-temporal dimension were implemented by String of Feature Graph Model. This model is able to recognize activities involving interaction between multiple objects. Recently, topic modeling has been used as an efficient solution for action representation based on visual words, coded from features by vector quantization techniques. Two models, Semilatent Diriclet Allocation (S-LDA) and Semilatent Correlated Topic Model (S-CTM), were suggested for human action recognition in [50]. By pushing the information provided by class labels of training data into these models, the latent topics were matched correctly class labels with quite high accuracy. A novel variant of LDA model [4] including two-layer topics, the mid-level topic describing the local spatial temporal patterns (STPs) and the top-level topic representing mixture distribution of STPs, was proposed by Yang et al. [55] for action group learning. Based on the original LDA [4], Xiao et al. [53] constructed a visual word vocabulary from the STIP and HOG3D features of the cuboids around interest points. A type-2 fuzzy topic model (T2 FTM) [5] was recommended to encode the higher-order uncertainty of each topic from 2D visual words. Although these topic models are impressive in single action recognition, they are sometimes inappropriate and restrictive for interaction because of the compact relations of individual objects and also interactive objects. Mapping directly the features as the visual words to the action as topics might ignore some intermediate states when two poses in an interaction are quite similar together in visualization. In summary, the use of STIP for activity recognition in current approaches could not guarantee the robustness because of STIP's fragility in practical environments. Posture information is really valuable to understand actions in the current frame; however, modeling the articulated-pose features effectively in the time dimension has been not presented in the most of discussed methods. In additions, describing the relations between pose features and actions by some principal modeling techniques cannot provide a deep understanding, especially with the interactive activities. Therefore, we believe that a novel topic model, able to explain the hierarchical relationships of features-postures-activities, is a feasible solution

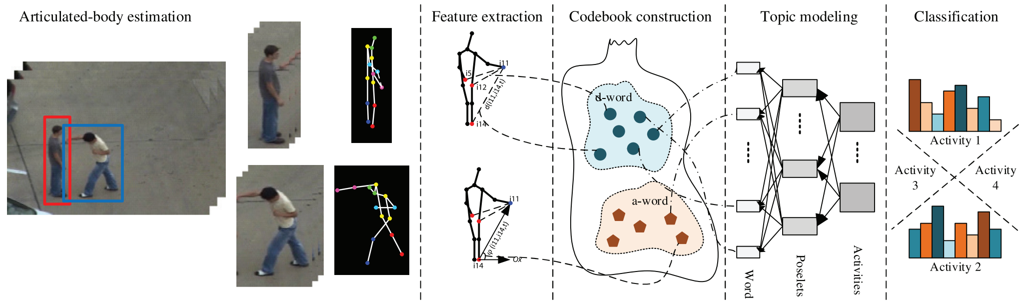


Fig. 1. The workflow of a proposed interaction recognition method using spatio-temporal relation features and topic model. The joint coordinates were achieved by Yang's estimation algorithm on each detected human object. The features used for representing spatio-temporal relation consist of joint distance and angle between pairs of joints. Then the codebook including two types of codewords, *d*-word and *a*-word corresponding to distance and angle feature, was constructed by *k*-mean clustering technique. A hierarchical topic model was suggested for describing the correlation between codeword, poselet, and activity. Finally, interactive activities were classified by a Multi-class SVM.

for remaining limitations. Nevertheless, an expensively computational resource may be needed for the modeling process and becomes a drawback in the comparison with the state-of-the-art methods.

3. Interactive activity recognition method

The proposed interaction recognition method consists of the following modules: articulated-body estimation, spatio-temporal relation feature extraction, codebook construction, topic modeling, and activity classification as Fig. 1.

3.1. Articulated-body estimation

In this work, the authors use an efficient articulated-body estimation algorithm, introduced by Yang et al., to locate the joint coordinates with two patterns of 14-part and 26-part [57] as shown in Fig. 2a–b. Given the bounding box of a human object, key points are evaluated by two criteria, PCK and APK, and then modeled into a tree graphical structure where the nodes of the graph represents body components, and the edges between components illustrate pairwise geometric relations. To detect human objects flexibly and search poses competently in images, a full core function was formulated by associating a compatibility function for part configuration evaluation with a corresponding configuration containing the information of part types and locations. Due to capturing dependence of local appearance on spatial geometry, Yang's model achieved better estimation speed and accuracy if compared with classic articulation models [13,16] using the pictorial structure [17] on the real-life datasets.

In order to boost the performance of articulated joint locating, multiple pose estimators are trained on the testing data sets hereafter to control variance among activities. In the training stage, samples of a particular activity in the Image Parse data set [35] are chosen as positive samples and remainders in the INRIA Person data set [11] are treated as negative samples. The INRIA Person data set comprises non-person images while the Image Parse data set contains 305 pose-annotated images of greatly articulated full-body human poses. In the testing stage, each estimator is tried one by one to select the best result with the maximum score. By this strategy, the estimation accuracy is fairly improved. The data set of 2D coordinates is obtained as the output of this phase.

3.2. Spatio-temporal relation feature extraction

In the interaction case, the human poses should be discriminated on different activities. The pose of individual human object are good enough for single action recognition; however, for interaction recognition, the relations between two objects have to be exploited due to the pose similarity. In the UT-Interaction dataset [39], Punching and Pushing, which present one human object in the standing pose and another in the acting pose should be differentiated through the active poses and the object-object relations. Because most of activities in the practical environments are performed within a time duration, monitoring objects in the time dimension is necessary, in which the temporal features describing the posture translation are extracted. The spatio-temporal relation features are therefore studied from the pose coordinate dataset. The authors calculate the distance of two joints and the angle between a joint vector and the horizontal axis. Fig. 2c–j describes eight feature types extracted from two objects.

Intra-spatio joint distance: The joint distance feature (see Fig. 2c) is defined as the Euclidean distance between a pair of two joints for each human object in a frame; therefore calculated as follows:

$$d_X(i,j,t) = \left\| p^t_{i,X} - p^t_{j,X} \right\| \tag{1}$$

where $p^t_{i,X} \in \Re^2$ is coordinate of joint i belongs to the human object X at the time $t \in T$ corresponding to the tth frame.

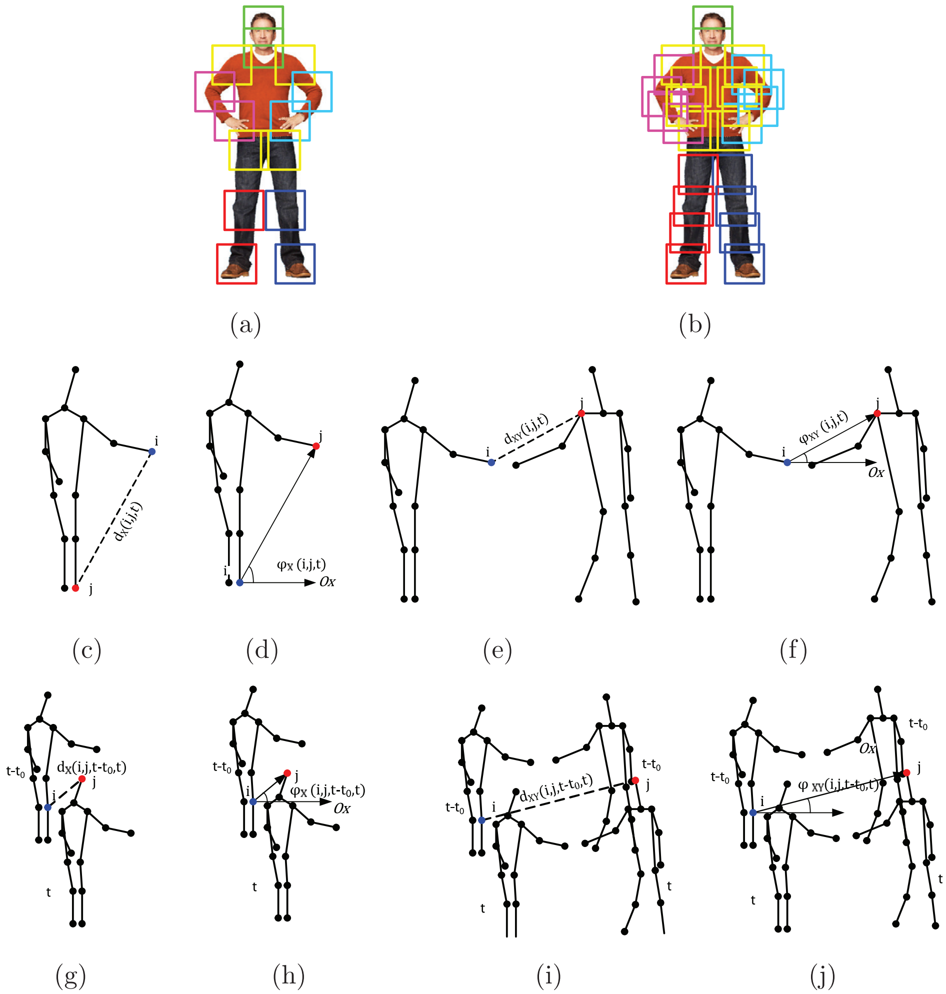


Fig. 2. Illustrations of two articulation patterns: (a) 14-part pattern, (b) 26-part pattern; and the extracted features using the joint coordinate dataset: (c) Intra-spatio joint distance, (d) Intra-spatio joint angle, (e) Inter-spatio joint distance, (f) Inter-spatio joint angle, (g) Intra-temporal joint distance, (h) Intra-temporal joint angle, (i) Inter-temporal joint distance, (j) Inter-temporal joint angle.

Intra-spatio joint angle: The joint angle feature (see Fig. 2d) is defined as the angle between the joint vector $\overrightarrow{p_i p_j}$ and the horizontal axis $\overrightarrow{Ox}$:

$$\varphi_X(i, j, t) = \angle\left(\overrightarrow{p^t_{i,X} p^t_{j,X}}, \overrightarrow{Ox}\right) \tag{2}$$

Inter-spatio joint distance: The inter-spatio joint distance feature (see Fig. 2e) is calculated by Eq. 1, where joints belong to different objects. Particularly, it is measured as follows:

$$d_{XY}(i, j, t) = \left\| p^t_{i,X} - p^t_{j,Y} \right\| \tag{3}$$

where $\left\{ p^t_{i,X}, p^t_{j,Y} \right\} \in \Re^2$ are the 2D location coordinates of joint i belongs to the human object X and joint i belongs to the human object Y at the tth frame.

Inter-spatio joint angle: The inter-spatio joint angle feature (see Fig. 2f) is developed from Eq. 2 for two objects:

$$\varphi_{XY}(i, j, t) = \angle\left(\overrightarrow{p^t_{i,X} p^t_{j,Y}}, \overrightarrow{Ox}\right) \tag{4}$$

Intra-temporal joint distance: The intra-temporal joint distance (see Fig. 2g) represents the Euclidean distance between pair of joints belonging to one human object at the current tth frame and the previous $(t - t_0)$th frame:

$$d_X(i, j, t - t_0, t) = \left\| p^{t-t_0}_{i,X} - p^t_{j,X} \right\| \tag{5}$$

where t_0 indicates the time length which is also understood as the number of frames.

Table 1
Category of extracted features.

Feature category	Term	Codebook size
Spatial distance	D^S_X, D^S_Y, D^S_{XY}	K
Temporal distance	$D^T_X, D^T_Y, D^T_{XY}, D^T_{YX}$	K
Spatial angle	$\Phi^S_X, \Phi^S_Y, \Phi^S_{XY}$	K
Temporal angle	$\Phi^T_X, \Phi^T_Y, \Phi^T_{XY}, \Phi^T_{YX}$	K
Spatio-temporal distance	$D^S_X, D^S_Y, D^S_{XY}, D^T_X, D^T_Y, D^T_{XY}, D^T_{YX}$	K
Spatio-temporal angle	$\Phi^S_X, \Phi^S_Y, \Phi^S_{XY}, \Phi^T_X, \Phi^T_Y, \Phi^T_{XY}, \Phi^T_{YX}$	K
Spatial distance-angle	$D^S_X, D^S_Y, D^S_{XY}, \Phi^S_X, \Phi^S_Y, \Phi^S_{XY}$	$2K$
Temporal distance-angle	$D^T_X, D^T_Y, D^T_{XY}, D^T_{YX}, \Phi^T_X, \Phi^T_Y, \Phi^T_{XY}, \Phi^T_{YX}$	$2K$
Merged feature	$D^S_X, D^S_Y, D^S_{XY}, D^T_X, D^T_Y, D^T_{XY}, D^T_{YX}$ $\Phi^S_X, \Phi^S_Y, \Phi^S_{XY}, \Phi^T_X, \Phi^T_Y, \Phi^T_{XY}, \Phi^T_{YX}$	$2K$

Intra-temporal joint angle: The intra-temporal joint angle (see Fig. 2h) describes the angle between the joint vector $\overrightarrow{p_i^{t-t_0}p_j^t}$ and the horizontal axis:

$$\varphi_X(i,j,t-t_0,t) = \angle\left(\overrightarrow{p_{i,X}t-t_0p^t_{j,X}}, \overrightarrow{Ox}\right) \tag{6}$$

Inter-temporal joint distance: The inter-temporal joint distance (see Fig. 2i) formulates the Euclidean distance between pairs of joints belonging to two different objects at different frames:

$$\begin{aligned} d_{XY}(i,j,t-t_0,t) &= \left\| p^{t-t_0}_{i,X} - p^t_{j,Y} \right\| \\ d_{YX}(i,j,t-t_0,t) &= \left\| p^{t-t_0}_{i,Y} - p^t_{j,X} \right\| \end{aligned} \tag{7}$$

where $d_{XY}(i,j,t-t_0,t)$ is the distance between joint i of the object X at the $(t-t_0)$th frame and joint j of the object Y at the current frame while an opposite case with $d_{YX}(i,j,t-t_0,t)$.

Inter-temporal joint angle: Similarly, the inter-temporal joint angle (see Fig. 2j) expresses the angle features between two different objects in different frames:

$$\begin{aligned} \varphi_{XY}(i,j,t-t_0,t) &= \angle\left(\overrightarrow{p^{t-t_0}_{i,X}p^t_{i,Y}}, \overrightarrow{Ox}\right) \\ \varphi_{YX}(i,j,t-t_0,t) &= \angle\left(\overrightarrow{p^{t-t_0}_{i,Y}p^t_{i,X}}, \overrightarrow{Ox}\right) \end{aligned} \tag{8}$$

Due to the difference in unit, distance and angle features have to be normalized as follows:

$$\widehat{d} = \frac{d - d_{\min}}{d_{\max} - d_{\min}} \tag{9}$$

$$\widehat{\varphi} = \frac{\varphi}{2\pi} \tag{10}$$

All features are obviously summarized and categorized into classes of feature types and dimensions as shown in Table 1, where the terms in categories are identified as follows:

$$\begin{aligned}
D^S_X &= \left\{ d(i,j,t) \middle| i \in X^t, j \in X^t, i \neq j \right\} \\
D^S_Y &= \left\{ d(i,j,t) \middle| i \in Y^t, j \in Y^t, i \neq j \right\} \\
D^S_{XY} &= \left\{ d(i,j,t) \middle| i \in X^t, j \in Y^t \right\} \\
D^T_X &= \left\{ d(i,j,t-t_0,t) \middle| i \in Xt-t_0, j \in X^t \right\} \\
D^T_Y &= \left\{ d(i,j,t-t_0,t) \middle| i \in Yt-t_0, j \in Y^t \right\} \\
D^T_{XY} &= \left\{ d(i,j,t-t_0,t) \middle| i \in Xt-t_0, j \in Y^t \right\} \\
D^T_{YX} &= \left\{ d(i,j,t-t_0,t) \middle| i \in Yt-t_0, j \in X^t \right\} \\
\Phi^S_X &= \left\{ \varphi(i,j,t) \middle| i \in X^t, j \in X^t, i \neq j \right\} \\
\Phi^S_Y &= \left\{ \varphi(i,j,t) \middle| i \in Y^t, j \in Y^t, i \neq j \right\} \\
\Phi^S_{XY} &= \left\{ \varphi(i,j,t) \middle| i \in X^t, j \in Y^t \right\} \\
\Phi^T_X &= \left\{ \varphi(i,j,t-t_0,t) \middle| i \in Xt-t_0, j \in X^t \right\} \\
\Phi^T_Y &= \left\{ \varphi(i,j,t-t_0,t) \middle| i \in Yt-t_0, j \in Y^t \right\}
\end{aligned}$$

$$\Phi_{XY}^{T} = \left\{ \varphi(i, j, t - t_0, t) \middle| i \in Xt - t_0, j \in Y^t \right\}$$
$$\Phi_{YX}^{T} = \left\{ \varphi(i, j, t - t_0, t) \middle| i \in Yt - t_0, j \in X^t \right\} \quad (11)$$

Compared with the spatial feature sets only describing the posture relations between two objects in the current frame, the temporal feature sets containing the information of body component translation are predicted to achieve a higher recognition accuracy; however, more computational resources may be needed for calculating temporal features. For example, the temporal distance set has four terms D_X^T, D_Y^T, D_XY^T, and D_YX^T instead of three terms in the spatial distance set as D_X^S, D_Y^S, and D_XY^T. If the 14-part pattern is used, D_X^S contains 91 distance values instead of 196 values in D_X^T. It is necessary to note that the 26-part pattern will rapidly increase the number of features, e.g., 1677 features are extracted from the 26-part pattern instead of 379 values extracted from the 14-part pattern.

3.3. Codebook construction

Some topic models, such as LDA [4] and PAM [27] rely on the existence of a codebook, constructed by a number of visual words. For codebook construction, the authors therefore utilize k-means clustering algorithm based on the Euclidean distance metric to quantize extracted features. The authors separate two types of codeword corresponding to distance and angle. Concretely, each element $\widehat{d}$ in the Spatio–Temporal Distance category is clustered as a distance codeword, denoted d-word, while each element $\widehat{\varphi}$ in the Spatio–Temporal Angle category is considered as an angle codeword, denoted a-word. The parameter K in the clustering algorithm, the number of clusters and also the size of the codebook (the number of vocabulary words), is set in advance for distance and angle feature category. If the method is investigated on the spatial or temporal category class of distance and angle, a mixture codebook of d-word and a-word will be created with $2K$ of size. Meanwhile, a codebook of either d-word or a-word is generated if the distance or angle feature category is utilized (see Table 1). In the testing phase, a frame containing two-object interactions will be represented by a collection of d-word and a-word by mapping from the codebook.

3.4. Four-level Pachinko Allocation Model

In the previous section, the features describing the interaction between two human objects in the spatio-temporal relation are computed and mapped to codewords. Fundamentally, they can be used for interactive action classification of a short period of time, however, the long time representation needs to be explored. Another issue is the high possibility of different activities comprising more similarly interactive features. This unexpected event might lead to the wrongly recognized label, especially with the complex activities, for example as Punching and Pushing. Therefore, in this section, the authors proposed a hierarchical model based on the Pachinko Allocation Model (PAM) to capture the correlation between the relational features, interaction poselets, and interactions. In order to represent and learn arbitrary, nested, and possibly sparse activity correlations, this model is constructed based on the arbitrary Directed Acyclic Graphs (DAGs).

Although PAM is fundamentally introduced with arbitrary DAGs, four-level hierarchy structure, a special case discussed in [27], consists of one root topic, u super topics at the second level $\mathcal{P} = \{\rho_1, \rho_2, \ldots, \rho_u\}$, v subtopics at the third level $\mathcal{Q} = \{\varrho_1, \varrho_2, \ldots, \varrho_v\}$ and N codewords at the bottom. According to the joint distance and angle features, the codebook comprises d-words and a-words which are described in the previous section. From natural language processing to computer vision, the super topic and subtopic in topic models are corresponding to the interactive activities and the interactive poselets, respectively. The root is associated to interactive activities; the interactive activities are fully connected to interactive poselets; and the interactive poselets are fully linked to the codewords at the bottom of the hierarchical structure as shown in Fig. 3a. The multinomials of the root and activities are sampled for each frame based on a single Dirichlet distribution $g_r(\delta_r)$ and $g_l(\delta_l)|_{l=1}^{u}$, respectively. The poselets are modeled with multinomial distributions $\phi_{\varrho_l}\big|_{k=1}^{v}$ and $\psi_{\varrho_l}\big|_{k=1}^{v}$ which are sampled from Dirichlet distribution $g(\beta)$ and $g(\gamma)$, which will be used for sampling the d-words and a-words in the PAM algorithm. The graphic model for four-level PAM is displayed in Fig. 3b. The particular notations used in PAM are summarized in the Table 2. According this model, a frame s in the sequence of T frames $\mathcal{S} = \{s_1, s_2, \ldots, s_T\}$, is generated by the following process:

1. Write a multinomial distribution $\theta_r^{(s)}$ from a Dirichlet prior $\delta_r^{(s)}$ for frame s.
2. For each interactive activity ρ_l, write a multinomial distribution $\theta_{\rho_l}^{(s)}$ over interactive poselets from a Dirichlet distribution $g_l(\delta_l)$, where δ_l is an appropriate Dirichlet prior.
3. Write multinomial distributions $\phi_{\varrho_k}\big|_{k=1}^{v}$ from a Dirichlet prior β for each interactive poselet ϱ_k.
4. Write multinomial distributions $\psi_{\varrho_k}\big|_{k=1}^{v}$ from a Dirichlet prior γ for each interactive poselet ϱ_k.
5. For each codeword w in the current frame s:
   - Write an interactive activity $\rho_{w,s}$ from $\theta_r^{(s)}$.
   - Write an interactive poselet $\varrho_{w,s}$ from $\theta_{\rho_{w,s}}^{(s)}$.
   - Write a codeword w from the multinomial $\phi_{\varrho_{w,s}}$ if w is a d-word and from the multinomial $\psi_{\varrho_{w,s}}$ if w is an a-word.

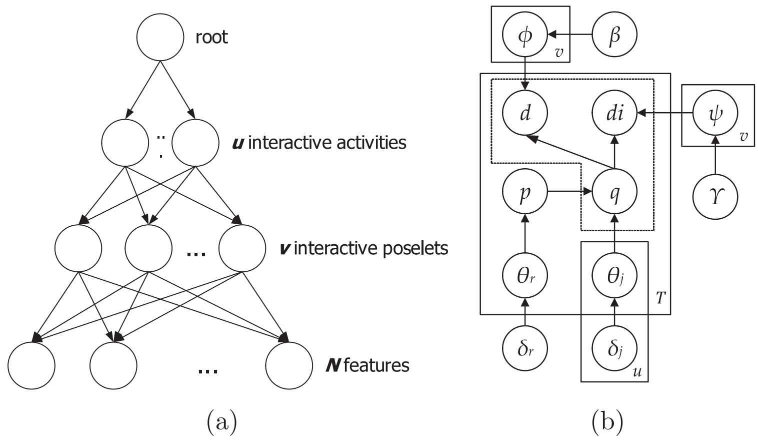


Fig. 3. Pachinko Allocation Model: (a) Hierarchical topic model, (b) Graphic model.

Table 2
Notations used in the PAM model.

Symbol	Description
u	Number of interactive activities
v	Number of interactive poselets
T	Number of frames
N	Number of unique codewords
$g_r(\delta_r)$	Dirichlet distribution associated with the root
$g_l(\delta_l)$	Dirichlet distribution associated with the lth activity
$g(\beta)$	Dirichlet distribution associated with poselet for distance features
$g(\gamma)$	Dirichlet distribution associated with poselet for motion featurres
$\theta_r^{(s)}$	The multinomial distribution sampled from $g_r(\delta_r)$ for the root in frame s
$\theta_{p_l}^{(s)}$	The multinomial distribution sampled from $g_l(\delta_l)$ for an activity in frame s
ϕ_ϱ	The multinomial distribution sampled from $g(\beta)$ for a poselet ϱ
ψ_ϱ	The multinomial distribution sampled from $g(\gamma)$ for a poselet ϱ
$\rho_{w,s}$	The interactive activity ρ associated with the codeword w in the frame s
$\varrho_{w,s}$	The interactive poselet ϱ associated with the codeword w in the frame s

Following the above process, the joint probability of generating the frame s, the interactive activity assignments $\rho^{(s)}$, the interactive posetlet assignments $\varrho^{(s)}$, and the multinomial distribution $\theta^{(s)}$ is calculated as follows:

$$P\big(s, \varrho(s), \rho(s), \theta^{(s)} \big| \delta, \beta, \gamma\big) = P(\theta_r|\delta_r)\prod_{l=1}^{u} P\big(\theta_{\rho_l}(s)|\delta_l\big) \times \prod_{w}\Big[P\big(\rho_w|\theta_r^{(s)}\big)P\big(\varrho_w|\theta_{\varrho_w}^{(s)}\big)P\big(w|\phi_{\varrho_w}, \psi_{\varrho_w}\big)\Big] \tag{12}$$

where $P\big(w\big|\phi_{\varrho_w}, \psi_{\varrho_w}\big) = P\big(w\big|\phi_{\varrho_w}\big)P\big(w\big|\psi_{\varrho_w}\big)$. Integrating out $\theta^{(s)}$ and summing over $\rho^{(s)}$ and $\varrho^{(s)}$, the marginal probability of a frame can be calculated as:

$$P(s|\delta, \beta, \gamma) = \int P\big(\theta_r^{(s)}\big|\delta_r\big)\prod_{l=1}^{u} P\big(\theta_{\rho_l}^{(s)}\big|\delta_l\big) \times \prod_{w}\sum_{\rho_w, \varrho_w}\Big[P\big(\rho_w|\theta_r^{(s)}\big)P\big(\varrho_w|\theta_{\rho_w}(s)\big)P\big(w|\phi_{\varrho_w}, \psi_{\varrho_w}\big)\Big]d\theta^{(s)} \tag{13}$$

The probability of generating the corpus $\mathcal{S}$ corresponding to the overall video is computed by:

$$P(\mathcal{S}|\delta, \beta, \gamma) = \int \prod_{k=1}^{v}\Big(P\big(\phi_{\varrho_k}\big|\beta\big) + P\big(\psi_{\varrho_k}\big|\gamma\big)\Big)\prod_{s} P(s|\delta, \beta, \gamma)d\phi d\psi \tag{14}$$

The joint distribution of the corpus $\mathcal{S}$ and the topic assignments is given by:

$$P(\mathcal{S}, \mathcal{P}, \mathcal{Q}|\delta, \beta, \gamma) = P(\mathcal{P}|\delta)P(\mathcal{Q}|\mathcal{P}, \delta)P(\mathcal{S}|\mathcal{Q}, \beta)P(\mathcal{S}|\mathcal{Q}, \gamma) \tag{15}$$

By integrating out the sampled multinomials, each term is calculated as follows:

$$P(\mathcal{P}|\delta) = \int \prod_{s} P\left(\theta_r^{(s)}|\delta_r\right) \prod_{w} P\left(\rho_w \middle| \theta_r^{(s)}\right) d\theta$$

$$P(\mathcal{Q}|\mathcal{P},\delta) = \int \prod_{s} \left(\prod_{l=1}^{u} P\left(\theta_{\rho_l}^{(s)}|\delta_l\right) \prod_{w} P\left(\varrho_w \middle| \theta_{\rho_w}^{(s)}\right) \right) d\theta$$

$$P(\mathcal{S}|\mathcal{Q},\beta) = \int \prod_{k=1}^{v} P\left(\phi_{\varrho_k}|\beta\right) \prod_{s} \left(\prod_{w} P\left(w \middle| \phi_{\varrho_w}\right) \right) d\phi$$

$$P(\mathcal{S}|\mathcal{Q},\gamma) = \int \prod_{k=1}^{v} P\left(\psi_{\varrho_k}|\gamma\right) \prod_{s} \left(\prod_{w} P\left(w \middle| \psi_{\varrho_w}\right) \right) d\psi \tag{16}$$

Finally, the approximate inference result of the condition distribution which samples the super topic and subtopic assignments for each codeword, is obtained as follows:

$$\begin{aligned} P(\rho_w, \varrho_w|\mathcal{D}, \mathcal{P}_{-w}, \mathcal{Q}_{-w}, \delta, \beta, \gamma) &\propto P(w, \rho_w, \varrho_w|\mathcal{D}_{-w}, \mathcal{P}_{-w}, \mathcal{Q}_{-w}, \delta, \beta, \gamma) \\ &= \frac{P(\mathcal{D}, \mathcal{P}, \mathcal{Q}|\delta, \beta, \gamma)}{P(\mathcal{D}, \mathcal{P}_{-w}, \mathcal{Q}_{-w}|\delta, \beta, \gamma)} \\ &= \frac{n_l^{(s)} + \delta_{rl}}{n_r^{(s)} + \sum_{l=1}^{u} \delta_{rl}} \frac{n_{lk}^{(s)} + \delta_{lk}}{n_l^{(s)} + \sum_{k=1}^{v} \delta_{lk}} \frac{n_{kz} + \beta_z}{n_k + \sum_{z=1}^{K} \beta_z} \frac{n_{kz} + \gamma_z}{n_k + \sum_{z=1}^{K} \gamma_z} \end{aligned} \tag{17}$$

where $n_r^{(s)}$ is the number of occurrences of the root r in the frame s; $n_l^{(s)}$ is the number of occurrences of activity ρ_l in the frame s; $n_k^{(s)}$ is the number of occurrences of poselet ϱ_k in s; $n_{lk}^{(s)}$ is the number of times that poselet ϱ_k is sampled from the activity ρ_l; $n_{kz}^{(s)}$ is the number of occurrences of codeword w_z in poselet ϱ_k. The notation $-w$ indicates the activity assignments except the codeword w. The hyper-parameters δ, β, and γ can be estimated via the Gibbs sampling algorithm which is described in [27]. By tagging the joint distance and joint angle features as codewords, the new data is generated as the output of PAM. The probability distribution is obtained as the implicit poselet – activity – frame sequence matrix from merging the same codewords in different video contents

3.5. Classification

The joint distance and joint angle features are viewed as codewords and assigned to particular interactive poselet and activity models by a four-level Pachinko Allocation Model. The interactive poselet and activity statistics in every frame sequence are gathered by PAM, then their frequency is observed. Hence, every sequence is represented by a matrix whose length is the number of interactive poselets and activities. The interaction recognition is performed on these matrices corresponding to interaction videos. To solve the N-class pattern recognition problem, the authors utilize the Binary Tree of SVM [15], or BTS for abbreviation, in which each node in the tree produces a binary decision using the original SVM. Based on the recursively dividing the classes into two disjoint groups in every node of the decision tree, the group of unknown sample will be identified by the SVM classifier. In the training phase, BTS has $N-1$ binary classifiers (N is the number of classes) while it requires only $\log_{4/3}\left(\frac{N+3}{4}\right)$ binary tests on average to make a decision.

4. Experiments and discussions

4.1. Dataset and experiment setup

In this paper, three experiments are performed on two well-known interaction datasets, BIT-Interaction dataset [24] and UT-Interaction dataset [39]. All of the experiments were performed on a desktop PC running Windows 7 Operating System with a 2.67-GHz Intel Core i5 CPU and 4GB of RAM. Matlab 2013a was used to make the simulations. The proposed method was evaluated using the 10-fold cross-validation for all experiments.

BIT-Interaction data set has eight classes of human interactive activity (bowing, boxing, hand shaking, high-five, hugging, kicking, patting, and pushing) as shown in Fig. 4a–h, with 50 short videos ($\sim$ 2 s) per class. Each video is recorded with a resolution of 320 × 240 and a rate of 30 fps (frame per second). Videos are captured in realistic scenes, included indoor and outdoor environments, with partial occluded body components, dynamic object movements, and different viewpoints in various illumination conditions.

UT-Interaction data set consists of six interactions (hand shaking, hugging, kicking, pointing, punching and pushing) as shown in Fig. 4i–n. Each interaction is presented by 10 videos whose lengths are around 1 min. Totally, there are 60 videos for six classes provided in the data set. Those videos are captured with a resolution of 720 × 480 and a frame rate of 30 fps with slightly different zoom rate and camera jitter.

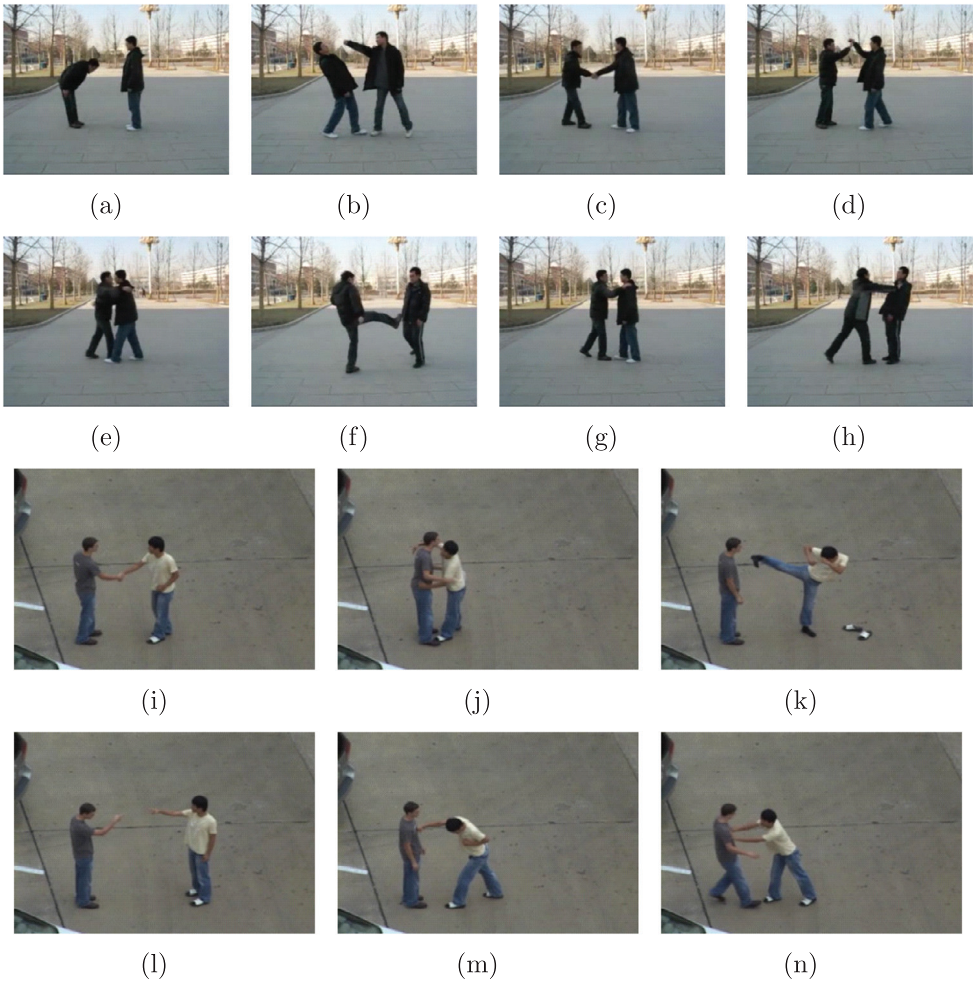


Fig. 4. Eight interactive activities in the BIT-Interaction data set: (a) Bowing, (b) Boxing, (c) Hand shaking, (d) High-five, (e) Hugging, (f) Kicking, (g) Patting, (h) Pushing. Six interactive activities in the UT-Interaction data set: (i) Hand shaking, (j) Hugging, (k) Kicking, (l) Pointing, (m) Punching, (n) Pushing.

In two data sets, the individual human objects in each frame were extracted by their bounding boxes supported in dataset owners for pose estimation. This strategy aim is to improve the estimation accuracy and computational speed because of searching body parts in a segmented area instead of a whole image. Moreover, the authors consider both the 14-part and 26-part patterns using the INRIA Person dataset [11] and the Image Parse data set [35] for pose learning. The parameter configuration for pose estimation algorithm [57] is set up with $\alpha = 0.2$, h and w as the height and width of the bounding box. A codebook with 1000 of size, consisted of 500 d-words and 500 a-words, is constructed using the k-mean clustering algorithm. In the four-level PAM model, the numbers of interactive activities u are defined to 8 and 6 for the BIT-Interaction data set and the UT-Interaction data set, respectively, and the number of interactive poselets v was set to 150 for both of them. The Dirichlet distribution over activities and poselets is produced with parameter 0.01. The Gibbs sampling process is performed with 1000 burn-in iterations and then 20 samples are drawn in the following 250 iterations. For BTS classifier, the authors utilize LibSVM [6] with RBF kernel to solve the multi-class classification problem.

Three experiments are explained in detail as follows:

- In the first experiment, the recognition method is validated for each data set on two articulation estimation patterns. This experiment investigates the influence of pose estimation performance on activity recognition accuracy.
- In the second experiment, the authors validate the feature types, distance and angle feature sets in the spatial and temporal dimension, using the 26-part pattern for pose estimation. This benchmark proves that recognition accuracy also depends on relation features.
- Finally, the proposed PAM-based hierarchical topic model was compared with LDA, a standard topic model, and the state-of-the-art interaction recognition methods on the same testing datasets.

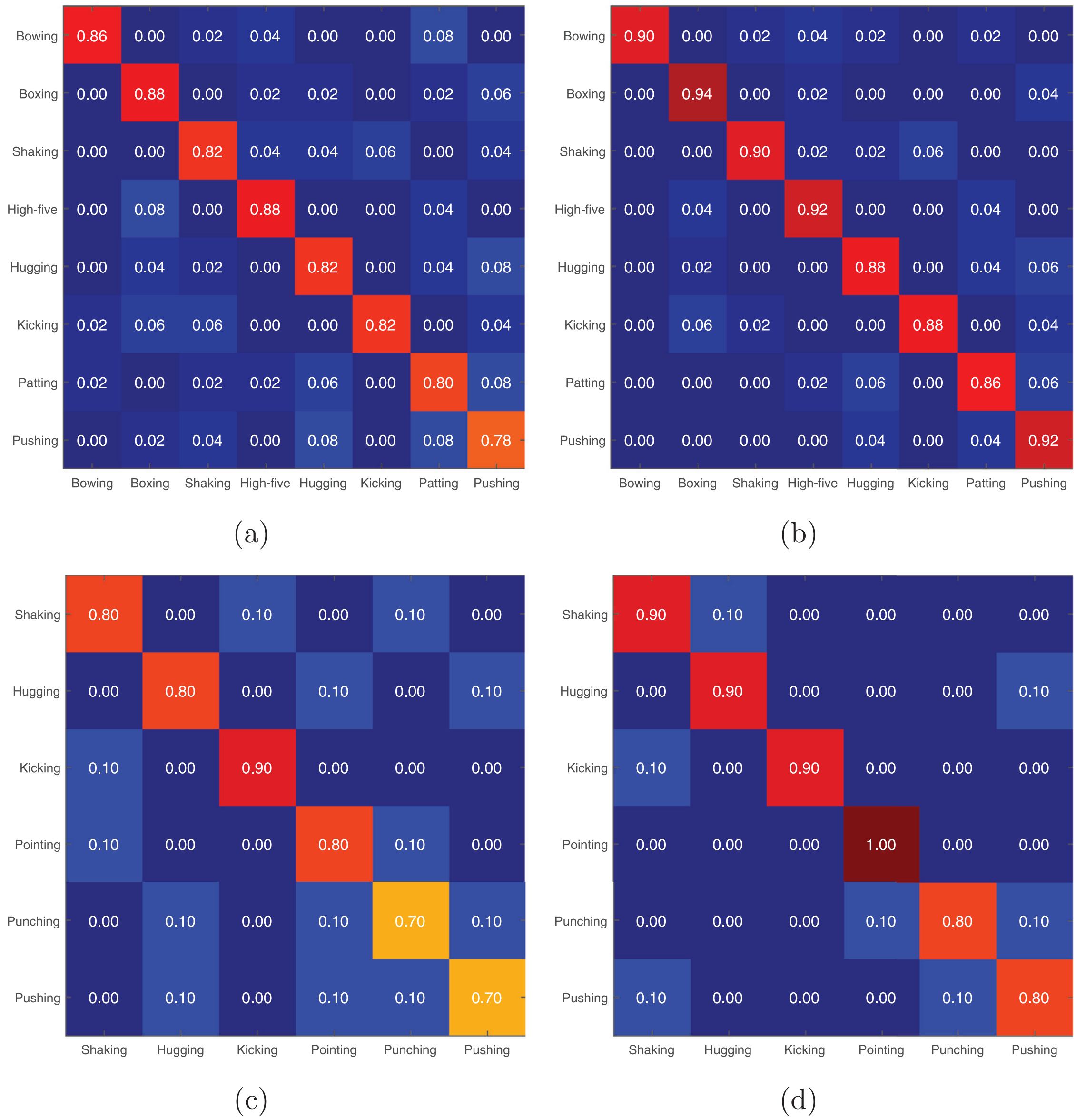


Fig. 5. Confusion matrices of the SVM classifier using spatio-temporal distance feature set on the BIT-Interaction data set: (a) 14-part pattern, (b) 26-part pattern; and on UT-Interaction data set: (c) 14-part pattern, (d) 26-part pattern.

4.2. Experiment results and discussions

The confusion matrices for the first experiment on two interaction datasets are shown in Fig. 5. The feature category used for the estimation pattern test is the spatio-temporal distance set (see Table 1). For the BIT-Interaction dataset (Fig. 5a–b), the 26-part pattern recognizes more accurate than 14-part pattern in most of activity classes. More valuable features which are extracted from the 26-part pattern are useful for the complex activity understanding. An improvement of classification accuracy is also obtained in the case of the UT-Interaction data set (Fig. 5c–d) where the 26-part pattern outperformed the 14-part pattern on five activities among six in total. However, more estimated parts (midway points between limbs, such as mid-upper arm, mid-lower arm, etc) significantly increase the computational cost of estimation and feature extraction processes, for instance with the spatial distance category, $(325 + 2 \times 676)$ features calculated from the 26-part pattern instead of $(91 + 2 \times 144)$ features calculated from the 14-part pattern.

In the second experiment, the authors investigate the proposed method on different feature categories using the 26-part pattern. The classification results on the BIT-Interaction data set are reported by the confusion matrices in Fig. 6 and summarized in Table 3. Totally, there are eight examined categories: the spatial distance set, temporal distance set, spatial

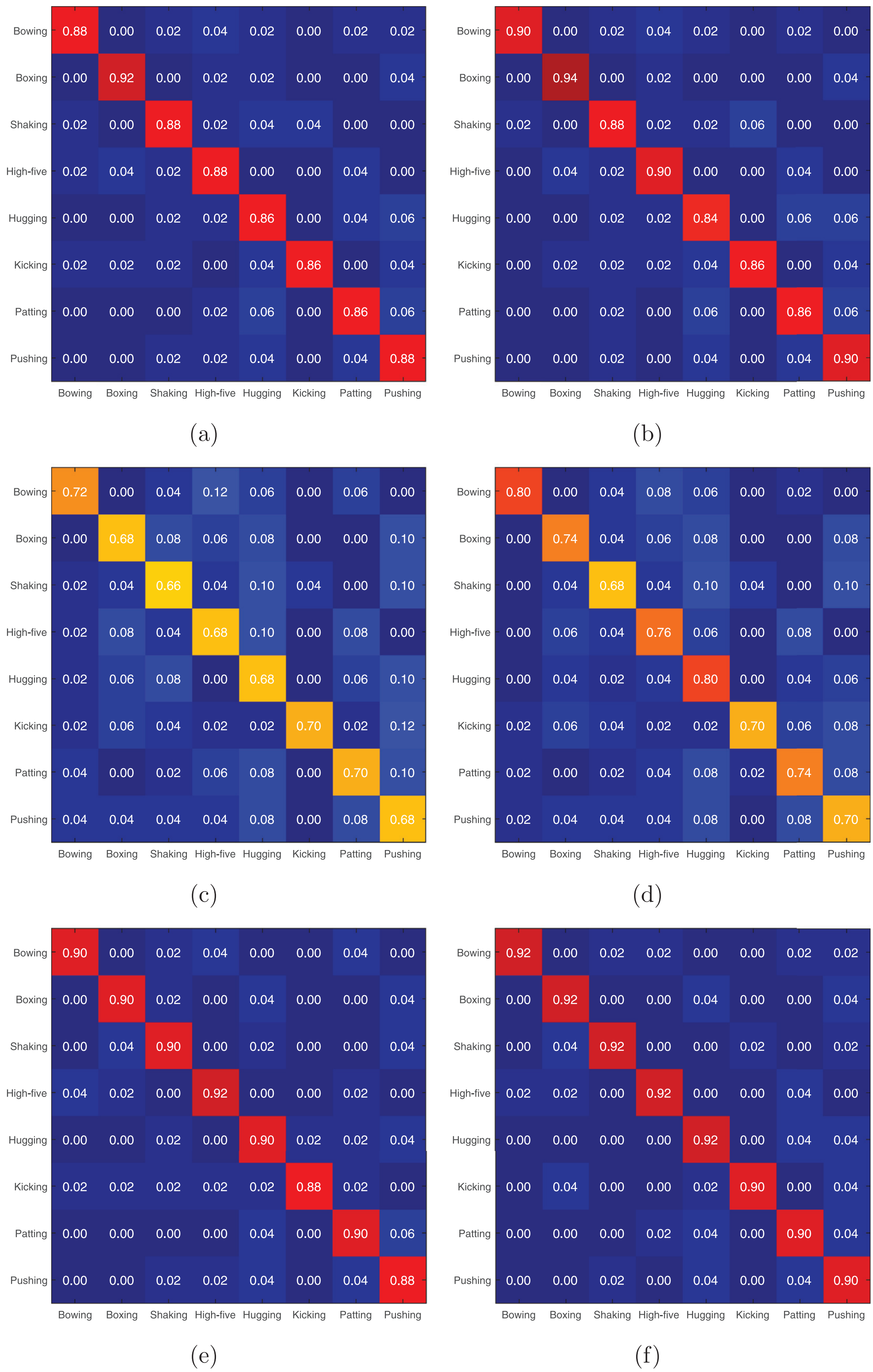


Fig. 6. Confusion matrices of the SVM classifier on the BIT-Interaction data set using 26-part pattern: (a) spatial distance feature, (b) temporal distance feature, (c) spatial angle feature, (d) temporal angle feature, (e) spatial distance-angle feature, and (f) temporal distance-angle feature.

Table 3
Recognition accuracy (%) of the proposed method using different feature types.

BIT-Interaction data set									
Features	Bowing	Boxing	Shaking	High-five	Hugging	Kicking	Patting	Pushing	**Overall**
Spatial distance	88.0	92.0	88.0	88.0	86.0	86.0	86.0	88.0	**87.8**
Temporal distance	90.0	94.0	88.0	90.0	84.0	86.0	86.0	90.0	**88.5**
Spatial angle	72.0	68.0	66.0	68.0	68.0	70.0	70.0	68.0	**68.8**
Temporal angle	80.0	74.0	68.0	76.0	80.0	70.0	74.0	70.0	**74.0**
Spatio-temporal distance	90.0	94.0	90.0	92.0	88.0	88.0	86.0	92.0	**90.0**
Spatio-temporal angle	82.0	74.0	70.0	74.0	78.0	74.0	78.0	76.0	**75.8**
Spatial distance-angle	90.0	90.0	90.0	92.0	90.0	88.0	90.0	88.0	**89.8**
Temporal distance-angle	92.0	92.0	92.0	92.0	92.0	90.0	90.0	90.0	**91.2**

UT-interaction data set							
Features	Shaking	Hugging	Kicking	Pointing	Punching	Pushing	**Overall**
Spatial distance	80.0	80.0	90.0	90.0	70.0	80.0	**81.7**
Temporal distance	80.0	90.0	90.0	90.0	80.0	80.0	**85.0**
Spatial angle	60.0	60.0	70.0	70.0	60.0	60.0	**63.3**
Temporal angle	70.0	60.0	70.0	70.0	70.0	70.0	**68.3**
Spatio-temporal distance	90.0	90.0	90.0	100.0	80.0	80.0	**88.3**
Spatio-temporal angle	70.0	80.0	70.0	80.0	80.0	70.0	**75.0**
Spatial distance-angle	90.0	90.0	90.0	90.0	80.0	90.0	**88.3**
Temporal distance-angle	90.0	90.0	90.0	100.0	90.0	90.0	**91.7**

angle set, temporal angle set, spatio-temporal distance set, spatio-temporal angle set, spatial distance-angle set, and temporal distance-angle set, that were collected from the intra and inter-object distance and angle features in the spatial and temporal dimension. In Fig. 6a–b representing results on the distance feature sets, the proposed method achieves a greater accuracy with the temporal set over the spatial set in most of activities. Compared with the spatial distance set, the temporal distance set contains more information of pose translation. This strategy is repeated on the angle feature sets (Fig. 6c–d) and the merging feature sets of distance and angle (Fig. 6e–f). The above results indicate that the temporal feature sets hold more relational body-part information about object movements. Furthermore, compared with distance feature, the angle information between joint pairs is less useful (68.8% versus 87.8% for spatial feature sets and 74.0% versus 88.8% for temporal feature sets in overall accuracy) because angle feature is quite fragile to noise from the estimation process. When merging distance and angle features following spatial and temporal dimension, the performance in accuracy is differentially improved (see Table 3). However, it is important to note that highly expensive computation is required for merged feature sets. From Table 3, the spatio-temporal distance feature, the spatial distance-angle feature, and the temporal distance-angle feature sets provide the highest overall accuracy results among eight feature categories. According confusion matrices, Hugging and Patting are the most confused together. Hugging and Patting are mostly confused with Pushing (4–6%), and on the contrary. Moreover, Pushing is mostly misunderstood with Hugging and Patting. These activities get some challenges in the pose tracking and locating due to the body-part overlapping. The confusion matrices and the overall accuracy results for the UT-Interaction data set are shown in Fig. 7 and Table 3, respectively. The angle feature is not compatible for this data set with low accuracy (less than 64% and 69% for spatial and temporal angle set, respectively), even if the temporal angle feature category is used. Combining distance and angle features to merging sets in the spatial and temporal dimension sometimes does not bring accuracy improvement at all, for instance, the recognition results of Kicking and Pointing using spatial distance feature in Fig. 7a are equal to results of spatial distance-angle feature in Fig. 7e. In the best performance case using the temporal distance-angle feature set, Pointing is recognized correctly with 100% in accuracy. Punching and Pushing are confused each other due to some resemblances of interactive poselets in the beginning and ending period of activities.

In the last experiment, the proposed method is validated using the merged feature set (see Table 1) that contained information of joint distance and angle features extracted in the spatio-temporal dimension. Two confusion matrices of the SVM classifier corresponding to two interaction data sets are presented in Fig. 8. Compared with feature sets in the second experiment, the proposed method provides higher recognition rates with merged feature sets on the BIT-Interaction dataset (Bowing, Boxing, Hugging, Kicking, Patting, and Pushing) and the UT-Interaction data set (Kicking and Pointing). Nevertheless, the confusions are still occurred with activities involving occlusions, such as Hugging, Patting, Punching, and Pushing. In this experiment, the authors further compare the proposed PAM-based method with the LDA-based approach. Although LDA is constructed on the DAGs structure with Dirichlet distribution, it is only capable to capture the correlation among the features (as codewords) to support directly to the high level information (as activities) without intermediate knowledge from interactive poselets. According to the benefits from capturing correlations among relational features, as well as among interactive poselets and activities, PAM outperforms LDA, greater than 10% and 13% of overall accuracy on two test data sets. Moreover, the authors do an accuracy competition between the proposed method with existing interaction recognition methods, concretely, Lan et al. [26], Ryoo et al. [39], Yu et al. [59], Ryoo et al. [38], Kong et al. [23], and Kong et al. [24] on the same data sets. The recognition accuracy results are presented in Table 4 for the BIT-Interaction data set and the

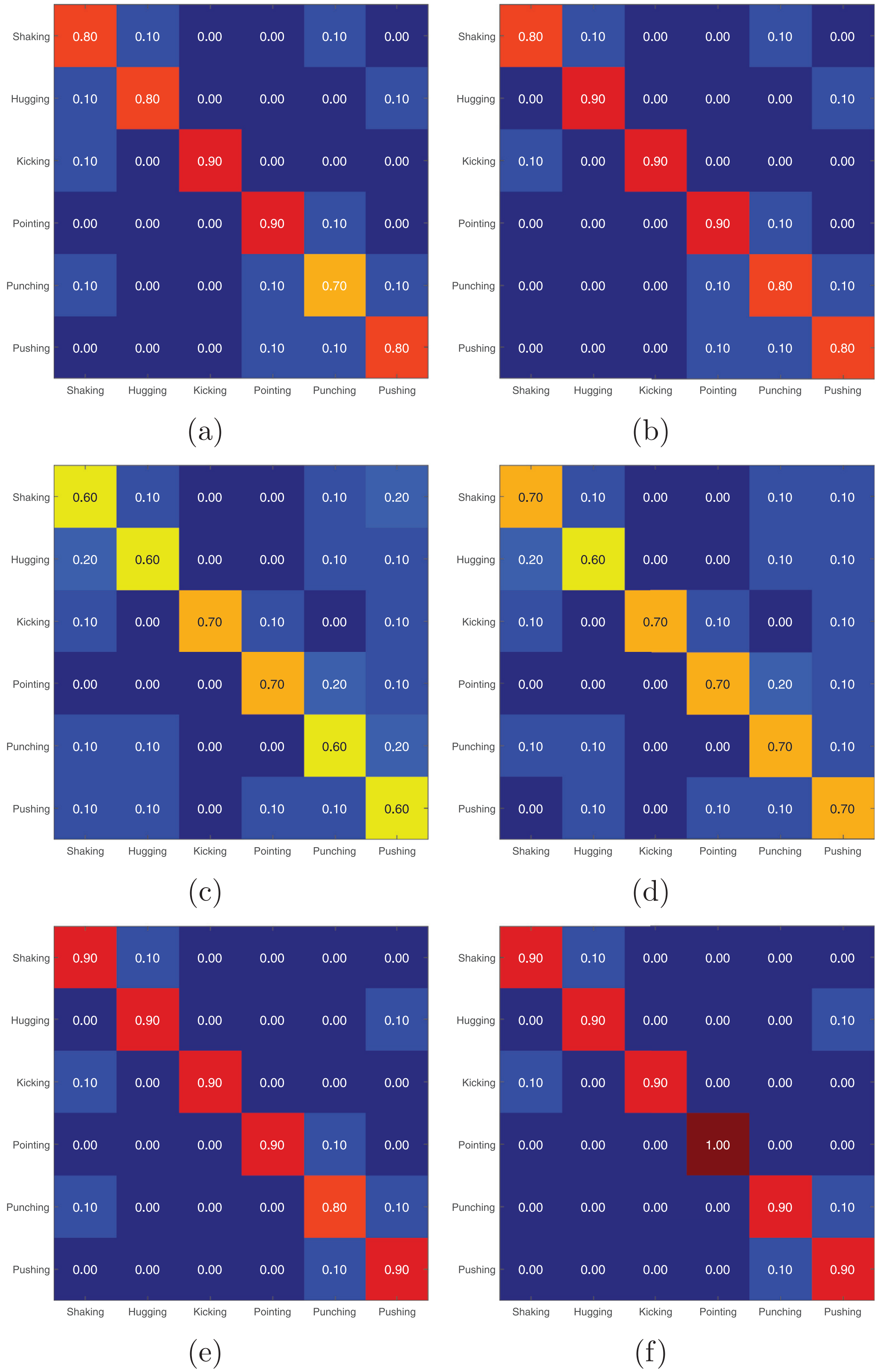


Fig. 7. Confusion matrices of the SVM classifier on the UT-Interaction data set using 26-part pattern: (a) spatial distance feature, (b) temporal distance feature, (c) spatial angle feature, (d) temporal angle feature, (e) spatial distance-angle feature, and (f) temporal distance-angle feature.

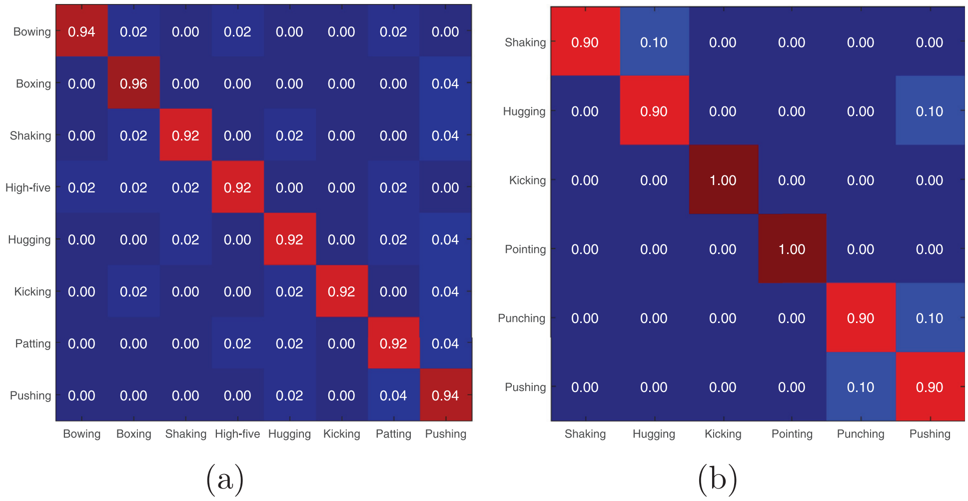


Fig. 8. Confusion matrices of the SVM classifier using 26-part pattern with merged feature category: (a) BIT-Interaction data set, (b) UT-Interaction data set.

Table 4
Comparing recognition accuracy (%) of the proposed method with existing methods.

BIT-interaction data set									
Features	Bowing	Boxing	Shaking	High-five	Hugging	Kicking	Patting	Pushing	Overall
LDA [4]	84.0	88.0	84.0	88.0	84.0	82.0	76.0	78.0	**83.0**
Lan et al. [26]	82.0	76.0	80.0	88.0	88.0	82.0	82.0	80.0	**82.3**
Yu et al. [59]	86.0	84.0	80.0	84.0	82.0	86.0	84.0	80.0	**83.3**
Ryoo et al. [38]	88.0	88.0	80.0	88.0	84.0	88.0	80.0	76.0	**84.0**
Kong et al. [23]	82.0	80.0	82.0	94.0	94.0	80.0	82.0	88.0	**85.3**
Kong et al. [24]	94.0	88.0	94.0	94.0	94.0	88.0	88.0	88.0	**91.0**
Proposed method	94.0	96.0	92.0	92.0	92.0	92.0	92.0	94.0	**93.0**
UT-interaction data set									
Features		Shaking	Hugging	Kicking	Pointing	Punching	Pushing	**Overall**	
LDA [4]		80.0	70.0	90.0	90.0	80.0	70.0	**80.0**	
Lan et al. [26]		80.0	80.0	100.0	80.0	70.0	70.0	**80.0**	
Yu et al. [59]		100.0	80.0	70.0	100.0	80.0	70.0	**83.3**	
Ryoo et al. [38]		80.0	90.0	90.0	90.0	80.0	80.0	**85.0**	
Kong et al. [23]		80.0	80.0	100.0	90.0	90.0	90.0	**88.3**	
Kong et al. [24]		100.0	90.0	100.0	80.0	90.0	90.0	**91.7**	
Proposed method		90.0	90.0	100.0	100.0	90.0	90.0	**93.3**	

UT-Interaction data set. According to the experimental outcomes, the proposed method outperforms the others in most of testing activities. The activity co-occurrence based method, proposed by Lan et al. [26], combines the adaptive structure and the HOG-based action context descriptor to model the person-person interaction. Due to spatial relation exploration, Lan's model is restricted to deeply understand more complex interactive activities which are generally required more temporal information. Ryoo et al. [38] introduce a novel methodology for activity prediction and recognition based on a dynamic bag-of-words. Although Ryoo's method is capable to fairly handle noise, it is inhibited by overlapping interactions, such as Patting and Pushing because of outliers from the spatio-temporal feature extractor [12]. Building on the work of Ryoo et al. [39], Yu et al. [59] propose Pyramid Spatio–Temporal Relationship Match (PSRM) to combine with Semantic Texton Forest (STFs) to upgrade recognition performance. Video-FAST descriptors provide good performance in processing speed, however, they are corruptible in practical environments containing more dynamic motions. This drawback explains for quite poor accuracy of Yu's method at Pushing, Shaking, and Hugging. Two approaches proposed by Kong et al. [23,24] significantly exceed previous works. According to high-level descriptors, called interactive phrases, Kong et al. formulate binary semantic relationships between interacting people [23]. Concretely, each interactive phrase, detected by an attribute model, is associated with only one attribute belonging to corresponding interactive person to describe motion relationships. Understanding co-occurrence relationships between pairs of interactive phrases therefore addresses motion ambiguity and partial occlusion. In [24], Kong et al. improve recognition performance by a data-driven attribute model and a new learning formulation. The extended version brought some improvements in classification accuracy when compared with original [23], for instance, 91.0% versus 85.3% for the BIT-Interaction data set and 91.7% versus 88.3% for UT-Interaction data set in overall accuracy. Kong's improvement approach recognizes more accurate than the proposed method at High-five, Shaking, and Hugging activities.

Table 5
Processing time result (Mins) of the proposed method.

Step	BIT-interaction (~28,000 frames)	UT-interaction (~10,800 frames)
Pose estimation	20	32
Feature extraction and Codebook construction	58	35
Activity modeling and Classification	212	168
Total	~ 290	~ 235

Learning data-driven phrases using the information bottleneck technique [44] is able to extract discriminative phrases for differentiating interactive activities. In additions, semantic descriptors are able to handle the motion ambiguity and partial occlusion in the interactions while the proposed method is fairly depended on the articulation estimation outcomes. However, the limitation is that the method did not consider dependencies of phrases and attributes in the temporal dimension to lead to some misperception at Pushing, Patting, and Pointing. Different from existing approaches, the proposed method calculates joint distance and angle features from detected joint coordinates using an effective articulated-body estimation to describe intra and inter-person relation in spatio-temporal dimension. The PAM-based hierarchical topic model provides full and flexible correlations of feature-poselet-interaction to maximize explicitness between activities through interactive poselets. Each step in the method is processed separably, the processing time is therefore measured individually and then accumulated for the total. The processing time results are detail listed in Table 5. It is necessary to note that the 26-part pattern and the merged feature set are installed as the default setting for the timing experiment. The processing time for pose estimation depends on the number of frames and the frame resolution. The Gibbs sampling used in the 4-Level PAM for activity modeling is run with 1000 burn-in iterations and its time varies on the number of activities and number of interactive poselets.

5. Conclusion

We proposed a four-level topic model, developed from Pachinko Allocation Model, for the interactive activity recognition, in which the relationships between the relation features and the interactions are fully described through the interactive poselets. In our approach, the intra and inter-person joint features of distance and angle are calculated in the spatio-temporal dimension from the pose estimation outcome. The poselet layer is composed by two types of codeword, d-word and a-word corresponding to joint distance and angle feature, to differentiate the complex interactions. Compared with the 14-part pattern, the proposed method achieves the better accuracy with the 26-part pattern used for pose estimation. Among testing feature categories, the merged feature set is reported as the best results for two benchmark interaction datasets. Moreover, we compare our approach with the standard LDA model used for topic modeling and the state-of-the-art approaches to demonstrate remarkable efficiency in the challenge of interaction recognition.

Because PAM is originally built as a parameter model, the numbers of super topic and sub-topic have to be predefined in advance. The hierarchical Dirichlet process (HDP) [46] can be used as a nonparametric prior for learning the number of topic. Another problem is the high feature dimension that comes from the feature type and the body pattern used in the joint estimation Some advance feature selection algorithms and dimensional reduction techniques are capable to apply for this task without degradation of recognition accuracy.

References

[1] R. Alazrai, Y. Mowafi, C.G. Lee, Anatomical-plane-based representation for humanhuman interactions analysis, Pattern Recognit. 48 (8) (2015) 2346–2363.
[2] M. Andriluka, S. Roth, B. Schiele, Pictorial structures revisited: People detection and articulated pose estimation, in: Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on, 2009, pp. 1014–1021.
[3] F.I. Bashir, A.A. Khokhar, D. Schonfeld, Object trajectory-based activity classification and recognition using hidden markov models, IEEE Trans. Image Process. 16 (7) (2007) 1912–1919.
[4] D.M. Blei, A.Y. Ng, M.I. Jordan, Latent dirichlet allocation, J. Mach. Learn. Res. 3 (2003) 993–1022.
[5] X. Cao, Z. Liu, Type-2 fuzzy topic models for human action recognition, Fuzzy Syst. IEEE Trans. PP (99) (2014). 1–1
[6] C.-C. Chang, C.-J. Lin, Libsvm: A library for support vector machines, ACM Trans. Intell. Syst. Technol. 2 (3) (2011) 27:1–27:27.
[7] Y. Chen, Z. Li, X. Guo, Y. Zhao, A. Cai, A spatio-temporal interest point detector based on vorticity for action recognition, in: Multimedia and Expo Workshops (ICMEW), 2013 IEEE International Conference on, 2013, pp. 1–6.
[8] Z. Cheng, L. Qin, Q. Huang, S. Yan, Q. Tian, Recognizing human group action by layered model with multiple cues, Neurocomputing 136 (2014) 124–135.
[9] S. Cho, H. Byun, A space-time graph optimization approach based on maximum cliques for action detection, IEEE Trans. Circuits Syst. Video Technol. 26 (4) (2016) 661–672.
[10] W. Choi, S. Savarese, Understanding collective activitiesof people from videos, Pattern Anal. Mach. Intell. IEEE Trans. 36 (6) (2014) 1242–1257.
[11] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), vol. 1, 2005, pp. 886–893.
[12] P. Dollar, V. Rabaud, G. Cottrell, S. Belongie, Behavior recognition via sparse spatio-temporal features, in: Visual Surveillance and Performance Evaluation of Tracking and Surveillance, 2005. 2nd Joint IEEE International Workshop on, 2005, pp. 65–72.
[13] M. Eichner, V. Ferrari, Better appearance models for pictorial structures, in: British Machine Vision Conference, 2009.
[14] A. Eweiwi, M.S. Cheema, C. Bauckhage, Action recognition in still images by learning spatial interest regions from videos, Pattern Recognit. Lett. 51 (2015) 8–15.

[15] B. Fei, J. Liu, Binary tree of svm: a new fast multiclass training and classification algorithm, Neural Netw. IEEE Trans. 17 (3) (2006) 696–704.
[16] P.F. Felzenszwalb, D.P. Huttenlocher, Pictorial structures for object recognition, Int. J. Comput. Vis. 61 (1) (2005) 55–79.
[17] M.A. Fischler, R. Elschlager, The representation and matching of pictorial structures, Comput. IEEE Trans. C-22 (1) (1973) 67–92.
[18] U. Gaur, Y. Zhu, B. Song, A. Roy-Chowdhury, A string of feature graphs model for recognition of complex activities in natural videos, in: Computer Vision (ICCV), 2011 IEEE International Conference on, 2011, pp. 2595–2602.
[19] I. Gori, J.K. Aggarwal, L. Matthies, M.S. Ryoo, Multitype activity recognition in robot-centric scenarios, IEEE Robot. Autom. Lett. 1 (1) (2016) 593–600.
[20] S. Jones, L. Shao, Content-based retrieval of human actions from realistic video databases, Inf. Sci. 236 (2013) 56–65.
[21] Y. Kong, Y. Fu, Close human interaction recognition using patch-aware models, IEEE Trans. Image Process. 25 (1) (2016) 167–178.
[22] Y. Kong, Y. Jia, A hierarchical model for human interaction recognition, in: Multimedia and Expo (ICME), 2012 IEEE International Conference on, 2012, pp. 1–6.
[23] Y. Kong, Y. Jia, Y. Fu, Learning human interaction by interactive phrases, in: Computer Vision ÈóÃÇÃô ECCV 2012, in: Lecture Notes in Computer Science, volume7572, 2012, pp. 300–313.
[24] Y. Kong, Y. Jia, Y. Fu, Interactive phrases: Semantic descriptionsfor human interaction recognition, Pattern Anal. Mach. Intel. IEEE Trans. 36 (9) (2014) 1775–1788.
[25] Y. Kong, W. Liang, Z. Dong, Y. Jia, Recognising human interaction from videos by a discriminative model, Comput. Vis. IET 8 (4) (2014) 277–286.
[26] T. Lan, Y. Wang, W. Yang, S. Robinovitch, G. Mori, Discriminative latent models for recognizing contextual group activities, Pattern Anal. Mach. Intel. IEEE Trans. 34 (8) (2012) 1549–1562.
[27] W. Li, A. Mccallum, Pachinko allocation: Scalable mixture models of topic correlations, of Machine Learning Research. Submitted, 2008.
[28] A. Liu, Human action recognition with structured discriminative random fields, Electron. Lett. 47 (11) (2011) 651–653.
[29] L. Liu, L. Shao, X. Zhen, X. Li, Learning discriminative key poses for action recognition, Cybern. IEEE Trans. 43 (6) (2013) 1860–1870.
[30] L. Liu, L. Shao, F. Zheng, X. Li, Realistic action recognition via sparsely-constructed gaussian processes, Pattern Recognit. 47 (12) (2014) 3819–3827.
[31] F. Lv, R. Nevatia, Single view human action recognition using key pose matching and viterbi path searching, in: Computer Vision and Pattern Recognition, 2007. CVPR '07. IEEE Conference on, 2007, pp. 1–8.
[32] L. Meng, L. Qing, P. Yang, J. Miao, X. Chen, D. Metaxas, Activity recognition based on semantic spatial relation, in: Pattern Recognition (ICPR), 2012 21st International Conference on, 2012, pp. 609–612.
[33] R. Messing, C. Pal, H. Kautz, Activity recognition using the velocity histories of tracked keypoints, in: Computer Vision, 2009 IEEE 12th International Conference on, 2009, pp. 104–111.
[34] B.T. Morris, M.M. Trivedi, Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach, IEEE Trans. Pattern Anal. Mach. Intel. 33 (11) (2011) 2287–2301.
[35] D. Ramanan, Learning to parse images of articulated bodies, in: Proceedings of Advances in Neural Information Processing System, 2007.
[36] M. Ramanathan, W.-Y. Yau, E.K. Teoh, Human action recognition with video data: Research and evaluation challenges, Human Mach. Syst. IEEE Trans. 44 (5) (2014) 650–663.
[37] M. Rodriguez, C. Orrite, C. Medrano, D. Makris, A time flexible kernel framework for video-based activity recognition, Image Vis. Comput. 48–49 (2016) 26–36.
[38] M. Ryoo, Human activity prediction: Early recognition of ongoing activities from streaming videos, in: Computer Vision (ICCV), 2011 IEEE International Conference on, 2011, pp. 1036–1043.
[39] M. Ryoo, J. Aggarwal, Spatio-temporal relationship match: Video structure comparison for recognition of complex human activities, in: Computer Vision, 2009 IEEE 12th International Conference on, 2009, pp. 1593–1600.
[40] S. Samanta, B. Chanda, Space-time facet model for human activity classification, Multimed. IEEE Trans. 16 (6) (2014) 1525–1535.
[41] B. Sapp, C. Jordan, B. Taskar, Adaptive pose priors for pictorial structures, in: Computer Vision and Pattern Recognition, 2010. CVPR 2010. IEEE Conference on, 2010, pp. 422–429.
[42] B. Sapp, A. Toshev, B. Taskar, Cascaded models for articulated pose estimation, in: Proceedings of the 11th European Conference on Computer Vision: Part II, in: ECCV'10, Springer-Verlag, Berlin, Heidelberg, 2010, pp. 406–420.
[43] P. Scovanner, S. Ali, M. Shah, A 3-dimensional sift descriptor and its application to action recognition, in: Proceedings of the 15th International Conference on Multimedia, in: MULTIMEDIA '07, 2007, pp. 357–360.
[44] N. Slonim, N. Tishby, Document clustering using word clusters via the information bottleneck method, in: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, in: SIGIR '00, ACM, 2000, pp. 208–215.
[45] L. Sun, H. Ai, S. Lao, Localizing activity groups in videos, Comput. Vis. Image Understand. 144 (2016) 144–154.
[46] Y.W. Teh, M.I. Jordan, M.J. Beal, D.M. Blei, Hierarchical dirichlet processes, J. Am. Stat. Assoc. 101 (2004).
[47] J. Tian, L. Li, W. Liu, A robust framework for 2D human pose tracking with spatial and temporal constraints, in: Digital Image Computing: Techniques and Applications (DICTA), 2014 International Conference on, 2014, pp. 1–8.
[48] H. Wang, C. Yuan, W. Hu, H. Ling, W. Yang, C. Sun, Action recognition using nonnegative action component representation and sparse basis selection, Image Process. IEEE Trans. 23 (2) (2014) 570–581.
[49] L. Wang, L. Cheng, T.H. Thi, J. Zhang, Human action recognition from boosted pose estimation, in: Digital Image Computing: Techniques and Applications (DICTA), 2010 International Conference on, 2010, pp. 308–313.
[50] Y. Wang, G. Mori, Human action recognition by semilatent topic models, Pattern Anal. Mach. Intel. IEEE Trans. 31 (10) (2009) 1762–1774.
[51] E.-J. Weng, L.-C. Fu, On-line human action recognition by combining joint tracking and key pose recognition, in: Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on, 2012, pp. 4112–4117.
[52] X. Wu, D. Xu, L. Duan, J. Luo, Y. Jia, Action recognition using multilevel features and latent structural svm, Circuits Syst. Video Technol. IEEE Trans. 23 (8) (2013) 1422–1431.
[53] Q. Xiao, J. Cheng, Human action recognition using topic model, in: Information Science and Technology (ICIST), 2014 4th IEEE International Conference on, 2014, pp. 694–697.
[54] J. xin Cai, X. Tang, G. Feng, Learning pose dictionary for human action recognition, in: Pattern Recognition (ICPR), 2014 22nd International Conference on, 2014, pp. 381–386.
[55] S. Yang, C. Yuan, W. Hu, X. Ding, A hierarchical model based on latent dirichlet allocation for action recognition, in: Pattern Recognition (ICPR), 2014 22nd International Conference on, 2014, pp. 2613–2618.
[56] W. Yang, Y. Wang, G. Mori, Recognizing human actions from still images with latent poses, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010, pp. 2030–2037.
[57] Y. Yang, D. Ramanan, Articulated human detection with flexible mixtures of parts, Pattern Anal. Mach. Intel. IEEE Trans. 35 (12) (2013) 2878–2890.
[58] B. Yao, L. Fei-Fei, Modeling mutual context of object and human pose in human-object interaction activities, in: Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on, 2010, pp. 17–24.
[59] T.-H. Yu, T.-K. Kim, R. Cipolla, Real-time action recognition by spatiotemporal semantic and structural forest, in: Proceedings of the British Machine Vision Conference, 2010, pp. 52.1–52.12.
[60] S. Zhang, H. Yao, X. Sun, K. Wang, J. Zhang, X. Lu, Y. Zhang, Action recognition based on overcomplete independent components analysis, Inf. Sci. 281 (2014) 635–647.
[61] Y. Zheng, Y.-J. Zhang, X. Li, B.-D. Liu, Action recognition in still images using a combination of human pose and context information, in: Image Processing (ICIP), 2012 19th IEEE International Conference on, 2012, pp. 785–788.
[62] M. Ziaeefard, R. Bergevin, Semantic human activity recognition: A literature review, Pattern Recognit. 48 (8) (2015) 2329–2345.

Section 1.4

Information Curation Layer (ICL) High level Context Awareness Paper

sensors

MDPI

Article

Ontology-Based High-Level Context Inference for Human Behavior Identification

Claudia Villalonga [1,2], Muhammad Asif Razzaq [1], Wajahat Ali Khan [1], Hector Pomares [2], Ignacio Rojas [2], Sungyoung Lee [1,*] and Oresti Banos [1,3]

1 Ubiquitous Computing Lab, Department of Computer Engineering, Kyung Hee University, 1 Seocheon-dong, Giheung-gu, Yongin-si, Gyeonggi-do 446-701, Korea; cvillalonga@oslab.khu.ac.kr (C.V.); asif.razzaq@oslab.khu.ac.kr (M.A.R.); wajahat.alikhan@oslab.khu.ac.kr (W.A.K.); oresti@oslab.khu.ac.kr (O.B.)

2 Department of Computer Architecture and Computer Technology, Research Center for Information and Communications Technologies—University of Granada (CITIC-UGR), C/Periodista Rafael Gomez Montero 2, Granada 18071, Spain; hector@ugr.es (H.P.); irojas@ugr.es (I.R.)

3 Telemedicine Group, Center for Telematics and Information Technology, University of Twente, 7500 AE Enschede, The Netherlands

* Correspondence: sylee@oslab.khu.ac.kr; Tel.: +82-31-201-2514; Fax: +82-31-202-2520

Academic Editors: Vladimir Villarreal and Carmelo R. García

Received: 30 April 2016; Accepted: 22 September 2016; Published: 29 September 2016

Abstract: Recent years have witnessed a huge progress in the automatic identification of individual primitives of human behavior, such as activities or locations. However, the complex nature of human behavior demands more abstract contextual information for its analysis. This work presents an ontology-based method that combines low-level primitives of behavior, namely activity, locations and emotions, unprecedented to date, to intelligently derive more meaningful high-level context information. The paper contributes with a new open ontology describing both low-level and high-level context information, as well as their relationships. Furthermore, a framework building on the developed ontology and reasoning models is presented and evaluated. The proposed method proves to be robust while identifying high-level contexts even in the event of erroneously-detected low-level contexts. Despite reasonable inference times being obtained for a relevant set of users and instances, additional work is required to scale to long-term scenarios with a large number of users.

Keywords: context recognition; context inference; ontologies; ontological reasoning; human behavior identification; activities; locations; emotions

1. Introduction

A revolutionary wave of smart systems has recently emerged to enable the automatic identification of human behavior. Proposed solutions are generally intended to recognize the primal dimensions of human behavior and context, including people's activities, emotions and locations. Video systems [1] and on-body sensors [2] have extensively been considered for the recognition of physical activity. Other works have used positioning technologies to track the user location and derive movement patterns [3]. Video, audio or a combination of both modalities have also been proposed for analyzing speech and facial expressions in order to recognize some emotional states [4,5]. Data-driven, knowledge-driven and hybrid methods are normally considered for processing these sensory data to identify human behavior. In data-driven approaches, machine learning techniques are used to detect patterns matching some known contexts describing behavior [6–8]. In knowledge-driven approaches, ontologies and rules are utilized to model and infer different contexts [9–11]. Both data-driven and knowledge-driven techniques are further combined in hybrid methods to determine various components of human behavior [12,13].

Despite the value of the contributions made for the automatic identification of human behavior, it is fair to say that most existing solutions tend to apply to a sole dimension of behavior. In other words, most systems are only capable of identifying activities, locations or emotions, but generally not a combination of them. While these primitives could be considered in isolation for a preliminary analysis of a person's behavior, their appropriate combination can lead to more meaningful and richer expressions of context for behavior understanding. Hence, there is a clear opportunity for developing new methods for the automatic identification of richer human behavior information.

This work presents an ontology-based method to intelligently combine cross-domain behavior primitives, also referred to as low-level contexts, in order to infer more abstract human context representations, hereafter high-level contexts. The proposed method particularly extends beyond the state-of-the-art while uniting emotion information as a novel behavioral component together with activity and location data to derive more meaningful contextual information. This paper contributes with an open ontology, the so-called Mining Minds Context Ontology, integrating all of the contextual definitions to link both low and high-level context domains. This work further contributes with the design, implementation and evaluation of a framework, namely the Mining Minds High-Level Context Architecture, which builds on the Mining Minds Context Ontology and reasoning techniques to enable the inference of high-level context from low-level context primitives in real time. The Mining Minds High-Level Context Architecture constitutes the core engine for the inference of high-level behavioral information in the Mining Minds platform [14,15]. Despite the proposed framework being originally devised to serve this platform, the Mining Minds High-Level Context Architecture has been defined in a way so it can be used independently for determining high-level context information from other similar sources of low-level context data.

The rest of the paper is organized as follows. Section 2 presents the related work. Section 3 describes the Mining Minds Context Ontology and presents some examples of context to illustrate the different modeling principles and the inference logic. Section 4 presents the Mining Minds High-Level Context Architecture and describes the inference method for the identification of the user context based on the Mining Minds Context Ontology. Section 5 describes the implementation of the Mining Minds High-Level Context Architecture and the results of its evaluation. Finally, the main conclusions and future steps are presented in Section 6.

2. Related Work

A number of surveys have reviewed the use and foundations of ontologies for context modeling. For example, a survey on context-aware systems [16] describes the basic design principles of context-aware architectures and depicts the different context models. Special focus is placed in this survey on the analysis and comparison of several approaches using ontologies. Another review of context modeling and reasoning techniques [17] discusses the requirements for modeling different context information and introduces the concept of high-level context abstractions. This survey describes and compares several ontology-based models of context information. Finally, a more recent survey on context-aware computing for the Internet of Things [18] evaluates 50 projects including the majority of research and commercial solutions proposed in the field of context-aware computing from 2001 to 2011. An extensive evaluation of research prototypes, systems and approaches building on ontology-based modeling and reasoning solutions is presented in this survey.

Many ontologies have been specifically proposed to model and recognize user context. The most well-known context ontologies and ontology-based context frameworks are described in the following. One of the most prominent ontologies for modeling context in pervasive environments is SOUPA (Standard Ontologies for Ubiquitous and Pervasive Applications) [19]. The core of the SOUPA ontology defines generic vocabularies for several domains: person, agent, belief-desire-intention, action, policy, time, space and event. Similarly, CONON (CONtext ONtology) [20] is a noticeable ontology for smart home environments. The CONON upper ontology captures the general features of different context entities: person, activity, computational entity and location. Both SOUPA and CONON ontologies are generic and can be extended to describe the context in the application-specific domain. For example, the Context Broker Architecture (CoBrA) [21] adopts the SOUPA ontology, whereas the SOCAM (Service-oriented Context-Aware Middleware) [22] builds on the CONON ontology. The CoBrA ontology describes places, agents and events in an intelligent meeting room. The ontology proposed in SOCAM models persons, activities, locations and devices for smart home and vehicle environments.

Apart from these early well-known solutions, more recent context ontologies and ontology-based context frameworks have been proposed. The Pervasive Information Visualization Ontology (PIVOn) [23] is composed of four ontologies for the description of intelligent environments: user, device, environment and service. The user model describes the static characteristics of the users, their agenda and their situation, including the user location, the current task and goals. The mIO! ontology [24] models context-related knowledge for the adaptation of applications in mobile environments. This ontology defines concepts like information on location and time, user information and its current or planned activities, as well as devices located in his or her surroundings. The Context Aggregation and REasoning (CARE) middleware [25] performs ontological and statistical reasoning to support the context-aware adaptation of Internet services in a mobile computing environment. The ontology, which models the user context within the CARE middleware, describes the user activities (actions and movements), interests, contacts, calendar items and places. For example, the context business meeting is defined as including any activity performed in a conference room within a company building and having at least two actors, each of which is an employee. Thus, the ontology in the CARE middleware models context based on activities and locations.

Some other works focus on the detection of a specific category of context, mainly activities, sometimes utilizing in their definition other types of contexts, such as locations. ActivO is the ontology used in COSAR [26], an activity recognition system that supports hybrid statistical and ontological reasoning. The ActivO ontology models a set of activities and the context data required to recognize them (the person performing the activity, the location of the activity and the time extent in which the activity takes place). The authors of the ActivO ontology have also proposed a very similar approach, but using OWL2 for modeling and reasoning [13]. Furthermore, some activities involve the interaction with objects. Thus, contextual information about the interaction (time and location) can be used to model and infer the activities. An ontology-based approach is used to model activities for smart homes in [9]. The proposed ontology models activities based on a sequence of user-object interactions and the location of the objects. For instance, the activity *making tea* is composed of the primitives *get cup*, *get tea*, *pour water*, *get milk* and *get sugar*, which take place in the *kitchen* . Composite activities in smart homes are modeled and recognized in [27]. Ontological and temporal knowledge modeling formalisms are combined to describe composite activities, like, for example, *make tea and then wash hands* . The work in [28] describes an ontology-based technique for multilevel activity recognition. The proposed ontology models atomic gestures (actions that cannot be decomposed), manipulative gestures (execution of simple atomic gestures), simple activities (temporal sequences of manipulative gestures) and complex activities (concurrent execution of simple activities). One example of a complex activity could be *clean up*, which is composed of the simple activities *put in dishwasher* and *clean table*. Finally, [29] proposes a fuzzy ontology for the representation of activity and the reasoning on vague, incomplete and uncertain knowledge. The ontology core models three domains: users, environment including locations and actions, activities and behaviors. Actions are atomic events, activities can be a single action or a composed set of actions, and behaviors are a sequence of activities and/or actions. For example, the behavior *coffee break* includes the action *exit office*, the activity *make coffee* or *take coffee* and the action *enter office*.

3. Mining Minds Context Ontology

There are multiple reasons supporting the choice of ontologies for context modeling and inference: ontologies surpass non-semantic models for context modeling in terms of flexibility, extensibility, generality, expressiveness and decoupling of the knowledge from the code [30–32]. More specifically, ontologies provide implicit semantics that enable the derivation of new information from existing ones, a key characteristic to procure interoperability among different systems. Moreover, the hierarchical structure of ontologies, with subclasses inheriting the properties from their ascendant classes, facilitates its evolvability and maintenance. In fact, new concepts can be easily added to the ontology and related to the existing ones, with multiple existing methods for the automatic validation of conflicts and semantic inconsistencies. Several reasoners are also available and can be used with the defined ontologies. Ontological reasoning can be inherently applied on ontology-based models; thus, simplifying the inference task and avoiding the need for the explicit definition of rules. Finally, using ontologies to infer context is also beneficial from the implementation perspective since no changes are required in the architecture and/or implementation whenever the model is extended; thus, only requiring the adaptation of the ontology itself.

The Mining Minds Context Ontology models context for human behavior identification in order to enable the provision of personalized health and wellness services in Mining Minds [14,15]. Since Dey proposed the first widely-accepted definition of context [33], many different interpretations of context have arisen. In Mining Minds, human context is defined as any information characterizing the physical, mental and social situation of a person that enables the identification of their behavior. Furthermore, human context is here categorized into two different levels of abstraction: low-level context and high-level context. Low-level context is defined as primitive context, i.e., contexts that can be directly identified from user data and do not require any other type of context information to be derived. Specifically, activities, locations and emotions are here considered as the three categories of low-level context. Activities can be normally identified from the body movement; locations can be directly derived from the user position; and emotions can be obtained from the user sentiments or physiological responses. High-level context is the context that requires several contexts of a diverse nature in order to be identified. This means that a high-level context builds on a combination of low-level contexts. Therefore, high-level contexts are more complex and abstract contexts.

The Mining Minds Context Ontology aims at comprehensively modeling the most commonplace and widely-used contexts for health and wellness services, such as the ones supported by Mining Minds. These contexts are typically observed for both sedentary and active lifestyles. Specifically, the high-level contexts include daily contexts like *office work*, *sleeping*, *house work*, *commuting*, *amusement*, *gardening*, *exercising*, *having meal* and *inactivity*. The low-level contexts required to compose the description of the high-level context have to be automatically recognizable. Thus, very simple low-level contexts in the domains of activities, locations and emotions are defined. Low-level contexts describing activities include sedentary activities associated with unhealthy habits, mild activities of daily living and some vigorous ones related to sport and fitness practices. Namely, the modeled activities are *lying down*, *standing*, *sitting*, *riding escalator*, *riding elevator*, *walking*, *running*, *jumping*, *hiking*, *climbing stairs*, *descending stairs*, *cycling*, *stretching*, *dancing*, *sweeping* and *eating*. Similarly, the low-level contexts describing the locations comprise the places where the user spends their daily life, i.e., *home*, *office*, *yard*, *gym*, *mall*, *restaurant*, *outdoors* and *transport*. The low-level contexts describing the emotions embrace the most prominent moods or states of mind, which are *anger*, *happiness*, *neutral*, *sadness*, *fear*, *disgust*, *surprise* and *boredom*. The specific combinations of low-level contexts that compose each high-level context are derived from the experience of the Mining Minds behavioral scientists. Figure 1 graphically represents these definitions of high-level context, which are modeled in the Mining Minds Context Ontology. The considered contexts are intended to represent a wide spectrum of situations and actions in a person's life; however, it must be noted that this list can certainly be extended in view of potential future applications while considering other less recurrent contexts.

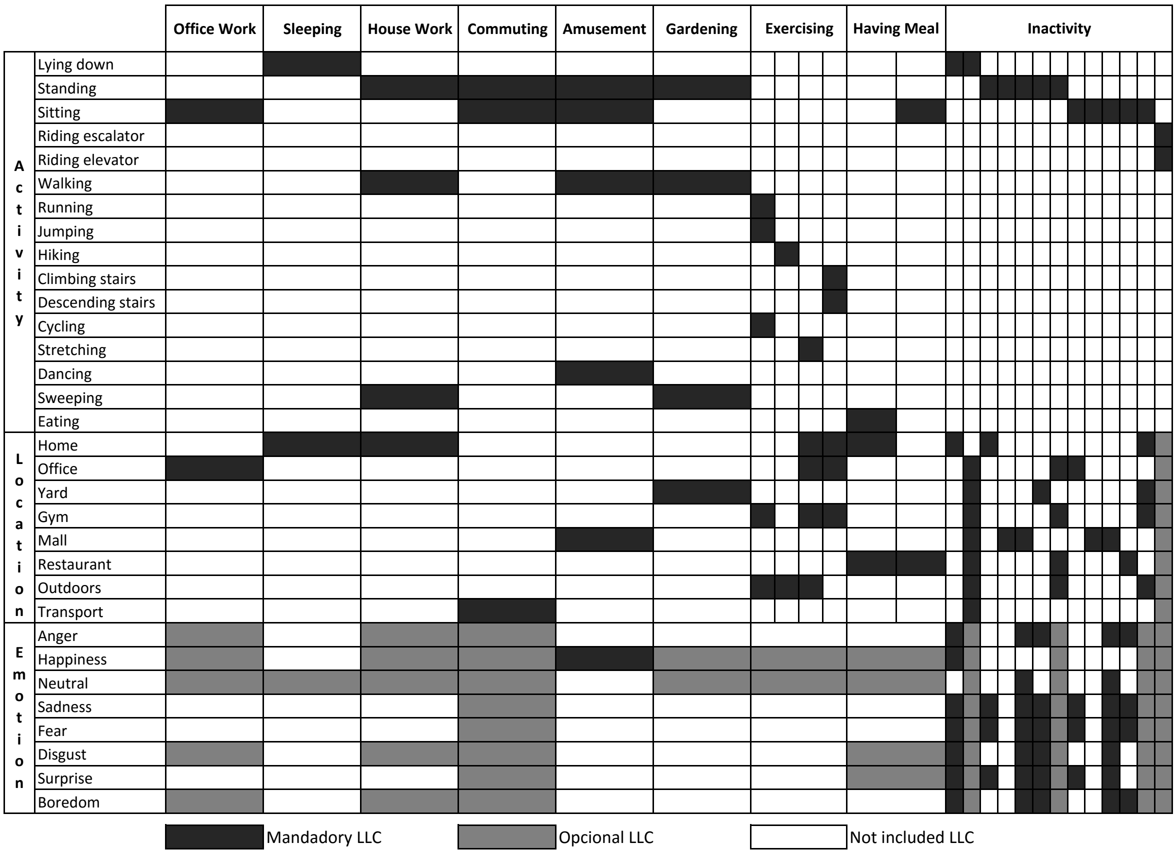


Figure 1. Graphical representation of the combination of low-level contexts that compose the high-level contexts modeled in the Mining Minds Context Ontology.

In broad strokes, the main novelties of the Mining Minds Context Ontology are a more comprehensive description of context using a two-level model and the incorporation of emotion information to detect some high-level contexts. First, a layered approach is followed in which high-level contexts build on a combination of low-level contexts. Current approaches model context in different dimensions, for example the user is performing an activity, has a location and has a mood. However, in these models, there is no clear link between the different dimensions of context, neither are they used to derive other contexts. Thus, some valuable information for the identification of human behavior is lost when using a one-level model. Second, the emotions enable the definition of new high-level contexts, which can only be identified whenever a specific emotion takes place. This is the case of the high-level context *amusement*, which must imply that the person is happy and having fun. For this context, it is not enough to know that the person is *sitting* in the *mall*, but also that their emotion is *happiness* in order to infer that the context refers to *amusement*. Therefore, in some cases, the activity and the location might not be enough to detect the high-level context, and the emotion enables the identification of more diverse high-level contexts. The Mining Minds Context Ontology is an OWL 2 ontology [34] and is publicly available at [35].

3.1. Terminology for the Definition of Context

The Mining Minds Context Ontology defines the concept of user context. The context is associated with a given user and has a start and an end. While a context has necessarily a start referring to the time in which the context initiates, the finalization of the context is not strictly necessary. This is motivated by the fact that the context may be prolonged over time and be still valid at the present time. A given context can refer to either low or high-level context. Low-level contexts represent either activities, locations or emotions, which can further compose a high-level context. In some cases, only one category of the low-level context is enough to determine the high-level context. This is the case of *inactivity*, where a sole sedentary activity like *sitting* defines this context. In some other cases, a specific category of low-level context is essential in order to identify the high-level context. For example, *amusement* can only be detected if the emotion is of type *happiness*. Accordingly, the ontology has been designed to support any combination of low-level contexts to define a specific high-level context. Given the seldom availability of emotion data, the ontology has been designed to procure the identification of some high-level contexts, even in the absence of emotion information. The modeling of this concept of context using the formal ontological description is presented in the following.

The main concept of the Mining Minds Context Ontology is the class *Context*, which represents the context of a user in an interval of time. Several necessary conditions are described for this class to model the concept of context (Figure 2). The existential and universal restrictions on the object property *isContextOf* ensure that any individual of the class *Context* is linked to an individual of the class *User* representing the user to which the context belongs. The existential and universal restrictions on the functional data property *hasStartTime* state that all of the individuals of the class *Context* must be related along this property to a unique *dateTime* data type of the W3C XML Schema Definition Language (XSD) [36] representing the instant of time in which the context starts. The universal restriction on the functional data property *hasEndTime* indicates that if there is a relationship of an individual of the class *Context* along the property *hasEndTime*, it has to be to a member of the XSD *dateTime* data type representing the end time of the interval in which the context is valid.

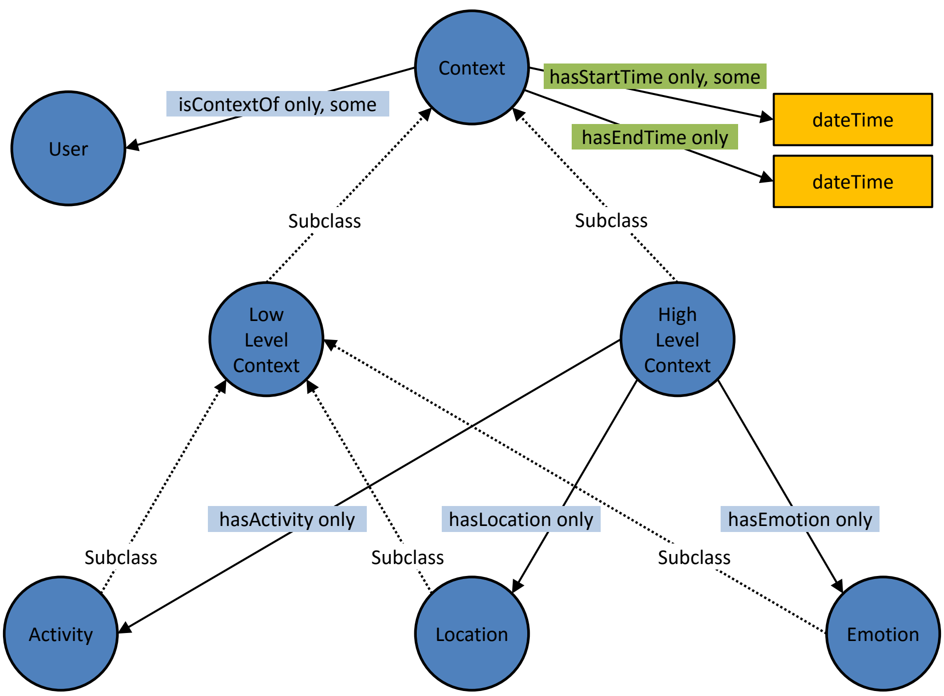


Figure 2. Mining minds context ontology: the class *Context*, its subclasses and the relations among them.

The class *LowLevelContext* represents the basic categories of low-level contexts via the classes *Activity*, *Location* and *Emotion*. The class *HighLevelContext* models the concept of high-level context. The universal restrictions on the object properties *hasActivity*, *hasLocation* and *hasEmotion* model the relationship between the individuals of the class *HighLevelContext* and the individuals of the different subclasses of *LowLevelContext*, which compose the high level context. The different types of high-level contexts are modeled via ten subclasses of the class *HighLevelContext*. Their equivalent anonymous classes as defined in Protégé [37] are presented in Figure 3.

```
HighLevelContext
 and (hasActivity some Sitting)
 and (hasLocation some Office)
 and (hasActivity only Sitting)
 and (hasEmotion only (Anger or Boredom or Disgust or Happiness or Neutral))
 and (hasLocation only Office)
```

(**a**)

```
HighLevelContext
 and (hasActivity some LyingDown)
 and (hasLocation some Home)
 and (hasActivity only LyingDown)
 and (hasEmotion only Neutral)
 and (hasLocation only Home)
```

(**b**)

```
HighLevelContext
 and (hasActivity some (Standing or Sweeping or Walking))
 and (hasLocation some Home)
 and (hasActivity only (Standing or Sweeping or Walking))
 and (hasEmotion only (Anger or Boredom or Disgust or Happiness or Neutral))
 and (hasLocation only Home)
```

(**c**)

```
HighLevelContext
 and (hasActivity some (Sitting or Standing))
 and (hasLocation some Transport)
 and (hasActivity only (Sitting or Standing))
 and (hasLocation only Transport)
```

(**d**)

```
HighLevelContext
 and (hasActivity some (Dancing or Sitting or Standing or Walking))
 and (hasEmotion some Happiness)
 and (hasLocation some Mall)
 and (hasActivity only (Dancing or Sitting or Standing or Walking))
 and (hasEmotion only Happiness)
 and (hasLocation only Mall)
```

(**e**)

```
HighLevelContext
 and (hasActivity some (Standing or Sweeping or Walking))
 and (hasLocation some Yard)
 and (hasActivity only (Standing or Sweeping or Walking))
 and (hasEmotion only (Happiness or Neutral))
 and (hasLocation only Yard)
```

(**f**)

```
HighLevelContext
 and
  (((hasActivity some Hiking)
    and (hasLocation some Outdoors)
    and (hasActivity only Hiking)
    and (hasLocation only Outdoors))
  or
  ((hasActivity some Stretching)
    and (hasLocation some (Gym or Home or Office or Outdoors))
    and (hasActivity only Stretching)
    and (hasLocation only (Gym or Home or Office or Outdoors)))
  or
  ((hasActivity some (ClimbingStairs or DescendingStairs))
    and (hasLocation some (Gym or Home or Office))
    and (hasActivity only (ClimbingStairs or DescendingStairs))
    and (hasLocation only (Gym or Home or Office)))
  or
  ((hasActivity some (Cycling or Jumping or Running))
    and (hasLocation some (Gym or Outdoors))
    and (hasActivity only (Cycling or Jumping or Running))
    and (hasLocation only (Gym or Outdoors))))
 and (hasEmotion only (Happiness or Neutral))
```

(**g**)

```
HighLevelContext
 and
  (((hasActivity some Eating)
    and (hasLocation some (Home or Restaurant))
    and (hasActivity only Eating)
    and (hasLocation only (Home or Restaurant)))
  or
  ((hasActivity some Sitting)
    and (hasLocation some Restaurant)
    and (hasActivity only Sitting)
    and (hasLocation only Restaurant)))
 and (hasEmotion only
      (Disgust or Happiness or Neutral or Surprise))
```

(**h**)

```
HighLevelContext
 and (not (Amusement or Commuting or Exercising or Gardening or HavingMeal
           or HouseWork or OfficeWork or Sleeping))
 and (hasActivity some
        (LyingDown or RidingElevator or RidingEscalator or Sitting or Standing))
 and (hasActivity only
        (LyingDown or RidingElevator or RidingEscalator or Sitting or Standing))
```

(**i**)

```
HighLevelContext
 and (not (hasActivity some Activity))
 and (not (hasEmotion some Emotion))
 and (not (hasLocation some Location))
```

(**j**)

Figure 3. Mining minds context ontology: definition of the ten subclasses of *HighLevelContext*. (**a**) *OfficeWork*; (**b**) *Sleeping*; (**c**) *HouseWork*; (**d**) *Commuting*; (**e**) *Amusement*; (**f**) *Gardening*; (**g**) *Exercising*; (**h**) *HavingMeal*; (**i**) *Inactivity*; (**j**) *NoHLC*.

In order to be a member of the defined class *OfficeWork* (Figure 3a), an individual of the class *HighLevelContext* must have a property of type *hasActivity* which relates to an individual of the class *Sitting*, and this property can only take as a value an individual of the class *Sitting*. Moreover, the individual of the class *HighLevelContext* must also have a property of type *hasLocation*, which relates to an individual of the class *Office* and only to an individual of the class *Office*. Finally, in case the individual of the class *HighLevelContext* has a property of type *hasEmotion*, this property must relate to an individual of the class *Anger*, the class *Boredom*, the class *Disgust*, the class *Happiness* or the class *Neutral*. This universal restriction does not specify that the relationship along the property *hasEmotion* must exist, but if it exists, it must link to the specified class members.

3.2. Instances of Context

An illustrative scenario is presented here to showcase the representation of instances of low-level contexts and high-level contexts in the Mining Minds Context Ontology (Figure 4). Let us imagine that it is 10 November 2015, and the user with identifier 9876 enters at 11:03:55 the office building of her or his working place. This event is detected by a location detector, a positioning system that interprets the coordinates of the user as the location of her or his office. Therefore, the low-level context of category location is identified as being of type *office* at 11:03:55. She or he starts talking on the phone, and a system capable of recognizing emotions detects from the tone of her or his voice that the user is bored. Thus, the low-level context of category emotion is identified as being of type *boredom* at 11:05:05. The phone call finalizes at 11:06:40, and then, no emotion is detected anymore. Meanwhile, at 11:05:25, the user sits down at her or his workplace. This event is detected by an activity recognizer that continuously measures her or his body motion. The low-level context of category activity is identified as being of type *sitting* at 11:05:25. It should be noted that every change in any of the low-level contexts may potentially lead to a new high-level context. For example, at 11:05:05, the combination of the activity *sitting*, the location *office* and the emotion *boredom* creates a high-level context that is classified as *office work*. At 11:06:40, when the emotion is no longer available, but the activity remains as *sitting* and the location as *office*, the high-level context for this user continues being identified as *office work*. Some combinations of low-level contexts do not constitute a known class of high-level context, based on the defined ontology. This is the case of the two high-level contexts at the beginning of this scenario. Namely, only location or the combination of the location *office* and the emotion *boredom* turn out to be not enough to identify a more abstract high-level context. Each context has associated a name, which serves as a unique identifier. These names are automatically created by the system whenever a new context is detected and are composed of the prefix *"llc_"* or *"hlc_"* and a sequential unique number. For the sake of simplicity, in this example, up to three digits are considered; however, large numbers are normally used by the system to procure unique identifiers. Furthermore, in order to make the example more understandable, for the low-level contexts, the membership of the instance to its name has been appended. For example, the context representing the activity *sitting* is named *llc_360_sitting*.

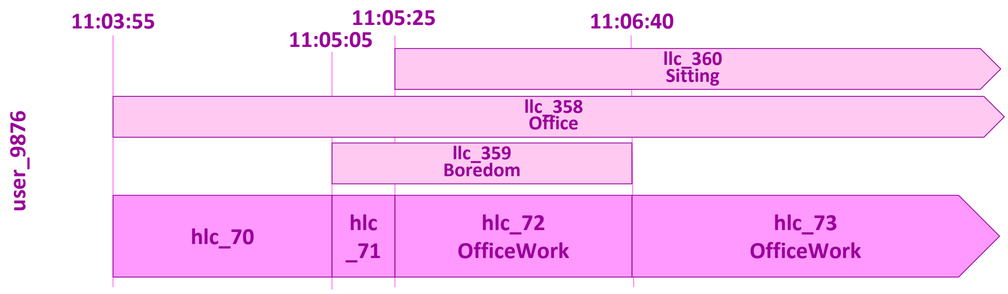


Figure 4. Exemplary scenario representing low-level contexts and high-level contexts.

The terminology described in Section 3.1 is utilized at this point to generate the instances of context resulting from this scenario. The instances of low-level context are directly created from the information provided by the activity recognizer, location detector or emotion recognizer. In Section 3.2.1, the generation of the low-level contexts is presented. High-level contexts can be created from the information of the low-level contexts which are part of it and which triggered its occurrence. In Section 3.2.2, the generation of the high-level contexts is introduced. High-level contexts can also be classified, i.e., the membership of the high-level context or the class to which a high-level context belongs can be determined. In Section 3.2.3, the inference of the membership of the high-level contexts is described. Since the process of inferring the membership of a high-level context is also called classification, the high-level contexts for which their membership has been inferred are hereafter called *classified* high-level contexts. Conversely, the high-level contexts for which their membership has not been inferred are hereafter called *unclassified* high-level contexts. Finally, it is possible that the classification of an unclassified high-level context does not result in any inferred statement. In other words, the high-level context does not belong to any of the classes of high-level context defined in the terminology. In this case, the high-level context, which has been intended to be classified, but does not belong to any known class, is called *unidentified* high-level context.

3.2.1. Instances of Low-Level Context

The low-level contexts are modeled as members of the subclasses of *LowLevelContext*: *Activity*, *Location* and *Emotion*. Figure 5 shows how the low-level contexts for the presented scenario are described in Protégé. *llc_358_office*, *llc_359_boredom* and *llc_360_sitting* are members of the classes *Office*, *Boredom* and *Sitting*, respectively. These instances model the low-level context of the user with identifier 9876. Thus, *llc_358_office*, *llc_359_boredom* and *llc_360_sitting* are related along the property *isContextOf* to the individual *user_9876*, which is a member of the class *User*. All of the individuals representing the low-level contexts have a relationship along the property *hasStartTime* to a value in the form of XSD *dateTime*, which represents the start time of the interval in which the low-level context is valid. For example, for the individual *llc_359_boredom*, the property *hasStartTime* links to the value *"2015-11-10T11:05:05"^^dateTime*, which indicates that this context started at 11:05:05 on 10 November 2015. Moreover, for this very individual, the property *hasEndTime* relates to the value *"2015-11-10T11:06:40"^^dateTime*, which means that this low-level context only occurred until 11:06:40 on 10 November 2015. Therefore, the individual *llc_359_boredom* models a low-level context of the type *boredom* for the user with identifier 9876 and which was valid in the period of time comprising from 11:05:05 to 11:06:40 on 10 November 2015.

Figure 5. Representation of the instances of low-level context for the exemplary scenario by using the Mining Minds Context Ontology in Protégé. (**a**) *llc_358_office* is a member of the class *Office*; (**b**) *llc_359_boredom* is a member of the class *Boredom*; and (**c**) *llc_360_sitting* is a member of the class *Sitting*.

3.2.2. Instances of Unclassified High-Level Context

The unclassified high-level contexts are modeled as members of the class *HighLevelContext* for which their properties and types are stated. Property assertions are used to define the low-level contexts that compose the unclassified high-level context. The properties *hasActivity*, *hasLocation* and *hasEmotion* relate to the individuals of the subclasses of the classes *Activity*, *Location* and *Emotion*, respectively. Reasoning in OWL is based on the Open World Assumption (OWA), which means that it cannot be assumed that something does not exist unless it is explicitly stated that it does not exist. Therefore, type assertions are used as closure axioms to indicate that an unclassified high-level context is composed of a unique and finite set of low-level contexts. Specifically, for each of the low-level context components of the high-level context, it is stated the type equivalent to the anonymous class represented by the universal restriction on the property *hasActivity*, *hasLocation* or *hasEmotion* where the value of the filler is the collection comprising only the low-level context. Furthermore, type assertions are also used as closure axioms to indicate that there is no low-level context of a specific category being part of the unclassified high-level context. In this case, for each of the categories of low-level context absent on the unclassified high-level context, it is stated that the type equivalent to the anonymous class is the negation class of the existential restriction on the property *hasActivity*, *hasLocation* or *hasEmotion* where the filler is the class representing the category of low-level context, *Activity*, *Location* or *Emotion*, respectively.

Figure 6 shows how the unclassified high-level contexts for the presented scenario are described in Protégé. *hlc_70*, *hlc_71*, *hlc_72* and *hlc_73* are members of the class *HighLevelContext*. Similarly as for the low-level contexts, the individuals representing the unclassified high-level contexts have relationships along the properties *isContextOf*, *hasStartTime* and *hasEndTime*. For the individual *hlc_72*, the property *hasActivity* relates to the individual *llc_360_sitting*, the property *hasLocation* to the individual *llc_358_office* and the property *hasEmotion* to the individual *llc_359_boredom*. Due to the OWA, *hlc_72* has been asserted as the type *hasActivity only ({llc_360_sitting})*, the type *hasLocation only ({llc_358_office})* and the type *hasEmotion only ({llc_359_boredom})*. These statements indicate that the individual *hlc_72* only has a *hasActivity* relationship to *llc_360_sitting*, a *hasLocation* relationship to *llc_358_office* and a *hasEmotion* relationship to *llc_359_boredom*. The individual *hlc_73* is composed of the same activity and location as *hlc_72*; however, no emotion is part of this unclassified high-level context. Therefore, *hlc_73* has been asserted as the type *not (hasEmotion some Emotion)*. This statement indicates that the individual *hlc_73* does not have any property of type *hasEmotion* linking to an individual of the class *Emotion*, i.e., this unclassified high-level context does not contain any emotion.

3.2.3. Instances of Classified High-Level Context

The classified high-level contexts are obtained using a reasoner that infers the membership of the unclassified high-level contexts. Thus, a classified high-level context is an individual of the class *HighLevelContext*, which is determined to be also a member of one of the ten subclasses of *HighLevelContext*: *OfficeWork*, *Sleeping*, *HouseWork*, *Commuting*, *Amusement*, *Gardening*, *Exercising*, *HavingMeal*, *Inactivity* or *NoHLC*. Figure 7 shows the classified high-level contexts for the working scenario and that have been inferred in Protégé using the Pellet reasoner [38]. The individuals *hlc_70* and *hlc_71* are not presented in the figure since they do not belong to any known class of high-level context, i.e., they are unidentified high-level contexts.

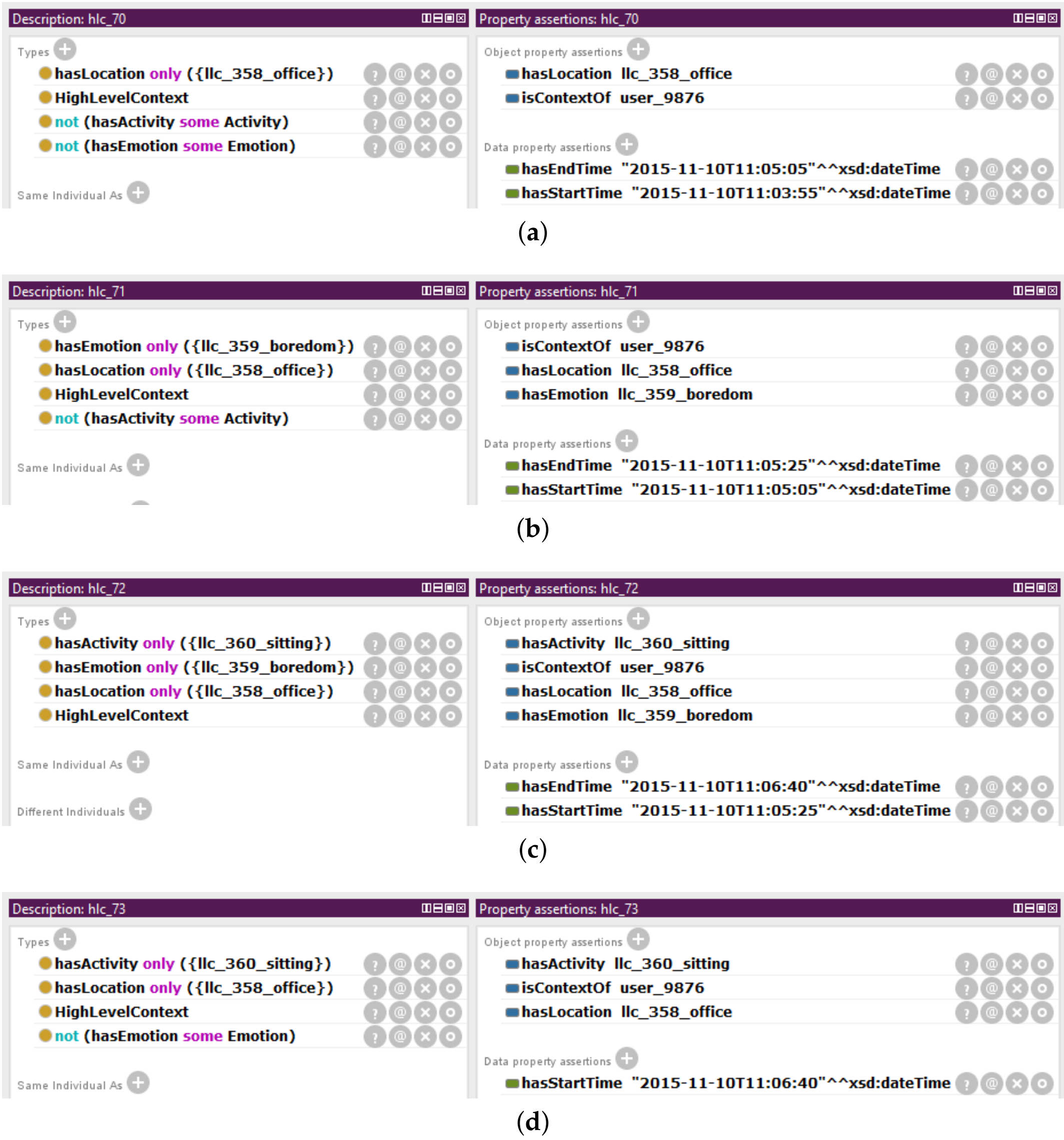
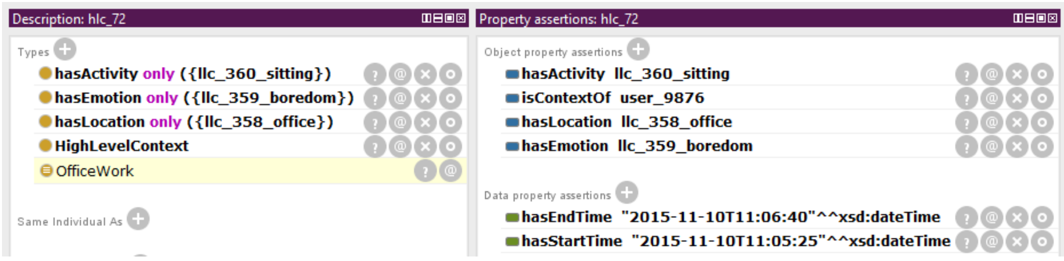
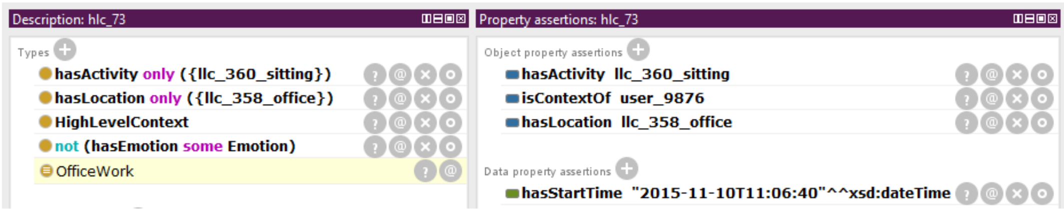


Figure 6. Representation of the instances of unclassified high-level context for the exemplary scenario by using the Mining Minds Context Ontology in Protégé. (**a**) *hlc_70*; (**b**) *hlc_71*; (**c**) *hlc_72*; and (**d**) *hlc_73* are composed of some of the low-level contexts *llc_358_office* (member of the class *Office*), *llc_359_boredom* (member of the class *Boredom*) and *llc_360_sitting* (member of the class *Sitting*).

The individual *hlc_72* is inferred by the reasoner to belong to the class *OfficeWork* (Figure 7a). Since this individual of the class *HighLevelContext* complies with the definition of the class *OfficeWork*, it is classified as being a member of this class. *hlc_72* fulfills the existential and universal restrictions on the property *hasActivity*, which state that a member of the class *OfficeWork* must have some *hasActivity* relationship to an individual of the class *Sitting* and only to a member of this class. These restrictions are met since the property *hasActivity* only links the individual *hlc_72* to the individual *llc_360_sitting*, which is a member of the class *Sitting*. Similarly, *hlc_72* also fulfills the existential and universal restrictions on the property *hasLocation*. Furthermore, *hlc_72* fulfills the universal restriction on the property *hasEmotion*, which states that in the case a member of the class *OfficeWork* has a *hasEmotion* relationship, it has to link to only an individual of the class *Boredom*. In fact, *hlc_72* is only related along the property *hasActivity* to the individual *llc_359_boredom*, which is a member of the class *Boredom*.



(**a**)



(**b**)

Figure 7. Representation of the instances of classified high-level context for the exemplary scenario by using the Mining Minds Context Ontology in Protégé. (**a**) *hlc_72*; and (**b**) *hlc_73*, which are both inferred to be members of the class *OfficeWork*, are composed of some of the low-level contexts *llc_358_office* (member of the class *Office*), *llc_359_boredom* (member of the class *Boredom*) and *llc_360_sitting* (member of the class *Sitting*).

The individual *hlc_73* is also classified by the reasoner as being a member of the class *OfficeWork* (Figure 7b). Similar to the classified high-level context *hlc_72*, the individual *hlc_73* also complies with the existential and universal restrictions on the properties *hasActivity* and *hasLocation*. However, the property *hasEmotion* about the individual *hlc_73* is not asserted. The universal restriction on the property *hasEmotion* does not state that the relationship must exist. In fact, it may not exist at all and the restriction still be fulfilled, as is the case for *hlc_73*. Thus, the individual *hlc_73* can be inferred as being a member of the class *OfficeWork*. The classification as members of the class *OfficeWork* of the two individuals of the class *HighLevelContext*, *hlc_72* and *hlc_73*, one with a *hasEmotion* relationship and another without it, proves the flexibility of the Mining Minds Context Ontology, which enables the identification of high-level contexts, even if one of the pieces of low-level information is missing. This is considered to be helpful in real-life scenarios where emotion recognition systems are not always available or may generate detection events in a less regular basis than activity recognizers or location detectors.

4. Mining Minds High-Level Context Architecture

The Mining Minds High-Level Context Architecture (HLCA) is the system architecture devised for the identification of the user context in Mining Minds. Conversely to most similar approaches, the HLCA supports the instance-based identification of context. The HLCA infers abstract context representations based on categories, such as physical activities, emotional states and locations. These categories, which are derived from the wide-spectrum of multimodal data obtained from the user interaction with the real- and cyber-world, are intelligently combined and processed at the HLCA in order to determine and track the user context. The inferred user context can be utilized by the Mining Minds entities and any other third party to enable the provision of personalized health and wellness services. The HLCA relies on the Mining Minds Context Ontology (Section 3) and applies ontological inference to identify the user context. Furthermore, the ontology is utilized in the representation of the context shared between the components of the HLCA. The HLCA consists of four main components (Figure 8): High-Level Context Builder (Section 4.1), High-Level Context Reasoner (Section 4.2), High-Level Context Notifier (Section 4.3) and Context Manager (Section 4.4).

In a nutshell, the operation of the HLCA is as follows. The High-Level Context Builder receives unstructured low-level information, namely activities, emotions and locations, yielded by the Low-Level Context Architecture, an independent Mining Minds entity that is in charge of identifying the user low-level context using data driven-approaches. Then, based on the received low-level context information, the High-Level Context Builder generates the ontological concepts representing the user context. The Context Mapper is in charge of interpreting the received low-level information and transforming it into the corresponding ontological concepts. The Context Synchronizer seeks concurrent low-level contexts, identifying other user contexts valid at the same moment in time. The Context Instantiator generates a new instance of an unclassified high-level context linking to the low-level contexts that compose it. The High-Level Context Reasoner receives the unclassified high-level context for its verification and classification. The Context Verifier checks the semantic and syntactic consistency of the unclassified high-level context. The Context Classifier identifies the membership of the unclassified high-level by applying ontological inference. The High-Level Context Notifier makes available the newly-classified high-level context to any third party application that registered for this type of information. During the context identification process, several components interact with the Context Manager, which provides the persistence of the Mining Minds Context Ontology, as well as supports the easy access to low-level context and high-level context information.

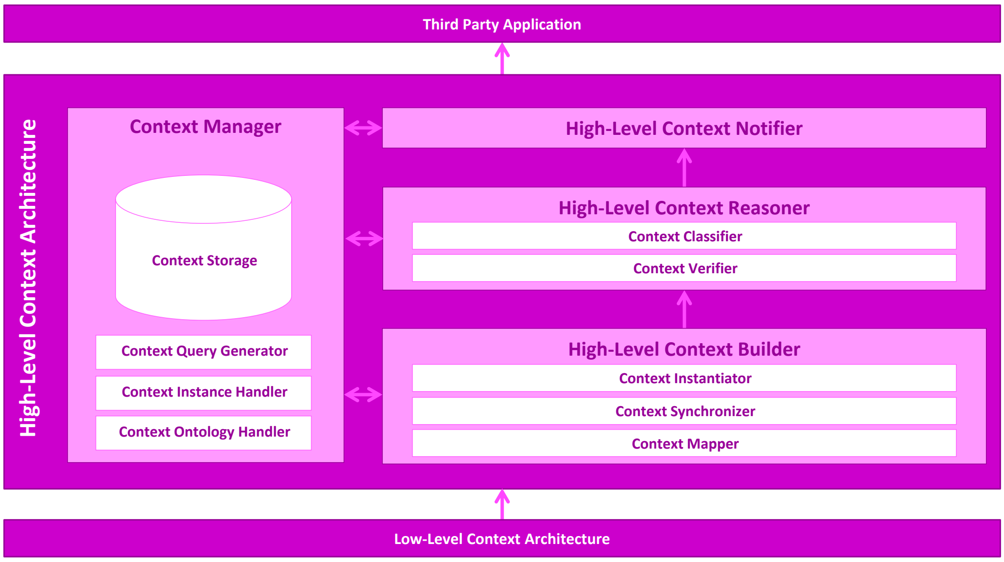


Figure 8. Mining Minds High-Level Context Architecture.

In the following, the different components of the HLCA are described in detail. For the sake of understanding, an example from the scenario presented in Section 3.2 is here considered to illustrate the operation of each component of the HLCA. Namely, the inference of a new high-level context at 11:05:25 on 10 November 2015 is considered. At that moment, a new low-level context of the category *sitting* for the user with identifier 9876 is detected by the Low-Level Context Architecture. This event triggers the operation of the HLCA, which after the processing identifies a new high-level context of type *office work* and serves it to the registered third party applications.

4.1. High-Level Context Builder

The High-Level Context Builder receives the low-level information, i.e., activities, emotions and locations, and generates the ontological concepts representing an unclassified high-level context associated with that information. The High-Level Context Builder has three subcomponents: the Context Mapper, the Context Synchronizer and the Context Instantiator.

4.1.1. Context Mapper

The Context Mapper interprets the received low-level information and transforms it into the corresponding ontological concepts. Specifically, it maps the labels plus metadata into ontological instances of low-level context. Whenever the Context Mapper gets a new label, it creates an instance of the subclass of the class *LowLevelContext* which represents the corresponding activity, location or emotion (as described in Section 3.2.1). The property *hasStartTime* is stated to relate this instance to the time in which the low-level context started and which is part of the received metadata. Furthermore, the user to which the context belongs is related along the property *isContextOf*. Once the low-level context instance has been created, it is stored in the Context Manager for its persistence (see Section 4.4.3) and it is notified to the Context Synchronizer.

For the working example, the Context Mapper receives at run-time the activity label "sitting" and several metadata, i.e., the identifier of the user "9876" and the time in which the context starts "2015-11-10T11:05:25". The Context Mapper generates an instance of low-level context and then asserts the properties about it. The instance *llc_360_sitting* of the class *Sitting* presented in Figure 5c is created. This instance has a *isContextOf* relationship to the individual *user_9876* and a *hasStartTime* relationship to the value *"2015-11-10T11:05:25"^^dateTime*.

4.1.2. Context Synchronizer

The Context Synchronizer searches for concurrent low-level contexts, whenever the Context Mapper has notified a newly detected low-level context instance. A change in the low-level context implies a new high-level context, comprising the new low-level context and the other low-level contexts still valid at the start of the new low-level context. The Context Synchronizer needs to determine the other low-level contexts of a given user which are valid a the start time of the new low-level context instance created by the Context Mapper. Therefore, one of the most important roles of the Context Synchronizer is to align concurrent low-level contexts of the same user which might have been received in an unordered manner due to the diverse delays introduced by the different components of the Low-Level Context Architecture. In order to search for the concurrent low-level contexts, the Context Synchronizer requests information stored in the Context Manager and accesses it through the Context Instance Handler (see Section 4.4.3). Once the Context Synchronizer has determined the low-level contexts concurrent to the one that triggered the process, the Context Instantiator is invoked.

In the considered example, when the Context Synchronizer is notified by the Context Mapper about the identification of the new low-level context represented by the instance *llc_360_sitting*, it searches for concurrent low-level contexts by querying the information stored in the Context Manager. The instances *llc_358_office* and *llc_359_boredom*, presented in Figure 5a,b, are found to be concurrent to the low-level context *llc_360_sitting*. These two low-level contexts belong to the same user, i.e., user with identifier 9876, and they are still valid at 11:05:25 on 10 November 2015, when the new low-level context *sitting* starts.

4.1.3. Context Instantiator

The Context Instantiator creates a new instance of an unclassified high-level context linking to the constituent low-level contexts. Whenever the Context Synchronizer detects a set of low-level contexts which are concurrent to a newly detected one, the Context Instantiator creates a new instance of an unclassified high-level context containing these low-level contexts (as described in Section 3.2.2). Therefore, an instance of the class *HighLevelContext* is created and the different low-level contexts which compose the high-level context are related to it along the properties *hasActivity*, *hasLocation*, and *hasEmotion*. Moreover, the closure axioms are established via type assertions on these properties. In case there is a low-level context of a particular type, the Context Instantiator generates the axiom stating that the property can only link to that given low-level context. Otherwise, if no low-level context has been determined for one of the categories-activities, locations or emotions-, the Context

Instantiator creates the axiom stating that there is no low-level context of that category. Furthermore, the Context Instantiator establishes a *hasStartTime* relationship to the time in which the high-level context change happened, i.e., the time in which the newly detected low-level context started and which triggered the creation of the new unclassified high-level context. Moreover, the user to which the high-level context belongs is related along the property *isContextOf*. The identifier of the user to which the high-level context belongs is the same than the one associated to the low-level contexts which compose the high-level context. Once the Context Instantiator has created the instance of an unclassified high-level context, this is served to the High-Level Context Reasoner (see Section 4.2) for its verification and classification.

For the working example, the Context Instantiator receives from the Context Synchronizer the newly detected low-level context represented by the instance *llc_360_sitting* and the concurrent low-level contexts *llc_358_office* and *llc_359_boredom*. The Context Instantiator creates the instance *hlc_72* of the class *HighLevelContext* (see Figure 6c) and links it to the low-level contexts which compose it. Therefore, the properties *hasActivity*, *hasLocation*, and *hasEmotion* relate, respectively, to the instances *llc_360_sitting*, *llc_358_office*, and *llc_359_boredom*. The closure axiom *hasActivity only ({llc_360_sitting})* indicates that the individual *hlc_72* only has a *hasActivity* relationship to the individual *llc_360_sitting*. Similarly, the other two closure axioms, *hasLocation only ({llc_358_office})* and *hasEmotion only ({llc_359_boredom})*, state the uniqueness of the relationships. The Context Instantiator also specifies that the instance *hlc_72* has a *isContextOf* relationship to the individual *user_9876* which is the owner of the different low-level contexts composing the high-level context. Finally, the Context Instantiator creates a relationship along the property *hasStartTime* to the moment in which the change in the low-level context triggered the identification of the new high-level context. The start time of the high-level context *hlc_72* is the start time of the low-level context *llc_360_sitting*. Thus, for the instance *hlc_72* the property *hasStartTime* links to the value *"2015-11-10T11:05:25"^^dateTime*.

4.2. High-Level Context Reasoner

The High-Level Context Reasoner performs a consistency check on the unclassified high-level context instance created by the High-Level Context Builder (see Section 4.1). In case the instance is valid, the High-Level Context Reasoner identifies the context type to which the high-level context belongs, i.e., it classifies the high-level context instance. In order to perform these tasks, the High-Level Context Reasoner applies ontological inference supported by the formal description of context in the Mining Minds Context Ontology (see Section 3.1). The High-Level Context Reasoner comprises two subcomponents: the Context Verifier and the Context Classifier.

4.2.1. Context Verifier

The Context Verifier checks the semantic and syntactic consistency of the unclassified high-level context provided by the High-Level Context Builder. Therefore, the instance of unclassified high-level context is validated and verified versus the Mining Minds Context Ontology, which is stored in the Context Manager and can be accessed through the Context Ontology Handler (see Section 4.4.2). During the consistency check, non-logical or malformed high-level contexts can be detected. For example, the high-level contexts which do not contain the necessary property *hasStartTime* or the ones composed from multiple different instances of low-level contexts of the same type. Once the Context Verifier has ensured that the unclassified high-level context is valid, this instance is provided to the Context Classifier for further processing.

In the described example, the Context Verifier receives from the Context Instantiator the newly created high-level context *hlc_72*. This instance is checked for its semantic and syntactic consistency, it is considered to be valid, and it is then served to the Context Classifier.

4.2.2. Context Classifier

The Context Classifier identifies the type of high-level context to which the unclassified high-level context belongs; thus, converting the unclassified instance into a classified high-level context. The classification of the unclassified high-level context instance into one of the defined high-level context classes is based on the inference functionalities provided by the Mining Minds Context Ontology. Specifically, one of the key features of this ontology is that it can be processed by a reasoner which can automatically perform the classification process. This means that the unclassified high-level context instance is compared versus the definitions of the different high-level context classes to determine whether it complies with the conditions that define the class. In case it complies, the instance is inferred to belong to that class. The classification process is triggered every time the Context Classifier receives a new valid instance of high-level context from the Context Verifier. After the membership of the unclassified high-level context instance has been determined, the Context Classifier adds to the unclassified high-level context instance the axiom stating that this instance belongs to a specific type of high-level context. Therefore, the instance of the class *HighLevelContext* which models the classified high-level context is related along the property *rdf:type* to the subclass of the class *HighLevelContext* representing the high-level context of which the instance is a member. It is possible that the unclassified high-level context does not belong to any of the known classes described in the Mining Minds Context Ontology. This means that no membership is inferred and the unclassified high-level context is considered to belong to an unidentified type of high-level context. In this case, the classified high-level context has the same exact representation than the corresponding unclassified high-level context. Finally, the Context Classifier serves the classified high-level context to the High-Level Context Notifier (see Section 4.3).

For the working example, the Context Classifier receives from the Context Verifier the high-level context *hlc_72*. The Context Classifier applies the classification method to this unclassified high-level context in order to determine its membership. The individual *hlc_72* is inferred to belong to the class *OfficeWork* since it complies with the definition of the class *OfficeWork* (as described in Section 3.2.3). Therefore, the Context Classifier creates the axiom *hlc_72 rdf:type OfficeWork* which indicates that the individual *hlc_72* is a member of the class *OfficeWork*. The classified high-level context instance *hlc_72* is provided to the High-Level Context Notifier for its notification.

4.3. High-Level Context Notifier

The High-Level Context Notifier makes available to third party applications the newly identified high-level contexts. The High-Level Context Notifier receives from the High-Level Context Reasoner a classified high-level context instance and notifies the subscribed third parties about the detection of a new high-level context. This notification is only conducted if the new instance belongs to a high-level context type different than the previous one. Only changes in the high-level context type are notified, this means that differences in the low-level context composition which do not imply a change on the type of high-level context are not communicated to the third parties. Furthermore, the High-Level Context Notifier stores the new high-level context into the Context Manager for its persistence via the Context Instance Handler (see Section 4.4.3) and gets as an answer from this component the previous valid high-level context.

For the described example, the High-Level Context Notifier receives from the High-Level Context Reasoner the high-level context *hlc_72* which has been classified as *OfficeWork*. The High-Level Context Notifier contacts the Context Instance Handler for the persistence of the instance *hlc_72* into the Context Storage. Moreover, the High-Level Context Notifier receives from the Context Instance Handler the previous valid instance of high-level context *hlc_71*. The High-Level Context Notifier compares the membership of *hlc_72* to the membership of the previous valid high-level context *hlc_71*. The High-Level Context Notifier determines that there has been a change in the type of high-level context, the previous instance *hlc_71* was unidentified and the new instance *hlc_72* is *office work*.

Therefore, the third parties are notified about the change in the high-level context modeled as the instance *hlc_72*.

4.4. Context Manager

The Context Manager persists the Mining Minds Context Ontology, including the terminology for the definition of context and the instances of context. Furthermore, this component eases the interactions with the persisted context information by facilitating the exchanges with the storage infrastructure. The Context Manager has four subcomponents: the Context Storage, the Context Ontology Handler, the Context Instance Handler and the Context Query Generator.

4.4.1. Context Storage

The Context Storage is a database which provides persistence for the storage of the Mining Minds Context Ontology, including both the context definition terminology and the context instances. Since the context is modeled via an ontology and the context instances are represented as ontological instances, this storage is devised to be a database of the type triple store. Moreover, the Context Storage also provides read and write functionalities for the Mining Minds Context Ontology. However, this storage cannot be directly accessed and all the interactions are handled through the Context Ontology Handler and the Context Instance Handler.

4.4.2. Context Ontology Handler

The Context Ontology Handler provides the management functionalities to interact with the Mining Minds Context Ontology terminology stored in the Context Storage. This component enables loading the context ontology to the Context Storage at the system start time. The Context Ontology Handler also supports the retrieval of the context ontology which is stored in the Context Storage, so that the rest of components of the HLCA have access to the latest version of the ontological terminology. Furthermore, the Context Ontology Handler enables the extension at runtime of the context ontology. The extensibility is required to evolve the context ontology, therefore, including new types of low-level contexts and new definitions for the high-level contexts. Every time the ontology is updated, the rest of components of the HLCA making direct use of the context ontology are notified to obtain an updated version of the terminology.

4.4.3. Context Instance Handler

The Context Instance Handler deals with the retrieval and storage of context information in the Context Storage. The Context Instance Handler offers three different functionalities: storage of a newly mapped low-level context, retrieval of concurrent low-level contexts, and storage of a newly inferred high-level context while retrieving the previous valid high-level context. The Context Instance Handler poses to the Context Storage the SPARQL queries [39] created by the Context Query Generator in order to retrieve the persisted context information. Specifically, the logic of the Context Instance Handler for the storage of a newly inferred high-level context is as follows. The identification of a new high-level context implies that the previous context for the given user is not valid anymore. Therefore, the storage process includes the finalization of the previous valid high-level context instance. This operation entails to set the value of the end time of the previous valid high-level context stored in the Context Storage. In order to find the previous valid high-level context, the Context Instance Handler needs to pose the appropriate SPARQL queries to the Context Storage. The Context Query Generator is invoked to create the queries for the previous valid high-level context based on the newly inferred high-level context instance (see Section 4.4.4). Furthermore, it must be noted that a an earlier new high-level context could be inferred after the classification of a posterior one. This scenario is not very common but could happen due to the different delays in the data-driven recognition process for the low-level contexts. If this situation occurs, the newly inferred high-level context is only valid until the start time of the posterior high-level context already stored in the Context Storage. Therefore, the storage process also includes the finalization of the newly inferred high-level context instance.

In the considered example, the High-Level Context Notifier interacts with the Context Instance Handler to persist the newly classified high-level context instance *hlc_72* and to retrieve the previously valid instance of high-level context. Therefore, the Context Instance Handler stores the instance *hlc_72* into the Context Storage. Moreover, the Context Instance Handler retrieves from the Context Storage the previously valid instance of high-level context. The previous high-level context is here an individual of the class *HighLevelContext* modeling the context of the user represented by the individual *user_9876* and which is valid at at 11:05:25 on 10 November 2015. In order to retrieve the previous high-level context for the instance *hlc_72*, the Context Instance Handler invokes the Context Query Generator which creates the SPARQL queries presented in Listing 1. This query is posed to the Context Storage which returns as the matching result the high-level context *hlc_71*. Then, the Context Instance Handler finalizes the previous high-level context instance *hlc_71*. This means that the individual *hlc_71* is related along the property *hasEndTime* to the value *"2015-11-10T11:05:25"^^dateTime*, which is the value for the property *hasStartTime* of the newly identified high-level context *hlc_72*. In this exemplary scenario, it is assumed that there are no delays in the recognition of the low-level contexts and therefore, there are no high-level contexts posterior to *hlc_72* which had already been detected.

4.4.4. Context Query Generator

The Context Query Generator is the component which generates the SPARQL queries [39] required by the Context Instance Handler in order to find the matching context instances stored in the Context Storage. The SPARQL queries are automatically created based on some information derived from the context instance that the Context Instance Handler provides to the Context Query Generator. The Context Query Generator is capable of generating several different SPARQL queries depending on the expected outcome required for each specific use case scenario. The Context Query Generator creates SPARQL queries for the identification of a low-level context still valid at the start time of a newly recognized low-level context, which belongs to the very user and which is of the same context category. The Context Query Generator also creates SPARQL queries for the identification of the start time of the next posterior low-level context which belongs to the actual user and which is of the same context category. The Context Query Generator can also create SPARQL queries for the identification of low-level contexts of a given user which are concurrent at the start time of a newly recognized low-level context instance. In addition, the Context Query Generator creates SPARQL queries for the identification of a high-level context which is still valid at the start time of a new high-level context and which belongs to the same user. Finally, the Context Query Generator creates SPARQL queries for the identification of the start time of the next posterior high-level context belonging to the same user.

The logic for the creation of SPARQL queries for the identification of a high-level context which is still valid at the start time of a new high-level context and which belongs to the same user is the following. There are two cases in which the previous high-level context is still valid, either it does not have an end time or its end time is posterior to the start time of the new high-level context. In the first case, the SPARQL needs to match a high-level context for the same user which has a start time previous to the start time of the new high-level context but does not have an end time. In the second case, the SPARQL needs to match a high-level context for the same user which has a start time previous to the start time of the new high-level context and an end time posterior to the start of the new high-level context.

The specific SPARQL query to request the previous high-level context for the instance *hlc_72* is presented in Listing 1. In the considered example, the previous high-level context for *hlc_72* is an individual of the class *HighLevelContext* which belongs to the user represented by the individual *user_9876* and which is valid at 11:05:25 on 10 November 2015. Therefore, the matching individual has to be a member of the class *HighLevelContext*, must have a *isContextOf* relationship to the individual *user_9876*, must have a *hasStartTime* relationship to a value less than or equal to *"2015-11-10T11:05:25"^^dateTime*, and must not have any *hasEndTime* relationship.

```
SELECT ?hlc
WHERE
 { ?hlc rdf:type HighLevelContext ;
     isContextOf user_9876 ;
     hasStartTime ?starttime .
  FILTER NOT EXISTS ?hlc hasEndTime ?endtime .
  FILTER ( ?starttime <= "2015-11-10T11:05:25"^^xsd:dateTime )
 }
```

Listing 1. SPARQL query to request the previous high-level context for the instance *hlc_72*.

5. Evaluation

This section analyzes both the robustness and performance of the Mining Minds High-Level Context Ontology and Architecture. Section 5.1 explores the tolerance offered by the Mining Minds Context Ontology for the inference of high-level contexts under the presence of low-level context errors. Section 5.2 studies the performance of the Mining Minds High-Level Context Architecture with respect to processing time and management of context instances.

5.1. Robustness of the Mining Minds Context Ontology

The proposed Mining Minds Context Ontology has been evaluated to determine how robust the identification of high-level contexts can be in the event of having erroneously detected low-level contexts. In other words, this evaluation aims at measuring the level of resilience of the high-level context level against errors originated at the low-level context level. Pellet (v2.3.2) [38], an open source OWL DL reasoner for Java has been used in the evaluation test program. First, a set of 1800 instances representing all the possible combinations of low-level contexts, i.e., activities, locations and emotions, have been generated. Then, the instances have been posed to the reasoner and the corresponding high-level contexts have been inferred. The resulting array of high-level contexts represents the ground-truth for this evaluation. Subsequently, various scenarios with increasing levels of error in the low-level contexts have been defined. Namely, 5, 10, 20 and 50 per cent of errors have been respectively introduced in the 1800 instances as to emulate potentially erroneous low-level contexts. For example, in the case of having a 10% of affected instances a total of 180 randomly selected instances are deliberately modified. The error has been introduced by replacing the supposedly affected low-level context with a new value randomly selected from the remaining contexts in the affected category (activity, location or emotion). Thus for example, if the original instance of high-level context is composed of the contexts *sitting*, *office* and *boredom*, and the affected context is the activity, the newly generated instance could contain the contexts *running*, *office* and *boredom*. Moreover, in order to evaluate the prominence of each specific context category or combination thereof, the analysis has been formulated for all the combinations of low-level categories, i.e., introducing errors in solely the activity, location, emotion, or combination of activity and location, activity and emotion, location and emotion, and all activity, location and emotion. The instances resulting from all these scenarios have been posed to the reasoner and the resulting high-level contexts have been compared against the ground truth to determine the accuracy of the model. Each of the experiments has been repeated one hundred times in order to ensure the statistical robustness. The average and standard deviation accuracy is presented in Table 1 for each corresponding study.

Table 1. Mean and standard deviation of the accuracy of the high-level context recognition under different levels of errors in the detected low-level contexts.

	5%	10%	20%	50%
Activity	97.60 ± 0.05	95.13 ± 0.05	90.39 ± 0.04	75.32 ± 0.20
Location	99.45 ± 0.02	98.82 ± 0.05	97.61 ± 0.15	93.93 ± 0.02
Emotion	99.63 ± 0.02	99.18 ± 0.05	98.32 ± 0.05	96.04 ± 0.07
Act & Loc	97.08 ± 0.10	94.27 ± 0.16	88.48 ± 0.11	72.63 ± 0.10
Act & Emo	97.16 ± 0.12	94.22 ± 0.06	89.60 ± 0.10	73.53 ± 0.30
Loc & Emo	99.00 ± 0.05	98.02 ± 0.09	96.24 ± 0.05	91.25 ± 0.09
Act & Loc & Emo	96.56 ± 0.06	93.10 ± 0.30	87.52 ± 0.11	71.60 ± 0.13

From an overall analysis of the obtained results it can be concluded that the impact of the error introduced in the low-level context is generally lower at the high-level context. For example, in the case of introducing a 5% error, the accuracy drops approximately no more than 0.4% at best and 3.5% in the worst case scenario. Similarly, for the 10%, 20% and 50% error cases the minimum and maximum accuracy drops are below the corresponding level of error. Experiencing a lesser impact is generally due to the fact that not all the misrecognitions at the low-level context lead to an inference error at the high-level context. For example, if the activity *running* is recognized instead as *climbing stairs*, and provided the rest of low-level contexts to be *gym* for the location and *neutral* for the emotion, the inferred high-level context remains to be *exercising*. Similar examples in which the error in the low-level context is not propagated to the high-level context can be found for the case of erroneous locations and emotions. It can also be observed that the activity is the most prevalent category in terms of error impact, which is certainly as a consequence of the importance given to the activity context in the definition of high-level contexts. Conversely, the location and especially the emotion tend to show a lower effect on the high-level context. In fact the definition of some high-level contexts allows for a good level of resilience against errors in the locations and the emotions. This is the case of the high-level context *inactivity*, which is determined from a sole sedentary activity, like *lying down*, and nearly any location and emotional state. Therefore, even if an the location is erroneously detected, the inferred high-level context would result in *inactivity*. The only exception to this case would happen if the location is misrecognized as *home*, since *lying down* at *home* and with a *neutral* emotional state is identified as the high-level context *sleeping*. Moreover, errors simultaneously present in various low-level contexts generally increase the chance of misidentification of the actual high-level context. Therefore, the combinations of errors in several low-level categories report a lower accuracy in the high-level context recognition than in the case of having only errors in a single category. As it was expected, the highest impact is observed when all three low-level contexts are subject to error. Either way, the error in the recognition of the high-level context remains below the level of error introduced in the considered low-level contexts. Finally, it must be noted that owing to the descriptive logic characteristic of ontologies, and conversely to probabilistic classification models, combinations of correct low-level contexts will always lead to a correctly inferred high-level context.

5.2. Performance of the Mining Minds High-Level Context Architecture

The HLCA has been implemented and benchmarked in order to assess its performance. The HLCA has been implemented in Java using available open source libraries. All the components of the HLCA build on Apache Jena (v2.11.2) [40], a semantic web framework which includes some APIs for handling RDF [41], OWL [34], and SPARQL [39]. In the implementation of the High-Level Context Reasoner, an off-the-shelf open source reasoner, namely Pellet (v2.3.2) [38], has been utilized in combination with Jena to enable the ontological inference functionalities. Furthermore, in the Context Manager, the Jena Triple Store (TDB) has been used as the Context Storage for the persistence of the Mining Minds Context Ontology. The communication between the Low-Level Context Architecture and the HLCA has been implemented by means of RESTful web services [42] and establishing service contracts among these two architectures. The same mechanism applies to the communication of the HLCA with the third party applications registered to get access to the newly identified high-level contexts.

For the evaluation, this implementation of the HLCA has been executed on a laptop operating Windows 10 with a 1.80 GHz Intel Core i7 CPU, 8GB RAM, and a HDD with 5400-RPM spindle speed, I/O data-transfer rate up to 6 Gb/s and 16 MB buffer. Using a test Java application the Low-Level Context Architecture has been emulated. The evaluation has consisted in the generation of 250,000 random low-level contexts belonging to 100 different users and which represented their context information for a time span of 16 days. First the category of the low-level context (activity, location or emotion) has been randomly selected and then one of the types for that category has also been randomly chosen. After that, the metadata associated to the low-level context label has been generated. The low-level context has been randomly assigned to one of the 100 users. The start time of each low-level context has also been randomly selected between 1 and 10 s after the start time of the previous low-level context. The generated low-level contexts, including the labels and the metadata, have been input one at a time to the HLCA for their mapping, synchronization, instantiation, verification, classification and notification. It is important to notice that the low-level contexts are served to the HLCA sequentially and at their simulated occurrence time. Thus, the HLCA works at real-time and processes each single instance on-the-fly right after receiving it. Concurrency is procured through user-based multithreading, thus supporting simultaneous processing of low-level contexts from different users taking place at the same time. Some resources such as the Context Storage are shared among threads (users). During the evaluation the time required for the context identification has been calculated and the volume of information generated and stored on the Context Storage has further been determined.

Figure 9 shows the time invested by each of the HLCA components and the system as a whole in the context identification process. The number of instances indicates the number of high-level contexts which have already been processed by the HLCA when the recognition process is triggered due to a change in the low-level context. Even if the context recognition process is performed instance-wise, the number of previously processed instances is important because of the volume of information generated by the system during the recognition process and persisted in the Context Storage. The processing times are further averaged to have an overall figure summarizing the time taken by the each component of the HLCA. Table 2 presents the mean and standard deviation of these times as well as the percentage of these times devoted to the interaction of the component with the Context Manager. This interaction is particularly relevant because the Context Manager hosts the Context Storage, the shared resource which persists and loads the context information.

Table 2. Mean and standard deviation of the processing time invested by each of the HLCA components in the context identification, as well as the percentage of these times devoted to the interaction with the Context Manager.

	Context Mapper	Context Synchronizer	Context Instantiator	Context Verifier	Context Classifier	Context Notifier
Mean (s)	0.986	2.188	0.001	0.032	0.046	1.012
Standard Deviation (s)	0.348	1.670	0.000	0.014	0.019	0.268
Context Manager (%)	99.53	99.97	0.00	0.00	0.00	99.99

One can observe the differences of scale in the processing times for each of the components of the HLCA and the disparate tendencies of these times when the number of recognized context instances increases. The processes in which the HLCA component does not have any interaction with the Context Storage take much less time than the ones involving it. Furthermore, in the cases where the Context Storage is not involved, the processing time does not increase with the number of identified context instances. The Context Classifier and the Context Verifier take only some milliseconds to verify and classify the high-level context instance. This time is quite small due to the architectural design principle for which each single instance of high-level context is reasoned separately on-the fly at run-time. The Context Instantiator does not access either the Context Storage, since the required interactions to find the concurrent low-level contexts are performed by the Context Synchronizer. Therefore, the Context Instantiator takes only one millisecond to create a new instance of high-level context and this time does not increase with the number of instances because of the independence of the process from any other high-level context.

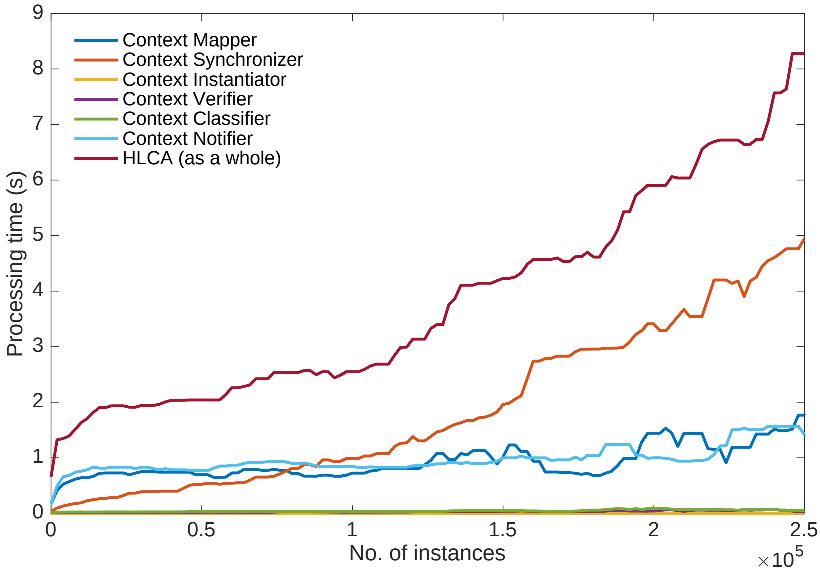


Figure 9. Processing time invested by each of the HLCA components in the context identification. The number of instances indicates the amount of previously processed high-level contexts when the recognition process is triggered.

In case the components of the HLCA invoke the Context Storage, the processing times rise and the interactions with the Context Storage tend to represent most of the computational time, specifically more than 99%. This means that the actual component is relatively quick to perform its job but the context read and write processes which involve the Context Manager delay the complete process. The processing time for the Context Mapper and the High-Level Context Notifier follow a similar pattern. These processing times increase with the number of instances, at the beginning and with very few instances the times rocket, but then they stabilize and reach values around one second. The similarity in the evolution of the processing times for these two components is normal because their interactions with the Context Manager are of the same type. In the first case, the Context Mapper stores the new low-level context instance, retrieves the previous low-level context and after updating it, stores it again into the Context Manager. In the second case, the High-Level Context Notifier stores the new high-level context instance, retrieves the previous high-level context, compares them and after updating the previous instance, stores it again into the Context Manager. Therefore, the evolution of the processing time for operations that involve read and write to the Context Manager can be observed in the times for the Context Mapper and the High-Level Context Notifier. The processing performed by the Context Synchronizer in order to request concurrent low-level context instances is the most time demanding process of the HLCA. In this case, most of the time is devoted to the execution of the SPARQL queries and the retrieval of the matching solutions from the Context Manager. The processing time for the Context Synchronizer increases almost lineally with the number of instances. In fact, for few instances this time is below the processing time for the Context Mapper and High-Level Context Notifier, but then it becomes much higher. Therefore, the Context Synchronizer is the bottle neck of the HLCA, with a clear impact on the evolution of the time required for the context identification.

The relevance of the time invested by the HCLA to recognize a high-level context fairly depends on the application domain. Thus for example, if an alert has to be sent right away or a prompt action be taken based on the detected high-level context, then this time might be arguably long. However, if the identified information is rather used for analyzing the trajectories of behavior over time, then this time turns to be hardly relevant. Under these considerations, the processing time for the recognition of high-level contexts could be the main limitation of the actual implementation of the HLCA and should be improved in future work. A potential solution could consist in introducing a cache system into the High-Level Context Builder to save temporarily only the latest instances of low-level context and periodically persist them into the Context Manager. With such a solution the Context Synchronizer would not need to interact with the Context Manager and could pose the SPARQL queries directly to the cache; thus, retrieving the low-level context instances from a much smaller store. The Context Mapper, also part of the High-Level Context Builder, could share this very cache with the Context Synchronizer and increase its performance as well. If the cache would prove to be a good solution, such a system could also be introduced in the Context Notifier. This component has a similar behavior than the Context Mapper and its processing time could be reduced as well. Alternate solutions for accelerating the processing time for the identification of high-level contexts could include parallelizing tasks, defining different levels of cache-memory or simply scaling the infrastructure through cloud-based services.

Finally, Figure 10 depicts the size of the Context Storage in the Context Manager increasing lineally with the number of stored high-level context instances. The initialization of the Context Storage, i.e., storing the terminology defining the Mining Minds Context Ontology, requires only 408.5 KB on disc. The storage of each new high-level context instance, which has associated the storage of the low-level context instance which triggered its creation, increases the size of the Context Storage in 17 KB, in average. Thus, for the previous simulation of 250,000 changes in the context, which leads to a total of 500,000 context instances on disc (i.e., 250,000 high-level context instances and 250,000 low-level contexts instances), the Context Storage reached a size of 4.06 GB. Despite the Context Manager proves to fairly handle this volume of data, the increasing time observed for I/O operations in long-term scenarios with several users demands for some of the aforementioned solutions.

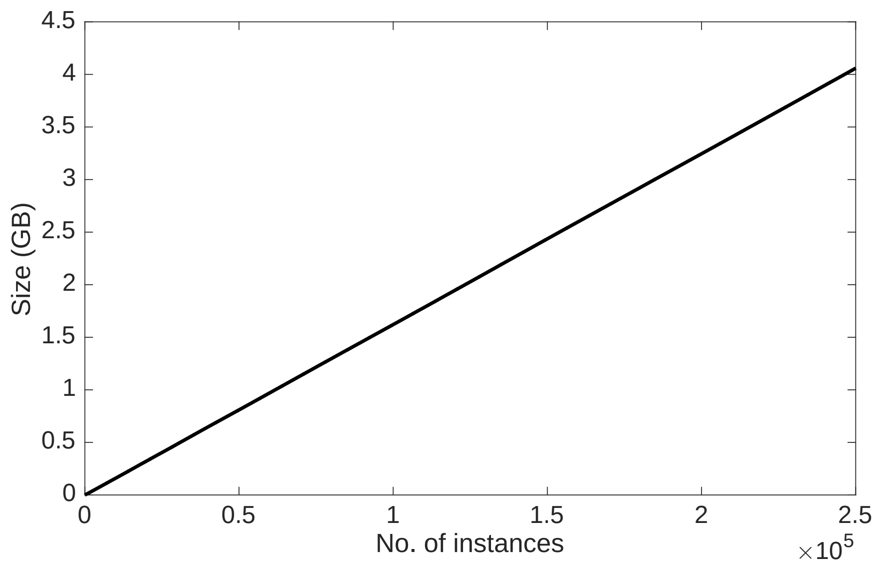


Figure 10. Size of the Context Storage depending on the number of persisted instances of high-level context. It must be noted that the storage of each high-level context instance has associated the storage of the low-level context instance which triggered its creation. Thus, for example, 250,000 instances in the X-axis represent 250,000 high-level contexts plus 250,000 low-level contexts stored on disc.

6. Conclusions

This work has presented an ontology-based method for deriving high-level context information out of the combination of cross-domain low-level context primitives, namely activities, locations and emotions. The modeling of the low and high-level contexts is defined through the so-called Mining Minds Context Ontology, while the processing and inference of contexts is performed by the Mining Minds High-Level Context Architecture, both contributions of this work. The Mining Minds Context Ontology has been designed to support any combination of low-level contexts to define a specific high-level context. The unprecedented incorporation of emotions in the context definition enables the representation of new high-level contexts that can only be identified whenever a specific emotion takes place. The Mining Minds Context Ontology has also been designed to procure the identification of some high-level contexts even in the absence of emotion information. The Mining Minds High-Level Context Architecture builds on the Mining Minds Context Ontology and reasoning techniques to enable the inference of high-level context from low-level context primitives in real time. The evaluation of the implemented architecture proves the reasonably good robustness properties of the Mining Minds Context Ontology against potentially erroneous low-level contexts. In fact, the results have showed that the impact of the error introduced in the low-level context is always lower at the high-level and that the activity is the most prevalent category in terms of error impact, while the location and especially the emotion tend to show a lesser effect on the high-level context. The current prototype implementation of the Mining Minds High-Level Context Architecture has been proven to perform well with respect to processing time and management of context instances. However, in order to ensure the scalability of the Mining Minds High-Level Context Architecture, the synchronization and database transactions management needs to be improved. Future work includes modifications in the database management to accelerate the inference time, the evaluation of this architecture with real users and the evolution of the Mining Minds Context Ontology in order to include more types of low-level context and new identifiable high-level contexts.

Acknowledgments: This work was supported by the Industrial Core Technology Development Program (10049079, Develop of mining core technology exploiting personal big data) funded by the Ministry of Trade, Industry and Energy (MOTIE, Korea). This work was also partially supported by the Spanish Ministry of Economy and Competitiveness (MINECO) Project TIN2015-71873-R together with the European Fund for Regional Development (FEDER).

Author Contributions: C.V. and O.B. are the principal researchers and main authors of this work. C.V. has designed the ontology and implemented the architecture. C.V. and O.B. have written the paper. M.A.R. and W.A.K. have contributed to the ideation phase. H.P., I.R. and S.L. have reviewed the manuscript for scientific content. All authors have read and approved the final manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Ke, S.R.; Thuc, H.L.U.; Lee, Y.J.; Hwang, J.N.; Yoo, J.H.; Choi, K.H. A review on video-based human activity recognition. *Computers* **2013**, *2*, 88–131.
2. Lara, O.D.; Labrador, M.A. A survey on human activity recognition using wearable sensors. *IEEE Commun. Surv. Tutor.* **2013**, *15*, 1192–1209.
3. Fallah, N.; Apostolopoulos, I.; Bekris, K.; Folmer, E. Indoor human navigation systems: A survey. *Interact. Comput.* **2013**, *25*, 21–33.
4. Fasel, B.; Luettin, J. Automatic facial expression analysis: A survey. *Pattern Recognit.* **2003**, *36*, 259–275.
5. Koolagudi, S.G.; Rao, K.S. Emotion recognition from speech: A review. *Int. J. Speech Technol.* **2012**, *15*, 99–117.
6. Liao, L.; Fox, D.; Kautz, H. Extracting places and activities from gps traces using hierarchical conditional random fields. *Int. J. Robot. Res.* **2007**, *26*, 119–134.
7. Banos, O.; Damas, M.; Pomares, H.; Prieto, A.; Rojas, I. Daily living activity recognition based on statistical feature quality group selection. *Expert Syst. Appl.* **2012**, *39*, 8013–8021.

8. Pan, Y.; Shen, P.; Shen, L. Speech emotion recognition using support vector machine. *Int. J. Smart Home* **2012**, *6*, 101–108.
9. Liming, C.; Nugent, C.; Okeyo, G. An Ontology-Based Hybrid Approach to Activity Modeling for Smart Homes. *IEEE T. Hum. Mach. Syst.* **2014**, *44*, 92–105.
10. Khanna, P.; Sasikumar, M. Rule Based System for Recognizing Emotions Using Multimodal Approach. *Int. J. Adv. Comput. Sci. Appl.* **2013**, *4*, 7.
11. Bae, I.H. An ontology-based approach to ADL recognition in smart homes. *Future Gener. Comput. Syst.* **2014**, *33*, 32–41.
12. BakhshandehAbkenar, A.; Loke, S.W. MyActivity: Cloud-Hosted Continuous Activity Recognition Using Ontology-Based Stream Reasoning. In Proceedings of the IEEE International Conference on Mobile Cloud Computing, Services, and Engineering, Oxford, UK, 8–11 April 2014.
13. Riboni, D.; Bettini, C. OWL 2 Modeling and Reasoning with Complex Human Activities. *Pervasive Mob. Comput.* **2011**, *7*, 379–395.
14. Banos, O.; Bilal-Amin, M.; Ali-Khan, W.; Afzel, M.; Ali, T.; Kang, B.H.; Lee, S. The Mining Minds Platform: A Novel Person-Centered Digital Health and Wellness Framework. In Proceedings of the 9th International Conference on Pervasive Computing Technologies for Healthcare, Istanbul, Turkey, 20–23 May 2015.
15. Banos, O.; Bilal Amin, M.; Ali Khan, W.; Afzal, M.; Hussain, M.; Kang, B.H.; Lee, S. The Mining Minds digital health and wellness framework. *BioMed. Eng. OnLine* **2016**, *15*, 165–186.
16. Baldauf, M.; Dustdar, S.; Rosenberg, F. A survey on context-aware systems. *Int. J. Ad Hoc Ubiquitous Comput.* **2007**, *2*, 263–277.
17. Bettini, C.; Brdiczka, O.; Henricksen, K.; Indulska, J.; Nicklas, D.; Ranganathan, A.; Riboni, D. A survey of context modelling and reasoning techniques. *Pervasive Mob. Comput.* **2010**, *6*, 161–180.
18. Perera, C.; Zaslavsky, A.; Christen, P.; Georgakopoulos, D. Context Aware Computing for The Internet of Things: A Survey. *IEEE Commun. Surv. Tutor.* **2014**, *16*, 414–454.
19. Chen, H.; Finin, T.; Joshi, A. The SOUPA Ontology for Pervasive Computing. In *Ontologies for Agents: Theory and Experiences*; Tamma, V., Cranefield, S., Finin, T., Willmott, S., Eds.; BirkHauser: Basel, Switzerland, 2005; pp. 233–258.
20. Wang, X.H.; Zhang, D.Q.; Gu, T.; Pung, H.K. Ontology based context modeling and reasoning using OWL. In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops, Orlando, FL, USA, 14–17 March 2004; pp. 18–22.
21. Chen, H.; Finin, T.; Joshi, A. An ontology for context-aware pervasive computing environments. *Knowl. Eng. Rev.* **2003**, *18*, 197–207.
22. Gu, T.; Pung, H.K.; Zhang, D.Q. A Middleware for Building Context-Aware Mobile Services. In Proceedings of IEEE Vehicular Technology Conference (VTC), Milan, Italy, 17–19 May 2004.
23. Hervás, R.; Bravo, J.; Fontecha, J. A Context Model based on Ontological Languages: A Proposal for Information Visualization. *J. UCS* **2010**, *16*, 1539–1555.
24. Poveda Villalon, M.; Suárez-Figueroa, M.C.; García-Castro, R.; Gómez-Pérez, A. A context ontology for mobile environments. In Proceedings of the Workshop on Context, Information and Ontologies (CEUR-WS), Lisbon, Portugal, 11–15 October 2010.
25. Agostini, A.; Bettini, C.; Riboni, D. Hybrid Reasoning in the CARE Middleware for Context Awareness. *Int. J. Web Eng. Technol.* **2009**, *5*, 3–23.
26. Riboni, D.; Bettini, C. COSAR: Hybrid Reasoning for Context-aware Activity Recognition. *Pers. Ubiquitous Comput.* **2011**, *15*, 271–289.
27. Okeyo, G.; Chen, L.; Wang, H.; Sterritt, R. Dynamic sensor data segmentation for real-time knowledge-driven activity recognition. *Pervasive Mobile Comput.* **2014**, *10*, 155–172.
28. Helaoui, R.; Riboni, D.; Stuckenschmidt, H. A Probabilistic Ontological Framework for the Recognition of Multilevel Human Activities. In Proceedings of the 2013 ACM International Joint Conference on Pervasive and Ubiquitous Computing (UbiComp '13), Zurich, Switzerland, 8–12 September 2013; ACM: New York, NY, USA, 2013; pp. 345–354.
29. Rodriguez, N.D.; Cuellar, M.P.; Lilius, J.; Calvo-Flores, M.D. A fuzzy ontology for semantic modelling and recognition of human behaviour. *Knowl. Based Syst.* **2014**, *66*, 46–60.
30. Korpipaa, P.; Mantyjarvi, J.; Kela, J.; Keranen, H.; Malm, E.J. Managing context information in mobile devices. *IEEE Pervasive Comput.* **2003**, *2*, 42–51.

31. Strang, T.; Linnhoff-Popien, C. A context modeling survey. In Proceedings of the Sixth International Conference on Ubiquitous Computing, Workshop on Advanced Context Modelling, Reasoning and Management, Nottingham, UK, 7–10 September 2004.
32. Chen, H.; Finin, T.; Joshi, A.; Kagal, L.; Perich, F.; Chakraborty, D. Intelligent agents meet the semantic Web in smart spaces. *IEEE Int. Comput.* **2004**, *8*, 69–79.
33. Dey, A.K. Understanding and Using Context. *Pers. Ubiquitous Comput.* **2001**, *5*, 4–7.
34. W3C OWL Working Group. OWL 2 Web Ontology Language: Document Overview (Second Edition); W3C Recommendation, 11 December 2012. Available online: http://www.w3.org/TR/owl2-overview/ (accessed on 10 March 2016).
35. Mining Minds Context Ontology Version 2. Available online: http://www.miningminds.re.kr/icl/context/context-v2.owl (accessed on 10 March 2016).
36. Peterson, D.; Gao, S.; Malhotra, A.; Sperberg-McQueen, C.M.; Thompson, H.S. W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes; W3C Recommendation, 5 April 2012. Available online: http://www.w3.org/TR/xmlschema11-2/ (accessed on 10 March 2016).
37. Protégé. Available online: http://protege.stanford.edu/ (accessed on 10 March 2016).
38. Sirin, E.; Parsia, B.; Grau, B.C.; Kalyanpur, A.; Katz, Y. Pellet: A practical OWL-DL reasoner. *J. Web Semant.* **2007**, *5*, 51–53.
39. Harris, S.; Seaborne, A. SPARQL 1.1 (SPARQL Query Language for RDF); W3C Recommendation, 21 March 2013. Available online: http://www.w3.org/TR/sparql11-query/ (accessed on 10 March 2016).
40. Apache Jena. Available online: https://jena.apache.org/ (accessed on 10 March 2016).
41. Brickley, D.; Guha, R.V. RDF Schema 1.1; W3C Recommendation, 25 February 2014. Available online: https://www.w3.org/TR/rdf-schema/ (accessed on 10 March 2016).
42. Richardson, L.; Ruby, S. Restful Web Services. Available online: http://transform.ca.com/LAC-Creating-RESTful-web-services-restful-interface.html?mrm=508752&cid=NA-SEM-API-ACF-000011-00000946-000000110 (accessed on 10 March 2016).

Section 1.5

Service Curation Layer (SCL) Paper

Computers in Biology and Medicine 69 (2016) 10–28

Contents lists available at ScienceDirect

Computers in Biology and Medicine

journal homepage: www.elsevier.com/locate/cbm

Multimodal hybrid reasoning methodology for personalized wellbeing services

Rahman Ali [a], Muhammad Afzal [a], Maqbool Hussain [a], Maqbool Ali [a], Muhammad Hameed Siddiqi [a], Sungyoung Lee [a,*], Byeong Ho Kang [b]

[a] Department of Computer Engineering, Kyung Hee University, Seocheon-dong, Giheung-gu, Yongin-si 446-701, Gyeonggi-do, Republic of Korea
[b] Department of Computing and Information Systems, University of Tasmania, Hobart, Tasmania 7005, Australia

A R T I C L E I N F O

Article history:
Received 10 August 2015
Accepted 24 November 2015

Keywords:
Multimodal reasoning
Hybrid reasoning
Case-based reasoning (CBR)
Hybrid-CBR
Rule-based reasoning (RBR)
Preference-based reasoning (PBR)
Physical activity recommendation
Wellness services

A B S T R A C T

A wellness system provides wellbeing recommendations to support experts in promoting a healthier lifestyle and inducing individuals to adopt healthy habits. Adopting physical activity effectively promotes a healthier lifestyle. A physical activity recommendation system assists users to adopt daily routines to form a best practice of life by involving themselves in healthy physical activities. Traditional physical activity recommendation systems focus on general recommendations applicable to a community of users rather than specific individuals. These recommendations are general in nature and are fit for the community at a certain level, but they are not relevant to every individual based on specific requirements and personal interests. To cover this aspect, we propose a multimodal hybrid reasoning methodology (*HRM*) that generates personalized physical activity recommendations according to the user's specific needs and personal interests. The methodology integrates the rule-based reasoning (*RBR*), case-based reasoning (*CBR*), and preference-based reasoning (*PBR*) approaches in a linear combination that enables personalization of recommendations. *RBR* uses explicit knowledge rules from physical activity guidelines, *CBR* uses implicit knowledge from experts' past experiences, and *PBR* uses users' personal interests and preferences. To validate the methodology, a weight management scenario is considered and experimented with. The *RBR* part of the methodology generates goal, weight status, and plan recommendations, the *CBR* part suggests the top three relevant physical activities for executing the recommended plan, and the *PBR* part filters out irrelevant recommendations from the suggested ones using the user's personal preferences and interests. To evaluate the methodology, a *baseline-RBR* system is developed, which is improved first using ranged rules and ultimately using a *hybrid-CBR*. A comparison of the results of these systems shows that *hybrid-CBR* outperforms the *modified-RBR* and *baseline-RBR* systems. *Hybrid-CBR* yields a 0.94% recall, a 0.97% precision, a 0.95% *f*-score, and low *Type I* and *Type II* errors.

1. Introduction

An individual's healthy lifestyle impacts the overall health of a population and results in a healthier society [1]. Without a healthy lifestyle, *i.e.,* proper diet, exercise, and controlled body mass index, individuals are prone to various diseases [2] that include lifestyle as an important cofactor [3]. Adopting physical activity is one of the key responses of individuals that helps in promoting a healthier lifestyle [4]. Similarly, wellness guidelines and automatic wellness recommendation systems play roles in public health promotion. These systems provide support for wellness experts in recommending the appropriate physical activity to individuals according to their personal requirements [5]. A healthier lifestyle involves a balanced combination of physical activity, mental behavior, and social interaction with other community members [6–8]. In this study, we focus on the physical activity aspect of a healthier lifestyle. We also focus on the development of a physical activity recommendation system to motivate users to keep their life active by involving themselves in various types of physical activities. Traditional physical activity recommendation systems provide general guidelines in the form of recommendations, which do not provide user-centric recommendations. To fulfill the

* Corresponding author. Tel.: +82 1073451441; fax: +82 31 202 2520.
E-mail addresses: rahmanali@oslab.khu.ac.kr (R. Ali), muhammad.afzal@oslab.khu.ac.kr (M. Afzal), maqbool.hussain@oslab.khu.ac.kr (M. Hussain), maqbool.ali@oslab.khu.ac.kr (M. Ali), siddiqi@oslab.khu.ac.kr (M.H. Siddiqi), sylee@oslab.khu.ac.kr (S. Lee), byeong.kang@utas.edu.au (B. Ho Kang).

http://dx.doi.org/10.1016/j.compbiomed.2015.11.013

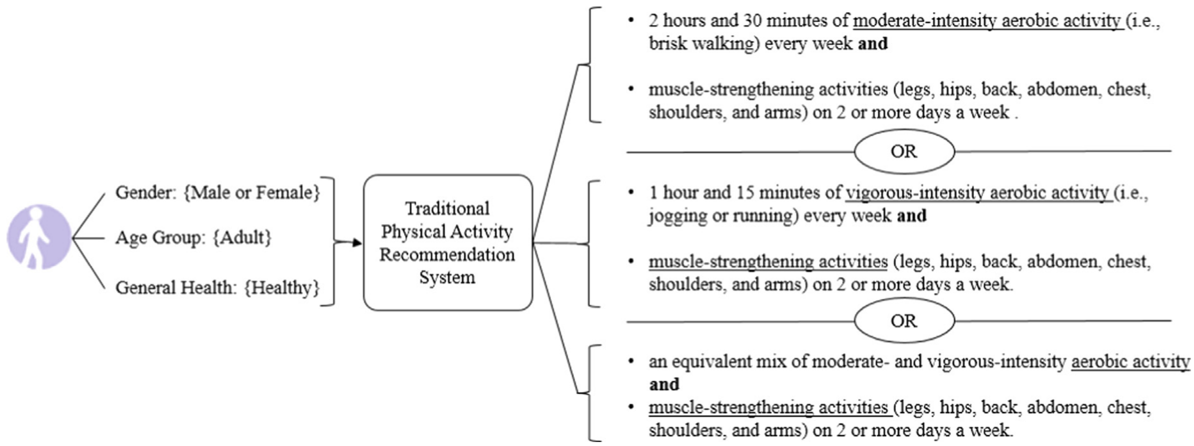


Fig. 1. General physical activity recommendations adopted from the Center for Disease Control and Prevention (CDC) guidelines [9].

personal needs of different users, a personalized physical activity recommendation system is required.

We illustrate the concept of personalized physical activity recommendation using an example in which an overweight, 30-year-old person affected with asthma is interested in personalized physical activity recommendation. The goal is to recommend an appropriate physical activity to this person according to his health needs as well as personal interests. If the recommended activity reflects his requirements, then it will be accepted; otherwise, it will be rejected. Existing physical activity recommendations, proposed by *CDC* [9], *WHO* [10], *AHA* [7], among others, recommend general physical activity for the whole community of users. These recommendations are abstract and exploit limited personal information of the users. An example of the *CDC* recommendations is shown in Fig. 1.

Fig. 1 shows that gender, age and health conditions are taken into consideration while suggesting options of physical activities. The following important questions arise:

A. Are the suggested recommendations appropriate for the person considered in the example?
B. Are the provided recommendations based on the user personal information (*e.g., BMI*), health, physical activities routines and preference list?

The answers to these questions are ‘no’, which mean that the system provides general guidelines and the user has to look into his personal information, daily routines, and preferences and choose appropriate physical activity for himself. Generally, this should not be the case and the recommendations shall reflect the person's specific needs. The system needs to be intelligent enough to first reason on the user personal profile information and calculate the user's weight status, target weight (goal status), and plan to achieve the goal. Based on these assessments, appropriate physical activity should be recommended according to the user current and past routines of activities and preference list.

To achieve the above stated goal, we are working on a personalized wellness platform called Mining Minds[1] [11] (see Section 3). Mining Minds is a collection of services, tools, and techniques for collaboratively investigating and analyzing the user's personal profile and daily routines for providing personalized wellbeing services. These services are generated by executing knowledge rules using the Mining Minds (*MM*) reasoning engine. This study focuses on the reasoning methodology adopted by the reasoning engine to generate daily physical activity recommendations. A multimodal hybrid reasoning methodology (*HRM*) is proposed, which plays an important role in interpreting the user's profile, physical activity routines, and personal preferences for generating personalized physical activity recommendations. *HRM* integrates the rule-based reasoning (*RBR*), case-based reasoning (*CBR*), and preference-based reasoning (*PBR*) methodologies for enabling the reasoning engine to personalize the recommendations. *RBR* of the proposed *HRM* exploits domain knowledge rules extracted from guidelines, *CBR* exploits implicit knowledge obtained from experts' past experience (successful cases), and *PBR* exploits users' personal preferences and interests to ensure accurate and personalized recommendations. The key ideas of *HRM* include the following: (i) exploitation of the diverse knowledge sources for personalized wellbeing recommendations using the integration of multiple reasoning methodologies, such as *RBR*, *CBR* and *PBR* in a linear combination to form *HRM*, (ii) reducing the bottlenecks of traditional single reasoning methodologies, which exploit only single knowledge sources for generating a single service at a time and (iii) enabling the generation of specific, relevant and personalized physical activity recommendations according to the user's specific requirements.

To validate the proposed *HRM*, a weight management scenario is considered, and a set of experiments are performed. The use of *HRM* for weight management is an innovative idea that guarantees specific and precise personalized physical activity recommendations. It is important to mention that our prescription of physical activities only focuses on healthy adults and not on people with disabilities, women who are pregnant and people who have medical complications.

The rest of the paper is structured as follows. Previous research is summarized in Section 2. In Section 3, an overview of the *MM* platform is provided. In Section 4, the proposed *HRM* is discussed from architectural, knowledge acquisition and reasoning perspectives. In Section 5, the experiments are performed, and the system is evaluated based on a weight management scenario. In Section 6, a discussion on the methodological aspects of the paper, different challenges faced and limitations of the approach is provided. Section 7 concludes the work performed and outlines some possible future extensions. Section 8 acknowledges the contributors and financial sponsors.

2. Related work

Human experts are limited in number and expensive in terms of healthcare and wellness services provided. Healthcare decision

[1] http://www.miningminds.re.kr/

support systems play effective roles in overcoming the shortage of human experts and improving quality of life with better services [12]. Decision support systems rely on automatic reasoning methodology for their decisions. Most of these systems are based on a single methodology for reasoning, such as *CBR* or *RBR* [13], among others. Nevertheless, a few use multiple reasoning approaches with a certain integration strategy. The integration of multiple reasoning methodologies in a single system has attracted increased attention in the research community due to the improved performance with respect to accuracy. The analogy of integration of reasoning methodologies is adopted from the decisions made by domain experts, who rely on multiple knowledge sources rather than a single source. Domain experts use information from general guidelines, clinical trials, and past successful cases to arrive at a final decision. In automatic reasoning systems, the concept of multimodal reasoning methodology evolved from the use of heterogeneous knowledge sources to generate the final decision [13]. The knowledge source, such as guidelines and past successful cases are modeled as knowledge rules and case bases that require *RBR* and *CBR* for their executions.

The integration of reasoning approaches can follow any set of strategies, such as *RBR* followed by *CBR*, *CBR* followed by *RBR* and *RBR* and *CBR* in parallel [13,14]. In the first strategy, *RBR* is used as the main methodology for making the decision. If *RBR* fails, CBR is used [15]. In the second strategy, *CBR* is used for the master reasoning process and *RBR* is used to refine the decision [16]. An example of this strategy is reasoning system for diabetes management [17]. The *CBR* refines the rules for the final outcome, specific to the patient's requirements. In other combinations, *CBR* and *RBR* are used in parallel, where either both outcomes are simply displayed or the best one is displayed based on some criteria. An example of parallel integration is the *WHAT* system [18,19], which is used for training beginning sports medicine students to design exercise regimens for patients with cardiac or pulmonary disorders. The regimens are produced by *RBR* and *CBR* in parallel and presented to the experts for choosing the best one. Other methodologies exist that closely cooperate with each other for generating final decisions [20,21]. Apart from *RBR* and *CBR*, filtration-based approaches, such as content-based filtration [22] and collaborative filtration [23,24] are also popular in the area of recommender systems for online shopping, product selection, and healthcare services. Preference-based recommender systems are used in e-applications such as e-commerce to offer alternative or cross-selling products to customers [25].

In the healthcare domain, hybrid reasoning approaches have been frequently used. In treatment planning for adolescent early intervention, hybrid *CBR* that uses *RBR* and fuzzy theory has been implemented [26]. For supporting physicians for the management of diabetes mellitus, integration of *CBR*, *RBR* and model-based reasoning (*MBR*)[27] and web-based *CBR* [28] has been proposed. For cancer decision support services, *CBR* has been integrated with *RBR*. The *CBR* part is used to adapt the production rules for decision making [21]. A recent study [29] integrates rough set theory and correlation analysis in a hybrid model, called *H2RM*, that predicts the diabetes type and manages patient observations for future trend analyses. Other similar studies can be found that focus on heart disease [12] and oncology [13], among others.

In the wellness field, the knowledge acquisition and reasoning engine (*KARE*) [5] is used in activity awareness for human-engaged wellness applications (*ATHENA*) [6] to promote active lifestyles. *KARE* uses the hybrid reasoning methodology by integrating the Random Forest, Naïve Bayes, and IB1 approaches. *KARE* generates food, physical activity, and music therapy recommendations for *ATHENA* users. For the elderly, an intelligent personalized exercise recommendations system is proposed [30] that utilizes the user's health status, goals and preference information. Similarly, a hybrid *CBR*/*RBR* approach has successfully been used for designing nutritional menus [31].

All of these methodologies have the common basis of being used in an exclusive manner. They do not guarantee a minimization of the shortcomings of *RBR* and *CBR*, which are discussed as follows:

- Conventional *RBR* systems lack the capability of specializing recommendations for individuals. In general, to deal with specific requirements of users and provide user-centric specialized recommendations, it is necessary to gradually increase the number of rules in the knowledge base. This approach not only results in knowledge base intractability problem, but also causes maintenance and combinatorial explosion issues [32].
- Standard *CBR* systems provide solutions for new problems using a large and unbiased case base as implicit knowledge. However, the requirement of a large case base is a difficult task and associated with a number of other issues, such as physical storage, proper indexing and computational complexities [33]. The preparation of the query cases to feed the *CBR* cycle for generating physical activity recommendations is a challenging task.
- There have been significant improvements in the integration of these methodologies in hybrid systems [34]; however, a number of challenging issues still need to be resolved for applying integration in the wellness domain.

The proposed *HRM* mitigates these problems by integrating *RBR*, *CBR*, and *PBR* in a sequential manner that exploits guideline rules, past successful experience cases and the personal preferences of users to enable personalization of recommendations.

3. Overview of the Mining Minds platform

Advancement in technology greatly impacts the means of service provisioning to the community by employing innovative and state-of-the-art techniques. This includes handling real-time data streaming by utilizing a big data infrastructure with cloud data storage and processing abilities. Our indigenously developed *MM* platform [11,35] provides a comprehensive picture of the usage of these technologies for monitoring users and collecting information that can facilitate the use of healthcare applications on a global scale. An abstract design view of the *MM* healthcare platform is shown in Fig. 2.

The overall *MM* platform is divided into four layers: *data curation layer (DCL), information curation layer (ICL), service curation layer (SCL)* and *supporting layer (SL)*. The *DCL* is responsible for curating the data. It consists of different modules for data streaming and communication, data representation and mapping and big data storage in a *Hadoop Distributed File System (HDFS). HDFS* addresses the volume, velocity and variety aspects of raw sensory data acquired using mobile sensors. The accelerometer raw data for low-level activities (*i.e.,* sitting, standing, moving in a bus, moving in a subway, walking, running, and cycling) are transferred to the *DCL* virtual machine, which is transformed to have a structured format and stored in a relational data model on the *DCL* server machine. The mobile device used in this case works as a gateway to connect to the *DCL* cloud server over the Internet. The stored data are fed to the *ICL* for activity recognition that leads to context formulation and behavior analysis of the users' daily activities. The information is stored back in the *HDFS* logs of the *DCL*. The processed activities, context, behavior information, and personal profile information are utilized by the *SCL* for reasoning and providing personalized physical activity recommendations. In *SCL*, knowledge bases are created by domain experts based on the online guidelines and experts' past experiences. This enables the process of provisioning personalized recommendations to users

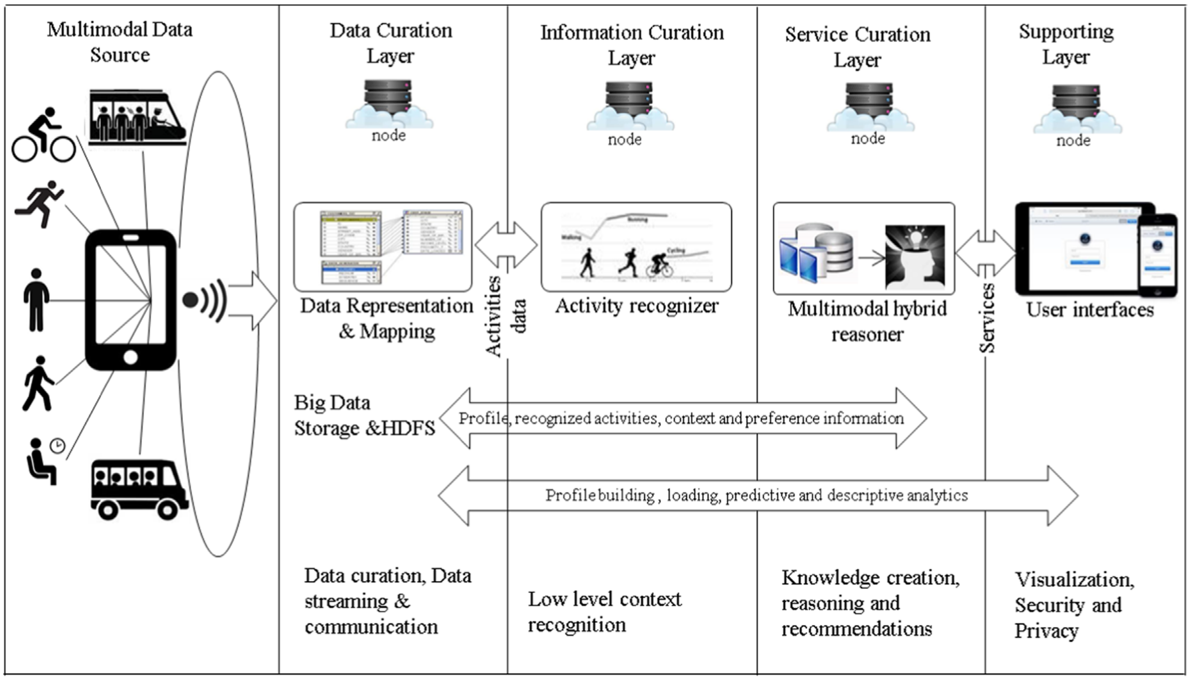


Fig. 2. Overview of the Mining Minds healthcare and wellness platform.

based on their needs, preferences, and interests. SL facilitates other layers by providing security, privacy, visualization and user interfaces. The user's personal profile information is collected using a mobile application and stored on the *DCL* server in a relational data model.

A multimodal hybrid reasoner is a key component of *MM* and plays the role of an intelligent service provisioning agent. It performs execution on the server side of the *SCL* and enables personalization of physical activity recommendations by integrating data and knowledge from diverse sources. The focus of this paper is on the reasoning methodology and its usefulness in *MM* for generating personalized physical activity recommendations.

4. Multimodal hybrid reasoning methodology (*HRM*)

For building an intelligent physical activity recommendation system, we moved beyond the traditional single reasoning methodology systems to a multiple reasoning methodology system. Our work integrates *RBR* and *CBR* with *PBR* into a single methodology called *multimodal hybrid reasoning methodology (HRM)*. *HRM* forms the basis of multimodal hybrid reasoner for the *MM* platform, which is the focus of this study. In *HRM*, these methodologies can be integrated in any of the following design strategies, shown in Fig. 3.

In Fig. 3(a–c), the sequence of the design strategy of *baseline-RBR* is as follows: level-1 *RBR* is followed by level-2 *RBR*, which is followed by level-3 *RBR* and *PBR*. The design strategy of the *modified-RBR* follows the same strategy as the *baseline-RBR*, except for the ranged-*METs*[2] rules, which are used at the level-3 *RBR*. The strategy of *hybrid-CBR* differs from those of the first two strategies at level-3, where *CBR* is used instead of *RBR*. In our study, we use the first strategy for building a baseline system to compare the results of the other strategies. The second strategy is the improved version of strategy 1, which is implemented in *MM* system (v1.0) but has its own limitations. To eradicate the shortcomings of the first two strategies, the third strategy of *hybrid-CBR* is used, which integrates *RBR*, *CBR,* and *PBR*. This strategy is experimented and realized outside the *MM* platform on a local set up in our lab.

Based on the idea illustrated above, we have defined the core components of the proposed multimodal hybrid reasoner and depicted them in the functional flow diagram shown in Fig. 4.

Fig. 4 shows high-level interactions of the different components of the reasoner along with the methodology used in each component. Like any other reasoning system, the core components of the proposed reasoner include the following: input/output interfaces, input data sources, knowledge bases, reasoning methodology and outputs. They are explained below as follows.

- *Input/output interfaces*: user's smart phone that runs the *MM* application works as the input/output interface for the reasoner.
- *Input data sources*: inputs of the reasoner include user requests, personal profile data, and daily physical activity data. The input data, except for the user requests, are stored in an intermediate database. The request for recommendation is received from the user's mobile application.
- *Knowledge base*: knowledge of the reasoner is composed of rules created from physical activity guidelines and past successful cases obtained from the implicit experience of the domain experts. The rules are stored in the rule base, while the past successful cases (*METs* index) are stored in the *METs case base (METCB).*
- *Reasoning methodology*: the reasoning methodologies include *RBR*, *CBR*, and *PBR*, which are integrated in a linear combination. The *RBR* methodology is applied at multiple levels: level-1, level-2, and level-3. At level-3, *RBR* is either used with distinct-*METs* rules or with ranged-*METs* rules. At the same level, *CBR* can also be used (using *METs* cases) as a counterpart of *RBR* for improved services. At the end, the multi-level filtration mechanism is applied in *PBR* to filter out irrelevant recommendations by utilizing the user's preferences and interests.
- *Outputs*: outputs of the reasoner include wellbeing recommendations for users, weight status, weight management plans, personalized physical activity recommendations and personalized filtered physical activity recommendations. These

[2] A metabolic equivalent, or *METs*, is a unit used to describe the energy expenditure of a specific physical activity. A *METs* is the ratio of the rate of energy expended during an activity to the rate of energy expended at rest (2008 Physical Activity Guidelines for Americans).

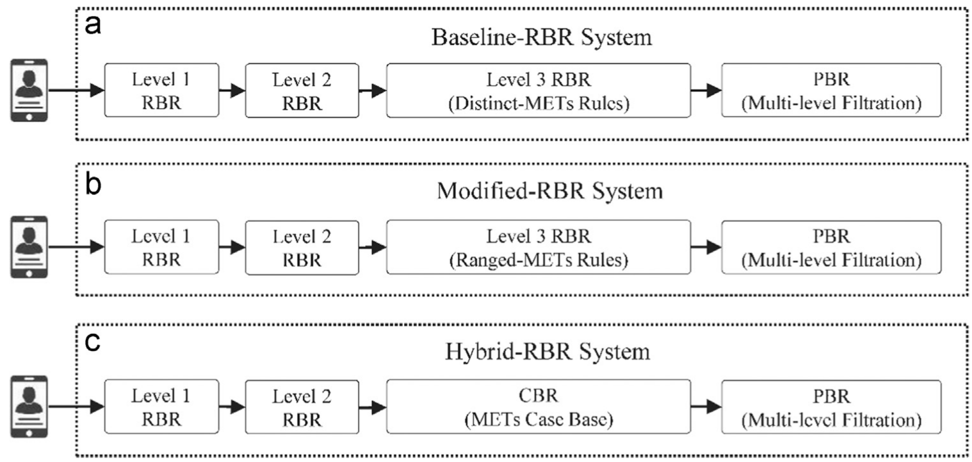


Fig. 3. Multiple design views of the proposed hybrid reasoning methodology on the basis of integration of different reasoning methodologies.

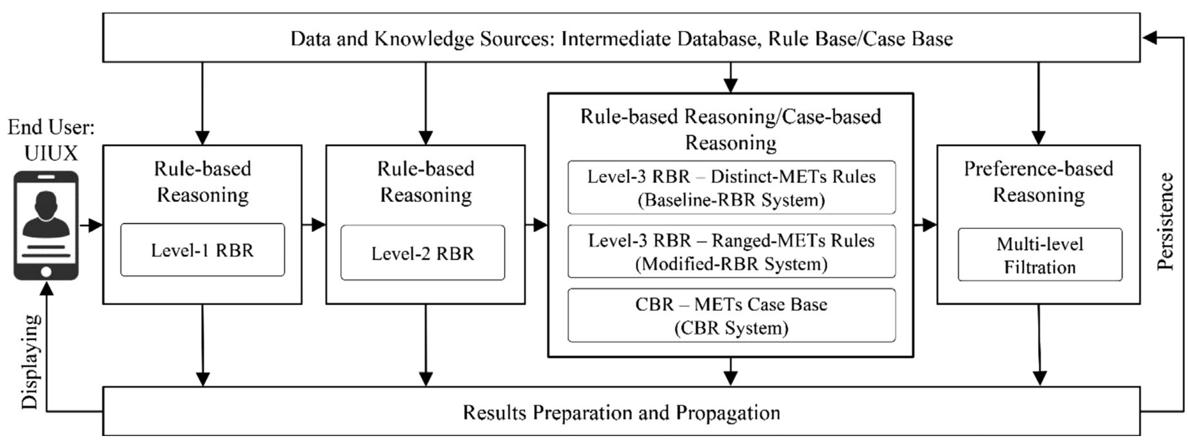


Fig. 4. Functional flow diagram of the proposed multimodal hybrid reasoning engine.

recommendations are aggregated and prepared by *Results Propagator* and then delivered to the end user and intermediate database. In the intermediate database, it is stored for future use as a successful case.

In the subsequent sub-sections, a detailed description of the architectural design of the proposed *HRM* is provided (Section 4.1), and then, the process of knowledge creation is discussed (Section 4.2); finally, the reasoning methodology is described in detail (Section 4.3).

4.1. Architectural design and workflow

A detailed data flow diagram of the multimodal hybrid reasoning engine illustrating communication is shown in Fig. 5.

The key components of the *HRM* are *service request handler (SRH), data loader and manipulator (DLM), knowledge base (KB), knowledge loader (KL), hybrid reasoner (HR)* and *result preparator and propagator (RP)*. The hybrid reasoner consists of *RBR*, *CBR*, and *PBR* modules along with the *PR* module. The *RBR*, *CBR*, and *PBR* modules work cooperatively in a linear combination for enhancing recommendations. *CBR* is the key reasoning methodology that is activated by the output of *RBR*. The output of *CBR* in turn activates the *PBR* methodology to personalize the recommended physical activity.

From the service execution perspective, when a user requests service, the *SRH* analyzes the request and activates the appropriate module of the reasoner. *SRH* supports the *MM* platform for multiple service generation. *SRH* forwards the request to *HR*, where the *RBR* (level-1, level-2), level-3 *RBR*/*CBR*, and *PBR* methodologies are sequentially executed. Outputs of the *HR* are forwarded to the *RP* module for final preparation and forwarded to the user mobile application interface (*UIUX*) for being displayed to the users.

For the weight management scenario, the multimodal hybrid reasoning methodology operates in the following flow.

- First, level-1 *RBR* is applied, which loads the *weight status rules (WSR)* (see Section 4.2) from the *KB* and the required personal profile data from the *intermediate database (IDB)* using the *data loader (DL)* component. The necessary computation on the personal data, *e.g.,* *BMI* calculation from height and weight information, is performed using the *data manipulator (DM)* and passed to the level-1 *RBR*. The level-1 *RBR* uses *RBR* methodology to recommend weight status recommendations (normal, overweight, underweight) as a service to the user and to level-2 *RBR* for further processing.
- Level-2 *RBR* receives the output of the level-1 *RBR* as input and performs the same reasoning procedure as level-1 *RBR* for recommending the goal state and associated calorie consumption plan and weight management plans. The level-2 *RBR* uses the *goal* and *plan recommendation rules (GPR)* loaded by *KL* from the *KB* and the personal profile data loaded by the *DL* from the *IDB*. The purpose is to generate goal and plan recommendations, which are provided to the users as a service and to level-3 *RBR*/*CBR* for further processing.
- Level-3 *RBR*/*CBR* receives the output of the level-2 *RBR* as input and further generates physical activity recommendations. Level-3

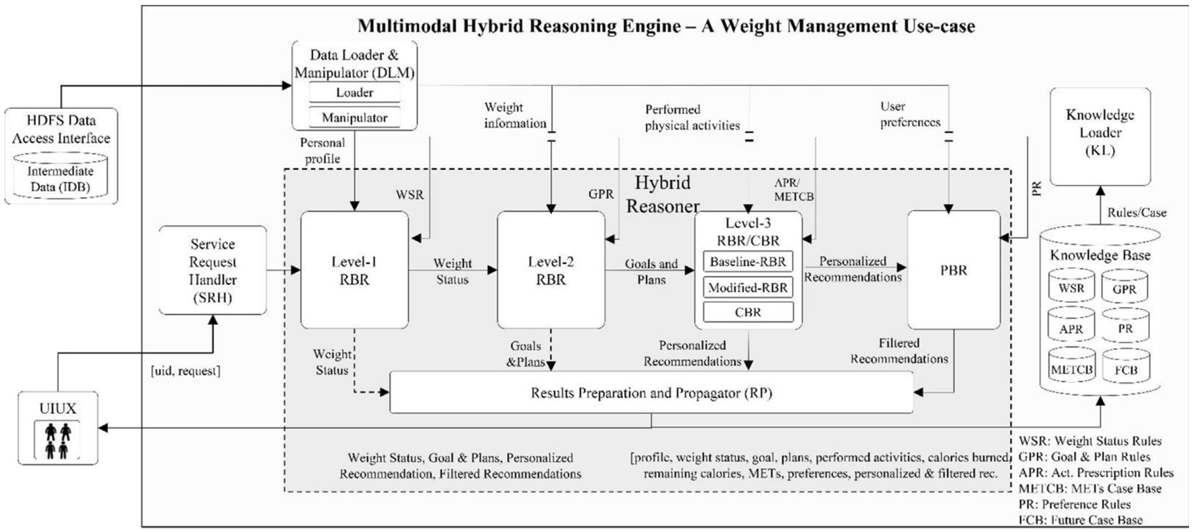


Fig. 5. Detailed data flow diagram of the proposed multimodal hybrid reasoning engine.

RBR/CBR supports both the *RBR* and *CBR* methodologies. The *RBR* results in *baseline-RBR* and *modified-RBR* systems. The *baseline-RBR* uses distinct-*METs* rules, while the *modified-RBR* uses ranged-*METs* rules that are loaded from the *KB* during the activity recommendation generation. The *CBR* methodology uses the *METCB* of the historical successful physical activity recommendations. In our case, we use the 2011 compendium of physical activity guidelines [36] as our key physical activity case base, which has physical activity recommendations associated with *METs* values. In either case (*i.e., baseline-RBR, modified-RBR* or *hybrid-CBR*), the list of all of the performed physical activities is loaded from the *IDB* and commutated for the duration, amount of consumed calories, remaining calories and corresponding metabolic equivalent (*METs*) value. The corresponding physical activities for the *METs* value are recommended and provided to the users. These physical activities are not filtered according to the preferences and interests of the users; therefore, they are forwarded to the *PBR* module for further personalization.

- *PBR* first receives the physical activities recommended by level-3 *RBR/CBR* and then loads the personal preferences and interests information from the *IDB*. The associated *preference-based rules (PR)* are loaded from the *KB* to apply multi-level filtration for filtering out irrelevant recommendations. The final filtered recommendations are personalized according to the user's personal preferences.
- The personalized recommendations are passed to the *RP* for proper preparation and packaging to be forwarded to the user application to be displayed on the user's mobile application.
- The user query, intermediate recommendations, and final personalized recommendations are stored in the *future case base (FCB)* for future use.

4.2. Knowledge acquisition

Knowledge is one of the most important ingredients of a reasoning system. This section describes how the knowledge used by the *HRM* is created. The key methodologies of *HRM* are *RBR* and *CBR*; therefore, we first need to create knowledge in the form of rules and cases. The process of knowledge acquisition is discussed below.

4.2.1. Rules base creation: translating guidelines

Wellness guidelines are the key source of information for improving the quality of life. Translating guidelines into computer-processable rules is a challenging task because it requires the involvement of knowledge engineers and domain experts [37]. In our study, knowledge from the unstructured guidelines of a weight management scenario is translated to rules with the help of *three* knowledge engineers and *two* domain experts. Based on the design of our study, the knowledge engineers first studied the weight management scenario, surveyed the weight management guidelines, indexed them, and categorized them into two groups: (a) standard equations to compute standard values and (b) indexes to be used in rule creation. An example of the first category is the calculation of calories burned/day, while an example of the second category is the *BMI* index. These rules are used by the *RBR* to generate physical activity recommendations. The process of guideline translation is described below.

4.2.1.1. Personal profile assessment. To classify users into underweight, normal or overweight states, personal profile assessment based on the standard *BMI* index is required [38]. The *BMI* index and personal profile information are combined together to form rules, which are shown in Table 1. For the *BMI* calculation, the standard *BMI* formula is used.

These rules are applicable for adults and used by level-1 *RBR* (see Section 4.3.1) for finding the weight status of the users.

4.2.1.2. Goal setting and plan management. A weight management system requires goals and the associated plans to achieve the goals. A goal can be either a local goal or global goal (*gloGoal*). A global goal is the final objective of the user to be achieved, while the local goal refers to a set of sub-goals to reach the global goal. For example, the total weight to be lost is considered a global goal, while weekly targets are considered local goals. To set a global goal in the context of the weight management scenario, first, an estimation of the ideal body weight (*idlWgt*) is required, which can be obtained using the Robinson JD [39] equation. The difference between the current weight (*curWgt*) and ideal weight

Table 1
Weight status rules (*WSR*) based on the standard Body Mass Index (*BMI*).

Gender	Age	*BMI* value	Weight status
M or F	> 20	< 18.5 kg/m^2	Underweight
M or F	> 20	> 18.5 and < 25 kg/m^2	Normal
M or F	> 20	> 25 and < 30 kg/m^2	Overweight

Table 2
Goals and weight management Plan Rules (GPR) for recommending the goal status and an appropriate plan.

Gender Male (M)/Female (F)	**Global Goal (gloGoal) - kg**	**Weight status (WS)**	**Plan prescription (PP)**
M or F	>0 (+ive)	Normal or Overweight	Weight Loss Plan (WLP): lose gloGoal(kg)
M or F	$=0$ (neutral)	Normal	Weight Maintenance Plan (WMP): motivational statements
M or F	<0 (−ive)	Underweight	Weight Gain Plan (WGP): gain gloGoal(kg)

yields the best estimation for the target goal in terms of the number of kg to be lost. The ideal body weight and global goal are computed using Eqs. 1 and 2.

$$idlWgt = 51.65\ kg + 1.85\ kg/inch\ over\ 5\ feet \quad (man)$$
$$idlWgt = 48.67\ kg + 1.65\ kg/inch\ over\ 5\ feet \quad (woman) \tag{1}$$

The ideal body weight is a debatable topic but has successfully been used in healthcare systems, such as drug dosage estimation [39] and cell transplantation [40]. Therefore, we have adopted it for the estimation of the global goal in our study.

$$gloGoal(kg) = curWgt(kg) - idlWgt(kg) \tag{2}$$

In our system, *gloGoal* by itself is a user service, but it is aimed towards devising plans for achieving the global goal. The rules defined for identifying appropriate plans, such as a weight loss plan, weight gain plan and weight maintenance plan *(GPR)*, are shown in Table 2.

In Table 2, we only focus on the first two cases.

Details of the suggested plan, *i.e.*, duration for achieving the global goal, can be computed using Eq. 3.

$$wghRedPlan(days) = roundup\left(\frac{7(days) * gloGoal(Kg)}{0.5(Kg)}\right) \tag{3}$$

In Eq. 3, a constant value of 0.5 represents the weight to be lost during one week. From this equation, local goals for weeks and months can be determined by subtracting a weight of 0.5 kg from the weight of the previous week (weekly plan). These plans can also be estimated in terms of the calories burnt (per day, per week, per month, *etc.*) using Eq. 4.

$$calToBurDay = \frac{gloGoal(kg) * Cal(1kgfat)}{wghRedPlan(days)} \tag{4}$$

In Eq. 4, *Cal* represents the number of calories equivalent to burning 1 kg of body fat.

All of these rules are used for setting the goal, devising plans, and managing weight and are used by level-2 *RBR*.

4.2.1.3. Physical activities assessment. Once a weight management plan is assessed, monitoring and recognition of the user's physical activities are required. Based on monitoring the previous day's activities, using the accelerometer sensor of the smartphone, the next day recommendations are planned. This process is performed in terms of the duration spent in each activity and the estimated amount of calories burnt. The amount of each activity (*amtAct*) is calculated by taking sum of all of the time slots (*timSlot*) during which the user performed that activity (*Act*), computed using Eq. 5.

$$amtAct_i = \sum_{j=1}^{t} Act_i.timSlot_j \tag{5}$$

The estimation of calories (*Cal*) for a specific activity (Act_i) in a specific time duration, $amtAct_i$, can be estimated by the product of the *METs* of that activity with the amount of activity and current weight of the subject. This calculation is shown in Eq. 6, which is adapted from the compendium of physical activities [36].

$$Act_i.Cal = Act_i.METs * amtAct_i(h) * weight(kg) \tag{6}$$

METs estimates the capacity and tolerance level of an individual to exercise in which he/she may participate safely without hurting him/herself [41]. We use it in our system to estimate calories from the physical activities and vice versa. In our calorie estimation process, we use the average *METs* rather than the exact value. The average *METs* for an activity (*e.g.*, walking) is calculated by considering all types of walking included in the *METs* guidelines [36] and taking the average. The same procedure is used for other activities that we consider (i.e., running, jogging, transportation, sitting, and standing). The rationale behind the average *METs* is the limitation of our current activity recognition system in recognizing the exact intensity of every sub-type of activity, for example, walking.

After applying Eq. 6, for all of the activities, Eq. 7 is used to sum all of the estimated calories.

$$totalBurnedCal = \sum_{i=1}^{a} Act_i.Cal \tag{7}$$

The remaining calories (*remCalToBurn*) for the rest of the day (in a daily calorie consumption plan) are computed using Eq. 8.

$$remCalToBurn = calToBurDay - totalBurnedCal \tag{8}$$

The aim of estimating the remaining calories is to recommend the appropriate physical activity using our reasoning system to meet the goals of the day. This recommendation requires the *METs* value computed from the *remCalToBurn* using Eq. 9 [36].

$$METs = \frac{remCalToBurn}{(amtAct = 1h) * weight(kg)} \tag{9}$$

We use the METs *value in both* RBR *and* CBR *to recommend the appropriate physical activity. For* RBR, *rules need to be created using the user's personal information and the required* METs *value. For* CBR, *a case base is to be prepared.*

4.2.1.4. Rules creations. Based on the estimated *METs* value and the user's personal information (*e.g.*, age), two types of rules are created. The first type is based on distinct-*METs* values, and the second type is based on ranged-*METs* value. The distinct-*METs* rules are used to build the *baseline-RBR* system, while the ranged-*METs* rules are used for building the *modified-RBR* system. When we considered distinct-*METs* and age together, we created a total of 122 rules for the 48 distinct-*METs* values. The distribution of the rules is as follows: 22 rules belong to the *Young* age group, 33 rules belong to the *Older Adults* group, and 47 belong to the *Adults* group. In the context of physical activity recommendation, age plays an important role; therefore, it is considered an essential

Table 3
Distinct-*METs* rules for recommending physical activity in the baseline rule-based reasoning system (*baseline-RBR*).

Rule ID	Age group	METs value	Activity prescription
R#1	Young	2	Walking, household
R#2	Older adults	6.5	Climbing hills with 0–9 lb load; Race walking; rock or mountain climbing
R#3	Young	7.8	Backpacking; hiking or organized walking with a daypack
–	–	–	–
R#122	Adult	15	Running; stairs up

Table 4
Ranged-METs rules for recommending personalized physical activity.

Rule ID	Age group	METs value	Activity prescription
R#1	Young, Adults, Older adults	<3	Light activity
R#2	Adults	≤ 23	Moderate – vigorous-intensity
R#3	Older adults	≤ 10.25	Moderate – vigorous (lower intensity level)
R#4	Young	≤ 7	Moderate

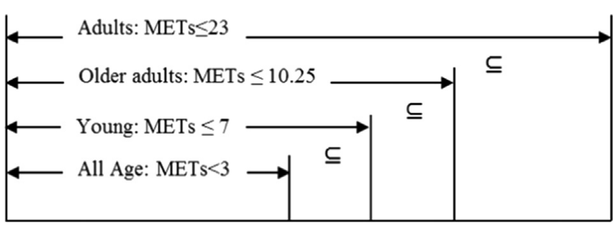


Fig. 6. Distribution of the subjects on the basis of age and intensity level of physical activities (*i.e., METs*).

part of the rules. The transformation of age to different age groups is supported by the guidelines from *WHO* [10] and *UK* [42]. These guidelines categorize users into three major age groups: *Young* (age 5–17), *Adults* (age 18–64), and *Older Adults* (age ≥ 65). A partial list of the distinct-*METs* rules is shown in Table 3.

In the *MM* implementation, we use ranged-*METs* rules; therefore, we first define ranges for the *METs* values used in these rules. According to the well-known physical activity guidelines from the center for disease control and prevention (*CDC*), American College of Sports Medicine (*ACSM*) [7], *WHO* [10], *US* [43] and *UK* [42], physical activities can be grouped into three categories: light (<3.0 *METs*), moderate (3.0–6.0 *METs*) and vigorous (>6.0 *METs*). According to these guidelines, moderate to vigorous-intensity physical activities are recommended to *Young*, *Adults* and *Older Adults*, but with slightly changed doses and patterns. For example, the *Young* group is recommended a physical activity of *METs* $\geq$ 3–7, and the *Adults* and *Older Adults* groups are recommended a physical activity of *METs* ≥ 3. However, the *Older Adults* group is recommended the same physical activities in the range of *METs* values for the *Adult* group but with lower intensity and dose due to their lower capabilities for exercise and physical activities. We have formulated these guidelines by considering the threshold value of *METs* ≤ 10.25 for *Older Adults*, *METs* ≤ 7 for *Young* and *METs* ≤ 23 for *Adults*. The light-intensity activities (*i.e., METs* <3) are appropriate for all age groups because they do not lead to injuries. Based on this grouping of the *METs* values by the age groups, the ranged-*METs* rules are defined and summarized in Table 4.

Table 5
Case base structure of the metabolic equivalent of tasks (*METs*) values by physical activities.

Attribute	Data type	Possible value	Description
Age group	Symbol	{All age, Young, Adults, Older adults}	Age of the subject
METs	Float	Min = 1.3, Max = 23.0	Metabolic Equivalents of Tasks one hour
Recommendations	String	Physical activities {running, walking, cycling, traveling-bus and subways, standing, sitting}	Physical activities

4.2.1.5. Case base creation. The *CBR* part of *HRM* operates based on well-established past successful cases to generate physical activity recommendations. The cases in the case base are adapted from the 2011 compendium of physical activity guidelines [36]. These guidelines contain a list of physical activities associated with *METs* values. We used the *METs* values and the associated physical activities as the two key attributes of our case base. We named this case base the *METs* case base *(METCB)*. Based on the discussion made in Section 4.2.1.4, we extended the number of attributes of the *METCB* to include an additional attribute, *age group*. The relationship between *age group* and *METs* ranges is represented in Eq. 10 and depicted in Fig. 6.

$$AgeGroup = AllAge \subseteq \mathbf{Y}oung \subseteq OlderAdults \subseteq Adults \quad (10)$$

In the above Eq. 10 and Fig. 6, it can be seen that we have added a fourth age group named *'All Age'* (*METs* <3). It is a subset of all of the other age groups because activities of this intensity are not injurious and can equally be recommended to any *age group*. The current *METCB* contains 119 instances, which may increase in the future. Table 5 presents the detailed characteristics of the *METCB*.

4.3. Hybrid reasoning

Hybrid reasoning is the key methodology implemented in the proposed reasoning engine that generates personalized physical activity recommendations in the *MM* system. It is composed of *RBR*, *CBR*, and *PBR* and is discussed in the subsequent sub-sections.

4.3.1. Multi-level rule-based reasoning (Multi-level-RBR)

In *HRM*, the *RBR* methodology works at three levels (level-1, level-2, and level-3). Its objectives include the following: (1) assessment of personal information and recommendation for weight status, (2) assessment of the ideal body weight and recommendations for goals and plans and (3) assessment of the performed physical activities and recommendations for appropriate physical activity. The recommendations of each level are provided to the user, on one end, and to the next level, on the other. For example, the first level of recommendations is provided to the user and to the level-2 *RBR*. This process involves a sequential flow, and finally, recommendations are generated, which are provided to the users on their mobile applications. The idea of provisioning intermediate results to the users is motivated from the fact that *MM* system supports the PULL service model, where users can subscribe either to a single service or a combination of services. Using this approach, some of the users subscribe only for *weight status recommendations*, while others subscribe for *goal and plan recommendations* and *physical activity recommendations*. If the MM system is constrained only to support the PUSH service model, then it may be enough for the users who require services on the subscription basis but will not support users who require customized subscription-based services.

4.3.1.1. Level-1 RBR. Once the user query arrives at the *HRM*, level-1 *RBR* gets activated, loads personal profile information, performs the necessary computations, retrieves the *WSR* (Table 1) and starts the rule-based reasoning process [44]. The outputs are provided to the end user and to the level-2 *RBR*. The steps of the level-1 *RBR* are listed in Algorithm 1.

Algorithm 1. Rule-based reasoning for the recommendations of weight status.

Input: UID:uid
Output: Weight Status (WS): List < Weight Status >
Begin
Let SID:sid = Weight Status Service Id
WSR: Set of Weight Status Rules, WSR = ∅
KB: Knowledge Base
1. **Foreach** RULE R in KB
    If (R ∈ sid)
        WSR:=WSR ∪ R;
    End If
  End for
2. **Foreach** RULE R in WSR
    WS := ExecuteWSRule(R, uid)
    IfWS ≠ "*Underweight*"
        PropgatWSResultsToUIUX(uid, WS);
        InvokeLevel2*RBR*(uid, WS); // See Algorithm 2
        Go to step 3
    Else
        PropgatWSResultsToUIUX(uid, educational & motivational statments for Weight Gain)
        Go to step 3;
    End If
  End for
3. *FCB*:=AddWStatus(uid, WS); // See discussion
4. Exit;
End

In first step of Algorithm 1, *WSR* are loaded from the knowledge base using an iterative loop process. The design of the knowledge base is based on the types of services, and rules are stored accordingly. Therefore, the type of service identifies the type of rules to be loaded. The type of service can be identified by the service Id (*sid*, in this case). Once the rules are loaded, the execution commences. The definition of *ExecuteSWRule*() is given in Function 1, and it loads the personal profile data of the user from the *IDB* and performs the necessary computations. The data loading process of the *IDB* uses a simple object access protocol (*SOAP*)-based service, defined in the *SCL*. Finally, the pattern matching process starts, and when a rule is matched, it is fired, and its corresponding weight status recommendations are generated. The results of this function are returned to Algorithm 1 for further processing.

Function 1. Rules execution for the weight status recommendations.

ExecuteWSRule(RULER, UIDuid)
Let WS = Weight Status, showing *BMI* status of the user
IDB: Intermediate Database
PPROF: Personal Profile
BMI: Body Mass Index
RHS: Right Hand Side
LHS: Left Hand Side
1. Load PPROF of uid from IDB;
2. Compute *BMI*;
3. **If***R*.LHS.values = (PPROF and *BMI*)
        WS:=RHS of R;
    EndIf
4. Return(WS)

When the weight status recommendations are received by Algorithm 1, they are forwarded to the user mobile application interface (*UIUX*) and to the level-2 *RBR*. The function *PropgatWSResultsToUIUX*() is responsible for providing the recommendations to the user while the function *InvokeLevel2RBR*() is used to invoke the level-2 *RBR*. The propagation function first communicates with the user's mobile application and then provides the generated intermediate recommendations along with some metadata for display purposes. In case the intermediate result of the level-1 *RBR* is the *underweight* status, then the system propagates motivational and educational statements using the *PropgatWSResultsToUIUX*() function (see Section 6).

4.3.1.2. Level-2 RBR. Level-2 *RBR* is activated by level-1 *RBR* for setting goals and prescribing the associated weight loss and calorie consumption plan recommendations. In level-2 *RBR*, the goal and plan rules (*GPR*) specified in Table 2 are used along with Eqs. 1–4. The algorithmic steps of level-2 *RBR* are given in Algorithm 2.

Algorithm 2. Rule-based reasoning algorithm for goals and plans prescription recommendations.

Input: UID:uid, WS
Output: Weight Loss Plan (WLP)
Begin
Let SID:sid = Weight Loss Service Id
GPR: Goal and Plan Rules, GPR = ∅
PP: Plan Prescription
1. **Foreach** RULE R in KB // KB: Knowledge Base
    If (R ∈ sid)
        GPR:= GPR ∪ R;
    End If
  End for
2. **Foreach** RULE R in GPR
    PP:=ExecuteGPRRule (RULE R,UID uid)
        If PP="WLP"
            Let wlPlan:= List < WLPlan > ;
        wl Plan=ComputeWLPlansInKgAndCalories(); // use Eq. 3 and 4
            PropgatWLPResultsToUIUX (uid,wlPlan);
            FCB:=AddRecommendedPlan (uid,wlPlan); // See discussion
            InvokeLevel3RBR-CBR (uid,wlPlan ["caloriesPlan"]); // See Algorithm 3
            Go to step 3;
    Else
            PropgatWMPResultsToUIUX(uid, educational & motivational statments for Weight Maintenance)
            Go to step 3;
        End if
    End for
3. **Exit;**
End

In Algorithm 2, the rules are loaded from the *KB* on the basis of service type (*sid*). The service is goal and plan recommendations, and the associated rules are the *GPR*. After the rules are loaded, Algorithm 2 executes *ExecuteGPRRule*() to generate the plan prescription (*PP*) recommendations. The definition of this function is shown in Function 2, which takes each rule from the *GPR* and retrieves the required personal profile data from *IDB* and computes the ideal body weight (*idlWgt*) and global goal (*gloGoal*). The pattern matching process then starts, and each attribute of the left hand side (*LHS*) of the rule *R* is checked against the loaded and computed values. When a match is found, rule *R* is fired, and its right hand side (*RHS*) is provided as the *PP* recommendation. These recommendations are returned to Algorithm 2 for further processing.

Function 2. Execution of the goal and plan rules for goal and plan recommendations.

ExecuteWMPPlanRule(RULER, UIDuid)
Let IDB: Intermediate Database
gloGoal: global Goal
idlWgt: ideal Weight
PPROF: Personal Profile
LHS: Left Hand Side
RHS: Right Hand Side
PP: Plan Prescription
1. Load PPROF of uid from IDB;
2. ComputeIdealWeight(idlWgt); //use Eq. 1
3. ComputeGlobalGoal(gloGoal); //use Eq. 2
4. **If** *R*.LHS.values = (PPROF, gloGoal)
    PP:=RHS of R;
  End if
5. Return (PP);

If the output retained in *PP* is weight loss plan (*WLP*), then the *Compute WL plans in kg and calories*() function is activated for computing daily, weekly, and monthly plans in terms of the number kg to lose and the associated calorie consumption plans. These plans are forwarded to the users and are displayed on their mobile application interface (*UIUX*) and are also forwarded to level-3 *RBR–CBR*. The functions responsible for these tasks are *Propgat WLP Results To UIUX*() and *Invoke Level* 2*RBR* – *CBR*(), respectively. In case the *PP* value is the weight maintenance plan (*WMP*), then educational and motivational statements are provided to the users using the *PropgatWMPResultsToUIUX*() function (see Section 6).

4.3.1.3. Level-3 RBR–CBR. In *HRM*, level-3 *RBR–CBR* uses either *baseline-RBR* or *modified-RBR* or *CBR* methodology. For these methodologies, an assessment of the performed physical activities is required in terms of the burned calories, remaining calories, and equivalent *METs* value. This assessment and the computations are performed using Eqs. 5–9. In the *baseline-RBR*, distinct-*METs* rules (Table 3) are used, while in the *modified-RBR*, ranged-*METs* rules (Table 4) are used to generate personalized physical activity recommendations. The algorithmic steps for both the *baseline-RBR* and *modified-RBR* are given in Algorithm 3 and are the same from the methodology perspective but different based on the nature of rules they use (for the level-3 *CBR*, see Section 4.3.2).

Algorithm 3. Assessment of physical activities and prescription of physical activity recommendations using rule-based reasoning.

Input: UID:uid, wlPlan
Output: Personalized Physical Activity Recommendations (PAR): List < Recommendations >
Begin
Let SID:sid = Personalized Physical Activity Recommendation Service APR : activity prescription rules and APR = $\varnothing$
1. **Foreach** RULE R in KB // KB: Knowledge Base
  If ($R \in$ sid)
    APR: = APR $\cup$ R;
   End if
  End for
2. **Foreach** RULE R in APR
   PAR : = ExecuteActPrescRule(RULER, UIDuid)
   If PAR $\neq \varnothing$
    Break;
  End If
3. **End for**
4. PropgatPARResultsToUIUX(uid, PAR);
5. *FCB*:=Add*RBR*PAR(uid, PAR); // See discussion
6. Invoke*PBR*(uid, PAR); // See Algorithm 5
7. Exit;
End

Algorithm 3 first loads the activity prescription rules (*ARP*) from the *KB* based on the service id, specified in the service request. For generating appropriate personalized physical activity recommendations (*PAR*), the *ExecuteActPrescRule*() function is used, the details of which are given in Function 3. The physical activities are recommended on the basis of the final computed *METs* values and the user's personal profile information. The *METs* value represents the intensity level of a physical activity. Within the same physical activity type, for example, walking, different intensity values exist that range from a *METs* value of 2.3 to a *METs* value of 12 [36]. Similar ranges exist for other activities as well, such as running, cycling, transportation, standing, and sitting. In the *METs* guidelines, a large number of distinct *METs* values are available, which makes it hard to define distinct *METs* rules. One of the solutions to this issue is to define range-based *METs* rules. In the *MM* implementation for the weight management scenario, *METs* range-based rules are used.

Function 3. Execution of distinct-*METs* and ranged-*METs* rules for physical activity recommendations.

ExecuteActPrescRule(RULER, UIDuid)
Let IDB: Intermediate Database
METs: Metabolic Equivalent of Task
PPROF: Personal Profile
AMTACT: Amount of Physical Activity Performed
PAR: Personalized Physical Activity Recommendations: List < Recommendations >
LHS: Left Hand Side
RHS: Right Hand Side
1. Load PPROF, AMTACT of uid from IDB;
2. Compute AMOUNT OF PHYSICAL ACTIVITY performed so far; //use Eq. 5
3. Compute CALORIES for each ACTIVITY; //use Eq. 6
4. Compute TOTAL BURNED CALORIES; //use Eq. 7
5. Compute REMAINING CALORIES; //use Eq. 8
6. Compute METs value; //use Eq. 9
7. **If** *R*.LHS.values = (PPROF, METs)
   PAR:=RHS of RULE;
   End if
8. Return(PAR)

Once *PAR* are generated, they are provided to the end users on their mobile application interface (*UIUX*) using the *Propgat PAR Results To UIUX*() function. The output of Algorithm 3 can be a list of physical activities that are generated either on the basis of ranged-*METs* rules or multiple physical activities against a single *METs* value in a rule. To filter this list of recommendations and personalize them to another level, they are provided to the

Table 6
Local similarity values of the attribute '*age group*' in the form of similarity matrix.

Age group	All age	Young	Older adults	Adults
All age	1	1	1	1
Young	1	1	0	0
Older adults	1	1	1	0
Adults	1	1	1	1

PBR methodology by using the *Invoke PBR*() function call of Algorithm 3 (see Section 4.3.3 for the *PBR* functionality).

4.3.2. Case-based reasoning (CBR)

To overcome the limitations of level-3 *RBR* implemented in the *MM* platform, we use *CBR* for generating more personalized recommendations. The *CBR* implementation is performed outside the *MM* implementation in our lab with the aim of enhancing the performance of *HRM*. The *CBR* methodology helps in recommending specific physical activity to users based on their gender information and required intensity for physical activity *i.e., METs* value. The *CBR* methodology is selected due to its capabilities of (1) recommending specific and precise physical activities to the user, (2) providing a list of top relevant physical activities as recommendations (*e.g.*, walking) with multiple similar alternatives (*e.g.*, running or cycling) and (3) refining the suggested recommendations based on the user's feedback for enhancing recommendation accuracy and specificity. *CBR* execution follows the standard *CBR* cycle (*retrieve, reuse, revise* and *retain*) to complete the process of suggesting and refining recommendations along with an incremental learning approach. In our study, we are unable to perform the *revise* step in *HRM* due to the limitation of the *MM* system in being unable to handle user feedback. This phase is left as future work.

4.3.2.1. Retrieve and reuse steps. In our *CBR* model, the case query contains two attributes, *age group* and *METs* value. The age value is retrieved from the personal profile of the user, which is transformed to the predefined *age group*. The value of the *METs* attribute is computed from the user's personal profile information and the physical activities the user performed so far. For this purpose, steps 1–6 of Function 3 are used. These values are provided to the *retrieve* step of the *CBR*, which starts retrieving similar cases from the *METCB.* For the retrieval of *age group* and *METs* values, two local similarity functions are defined, which are shown in Eqs. 11 and 12.

$$METSim_l(nC,eC) = \frac{d_g(Max_{MET}, Min_{MET}) - d_l(nC_{MET}, eC_{MET}) - 1}{d_g(Max_{MET}, Min_{MET})} \tag{11}$$

Here, $METSim_l$ calculates the similarity of the *METs* between the *new query case* (*nC*) and *existing cases* (*eC*) in the *METCB*. Similarly,d_g is the global distance function that calculates the distance between Max_{MET} (maximum *METs* value in the *METCB*, *i.e.*, 23 for running) and Min_{MET} (minimum *METs* value in the *METCB*, *i.e.,* 1.3 for resting). Here, d_l is the most important local similarity function that computes the distance between the *METs* values of *nC* and *eC*.

$$AGSim_l(nC,eC) = \left\{ \begin{array}{ll} AG_{ij} = 1 & for \forall (i \geq j) \text{ OR } (i = 0 \text{ OR } \ j = 1) \\ 0 & otherwise \end{array} \right\} \tag{12}$$

In Eq. 12, $AGSim_l$ is the local similarity function that matches the *METs* values of *eC* with *nC*. The similarity criterion used in the equation is the exact match, which is denoted as value 1. The interpretation of this value is that if the age group of the query case is similar to that of the existing case (*i.e.,*$AG_{nC} = AG_{eC}$), then this value will be 1; otherwise, it will be 0. The symmetric view of the local similarity function of this attribute is represented in a confusion matrix shown in Table 6.

In the above confusion matrix, the diagonal value of each age group is equal to 1, which shows the exact match relationship of each age group with itself. The age group, labeled as *All Age*, represents the list of *METs* values (less than 3) that can be equally recommended to the rest of the age groups; therefore its value is 1 for all of the other age groups. Similarly, the *METs* values of the age group *Young* (less or equal to 7) are also a subset of the *METs* values of the *Older Adults* and *Adults* age groups; therefore its value is 1 for all these age groups. This makes both the columns identical in the similarity matrix table.

After computing the local similarities, we use the weighted sum global similarity function, Sim_g, to compute the global distance between *nC* and *eC*, as shown in Eq. 13.

$$Sim_g(nC,eC) = \beta(AGSim_l(nC,eC)) + \gamma(METSim_l(nC,eC)) \tag{13}$$

Here, β denotes the weight value assigned to the attribute *age group* and γ denotes the weight value assigned to the *METs* attribute. The value of β is 0.1 (*i.e.,* $\beta = 0.1$), and the value of γ is 0.9 (*i.e.,* $\gamma = 0.9$). The higher value of γ represents the importance and contribution of the *METs* attribute in the final decision. For the selection of similar cases, we use k-NN [45] with k=3 to select the top three similar cases and reuse them as the suggested recommendations. In the *MM* system, the selection of the top three cases provides choices to the users for following any of the proposed recommendations based on their personal preferences and interests. The top recommended activities are of the same intensity or close to each other in intensity and have similar impacts on an individual's health. The acceptance of the top three recommendations is based on the threshold value (confidence), denoted by symbol μ. We set the threshold value to be greater than or equal to 95 (*i.e.,* $\mu \geq 95$). If a single case satisfies the threshold, only one recommendation is provided as the final physical activity recommendation. If more than 1 case is retrieved, then *PBR* is activated for further filtration and personalization of the suggested physical activity recommendations (see Section 4.3.3). The confidence value for the acceptance of recommendations is the threshold value, which is computed using Eq. 13. It is the aggregate score obtained from the local similarity values of Eqs. 11 and 12. The method used for aggregation is the weighted sum, which has a higher weight $\gamma = 0.9$ for the *METs* attribute and lower weight $\beta = 0.1$ for the *Age Group* attribute. To set the confidence/threshold value as $\mu \geq 95$(0.05threshhold), we were motivated by the well-known work [46–48] in the statistical community. The authors considered a 95% confidence interval or 0.05 threshold value as the acceptable value for accepting a hypothesis. The detailed working methodology of the proposed *CBR* is presented in Algorithm 4.

Algorithm 4. Case-based reasoning methodology for generating personalized physical activity recommendations.

Input: UID:uid, METCBurl, *nC*:= new Case, where *nC*∈ PPROF, METs and *nC* is computed using Eqs. 5–9

Output: Personalized Physical Activity Recommendations (PAR)): List <Recommendations>
Begin
Let PAR:= A set of top 3 relevant existing cases as the proposed recommendations
Sim_g[]:= Array of global similarities of existing cases
1. $METCB_r$:= ReteriveCaseBaseFromKB(METCBurl), Where $METCB_r$ is the matrix $eC_m \times A_n$, eC_m is the set of existing cases, i.e., $eC = eC_1, eC_2, eC_3, \ldots, eC_m$. Similarly, A_n is the set of attributes, i.e., $A_n = A_1, A_2, A_3, \ldots, A_n$
2. **For** $i=1$ to SizeOfCases($METCB_r$)
   Let Sim_l []:=Array of local similarities of attributes of individual cases
   For $j=1$ to SizeOfAttributes($METCB_r$)
      Sim_l [A_j]:=ComputeLocSim(nC.A_j,$METCB_r$ [i,j]); // use Eqs. 11 and 12
   End for
   $\mathbf{Sim_g}$ [eC_i]:=ComputeGlobSim ($\mathbf{Sim_l}$); // weighted sum method (Eq. 13)
3. End for
4. PAR:=ApplyKNN(Sim_g]); //where k=3
5. PropgateCBRResultsToUIUX(uid,PAR);
6. FCB:=RetainCBRPAR(uid,PAR); // See discussion
7. InvokePBR(uid,PAR); //See Algorithm 5
8. Exit;
End

Algorithm 4 begins execution when nC is input to the *CBR* algorithm. In the first step, the *ReteriveCaseBaseFromKB*() function is used to load the existing cases from *KB* to the $METCB_r$. For this purpose, the *URL* of *METCB*, *METCBurl,* is used. Each *eC* is matched against *nC*, and the distance is calculated using the local and global similarity functions (*i.e*., Eqs. 11 and 12). *k-NN* with $k=3$ is used to obtain the top three similar cases as the suggested physical activity recommendations. These recommendations are specific and precise compared with the results of the *baseline-RBR* and *modified-RBR* systems. The retrieved case(s) is/are passed to the end users as the proposed personalized physical activity recommendations with the help of the *PropgateCBRResults*() function. Similarly, this/these recommendations(s) is/are also forwarded to *PBR* using the *InvokePBR*() function to filter them according to the user's preferences and interests.

4.3.2.2. Retain steps. Once the *reuse* step suggests recommendation (s), the whole case needs to be retained in the case base as a new case. In the proposed *HRM*, we add this new case to a data store, called the future case base (*FCB*). If the *retrieve* step ends with a single recommendation, the whole case, including the user's personal profile and suggested activity, is stored in the *FCB*. However, if more recommendations are generated, the new case is stored in the *FCB* after applying the *PBR* methodology (see Section 4.3.3).

4.3.3. Preference-based reasoning (PBR)

The recommendations generated by the *RBR* and *CBR* methodologies are based on the knowledge created based on general guidelines, which are unable to reflect the user's personal interests and preferences. These recommendations are not personalized from the perspective of the user's personal interests and preferences; to satisfy them, another level of refinement and filtration of the suggested recommendations is required that is performed by the *PBR* methodology. The *PBR* mechanism exploits the user model, built on top of the user profile. A user model contains the user's personalized requirements, such as preferences and interests. This information is initially acquired from the user, during the registration process and updated thereafter. The recommendations provided by the *RBR* and *CBR* exploit data only from the user's personal profile and physical activity behaviors and do not take into account the preferences. When recommendations are provided on the basis of these methodologies, multiple interpretations can be made. For example, consider a scenario where a user *U* requires *X METs* of physical activity to burn an amount *Y* of calories. The *RBR* or *CBR* can generate the following set of recommendations for the mentioned scenario.

- Walking *M1* minutes OR Running *M2* minutes OR Cycling *M3* minutes OR Hiking *M4* minutes, *etc*.

These recommendations are equivalent and can meet the user's requirement mentioned in the scenario' however some of them may not fit the user's personal interests and preferences adequately. It may be that the user is interested in walking and cycling but not in running and hiking. Therefore, the final recommendations should only include walking and cycling.

To obtain the user's final preference-based personalized recommendations, we propose a multi-filter approach implemented as part of the *PBR*. According to this approach, filtered personalized physical activity recommendations (*FPAR*) are obtained from the list of generated personalized physical activity recommendations using the user preferences (UPrefrences). This process of filtration is shown in Algorithm 5.

Algorithm 5. Filtration of the personalized physical activity recommendations using user preferences.
Input: UID: uid, PAR
Output: Filtered Personalized Physical Activity Recommendations(FPAR): List <filteredRecommendations>
Begin
Let UPrefrences[]= List of user preferences
FCB:=Future Case Base
FPAR : = $\varnothing$
1. UPrefrences[]=loadUserPrefences(uid);//Load user presences from user profile in IDB
2. **Foreach** Recommendation Rec in PAR
   If (Rec $\in$ UPrefrences)
      FPAR:= FPAR $\cup$ Rec;
   End if
   End for
3. PropgatFilteredPARToUIUX(uid, FPAR);
4. *FCB*:=AddFPAR(uid, FPAR); // see discussion
5. Exit
End

The process of preference-based reasoning starts by loading the user's list of preferences, denoted by *UPrefrences*, from the intermediate database. The filtration process is performed in step 2 by taking each recommendation from the *PAR* and checking it against the preference list of the user. If the recommendation does not satisfy the user's preference, it is filtered out; otherwise, it is added to the filtered list *FPAR*. This process is continued till all of the recommendations in *PAR* are checked. Finally, the filtered personalized recommendations are provided to the user on his mobile application interface using the PropgatFilteredPARToUIUX() function. At the same time, the final *FPAR* are retained in the *FCB* as the recommended physical activity. This incrementally grows the *FCB*, which can be best used in future for successful cases of physical activity recommendations.

Table 7
Personal profile information of the volunteers, *WHO* participated in the evaluation of Mining Minds platform.

User ID	Gender: Male (M), Female (F)	Age (Years)	Height (Feet)	Weight (Kg)	Preferred activities
1	M	26	6.2	84.5	Running, walking
2	M	28	5.7	72.5	Running, walking, cycling
3	M	28	5.8	70.1	Walking
4	M	31	5.4	68	Running, cycling
5	M	31	5.6	71.9	Walking, traveling
6	M	32	6	85.9	Running
7	F	32	5.2	65	Walking, jogging
8	M	37	5.8	75	Walking, cycling
9	F	30	5.2	75	Walking running, cycling
10	M	38	5.8	71	Running, cycling

Table 8
Distribution of the physical activities in the *METs Case Base*.

S.No	Type of activity	Distribution
1	Running	25
2	Walking	56
3	Cycling	18
4	Standing	5
5	Sitting	4
6	Transportation	4
7	Volunteer	7
Total instances		119

5. Experiments and evaluation

For evaluating the performance of the proposed *HRM*, we performed the following set of tasks. Initially, we defined a weight management scenario, then set up a set of experiments, and finally performed the experiments and analyzed the results.

5.1. Case-study: weight management

We considered and implemented a weight management scenario for healthy individuals who are overweight or tend to overweight. After implementation of the methodology, we asked ten volunteers (ages 26–38 years) to use the system for a couple of weeks. The basic personal information of these individuals is shown in Table 7.

The individuals were asked to use the application during the specified period of time and follow the recommendations provided. During the user's physical activity, the mobile application collected the user's daily physical activity data using the accelerometer sensor of the smartphone. These activities included sitting, standing, moving in a bus, moving in a subway, walking, running and cycling, which are recognized by the *activity recognizer* module (in the *ICL*) of the Mining Minds platform (Fig. 2). For the detailed methodological process of recognition of these activities and the support of *ICL*, refer to the work of Han et al. [49], and Banos et al., [50]. The data are stored in the *DCL*, from where they are recognized by the *ICL* and provided to the *SCL* for recommending the appropriate physical activity for the remaining targets.

5.2. Experimental setup

To perform the experiments, we first set up the required environment, then specified the data and knowledge used for the experiments and finally defined the evaluation criteria.

5.2.1. Environment

The implementation of *HRM* was performed on a distributed framework in the Microsoft Azure public cloud environment. As described in Section 3, the *MM* platform is composed of four layers, and each layer is deployed on an individual virtual instance. The proposed *HRM* is part of *SCL*, which was hosted on a standard *A3 MS* Azure instance with Microsoft Windows Server 2012 *R2* as the guest Operating System (*OS*). *HRM* communicates with *DCL* and *SL* and communicates with *DCL* to load data for reasoning and storing final recommendations. With *SL*, *HRM* provides a recommendation service on the request and response model. The services in *SCL* are implemented as *SOAP*-based web services, and their accessibility is defined using service contracts between layers. Web services are implemented in Java and deployed on Glassfish server on virtual machine (*VM*).

For implementation of the third experiment, *hybrid-CBR*, which operates on *METCB*, we used myCBR[3], which is an open-source similarity-based retrieval tool. We used the Windows environment on a *PC* with an Intel Pentium Dual-CoreTM (2.5 GHz) with 4 GB of memory.

5.2.2. Data and knowledge (rules/case base)

As we evaluate our proposed *hybrid-CBR* methodology in terms of the performance of the *baseline-RBR* and *modified-RBR* systems, we therefore require data and knowledge on all of these systems. For the *baseline-RBR* and *modified-RBR* experiments, we used the user's personal profile, physical activity data and knowledge rules created based on the guidelines (Tables 3 and 4). For the *hybrid-CBR* experiments, we use *METCB*, prepared from *METs* guidelines [36]. The size of our '*METCB*' is 119 instances. It contains the activities we focus on in the *MM* platform. The distribution of these activities in *METCB* is shown in Table 8.

In the compendium of physical activity guidelines [36], "standing" and "sitting" are the sub-categories of volunteer physical activity. More details on the structure of *METCB* are given in Table 5. For the offline testing and evaluation of the methodology, we designed a *Test Case Base (TCB)* that contains 64 test instances. We prepared these test cases from the original *METCB*. The method used for defining the value of the *METs* attribute of the *TCB* was random value computation. The random value is computed from the *METs* attribute of the original *METCB* using Microsoft Excel [51]. The function used for the random value generation is shown in Eq. 14.

$$METs.value = randbetween(bottom, top) \tag{14}$$

Here, *bottom* represents the minimum value of the *METs* and *top* represents the maximum value of *METs* for the new test cases. We used $bottom = 1.3$ and $top = 23$. The values 1.3 and 23 are the

[3] http://mycbr-project.net/index.html

Table 9
Weight status and goal and plan recommendations generated by level-1- and level-2 rule-based reasoning of the proposed multimodal reasoning methodology.

User ID	Level-1 *RBR* (Algorithm 1) Results		Level-2 *RBR* (Algorithm 2) results				
	BMI	**Weight status**	**Ideal body weight** (Kg)	**Goal** (# of Kg to lose)	**Weight management plan**	**Duration plan** (weeks)	**Calories burning plan** (daily)
1	23.9	Normal	78.0	6.5	weight loss	13	550
2	25.02	Overweight	64.8	7.7	weight loss	15	550
3	23.5	Normal	66.6	3.5	weight loss	7	550
4	25.7	Overweight	59.1	8.9	weight loss	18	550
5	25.8	Overweight	62.9	9.0	weight loss	18	550
6	25.7	Overweight	74.2	11.7	weight loss	23	550
7	26.2	Overweight	52.0	13.0	weight loss	26	550
8	25.14	Overweight	66.6	8.4	weight loss	17	550
9	30.24	Obese	52.0	23.0	weight loss	46	550
10	23.8	Normal	62.1	8.9	weight loss	18	550

minimum and maximum values, respectively, of the *METs* attribute in the original *METCB*.

5.2.3. Evaluation criteria

To evaluate the proposed reasoning methodology, a group of system-centric evaluation criteria are used [52]. We evaluated the system using *Type I* (False positive-FP) and *Type II* (False negative-FN) errors, precision, recall, accuracy, and *f*-score criteria. We do not focus on a user-centric evaluation that addresses the user's satisfaction because in the current implementation, only a prototype of the *MM* platform is implemented. The *hybrid-CBR* experiments were performed in a closed environment in our lab; therefore, we leave user-centric evaluation as future work when the *MM* platform will be fully implemented with the feedback mechanism.

5.3. Experiments and analysis of the results

As the design of *HRM* is based on *RBR*-first followed by the *CBR* strategy, we therefore first evaluate the *RBR* and then tailor its results to *CBR*. During the *RBR* execution, the level-1 *RBR* is first executed for reasoning the weight status of all of the subjects using Algorithm 1 and presenting the output as recommendations to the users, as shown in Table 9. If the weight status is not underweight, the output is fed to level-2 *RBR* for setting goals and recommending weight loss and calorie consumption plans using Algorithm 2. The resulting recommendations of the level-2 *RBR* are also shown in Table 9.

These recommendations include the goal in terms of kg to lose, weight management plan, number of weeks to successfully execute the plan and daily calorie consumption plan. The volunteers were asked to follow these plan recommendations. The objective of *HRM* is to recommend appropriate physical activities for these plans. The *HRM* estimates *METs* values to materialize the plans. The *METs* estimation is required in two cases:

- At the start of plan, when *HRM* initially recommends the physical activity for starting the plan.
- During the plan, *i.e.*, the subject follows the plan and the system makes further recommendations.

In the first case, the *METs* estimation is performed only for the recommended 'daily calorie consumption plan', which is the output of the level-2 *RBR*. In the second case, the *METs* estimation is based on the remaining calories (see Eq. 8). Once the *METs* value is computed, the corresponding physical activity recommendations are generated. These recommendations can be generated using the *baseline-RBR, modified-RBR* and *hybrid-CBR* systems; therefore, we perform three different sets of experiments, which are discussed below.

Table 10
Physical activity recommendations generated by the baseline rule-based reasoning system.

User ID	METs	Personalized physical activity recommendations
1	6.5	i. Climbing hills with 0–9 lb load. ii. Race walking; rock or mountain climbing
2	7.6	X
3	7.8	i. backpacking; hiking or organized walking with a daypack
4	8.1	X
5	7.6	X
6	6.4	X
7	8.5	i. bicycling; BMX ii. bicycling; mountain; general iii. bicycling; 12 mph; seated; hands on brake hoods or bar drops; 80 rpm
8	7.3	i. climbing hills with 10–20 lb load
9	7.3	i. climbing hills with 10–20 lb load
10	7.7	X

5.3.1. Experiment 1: baseline-RBR system

The purpose of this experiment is to build the initial *baseline-RBR* system for comparing the results of the systems. The results of this experiment were generated prior to the implementation of the proposed idea in the *MM* platform. In level-3 *RBR*, distinct-*METs* rules, shown in Table 3, are used to generate physical activity recommendations using Algorithm 3 with exact match criteria. A few examples of the prescribed recommendations are shown in Table 10. These are based on the initial calorie consumption plan of the 10 volunteers.

While generating these recommendations, the first *METs* values for all volunteers are computed based on their calorie plans and then combined with the attribute *age group* to prepare the data for the rules. The symbol 'X' in Table 10 denotes that no recommendation is generated for these query cases. From Table 10, it is clear that five out of ten queries cases are unsuccessful and that recommendations could not be generated for them. These include the queries of users 2, 4, 5, 6 and 10. The reasons for the empty recommendations are that these queries do not match any rule described in Table 3. The distinct rules used in this experiment use *METs* values adopted from the *METs* guideline for physical activity, which does not include the values 7.6, 8.1, 7.6, 6.4, and 7.7. Therefore, no rule with these values exists in Table 3, and hence, no match is found during the reasoning process for the specified input query cases. For the detailed evaluation of the *baseline-RBR* system, the whole '*TCB*' is used as a test case. The results are calculated and presented in Figs. 7 and 8, which show that the recall

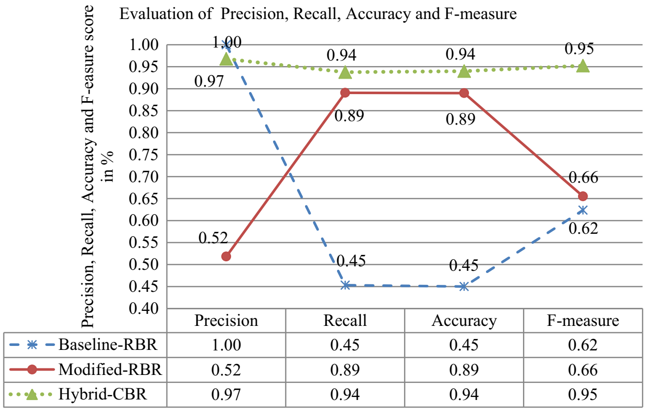


Fig. 7. Comparison of baseline-*RBR*, *modified-RBR* and *hybrid-CBR* system on the basis of precision, recall, accuracy, and *f*-measure.

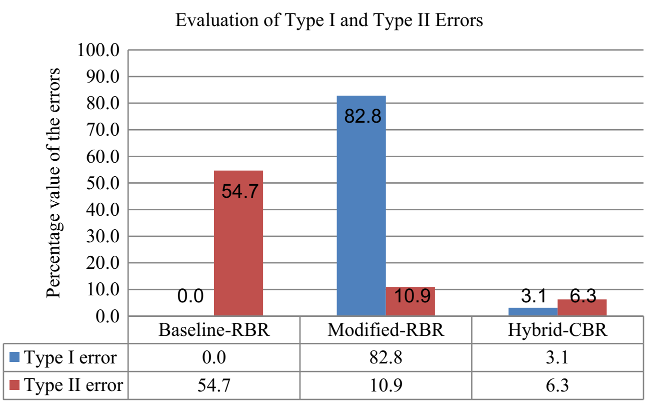


Fig. 8. Comparison of baseline-*RBR*, *modified-RBR* and *hybrid-CBR* for *Type I* and *Type II* errors.

of the *baseline-RBR* is very low (45%) and that the *Type II* errors are very high (54.5%). The limitations of this experiment are summarized as follows: (1) creation of distinct rules for each value of *METs* is a difficult task that results in a rule intractability problem, (2) the closest similar recommendations are overlooked if an exact match is not found, and (3) a high *Type II* error rate is observed.

5.3.2. Experiment 2: modified-RBR system

Based on the lesson learnt from the *baseline-RBR* system, level-3 *RBR* is implemented with *ranged-METs* rules (Table 4) in the *MM* platform. Algorithm 3 is used to execute these rules. To demonstrate the effectiveness of this experiment, we consider an example query for volunteer 4 (Table 7) with *age group* = *adults* and *METs* = 8.1 (see Table 10). The *modified-RBR* generates multiple recommendations for this query, though *baseline-RBR* fails to do so. To fully evaluate Algorithm 3, the whole '*TCB*' is applied, and the results produced are shown in Figs. 7 and 8. The recall and accuracy are increased from 0.45 to 0.89 and the *f*-score is increased from 0.62 to 0.66, while the Type II error rate is reduced from 54.7 to 10.9. The advantage of the *modified-RBR* system is that all queries are served and no query is returned with empty recommendation results. For example, when the query case with '*age group*' = *All Age* and *METs* = 2.7 is processed, a total of 17 recommendations are generated, as shown in Table 11. When the *baseline-RBR* is used for this query, no recommendation is generated because the *METs* value of the query case has no match with the *METs* values of the distinct rules. However, in the *modified-RBR*, the ranged-*METs* rule with a *METs* value less than 3 is satisfied, and hence, all of the associated recommendations are generated.

Table 11
Physical activity recommendations generated for a single query case using the modified rule-based reasoning system

Recommendation #	METs	Suggested physical activity recommendations
1	1.3	Riding in a car or truck
2	1.3	Riding in a bus or train
3	1.5	Sitting; meeting; general; and/or with talking involved
4	1.5	Sitting; light office work; in general
5	2.0	Walking; household
6	2.0	Walking; less than 2.0 mph; level; strolling; very slow
7	2	Sitting; child care; only active periods
8	2	Walking; less than 2.0 mph; very slow
9	2.3	Carrying 15 lb child; slow walking
10	2.3	Standing; light work (filing; talking; assembling)
11	2.5	Bird watching; slow walk
12	2.5	Walking from house to car or bus; from car or bus to go places; from car or bus to and from the worksite
13	2.5	Walking to neighbor's house or family's house for social reasons
14	2.5	Walking; to and from an outhouse
15	2.5	Sitting; moderate work
16	2.5	Automobile or light truck (not a semi) driving
17	2.8	Walking; 2.0 mph; level; slow pace; firm surface

Similarly, all of the queries yields results, and no query is unsuccessful.

The limitation of the system is its high *False Alarm* rate (*i.e., Type I* error), as shown in Table 11. From this table, we see that a list of 17) recommendations is generated for a single query. On average, 52 options of physical activities are provided as recommendations for each query, which is problematic. A summary of the *Type I error* for this experiment is shown in Fig. 8. The high *False Alarm* rate results in a wide scope of recommendations that may not fit well with the user's required physical activity. This effect is normalized in *PBR* when multiple filters are applied for filtering unnecessary and irrelevant recommendations.

5.3.3. Experiment 3: CBR system

The objective of using *CBR* is to minimize limitations of the *baseline-RBR* and *modified-RBR* systems. To overcome these problems, we performed the *CBR* experiment in a local set up without involving the *MM* setup. The outputs of level-1 *RBR* and level-2 *RBR* and the estimated *METs* value generate a query case for the *CBR* methodology. Algorithm 4 uses the local similarity function, global similarity function, k-NN with $k=3$ and a threshold $\mu >=95$ to generate appropriate physical activity recommendations. The *CBR* methodology has significantly improved *Type I* and *Type II* errors, as shown in Fig. 8. *CBR* offers the following advantages:

- *Type I* errors are reduced – k-NN with $k=3$ retrieves the top cases that are most relevant to the query case and specific to the user's requirement. Hence, the *False Alarm* rate is significantly reduced.
- *Type II* errors are reduced and recall is improved – the global similarity function of *CBR* with threshold $\mu >=95$ has reduced *Type II* errors. The retrieval of most similar recommendations minimized the *False Negative* cases and improved recall.
- Relevant and specific recommendations – the retrieve phase of *CBR* retrieves the top three recommendations that are either exactly the same as required by the user or close to the user's specific requirements for physical activity. Hence, the number of recommendations is reduced to an optimum level on the one hand and is closer to the user's specific requirements on the other.

Table 12
Physical activity recommendations generated using case-based reasoning methodology.

User ID	New case (METs value)	Retrieved cases (METs value)	Suggested physical activity recommendations
1	6.5	6.5	i. climbing hills with 0–9 lb load.
		6.5	ii. race walking; rock or mountain climbing
		6.3	iii. climbing hills; no load
2	7.6	7.3	i. climbing hills with 10–20 lb load
		7.5	ii. bicycling; general
		7.8	iii. backpacking; hiking or organized walking with a daypack
3	7.8	7.8	i. backpacking; hiking or organized walking with a daypack
		8	ii. running; training; pushing a wheelchair or baby carrier
		8	iii. running; marathon
4	8.1	8	i. running; training; pushing a wheelchair or baby carrier
		8	ii. running; marathon
		8	iii. carrying 25 to 49 lb load; upstairs
5	7.6	7.3	i. climbing hills with 10 to 20 lb load
		7.5	ii. bicycling; general
		7.8	iii. backpacking; hiking or organized walking with a daypack
6	6.4	6.3	i. climbing hills; no load
		6.5	ii. climbing hills with 0–9 lb load
		6.5	iii. race walking; rock or mountain climbing
7	8.5	8.5	i. bicycling;
		8.5	ii. bicycling; mountain; general
		8.5	iii. bicycling; 12 mph; seated; hands on brake hoods or bar drops; 80 rpm
8	7.3	7	i. walking; 4.5 mph; level; firm surface; very; very brisk
		7.3	ii. climbing hills with 10–20 lb load
		7.5	iii. bicycling; general
9	7.3	7	i. walking; 4.5 mph; level; firm surface; very; very brisk
		7.3	ii. climbing hills with 10–20 lb load
		7.5	iii. bicycling; general
10	7.7	7.5	i. bicycling; general
		7.8	ii. backpacking; hiking or organized walking with a daypack
		8	iii. bicycling; 12–13.9 mph; leisure; moderate effort

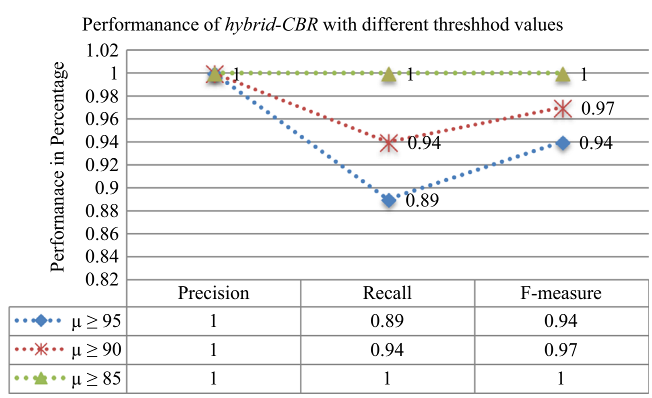


Fig. 9. Performance of basline-*RBR*, *modified-RBR* and *hybrid-CBR* with different threshold values.

To demonstrate the effectiveness of the *CBR* methodology for these objectives, we consider the case of 10 volunteers of the *MM* evaluation team and their estimated *METs* values (Table 10). The initial recommendations for the calculated *METs* values and *age group* = *adults* are shown in Table 12.

Table 12 shows that for each query case, the top three most relevant physical activity recommendations are provided, which fulfills the user's specific requirements. For the query *age group* = *Adults* and *METs* = 8.1, *baseline-RBR* failed to generate recommendations (see Table 10) and *modified-RBR* produced 59 possible recommendation options, but *CBR* produced only three recommendations (Table 12). The difference between the required *METs* values of the query case and the one using the rules is only 0.1, which is negligible; however, *baseline-RBR* fails to generate recommendations. This clearly shows the effectiveness of the proposed *CBR* methodology in *HRM*.

Moreover, to fully evaluate the *CBR* methodology, we apply the whole *'TCB'* to generate recommendations. The results are shown in Figs. 7 and 8. These results are significantly improved compared with those of the *baseline-RBR* and *modified-RBR* methodologies.

The green line at the top of the graph in Fig. 7 shows the performance of *hybrid-CBR*, which is superior to the other two approaches.

Fig. 8 pictorially shows that hybrid *RBR*/*CBR* has improved *Type I* and *Type II* error results compared with the other experiments. To present the results of *hybrid-CBR* with different threshold values i.e., $\mu \geq 95$, $\mu \geq 90$ and $\mu \geq 85$, we applied the '*TCB*' and calculated the results, which are shown in Fig. 9.

Fig. 9 shows that the proposed *hybrid-CBR* model produces 100% results for precision, recall, and *F*-score when the threshold μ is taken as 85.

Table 13
Physical activity recommendations for volunteer no. 8, generated using hybrid case-based reasoning methodology

User ID	Physical activity recommendations based on *hybrid-CBR*
8	i. walking; 4.5 mph; level; firm surface; very; very brisk ii. climbing hills with 10–20 lb load iii. bicycling; general

Table 14
Personalized filtered recommendations refined using the user's personal preferences.

User ID	Personalized filtered recommendations
8	i. To be healthy with normal body weight, you can take a *very brisk walk on firm surface with a speed of* 4.5 mph ii. To achieve today's goal for your required calories consumption, you can perform *physical activity of bicycling*

5.3.4. PBR (preference-based reasoning) results

We evaluated the *PBR* methodology results using an example and examined the filtration process, which filters the physical activity recommendations generated by level-3 *RBR*/*CBR*. Consider the physical activity recommendations, shown in Table 13, for volunteer no. 8.

As the preferences of volunteer no. 8 are walking and cycling (Table 7), *PBR* filters out the *climbing hills* recommendation. Similarly, all recommendations are filtered one by one, and the final filtered recommendations are sent to the *result propagator,* which forwards the recommendations in descriptive form to the end user. Table 14 shows the filtered recommendations in descriptive form.

6. Discussion

Physical activity recommendations help users adopt an active pattern of life. In this regard, the 2011 compendium of physical activities guidelines [36] suggests a wide range of activities with different intensity levels that are measured in terms of *METs* values. The study has described a hybrid multimodal reasoning methodology that has integrated *RBR, CBR,* and *PBR*. The *RBR* methodology is based on domain expert knowledge created from online guidelines for generating intermediate recommendations of goal setting, weight status and goal achieving plans that serve *CBR* to generate final physical activity recommendations. The goal of hybrid reasoning methodology is to ensure accurate and precise personalization of physical activity recommendations. A number of experiments are performed to demonstrate that the methodology achieves this goal. The results shows that the *hybrid-CBR* system outperformed the *baseline-RBR* and *modified-RBR* systems and had significantly improved precision, recall, accuracy, f-measure, and *Type I* and *Type II* errors. The *baseline-RBR* system was tested with 122 distinct-*METs* rules, and it exhibited specificity with exact match criteria, but suffered from a high *False Negative* rate, low accuracy and the rule intractability problem. The *modified-RBR* system was tested with a reduced number of ranged-*METs* rules and implemented in the MM platform and exhibited improved accuracy, but at the cost of low precision. A large number of recommendations were generated, with the majority being irrelevant to the user requirements. In the results, the correctness of recommendations was compromised by the *False Alarm* rate, which is generally unsuitable in the context of a personalized recommendation system. For minimizing the *Type I* and *Type II* errors and increasing the accuracy, *hybrid-CBR* was tested, and it outperformed the other two systems.

The challenging issue associated with *hybrid-CBR* was the design and preparation of the case base. We resolved this issue by creating a case base, *METCB,* from the compendium of physical activities guidelines [36], user personal profile information and general guidelines of physical activities from different organizations, such as *WHO*, *UK* and *CDC*. A *CBR* methodology has a complete cycle starting from *retrieval* to *reuse, revise* and *retain*; however, we did not focus on the *revise* step. The rational is that *revise* should be activated once any of the following conditions are obtained: (1) no existing case with confidence (nC, rC) ≥ 95 is found in *METCB* or (2) the *MM* feedback mechanism returns user remarks indicating dissatisfaction. As the *MM* feedback mechanism is not yet built, we did not focus on these options and have planned them for future work. Furthermore, the current '*METCB*' consists of only three attributes: age group, required *METs* value and recommended physical activity. This imposes the constraint of using multi-level *RBR* prior to applying *CBR*. The *RBR* refines the required high-level information from the basic profile and physical activity information to serve the *CBR* cycle for generating personalized physical activity recommendations. The *RBR* part of this methodology can be excluded and the complexity can be reduced if a case base with all of the required data, starting from the user's personal profile to the intermediate recommendations and final physical activity recommendations, are prepared and stored in a single case base. This case base will contain the user's personal profile, weight status, recommended plans for weight loss, required *METs* value, list of recommended physical activities, personal preference list and final filtered list of physical activities. To obtain a case base with the specified schema, we created a case base, named *FCB*, which incrementally adds new solved cases as successful cases for future use. For this purpose, in Algorithms 1–4, we added a statement that populates the respective attributes in the *FCB*. In the future, this case base will help in directly generating personalized physical activity recommendations. It will also help in validating results of other similar systems.

Regarding the scope of this study, using the weight management scenario, we only focused on a weight loss plan and did not address underweight and normal body weight cases. Therefore, the focus of recommendations is on *weight loss plan* rather than *weight gain plan* and *weight maintenance plan*. To smoothly tackle underweight and normal body weight cases, in Algorithm 1 and Algorithm 1, we display messages describing educational and motivational statements. However, we have not added details of these educational and motivational statements. In this study, we simply provide statements such as "eat high-fat foods or use protein powders to intake more calories", "maintain a reasonable amount of the exercise routine", and "you are doing good, keep it up" Furthermore, we also provide links to known online resources to educate the user about weight gain and weight maintenance.

We have partially implemented *PBR* with only a preference-based reasoning technique. The personalized recommendations are filtered one by one on the basis of the user's personal preferences and interests. A *PBR* system can provide more features; however, our interest lay only in filtering out irrelevant and unnecessary recommendations, and we therefore partially implemented the system in *HRM*. In the future, a complete user model may add more features to the system in terms of more personalized recommendations.

In the current technologically advanced era, a number of technologies (such as *CCTV* cameras etc.) can be used to monitor individuals' behaviors, specifically those of the elderly, to provide surveillance services [53]. This reduces the risk of a number of unobserved incidents that mostly occur among the elderly. The traditional surveillance mechanism can be avoided if "personal big data" are introduced to record the recognized daily physical activities of individuals and if analysis operations are enabled for them.

7. Conclusion and future work

This paper has presented *HRM* that effectively integrates multiple reasoning methodologies, such as *RBR, CBR,* and *PBR*, facilitating adoption and extension for different wellness services. The *hybrid-CBR* methodology achieves the objective of precise and specific personalized recommendation generation according to the user's specific needs. The application of *HRM* in a weight management scenario has proved that the precision, recall, accuracy, and f-score of personalized physical activity recommendations can be significantly improved if the integration of these methodologies is performed correctly. *Hybrid-CBR* achieves 0.97% precision, 0.94% recall, 0.94% accuracy, and a 0.95% f-score on the *TCB* with 64 test instance cases. Similarly, the *Type I* and *Type II* errors are significantly reduced. The significance of the proposed methodology is its preciseness in the recommendations made, which ensures personalization. Furthermore, the proposed methodology can be easily extended to other application areas, which will increase its worth.

In future, we plan to design and prepare an extended case base to host all the relevant information required for generation of personalized recommendations. This will enhance performance of *CBR* and *RBR* in *HRM*. Furthermore, it will reduce the complexity of the *HRM*. Moreover, the current personalized recommendations are merely based on *mode* and *intensity* features that lake '*amount*' and '*frequency*' characteristics. Hence, we plan to include these aspects in the future extensions. We also plan to extend *HRM* for recommending physical activity plans in dynamic way, using the user calendar and personal schedule information. Finally, we also plan to extend *PBR* part of the model by exploiting user model in comprehensive way to ensure more personalization.

Conflict of interest statement

Authors have no conflict of interests.

Acknowledgments

This work was supported by a grant from the Kyung Hee University in 2011 [KHU-20110219].

References

[1] O. Pakholok, The idea of healthy lifestyle and its transformation into health-oriented lifestyle in contemporary society, SAGE Open 3 (2013) 2158244013500281.
[2] M. Sharma, P.K. Majumdar, Occupational lifestyle diseases: an emerging issue, Indian J. Occup. Environ. Med. 13 (2009) 109.
[3] G.E. Fraser, D.J. Shavlik, Ten years of life: is it a matter of choice? Arch. Intern. Med. 161 (2001) 1645–1652.
[4] D.E.R. Warburton, C.W. Nicol, S.S.D. Bredin, Health benefits of physical activity: the evidence, Can. Med. Assoc. J. 174 (2006) 801–809.
[5] R. Ali, M.H. Siddiqi, S. Lee, B.H. Kang, KARE: A hybrid reasoning approach for promoting active lifestyle, in: Proceedings of the 9th International Conference of Ubiquitious Information Managment and Communication ICUIMC (IMCOM) Bali, Indonesia, 2015.
[6] M. Fahim, M. Idris, R. Ali, C. Nugent, B. Kang, E.-N. Huh, S. Lee, ATHENA: a personalized platform promote an active lifestyle wellbeing based physical, mental social health primitives, Sensors 14 (2014) 9313–9329.
[7] W.L. Haskell, I.M. Lee, R.R. Pate, K.E. Powell, S.N. Blair, B.A. Franklin, C.A. Macera, G.W. Heath, P.D. Thompson, A. Bauman, Physical activity and public health: updated recommendation for adults from the American College of Sports Medicine and the American Heart Association, Circulation 116 (2007) 1081.
[8] E. Verhagen, L. Engbers, The physical therapist's role in physical activity promotion, Br. J. Sport. Med. 43 (2009) 99–101.
[9] C. Centers for Disease, Prevention, Division of Nutrition, Physical Activity and Obesity, National Center for Chronic Disease Prevention and Health Promotion. Healthy weight, 2011.
[10] World Health, Organization, Global recommendations on physical activity for health, 2010.
[11] O. Banos, M. Bilal Amin, W. Ali Khan, M. Afzal, M. Ahmad, M. Ali, T. Ali, R. Ali, M. Bilal, m Han, J. Hussain, M. Hussain, S. Hussain, T.H. Hur, J.H. Bang, T. Huynh-The, M. Idris, D.W. Kang, S.B. Park, M. Siddiqui, L.B. Vui, M. Fahim, A.M. Khattak, B.H. Kang, S. Lee, An Innovative Platform for Person-Centric Health and Wellness Support, in: F. Ortuño, I. Rojas (Eds.), Bioinformatics and Biomedical Engineering, Springer International Publishing, Granada, Spain, 2015, pp. 131–140.
[12] J. Shen, J. Xing, M. Xu, Research on CBR-RBR fusion reasoning model and its application in medical treatment, in: Proceedings of the 21st International Conference on Industrial Engineering and Engineering Management Springer, 2014, pp. 431–434.
[13] D. Rossille, J.-Fo Laurent, A. Burgun, Modelling a decision-support system for oncology using rule-based and case-based reasoning methodologies, Int. J. Med. Inform. 74 (2005) 299–306.
[14] K. Khandelwal, D.P. Sharma, Hybrid Reasoning Model for Strengthening the problem solving capability of expert systems, Int. J. Adv. Comput. Sci. Appl. (IJACSA) 4 (2013).
[15] H.-J. Park, J.-S. Oh, D.-U. Jeong, K.-S. Park, Automated sleep stage scoring using hybrid rule-and case-based reasoning, Comput. Biomed. Res. 33 (2000) 330–349.
[16] G.J. Petot, C. Marling, L. Sterling, An artificial intelligence system for computer-assisted menu planning, J. Am. Diet. Assoc. 98 (1998) 1009–1014.
[17] S. Montani, R. Bellazzi, Supporting decisions in medical applications: the knowledge management perspective, Int. J. Med. Inform. 68 (2002) 79–90.
[18] K. Evans-Romaine, C. Marling, Prescribing exercise regimens for cardiac and pulmonary disease patients with CBR, in: Proceedings of the Workshop on CBR in the Health Sciences at 5th International Conference on Case-based Reasoning (ICCBR-03), Citeseer, 2003, pp. 45–62.
[19] P. Berka, NEST: a compositional approach to rule-based and case-based reasoning, Adv. Artif. Intell. 2011 (2011) 4.
[20] I. Bichindaritz, M.F. Siadak, J. Jocom, C. Moinpour, E. Kansu, G. Donaldson, N. Bush, M. Chapko, J.M. Bradshaw, K.M. Sullivan, CARE-PARTNER: a computerized knowledge-support system for stem-cell post-transplant long-term follow-up on the World-Wide-Web, in: Proceedings of the AMIA Symposium, American Medical Informatics Association, 1998, p. 386.
[21] J. Lieber, M. D'Aquin, P. Bey, B. Bresson, O. Croissant, P. Falzon, A. Lesur, Julien, V. Mollo, A. Napoli, The kasimir project: knowledge management in cancerology, in: Proceedings of the 4th International Workshop on Enterprise Networking and Computing in Health Care Industry-HealthCom, 2002, pp. 125–127.
[22] R. Van Meteren, M. Van Someren, Using content-based filtering for recommendation, in: Proceedings of the Machine Learning in the New Information Age: MLnet/ECML2000 Workshop, 2000, pp. 47–56.
[23] D.A. Davis, N.V. Chawla, N.A. Christakis, A.-Ls BarabÃ¡si, Time to CARE: a collaborative engine for practical disease prediction, Data Min. Knowl. Discov. 20 (2010) 388–415.
[24] S. Hasan, G.T. Duncan, D.B. Neill, R. Padman, Towards a collaborative filtering approach to medication reconciliation, AMIA Annu. Symp. Proc. (2008) 288–292.
[25] B. Satzger, M. Endres, W. KieÃŸling, A preference-based recommender system, in: K. Bauknecht, et al., (Eds.), E-Commerce and Web Technologies, Springer-Verlag Berlin, Heidelberg, 2006, pp. 31–40.
[26] W.M. Wang, C.F. Cheung, W.B. Lee, S.K. Kwok, Knowledge based treatment planning for adolescent early intervention of mental healthcare: a hybrid case based reasoning approach, Exp. Syst. 24 (2007) 232–251.
[27] S. Montani, P. Magni, R. Bellazzi, C. Larizza, A.V. Roudsari, E.R. Carson, Integrating model-based decision support in a multi-modal reasoning system for managing type 1 diabetic patients, Artif. Intell. Med. 29 (2003) 131–151.
[28] L. Nguyen, Z. Sun, A. Stranieri, S. Firmin, CWDM: A case-based diabetes management web system, in Proceedings of the 24th Australasian Conference on Information Systems (ACIS), RMIT University, 2013, pp. 1–11.
[29] R. Ali, J. Hussain, M.H. Siddiqi, M. Hussain, S. Lee, H2RM: A hybrid rough set reasoning model for prediction and management of diabetes mellitus, Sensors 15 (2015) 15921–15951.
[30] M.H. Nassabi H. op den Akker M. Vollenbroek. An ontology-based recommender system to promote physical activity for pre-frail elderly, in: Proceedings of the Mensch & Computer Walter De. Gruyter GmbH Co. KG, 2014, pp. 181–184.
[31] C. Marling G. Petot L. Sterling A CBR/RBR hybrid for designing nutritional menus, Multimodal Reasoning: Papers from them 1998 AAAI Spring Symposium 1998, AAAI Press, Menlo Park.
[32] D. Wang, M. Yu, C.B. Low, S. Arogeti, Model-Based Health Monitoring of Hybrid Systems, Springer Science+Business Media, New York, 2013.
[33] A. Aamodt, E. Plaza, Case-based reasoning: foundational issues, methodological variations, and system approaches, AI Commun. 7 (1994) 39–59.
[34] C. Marling, E. Rissland, A. Aamodt, Integrations with case-based reasoning, Knowl. Eng. Rev. 20 (2005) 241–245.
[35] O. Banos, M. Bilal Amin, W. Ali Khan, M. Afzal, T. Ali, Kang, B. H., S. Lee, The mining minds platform: a novel person-centered digital health and wellness framework, in: Proceedings of the 9th International Conference on Pervasive Computing Technologies for HealthcareIstanbul, Turkey, 2015.
[36] B.E. Ainsworth, W.L. Haskell, S.D. Herrmann, N. Meckes, D.R. Bassett Jr, C. Tudor-Locke, J.L. Greer, J. Vezina, M.C. Whitt-Glover, A.S. Leon, Compendium of physical activities: a second update of codes and MET values, Med. Sci. Sport. Exerc. 43 (2011) (2011) 1575–1581.

[37] D. Nassim, J. Marie-Christine, Clinical practice guidelines formalization for personalized medicine, Int. J. Appl. Evol. Comput. 4 (2013) 26–33.
[38] World Health Organization, BMI classification World Health Organization. URL: 〈http://apps.who.int/BMI/index.jsp〉, (2013).
[39] J.D. Robinson, S.M. Lupkiewicz, L. Palenik, L.M. Lopez, M. Ariet, Determination of ideal body weight for drug dosage calculations, Am. J. Health -Syst. Pharm. 40 (1983) 1016–1019.
[40] S. Singhal, L.I. Gordon, M.S. Tallman, J.N. Winter, A.O. Evens, O. Frankfurt, S.F. Williams, D. Grinblatt, L. Kaminer, R. Meagher, Ideal rather than actual body weight should be used to calculate cell dose in allogeneic hematopoietic stem cell transplantation, Bone Marrow Transplant. 37 (2006) 553–557.
[41] M. Jette, K. Sidney, G. Blümchen, Metabolic equivalents (METS) in exercise testing, exercise prescription, and evaluation of functional capacity, Clin. Cardiol. 13 (1990) 555–565.
[42] UK, Guidlines, Technical Report: Physical Activity Guidelines in the UK: Review and Recommendations, 2010.
[43] US, Guidelines, Physical activity guidelines advisory committee report, 2008, Washington, DC: US Department of Health and Human Services, 2008, pp. A1–H14.
[44] J.-F. Baget, Improving the forward chaining algorithm for conceptual graphs rules, in: Proceeding of the International Conference on Principles of Knowledge Representation and Reasoning, 2004, pp. 407–414.
[45] T. Cover, P. Hart, Nearest neighbor pattern classification, Inform. Theory IEEE Trans. 13 (1967) 21–27.
[46] J. Neyman, Outline of a theory of statistical estimation based on the classical theory sof probability, Philosophical Transactions of the Royal Society of London, Ser. A, Math. Phys. Sci. 236 (1937) 333–380.
[47] D.G. Altman, S.M. Gore, M.J. Gardner, S.J. Pocock, Statistical guidelines for contributors to medical journals, Br. Med. J. (Clin. Res. Ed.) 286 (1983) 1489–1493.
[48] R.A. Fisher, Statistical Methods for Research Workers, Genesis Publishing Pvt Ltd, Edinburgh, Tweeddale Court London, 1925.
[49] M. Han, Y.-K. Lee, S. Lee, Comprehensive context recognizer based on multimodal sensors in a smartphone, Sensors 12 (2012) 12588–12605.
[50] O. Banos, J. Bang, T. Hur, M.H. Siddiqi, T. Hyunh-The, L.B. Vui, A. Khan, A.T. W., C. Villalonga, S. Lee, Mining Human Behavior for Health Promotion, in: Proceedings of the 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC 2015) Milano, Italy, 2015.
[51] Microsoft, Microsoft Excel 2010 [computer software], Version 14.0.4760.1000 (32-bit), Microsoft Corporation, 2010.
[52] P. Pu, L. Chen, R. Hu, Evaluating recommender systems from the user's perspective: survey of the state of the art, User Model. User -Adapt. Interact. 22 (2012) 317–355.
[53] S.C. Mukhopadhyay, N.K. Suryadevara, R.K. Rayudu, Are Technologies Assisted Homes, Safer for the elderly? Pervasive Mob. Sens. Comput. Healthc. 2 (2012) 51–68.

Rahman Ali is a Ph.D. student in Ubiquitous Computing Laboratory (UCLab), Department of Computer Engineering, Kyung Hee University, South Korea. He got his M.Phil Degree in Computer Science from Department of Computer Science, University of Peshawar, Pakistan in 2009. He secured his M.Sc Degree (in Computer Science) from Hazara University, Mansehra, Pakistan in 2005 and Bachelor Degree from Govt. Jahanzeb College Saidu Sharif, Swat, Pakistan back in 2002. He has been a lecturer in Computer Science at University of Peshawar since 2009. He also served IIT, University of Science and Technology, Bannu as a lecturer in computer science and the Laboratoire d'Informatique de l'Universit du Maine, France as a research assistant. His current research includes Artificial Intelligence, Machine Learning, Knowledge Acquisition and Reasoning.

Muhammad Afzal is a Ph.D. student in Ubiquitous Computing Laboratory (UCLab), Department of Computer Engineering, Kyung Hee University, South Korea. He got his M.S Degree in Information Technology from School of Electrical Engineering and Computer Science, National University of Sciences and Technology (NUST), Pakistan in 2009. He secured his B.S degree in Computer Science from Kohat University of Science and Technology (KUST). He has been serving as proctor of HL7 International for conducting HL7 Standard Certification Exams in Pakistan since 2010. His current research includes Evidence-based Knowledge Acquisition and Systems, Applications of Machine Learning, Text Processing, and Information Extraction.

Maqbool Hussain received his M.S. degree from the School of Electrical Engineering and Computer Science, National University of Sciences and Technology, Pakistan in 2009. He is now enrolled in Ph.D. program in the Department of Computer Engineering of Kyung Hee University, Republic of Korea. His research interest includes Artificial Intelligence, Healthcare Interoperability Standards and Clinical Decision Support Systems.

Maqbool Ali received his M.Sc. and M.S. degrees with distinctions in computer science from the University of Agriculture, Faisalabad, and NUST, Islamabad, Pakistan, in 2001 and 2013, respectively. From 2002 to 2005, he served as an instructor in the field of computer science and from 2005 to 2014, he has been a researcher in the same field. Since March, 2014, he has joined Ubiquitous Computing Lab. to start his Ph.D. at the Dept. of Computer Engineering, Kyung Hee University, Republic of Korea. His research interests include data mining, machine learning, and natural language processing.

Muhammad Hameed Siddiqi is Ph.D. student in Ubiquitous Computing (UC) Lab, Department of Computer Engineering, Kyung Hee University, South Korea. He did his Master of Engineering from the Department of Computer Engineering, Kyung Hee University, South Korea in 2012, and Bachelor of Computer Science (Hons) from Islamia College University of Peshawar, N-W.F.P, Pakistan in 2007. He was a Graduate Assistant at Universiti Teknologi PETRONAS, Malaysia from 2008 to 2009. His research interest focuses on Image Processing, Pattern Recognition, Machine Intelligence, Activity Recognition, and Facial Expression Recognition.

Byeong Ho Kang is an Associate Professor of Computing and Information Systems at the University of Tasmania, Australia. He received his Ph.D. degree from the University of New South Wales, Sydney, in 1996. His research interest includes Artificial Intelligence, Expert System, Knowledge Acquisition and Internet Applications.

Sungyoung Lee received his B.S. from Korea University, Seoul, Korea. He got his M.S. and Ph.D. degrees in Computer Science from Illinois Institute of Technology (IIT), Chicago, USA in 1987 and 1991 respectively. He has been a professor in the department of Computer Engineering, Kyung Hee University, Korea since 1993. He is a founding director of the Ubiquitous Computing Laboratory, and has been affiliated with a director of Neo Medical ubiquitous-Life Care Information Technology Research Center, Kyung Hee University since 2006. Before joining Kyung Hee University, he was an assistant professor in the Department of Computer Science, Governors State University, llinois, USA from 1992 to 1993. His current research focuses on Ubiquitous Computing and Applications, Wireless Ad-hoc and Sensor Networks, Context-aware Middleware, Sensor Operating Systems, Real-Time Systems and Embedded Systems, Activity and Emotion Recognition. He is a member of ACM and IEEE.

Section 1.6

Supporting Layer (SL) Paper

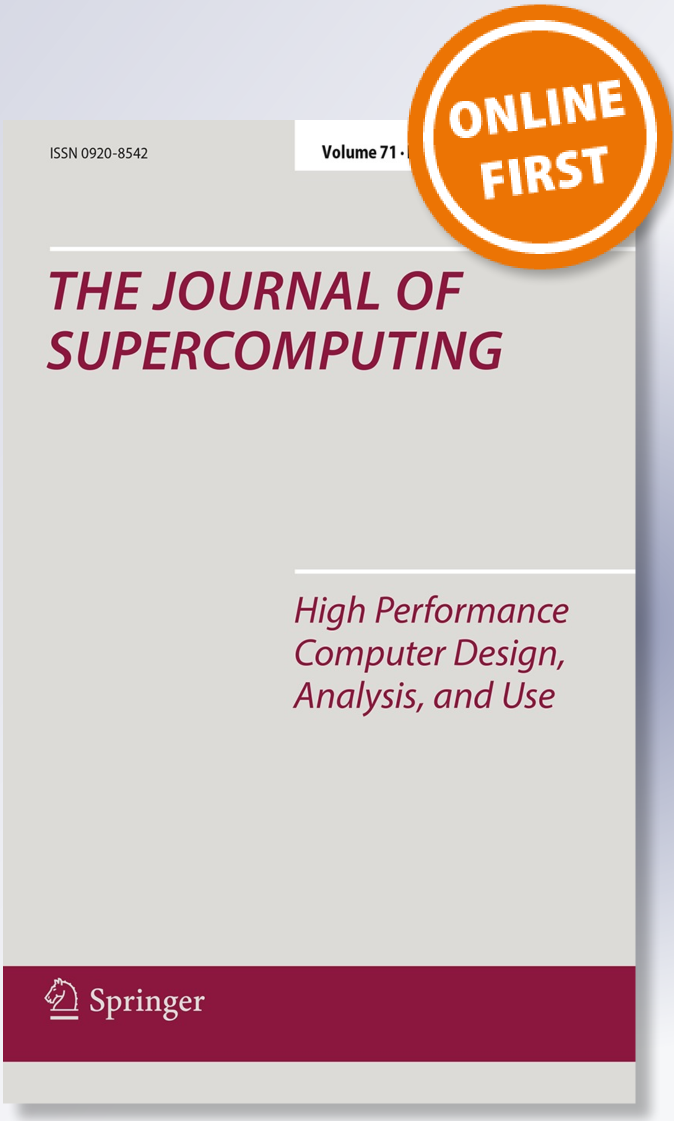
Health Fog: a novel framework for health and wellness applications

Mahmood Ahmad, Muhammad Bilal Amin, Shujaat Hussain, Byeong Ho Kang, Taechoong Cheong & Sungyoung Lee

The Journal of Supercomputing
An International Journal of High-Performance Computer Design, Analysis, and Use

ISSN 0920-8542

J Supercomput
DOI 10.1007/s11227-016-1634-x



Springer

J Supercomput
DOI 10.1007/s11227-016-1634-x

Health Fog: a novel framework for health and wellness applications

Mahmood Ahmad[1] · Muhammad Bilal Amin[1] · Shujaat Hussain[1] · Byeong Ho Kang[2] · Taechoong Cheong[1] · Sungyoung Lee[1]

Abstract In the past few years the role of e-health applications has taken a remarkable lead in terms of services and features inviting millions of people with higher motivation and confidence to achieve a healthier lifestyle. Induction of smart gadgetries, people lifestyle equipped with wearables, and development of IoT has revitalized the feature scale of these applications. The landscape of health applications encountering big data need to be replotted on cloud instead of solely relying on limited storage and computational resources of handheld devices. With this transformation, the outcome from certain health applications is significant where precise, user-centric, and personalized recommendations mimic like a personal care-giver round the clock. To maximize the services spectrum from these applications over cloud, certain challenges like data privacy and communication cost need serious attention. Following

✉ Mahmood Ahmad
rayemahmood@oslab.khu.ac.kr

✉ Taechoong Cheong
tcchung@khu.ac.kr

Muhammad Bilal Amin
mbilalamin@oslab.khu.ac.kr

Shujaat Hussain
shujaat.hussain@oslab.khu.ac.kr

Byeong Ho Kang
Byeong.Kang@utas.edu.au

Sungyoung Lee
sylee@oslab.khu.ac.kr

1 Ubiquitous Computing Lab, Department of Computer Engineering, Kyung Hee University, Seoul, Korea

2 School of Computing and Information Systems, University of Tasmania, Hobart, Australia

Published online: 29 January 2016

the existing trend together with an ambition to promote and assist users with healthy lifestyle we propose a framework of Health Fog where Fog computing is used as an intermediary layer between the cloud and end users. The design feature of Health Fog successfully reduces the extra communication cost that is usually found high in similar systems. For enhanced and flexible control of data privacy and security, we also introduce the cloud access security broker (CASB) as an integral component of Health Fog where certain polices can be implemented accordingly. The modular framework design of Health Fog is capable of engaging data from multiple resources together with adequate level of security and privacy using existing cryptographic primitives.

Keywords E-health and wellness applications · Big data · IoT · Cloud storage · Fog computing · Cloud access security broker

1 Introduction

In the past few years, health and wellness applications have emerged as a fast growing category of mobile applications. This increasing trend is considered as a prompt and useful resource for collecting users' data which are used for generating recommendations for a healthy lifestyle. Using smart phone features, applications like Microsoft Health, Apple Healthkit, Samsung S Health, and Google Fit collect users data by monitoring their daily activities, e.g., eating habits, sleeping patterns, and workout routines to generate certain recommendations which are helpful in maintaining a healthy lifestyle. To expand the spectrum of these recommendations, the data acquired from smart phones can be further synergized with other data resources like wearable sensors and a smart home environment. The processing on this integrated data acquired from various resources encompasses comprehension towards overall recommendations, thus creating the favorable environment to further engage other possible data resources including, but not limited to, social media and personal health records too. This expansion on data intake from various resources enables health and wellness applications to advise personalized and user-specific recommendations rather than giving general tips for a healthy lifestyle. Due to this reason the adaptation rate of such applications is on the rise with downloads in millions [1–4]. These applications offer a variety of features and plans like weight-loss, calorie-counter, women-health, and activity-recognition. To maximize the feature space of an application there is another trend of data-cross-sharing in which one application can share its data with the other, e.g., Fitocracy [5] can share its data with the RunKeeper [1].

The amount of data generated by smart phones and supportive need to include data from other resources make data volume enormous and its structure more complex. Although smart phones are sufficiently equipped with large memory size and computational resources for on-device storage and processing ability, however, to achieve increased battery life, data backup, centralized data storage, and to fulfill data-cross-sharing, there is another approach gaining momentum in a majority of applications which is the adoption of cloud services. Applications that used to store

Features / Systems	Data Source						Storage & Security				UI/UX			Services						Information Sharing			Knowledge Maintenance		
	Sensory Data	User profile	IOT	Other apps	Clinical data	Social media	User device	Cloud storage	Big data storage	Encrypted storage	User Experience	User Modelling	Adaptation of UI	Activity Recognition	Expert Svc	Wellness svc	Personal recommendation	Clinical svc	SDK/API	With other apps	Social media	Other users	Open Knowledge	Knowledge acquisition	Knowledge evolution
Google Fit	☑	☑	☑	☑	☒	☒	☑	☑	☑	☑	☒	☑	☒	☑	☑	☑	☒	☒	☑	☑	☑	☑	☑	☑	☒
Samsung S health	☑	☑	☑	☑	☒	☒	☑	☑	☑	☑	☒	☑	☒	☑	☒	☑	☑	☑	☑	☑	☑	☑	☑	☑	☒
Microsoft health	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☒	☑	☒	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑
Apple healthkit	☑	☑	☑	☑	☒	☒	☑	☑	☒	☑	☒	☑	☒	☑	☑	☑	☑	☒	☑	☑	☑	☑	☒	☑	☒
Open mhealth	☑	☑	☑	☑	☒	☒	☑	☑	☒	☒	☒	☑	☒	☑	☑	☑	☑	☒	☑	☒	☑	☑	☒	☑	☒
NoomCoach	☑	☑	☑	☑	☒	☒	☑	☑	☒	☒	☒	☑	☒	☑	☑	☑	☑	☒	☒	☑	☑	☒	☒	☑	☒
Argus	☑	☑	☑	☒	☒	☒	☑	☑	☒	☑	☒	☑	☒	☑	☑	☑	☒	☒	☒	☒	☑	☒	☒	☑	☒
Runtastic	☑	☑	☑	☑	☒	☒	☑	☑	☑	☑	☒	☑	☒	☑	☑	☑	☑	☒	☑	☑	☑	☒	☑	☑	☒
Runkeeper	☑	☑	☑	☑	☒	☒	☑	☑	☑	☑	☒	☑	☒	☑	☑	☑	☒	☒	☑	☑	☑	☒	☑	☑	☒
Zombie Run	☑	☑	☒	☒	☒	☑	☑	☑	☒	☒	☒	☑	☒	☑	☒	☑	☒	☒	☒	☒	☑	☒	☒	☑	☒
Mining Mind	☑	☑	☑	☒	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☑	☒	☒	☑	☑	☑	☑	☑

Fig. 1 Health and wellness applications: features survey and their categorization

data locally are swiftly moving towards centralized data storage through the cloud. Using cloud services, applications can conveniently afford inclusion of data from multiple resources and subsequent execution of substantial and complex computations. In addition, the application access also become more flexible and ubiquitous in comparison to the legacy approach where only a single device has the privilege to exploit application features on which it is installed. A user can now access these applications more seamlessly on a range of devices from mobile devices to fixed terminals and even across various applications. The problems of data loss in case of malfunctioning or stolen device is an allied advantage of using cloud services for convenient recovery.

Mobile health applications are being designed to promote fitness and track beneficial health metrics aiding healthy living. To monitor and track daily activities there is a huge rack of applications with variety of services. After selecting few of them, we organize and very briefly explain their features in certain groups to analyze growth and trend of offered services as shown in Fig. 1.

1.1 Data source

The integrated sensors within a smart phone are the initial and foremost sources for data acquisition. For personalized recommendations the sensory information is fused together with user profile like her age, gender, BMI index, weight, food preferences, and disability information. In addition, many applications also acquire and utilize data generated by physical devices including wearables and IoT (internet of things) like pedometer, heart rate monitor, surveillance camera, and smoke detector. The social media and clinical data are a gradual and influential addition being considered while generating the user-centric recommendations.

1.2 Storage and security

The data generated by health applications are usually stored on the user device. While opting for on-device storage, majority of applications lacks backup strategy of data and also do not employ encrypted data storage, thus considering physical possession of a device sufficient against data protection. This assumption of protection fails when a user is deprived of her device and loses it due to some reason. In this situation the device offers same privileged data access to any unauthorized user having the device possession. Also, the same situation leaves no alternate option for data recovery as well. Due to these reasons and realizing the necessity of protecting the applications data, its storage is deemed essential with encrypted storage along with opting for a storage location where data can be restored conveniently. Realization of these facts introduce the pivotal role of cloud computing as a growing trend amongst health applications. With ever increasing sophistication and integrated sensors in smart phones, health applications create enormous amount of data that are now more convenient to store and process with the induction of cloud. With this convenience, health applications can easily afford to engage and process data which are generated by other devices (IoT), wearables and third-party applications along with the data that are generated indigenously. This homogenization expands and evolves data into big data as current era of health applications deals with more volume, variety, and velocity of data used ever before. With this transformation, the data-driven health applications can now generate more realistic and personalized recommendations in comparison to legacy approach of generating general recommendation due to limited input of available data.

1.3 User interface (UI)/ user experience (UX)

The user interface (UI) has an important and profound role between the application and its user. The information on user interaction can be collected through user interface to maximize user experience. The importance of this aspect in modern applications is due to the fact that static interface lacks the ability to reflect user needs and satisfaction. In certain situations the interface has to be redesigned and adjusted with respect to specific needs of its users, e.g., increased font size for a user with week eyesight. In this regard, the information from user profile data (age, gender, week eyesight or any other disability) together with the contextual information (mode, time of the day, weather, etc.) can be used for customizing the application interface. The application designed on these features adapts itself according to the situation and makes user experience more comfortable while interacting with the application. In addition, involving user feedback through continuous monitoring of how she interacts with the application adds more refinement and perfection for enhanced user experience.

User experience is an evolutionary process to adapt and personalize the user interface. The personalization aspect is the most important factor and is achievable with user involvement where interface is dynamic and subject to user experience. The adaptive UI is managed by the UX, user feedbacks, and movement in the application which is then fused with the user profile data. The user movements encompass number of clicks, color schemes, text size, brightness and navigation within the application. All these parameters can be used in a user satisfaction module where suitable weight assignment is done for the collected parameters. The accumulated weight along with certain bounds in terms of threshold can be shared and further verified with domain expert. A similar system has been proposed by Hussain et al. [30] in which user feedback, web monitoring, and contextual information are used for adaptive UI with the help of UX.

1.4 Services

Majority of applications are capable of recognizing user activities (walking, jogging, running, sleeping) and present this information visually at the end of the day or whenever required by the user. Other than presenting the activities individually and their time duration, few applications also present the impact of one activity over the other, e.g., sleep efficiency with number of steps taken or changing heart rate due to running [3]. To promote the healthy lifestyle this information is also used to predict and anticipate certain trends, e.g., weight, calories deposit, or sleep efficiency. Sharing this information with health experts adds more comprehension and perfection towards user wellness with personalized recommendations. Clinical services and provisioning of SDK/API are also potential and growing features offered by few applications.

1.5 Information sharing

The data-driven landscape for health and wellness applications has revolutionized their core operations by expanding the boundaries for data/information sharing across various applications. Instead of making data silos as an independent island of information, its sharing is now more frequent involving other applications, devices, social media, and domain experts. The application of Fitcoracy [5] can sync with RunKeeper [1], DigifitIcardio [6] can use data generated by a heart-rate monitoring device Mio [7]. Similarly, users can share their data through social media within a community of their peers [4] and also with domain experts, e.g., clinicians, personal care-givers, and dietitians.

1.6 Knowledge maintenance

The knowledge which is derived from the personalized recommendations, user satisfaction level, and feedback analysis is preserved for later use as well as shared as an open knowledge. The discussion about knowledge acquisition, maintenance, and evolution in health applications is beyond the scope of this paper; therefore, it is just highlighted as a feature alone.

Figure 1 represents growing trend within health and wellness applications with respect to data and their utility. The importance of diverse data sources and induction of public cloud is obvious; however, it also introduces significant challenges of data heterogeneity resolution, implications associated with personal data on public cloud, and secure sharing of user data and information with other entities. With the ambition of promoting healthy living and considering the aforementioned challenges, we propose a framework aiming swift and secure services. The proposed framework employs the Fog Computing as a mediator layer between the system entities used for data acquisition and system consumers over public cloud. The rational behind using the Fog Computing is to bring computational resources close to data generating entities and preserving privacy aspect of users data during heterogeneity resolution of multivariate data. Considering data preprocessing as a demanding task with respect to computation and energy, it is not feasible to perform it on the user device; also, executing the same on public cloud can expose individual identity thus compromising individual identity with information exposure. These end-to-end limitations can be resolved with data preprocessing within Fog. At the same time, only relevant and useful information is uploaded into the public cloud after preprocessing, thus avoiding unnecessary communication overhead due to entire upload of data. In this paper we make following contributions:

- computational task for data heterogeneity resolution is done through Fog Computing[1] instead of public cloud, thus minimizing the information leakage during this process,
- services pool and data access policies and guidelines are staked in Health Fog giving more flexibility in terms of management and autonomous control,
- The processed and encrypted data are made available to authorized users obliviously using existing cryptographic standards.

2 Definitions and technical preliminaries

Before describing the HealthFog framework, we present some definitions and technical preliminaries.

2.1 Fog-computing

Fog computing, a Micro Datacenter paradigm, is a highly virtualized platform, which provides computation, storage, and networking services between the end nodes in an internet of things (IoT) and traditional clouds [8]. In contrast to the cloud, which is more centralized, Fog computing targets the services and applications with widely distributed deployments. Fog is aimed to deliver high-quality streaming to mobile nodes, like moving vehicles, through proxies and access points positioned accordingly, like, along highways and tracks. Fog suits applications with low latency requirements, emergency and healthcare-related services, video streaming, gaming, augmented reality, etc.

2.2 Cloud access security broker

Cloud Access Security Brokers (CASB) are quickly emerging as a must-have security solution for organizations looking to adopt cloud-based applications. CASBs are either on-premise, or cloud-based (or both) security policy enforcement points which are

[1] Depending upon the nature of proposed framework we will refer Fog computing as Health Fog.

placed between end users and the various cloud service providers. CASBs can inspect traffic, alert on anomalous behavior, and in most cases provide some level of data loss prevention (DLP) enforcement. Cloud Access Security Brokers can also consolidate multiple types of security policy enforcement, e.g., user authentication, single sign-on, authorization, credential mapping, device profiling, encryption, tokenization, logging, alerting, and malware detection/prevention [9,10].

2.3 Homomorphic encryption

Homomorphic encryption *HE* is a form of encryption where a specific algebraic operation performed on the plaintext is equivalent to another (possibly different) algebraic operation performed on the ciphertext. An encryption scheme is said to be additive homomorphic if and only if

$$E_H(m_1) \odot E_H(m_2) = E_H(m_1 + m_2),$$

where $\odot$ is an operator. Pascal Paillier cryptosystem [23] possesses the property of additive *HE* which is as follows:

- Key generation: Let $N = pq$ be the RSA-modulus and g be an integer of order αN module N^2 for some integer α. The public key is (N, g) and the private key is $\lambda(N) = lcm((p-1)(q-1))$.
- Encryption: The encryption of message $m \in Z_N$ is $E_h(m) = g^m r^N \bmod mod N^2$ where $r \in_R Z_N^*$
- Decryption: For ciphertext c, the message is computed from

$$m = \frac{L(c^{\lambda(N)} mod N^2)}{L(g^{\lambda(N)} mod N^2)}$$

A scheme is said to be multiplicative homomorphic if and only if

$$E_H(m_1) \odot E_H(m_2) = E_H(m_1 \times m_2)$$

The Goldwasser-Micali (GM) cryptosystem is a semantically secure scheme based on the quadratic residuosity problem. It has XOR homomorphic properties, in the sense that $E_H(b).E_H(b') = E(b \oplus b') mod N$, where b and b' are bits and N is the public key. A homomorphic encryption is said to be semantically secure if $E(H)$ reveals no information about m_1 and m_2; hence it is computationally infeasible to distinguish between the cases $m_1 = m_2$ and $m_1 \neq m_2$ [22].

Here is an example of how a homomorphic encryption scheme might work in cloud computing.

2.3.1 Example

Let us assume that a sensitive information comprising number 5 and 10 is encrypted and uploaded in the public cloud. For simplicity and understanding purpose, the corresponding ciphertext of 5 and 10 appears as 10 and 20 after applying the homomorphic encryption, i.e., the algorithm multiplies original values with 2. To perform any operation on these encrypted values the cloud will use 10 and 20, without knowing the original values. To utilize the computational services of cloud for addition purpose, it will use the features of homomorphic encryption and return the answer 30. On receiving end, the value of 30 will be decrypted as 15 using the decryption key.

2.4 Private comparison

Yaos classical millionaires problem involves two millionaires who wish to know who is richer. However, they do not want to find out inadvertently any additional information about each other's wealth. More formally, given two input values x and y, which are held as private inputs by two parties Alice and Bob, respectively. the problem is to securely evaluate the Greater Than (GT) condition through a predicate function f such that $f(x, y) = 1$ if and only if $x > y$, without exposing inputs. We used Fischlin protocol [11] for the private comparison because it allows comparing two ciphertexts encrypted with the GM cryptosystem using the same public key. Fischlin uses the GM-encryption scheme to construct a two-round GT protocol. The GM encryption scheme has the XOR, NOT, and re-randomization properties. They modified the scheme to get an AND property, which can be performed only once. The computation cost $O(n)$ for the server side is very efficient. Nevertheless, the overall computation cost for both the client and server sides are $O(nlogN)$, where N is the modulus. The scheme is as follows:

- Key generation: Let $N = pq$ be the RSA-modulus and z be a quadratic non-residue of Z_n^* with Jacobi symbol +1. The public key is (N, z) and the secret key is (p, q).
- Encryption: For a bit b, the encryption is $E(b) = z^r r^2 \bmod N$, where $r \in_R Z_N^*$.
- Decryption: For a ciphertext c, its plaintext is 1 if and only if c is a quadratic non-residue. If c is a quadratic residue in Z_N, c is quadratic residue in both Z_p^* and Z_q^*.
- xor-property: $E(b_1)E(b_2) = E(b_1 \oplus b_2)$.
- Not-property: $E(b) \times z = E(b \oplus 1 = E(\bar{b})$.
- Re-randomization: Randomization of ciphertext c can be done by multiplying an encryption of 0.

3 Proposed system overview

The proposed system is designed in a layered architecture comprising data generating entities DGE, Fog computing and end users. Hospitals and clinical institutes, smart home environment, and users equipped with wearables and smart-phone are main entities of DGE. For brevity, the following discussion is with respect to a single user 'Alice' who is a member in DGE and formally expressed as DGE_{Alice}. Alice is living in a smart home environment and using few wearables to monitor her daily activities as shown in Fig. 2. Her passion to achieve a healthy lifestyle is triggered

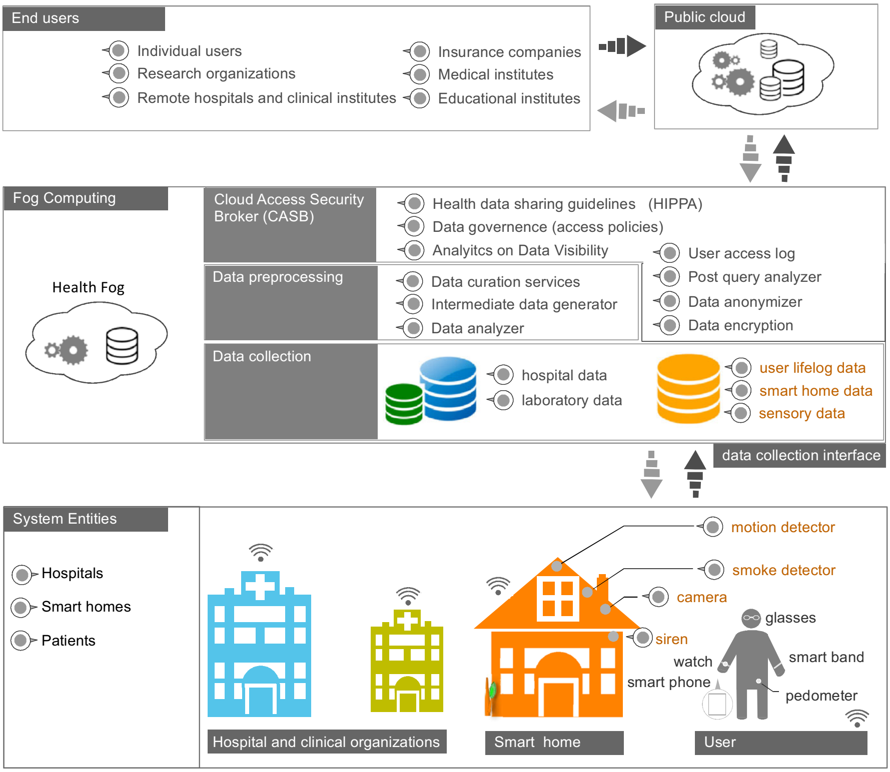


Fig. 2 Health Fog: system architecture

due to a recent discovery on her weight gain during a visit to a nearby hospital. The preliminary diagnosis highlights the sedentary life style caused by irregular routines for diet and sleep along with inadequate physical workout. The doctor advises her to change monotonous routines causing weight abnormality by inducing sufficient workout and that too with regular feedback and careful monitoring round the clock. Considering the high tendency of weight gain in her family tree, she also decides to avail expert services from a nutritionist for appropriate diet selection. To apprise the doctor on her lifestyle and to seek specialized advise from a domain experts, a mechanized system is required that should support data shareability with adequate security and high availability. With these requirements, Alice joins the Health Fog where hospitals, domain experts, and other people like her are already registered, collaboratively pursuing a healthy lifestyle.

4 Data outsourcing

Hospital and a user are two main data generating entities referred as DGE_H and DGE_U. A user living in a smart home environment using various devices d_i is represented as $DGE_U \langle d_1, d_2, \ldots, d_n \rangle$. For any device d_i that is required to be used as an active member of DGE_U is commissioned with Health Fog through Health Fog client stub. A client stub is an application running on a smart phone responsible for data uploads, activation or deactivation of a particular device, i.e., $d_i \in DGE$. Provisioning of activation and deactivation of a commissioned device is meant to control data uploads from user's side as per individual needs and concerns of privacy. The client stub is also responsible for providing end to end user services through personalized recommendations, alerts, statistical analysis on daily activities, and expert advice. Other than this independent link established between the client stub and the Health Fog, DGE_H also deposits its data by its own and separate link with the Health Fog. The data which are uploaded by the DGE_H are assumed to be preprocessed with the removal of identifiable information of a patient/visitor of that hospital.

5 Health Fog

The data arriving at the data collection layer of Health Fog are preserved under defined categories for user $DGE_U^{\circledast}$ and hospital $DGE_H^{\circledast}$. The data are then preprocessed for any missing value, noise reduction, erroneous values, duplicates, correct data labeling, and reduced as $DGE_U^{\odot}$ and $DGE_H^{\odot}$. The process Δ of data transformation for DGE_U and DGE_H is defined as

$$\begin{aligned} DGE_U^{\circledast}\circledast &\xrightarrow{\Delta_{Ш}^{\theta}} DGE_U^{\odot} \\ DGE_H^{\circledast}\circledast &\xrightarrow{\Delta_{Ш}^{\theta}} DGE_H^{\odot} \end{aligned} \tag{1}$$

Through process Δ, we claim two advantages of data reduction and privacy gain represented as $\Delta_{Ш}$ and Δ^{θ}, respectively. Collectively these two operations are represented as $\Delta_{Ш}^{\theta}$ in Eq. 1. With data reduction we mean the transformation of raw data into useful information, whereas the privacy gain is the information exposure of knowing individual identity while processing the raw data. Since these two operations are performed within Health Fog, communication overhead and information exposure for a curious cloud are marginally reduced which could appear otherwise.

The role of Cloud Access Security Brokers (CASB) is aimed to improve Health Fog security features by placing it in-between the public cloud and its consumers. CASB helps Health Fog to improve its monitoring, visibility, and control of user and data activity on public cloud. It also ensures that unauthorized parties do not gain access to corporate resources in the cloud by unification of consolidated polices ensuring their consistency and effectiveness at all operational frontiers. With CASB we propose following services.

5.1 Cloud access security broker (CASB)

The personal data which have to be shared through public cloud need to be in compliance with certain rules and regulations. The statutory body of these rules and regulations is accommodated within the CASB. For this purpose we refer to HIPPA as one of the policy guidelines while making data availability for the sharing purpose. Besides, polices that are defined at an organizational level can also be made part of

CASB to further meet local needs for data sharing and its access. Through CASB the visibility into user data and access is also provisioned through analytic and data visibility sub component. For each access that passes through CASB, it is logged for post analysis for anomaly detection in user query or data access pattern. The query log for post query analysis is used for discovering the data access trends. The access trends helps to improve data provisioning both in terms of refinement and improved security. To protect the individual identity from outsourced data without encryption, the anonymization is an option that can be used instead. With anonymization that involves techniques like generalization, suppression, and perturbation, to hide individual's identity is an option for data sharing without encryption, thus making data utility aspect more flexible with large number of users requiring no authentication. Although with anonymization the utility aspect on data becomes relatively minimal, however, it saves the cost of encryption/decryption and user management. For more details on anonymization readers may refer to [12,13].

The security and privacy issues on outsourced data in public cloud are not limited with the nature of data alone but also involve encryption techniques and access model as well. While achieving desirable security and privacy features on outsourced data, encrypted outsourcing in public cloud is highly recommended [14–16]. For this reason the data outsourcing from Health Fog into the public cloud is provisioned with encryption. Addressing the general issues while interacting with the encrypted data over cloud mainly includes user management and how the data are accessed or explored. With exploration, we refer retrieving the desired results from encrypted data against user query parameters. In this regard, the user access is controlled with the help of authorized credentials and a decryption key. In Health Fog, the process of user verification is done through CASB followed by request routing to public cloud where data are outsourced. The request arriving at cloud is then processed against the request parameters over encrypted data. Besides protecting the outsourced public data, protecting request parameters is equally important and emphasized. After processing the user query the result is sent back to the user where it is decrypted with authorized key. This whole process for data outsourcing, user accession, and response extraction is given in the following subsections.

5.1.1 Setup

Let $DGE_U^{\circledast}$ and $DGE_H^{\circledast}$ b the raw data collected from the user and hospital, respectively. After going through the process of Δ, these data are collectively represented as $D_*^{\odot}$

$$DGE_U^{\odot} + DGE_H^{\odot} = D_*^{\odot} \tag{2}$$

For quick data exploration on $D_*^{\odot}$, an index $\mathcal{I}$ is also created for all unique keywords against the outsourced data. Once $\mathcal{I}$ is generated, Health Fog initializes proxy re-encryption by generating Health Fog key (ω_o), user key (ω_u), and transformation key ($\omega_{o\rightarrow u}$). The owner key (ω_o) ensures the privacy of keywords within $\mathcal{I}$, whereas keyword frequencies are concealed with CASBs secret key (sk). The Health Fog key (ω_o) ensures the privacy of keywords within $\mathcal{I}$. The user key (ω_u) is used by the user to encrypt search criteria. The Health Fog only shares (ω_o) with the authorized users.

The transformation key ($\omega_{o \rightarrow u}$) is used by the cloud server to transform ciphertext (encrypted inverted index). Transformation of ciphertext ensures that the Health Fog does not need to outsource separate encrypted index for each authorized user. Each user is also provided with authorized credentials, i.e., user id and password that are used each time a user forwards her request through Health Fog.

5.1.2 Data outsourcing

For privacy aware data processing and oblivious request evaluation of user query on cloud, Health Fog encodes $\mathcal{I}_{kw_{0\ldots n}}$ using a publicly known encoding function denoted as $\mathcal{H}$, i.e., $\mathcal{H}\left(\mathcal{I}_{kw_{0\ldots n}}\right) \rightarrow \hat{\mathcal{I}}_{kw_{0\ldots n}}$. The encoded keywords $\left(\hat{\mathcal{I}}_{kw_{0\ldots n}}\right)$ are then encrypted with proxy reencryption algorithm using $\mathcal{E}_p\left(\hat{\mathcal{I}}_{kw_{0\ldots n}}\right) \rightarrow \hat{I}^{\omega_o}_{kw_{0,\ldots n}}$. To ensure that the cloud server cannot learn any information from the inverted index, Health Fog encrypts $\mathcal{I}_{f_{0,\ldots,n}}$ with CASB secret key, i.e., $\mathcal{E}\left(\mathcal{I}_{f_{0,\ldots,n}}, sk\right) \rightarrow \mathcal{I}^{sk}_{f_{0,\ldots,n}}$. After that, Health Fog encrypts (ω_u) with the public key of the user to whom it wants to grant searching capabilities over the outsourced data, i.e. $\mathcal{E}\left(\omega_u, k_{pub}\right) \rightarrow \omega_u^{k_{pub}}$. In a cloud storage system, outsourced data can be shared with multiple users each having its own access privileges over the outsourced data. With proxy reencryption Health Fog does not need to encrypt $\mathcal{I}_{kw_{0\ldots n}}$ separately to permit each authorized user to query $\hat{\mathcal{I}}^{\omega_o}_{kw_{0\ldots n}}$. An authorized user can submit its query encrypted with its proxy reencryption secret key $\left(\omega_{u_i}\right)$. Cloud server then transforms $\hat{\mathcal{I}}^{\omega_o}_{kw_{0\ldots n}}$ to $\hat{\mathcal{I}}^{\omega_{u_i}}_{kw_{0\ldots n}}$ using an appropriate transformation key ($\omega_o \rightarrow u_i$) provided by the Health Fog. Thus, the Health Fog only needs to encrypt $\hat{\mathcal{I}}^{\omega_o}_{kw_{0\ldots n}}$ once, and n authorized users can query it, without compromising privacy of the outsourced data.

5.1.3 Query generation

In order to privately search the cloud storage, a user obtains its proxy reencryption secret key from the Health Fog and deciphers it using the private key, i.e. $\mathcal{D}\left(\omega_u^{pub}, k_{pri}\right) = \omega_u$. The user then defines a search criteria $\left(\mathcal{C}_{kw_{0,\ldots l}}\right)$ that consist of a list of keywords $k_{w_0}, \ldots, k_{w_l}$. Then $\mathcal{C}_{kw_{0,\ldots l}}$ is encoded using a publicly known encoding function, i.e. $\mathcal{H}\left(\mathcal{C}_{kw_{0,\ldots l}}\right) \rightarrow \hat{\mathcal{C}}_{kw_{0,\ldots l}}$, where $\mathcal{H}$ is the same as used by the Health Fog during data outsourcing. To ensure confidentiality of the keywords, $\hat{\mathcal{C}}_{kw_{0,\ldots l}}$ is encrypted with proxy reencryption using the proxy reencryption secret key, i.e. $\mathcal{E}_p\left(\hat{\mathcal{C}}_{kw_{0,\ldots l}}, \omega_u\right) = \hat{\mathcal{C}}^{w_u}_{kw_{0,\ldots l}}$.

Once privacy of the search criteria is assured, it is send to CASB who uses it to model oblivious search query. On receiving $\hat{\mathcal{C}}^{w_u}_{kw_{0,\ldots l}}$ the CASB defines a polynomial ($P(x)$), such that each element of $\hat{\mathcal{C}}^{w_u}_{kw_{0,\ldots l}}$ is a root of $P(x)$, i.e.$P(x \in \hat{\mathcal{C}}^{w_u}_{kw_{0,\ldots l}}) = \sum_{i=0}^{l} \alpha_i x^i = 0$.

Once $P(x)$ is defined in accordance with $\hat{\mathcal{C}}^{w_u}_{kw_{0,\ldots l}}$, the CASB then initializes homomorphic encryption by generating a public key $\left(\sigma_{pk}\right)$ and secret key (σ_{sk}). The CASB encrypts the coefficients $\left(\alpha_{o,\ldots l}\right)$ of $P(x)$ with homomorphic encryption algorithm using σ_{sk}, i.e. $\mathcal{E}_H\left(\alpha_{0,\ldots l}, \sigma_{sk}\right) = \alpha^{\sigma_{sk}}_{0,\ldots l}$. Subsequently $\alpha^{\sigma_{sk}}_{0,\ldots l}$ and σ_{pk} are transferred to the cloud server. Encrypted coefficients $\alpha_{0,\ldots l}^{\sigma_{sk}}$ are used to execute search query over encrypted inverted index $\left(\hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_o}\right)$. In the context of search over encrypted data, set intersection can be used to execute search query by matching search criteria with the inverted index.

5.1.4 Searching

Cloud server hosts the encrypted inverted index as encrypted keywords $\left(\hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_o}\right)$ and their concealed frequencies $\mathcal{I}_{f_{0,\ldots,n}}^{sk}$ along with the outsourced data $\mathcal{F}$. Encrypted query $\alpha_{0,\ldots l}$ submitted by the CASB is evaluated against $\hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_o}$. On receiving $\alpha_{0,\ldots l}^{sk}$, cloud server transforms $\hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_o}$ with the respective users transformation key $\omega_{o\rightarrow u}$, provided by the Health Fog, i.e. $\mathcal{T}_p\left(\hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_o}, \omega_{o\rightarrow u}\right) \rightarrow \hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_u}$. Once the encrypted index is transformed, cloud server defines a polynomial $P(y)$, using each element of $alpha_{0,\ldots l}$ as a coefficient of $P(y)$. It then computes oblivious value $\left(\Delta_{y_i}\right)$ by evaluating $r.P(y_i)$, where $y_i \in \hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_u}$ and r is a random number, i.e. $\Delta_{y_i} = r.P(y_i)$.

As the query is concealed using homomorphic encryption, cloud server cannot learn any information from $P\left(y_i \in \hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_u}\right)$. Once the cloud server has evaluated $P\left(y_{0,\ldots,n} \in \hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_u}\right) = \Delta_{y_{0,\ldots,n}}$, it replies back the query evaluation resultlist of oblivious values along with the concealed keyword frequencies to the CASB, i.e. $\Delta_{y_{0,\ldots,n}}, \mathcal{I}_{f_{0,\ldots,n}}^{sk}$.

5.1.5 Response extraction

On receiving the cloud server response $\Delta_{y_{0,\ldots,n}}, \mathcal{I}_{f_{0,\ldots,n}}^{sk}$, CASB decrypts the oblivious values using the homomorphic secret key, i.e. $\mathcal{D}_H\left(\Delta_{y_i}, \sigma_{sk}\right) = \psi_i$, where ψ_i can be zero or a random number. As the search query was modeled as a polynomial having roots equal to the concealed search criteria, i.e. $P(x \in \hat{\mathcal{C}}_{kw_{0\ldots l}}^{\omega_u}) = \sum_{i=0}^{l} \alpha_i x^i$, query evaluation at cloud server can result either in a zero or a non-zero value shown in Eq. 3.

$$P(y_i) = \begin{cases} \psi_i = 0 \; if \left\{ y_i | y_i \in \hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_u} \wedge y_i \in \mathcal{C}_{kw_{0,\ldots,l}}^{\hat{\omega_u}} \right\} \\ \psi_i = 0 \; if \left\{ y_i | y_i \in \hat{\mathcal{I}}_{kw_{0,\ldots,n}}^{\omega_u} \wedge y_i \notin \mathcal{C}_{kw_{0,\ldots,l}}^{\hat{\omega_u}} \right\} \end{cases} \tag{3}$$

Zero value reveals that inverted index contains keyword that matches with the concealed search criteria specified by the user, i.e. $\hat{\mathcal{C}}_{kw_i}^{\omega_u} \in \mathcal{I}_{kw_{0\ldots n}}^{\hat{\omega_u}}$, whereas non-zero reveals that concealed search criteria do not match with any of the keyword in inverted index; consequently, CASB recovers r. Once encrypted keywords are identified, CASB deciphers the corresponding frequency index using the secret key, i.e.$\mathcal{D}_S\left(\mathcal{I}_{fi}^{sk}, sk\right) \rightarrow \mathcal{I}_{fi}$. After sorting the encrypted keywords based on their frequency count, CASB replies oblivious results to the user.

On receiving the CASBs response, a user deciphers the search criteria using its proxy reencryption secret key. Through decryption the user learns the keyword that matches with the encrypted index, i.e. $\mathcal{D}_P\left(\hat{\mathcal{C}}^{\omega_u}_{kw_{0\ldots k}}, \omega_u\right) = \widehat{\mathcal{C}_{kw_{0\ldots k}}}$, where k is the number terms that are identical between $\widehat{\mathcal{C}_{kw_{0\ldots l}}}$ and $\widehat{\mathcal{I}^{\omega_u}_{kw_{0\ldots n}}}$. During the query evaluation cloud server learns nothing about the inverted index or the search criteria, however; it accurately evaluates the search query and replies back the oblivious response.

6 Evaluation and results

For experiment purpose we have used the Samsung Galaxy S-III smart phone. The sensory input of accelerometer and GPS has been used for activity detection and subsequent recommendations by the domain experts, i.e., doctor and the nutritionist. The data generated by these sensors of smart phone are retrieved on a 3-s interval. For optimal communication between the client stub and Health Fog, the 3-s interval data are batched over a minute and sent to the Health Fog after 60 s. For activity detection, these data are deposited on Health Fog through client stub along with other sensory inputs collected from the smart home environment. The user data generated within the hospital like history, prescriptions, etc., are also centralized into the Health Fog. After collecting all data resources it is preprocessed and curated for activity detection and appear as intermediate data. This intermediate data are then shared with the doctor and the nutritionist or as preferred by the user. The calorie meter, activity detection, activity detail, and personalized recommendations are also shared with the users.

The data acquired from accelerometer and GPS followed by the detected activity against this data are shown in Fig. 3. Due to space limit the visibility of raw data has been restricted with fewer rows only followed by their transformation into useful information of activity on Health Fog. Depending upon the activity detection, the burned calories are calculated and logged. The final data for activity detection and calories are then deposited on the cloud and shared with the Hospital and nutritionist. The authorized entities can monitor the selected data as required. Figure 4 shows the total activities performed during a day to the user along with recommendations by the doctor and the nutritionist. The total amount of calory intake is shown as

Accelerometer Data

User Device ID	X-coordinate	Y-coordinate	Z-coordinate	timestamp
25	16	14	14	11/26/2014 8:16
25	16	14	15	11/27/2014 8:16
25	16	14	15	11/28/2014 8:16
25	16	14	17	11/29/2014 8:16
25	16	14	18	11/30/2014 8:16
25	16	14	21	12/1/2014 8:16
25	16	14	22	12/2/2014 8:16
25	16	14	24	12/3/2014 8:16
25	16	14	25	12/4/2014 8:16
25	16	14	27	12/5/2014 8:16
25	16	14	28	12/6/2014 8:16
25	16	14	29	12/7/2014 8:16
25	16	14	29	12/8/2014 8:16
25	16	14	32	12/9/2014 8:16

GPS Data

User Device ID	latitude	longitude	speed	timestamp
25	12	16	16	11/26/2014 8:16
25	12	16	18	11/27/2014 8:16
25	12	16	18	11/28/2014 8:16
25	12	16	19	11/29/2014 8:16
25	12	16	19	11/30/2014 8:16
25	12	16	21	12/1/2014 8:16
25	12	16	22	12/2/2014 8:16
25	12	16	23	12/3/2014 8:16
25	12	16	23	12/4/2014 8:16
25	12	16	24	12/5/2014 8:16
25	12	16	25	12/6/2014 8:16
25	12	16	25	12/7/2014 8:16
25	12	16	26	12/8/2014 8:16
25	12	16	26	12/9/2014 8:16

Activity Recognition

User Device ID	Activity	Start Time	End Time
25	Cycling	11/26/2014 8:16	11/26/2014 8:20
25	Jogging	11/26/2014 8:35	11/26/2014 8:55
25	Running	11/26/2014 9:35	11/26/2014 9:40
25	Jogging	11/26/2014 10:00	11/26/2014 10:10
25	Walking	11/26/2014 10:20	11/26/2014 10:25
25	Walking	11/27/2014 9:35	11/27/2014 9:50
25	Cycling	11/27/2014 10:00	11/27/2014 10:20

Fig. 3 Raw data for accelerometer and GPS along with activity recognition data

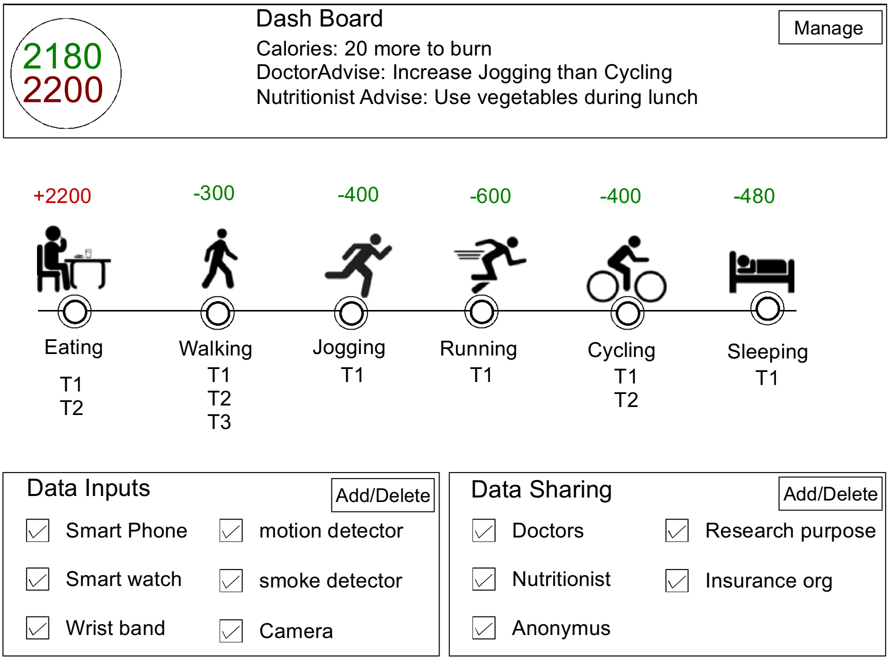


Fig. 4 Health Fog: user interaction

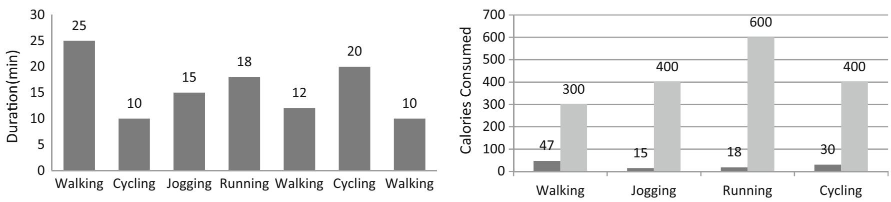


Fig. 5 Calories' consumption

2200, whereas calorie consumption by each activity is shown separately. The number of times an activity is performed during a day is shown from ($T_1, T_2, \ldots, T_n$). The activities performed along with the food intake are then shared with the doctor on regular basis for appropriate advice and cure. The data input resources can be selected as per user's own preferences. Likewise, the control of sharing user's personal data (without disclosing her identity) with other entities is also under the discretion of user. The daily breakup of information comprising calories' break down, activity detection, duration, and recommendations by the doctors and nutritionist is shown to the user as appearing in Fig. 5.

The information which is uploaded from the Health Fog to the cloud is evaluated on local machine and Google App Engine [17] as cloud server. The local machine specification comprises on Intel(R) Core(TM) i3 processor and 4GB of RAM, whereas Microsoft(R) Windows7(TM) X 64bit is the OS installed.

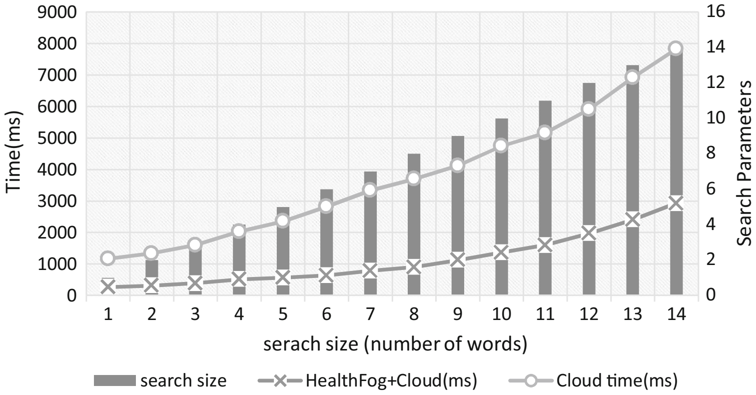


Fig. 6 Comparison: execution time

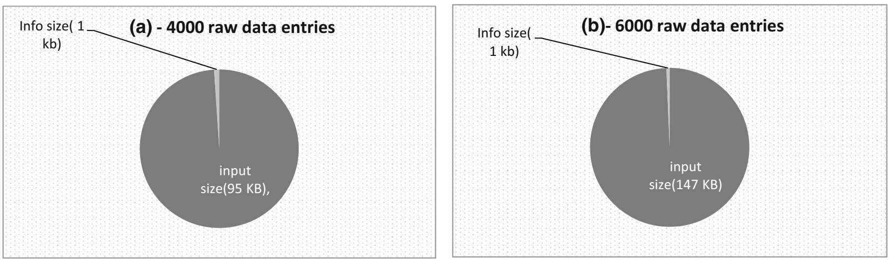


Fig. 7 Data reduction

The performance gain with respect to execution time on the information which is preprocessed through Health Fog and without its preprocessing is shown in Fig. 6. Here different selection parameters like user age, gender, BMI index have been used for the test purpose. This performance gain is achieved due to the transformation of raw data into required information thus eliminating unnecessary communication. The data reduction is shown in Fig. 7 where 4000 raw data entries occupying 95 Kb of space are replaced with only 1 Kb of actual information. Similarly, 147 Kb of space for 6000 Kb raw data entries is again transformed into 1 Kb of actual information thus saving notable communication overhead due to unnecessary movement of raw data.

7 Discussion

The ever-growing demand for ubiquitous healthcare systems to improve human health and well-being has suitably engaged advanced technologies, namely cloud computing, IoT, sensory devices, and wearables. With this adoption, the opportunity of including multivariate data in healthcare applications has been made possible for personalized and patient centric services. For real-time provisioning of data from this technology, their omnipresence has been made available through smart phones and wearables for continuous monitoring. With the development of internet of things (IoT) and smart devices the data nodes are increasing at an exponential rate inviting big data to be stored and processed on cloud [18] due to certain reasons. These reasons primarily address the issues for central storage, complex computation, and information sharing. In an effort to optimize the whole process, Fog computing, which is an evolving paradigm shift, can facilitate such systems as a gateway between the end user and cloud. The constructive control of Fog computing effectively minimize the unnecessary communication from data generating nodes to cloud. In addition, certain policies and rules can be integrated within the Fog to ensure data privacy and security. In conventional applications that are solely dependent on cloud, the latency increases and required quality of services degrades. For this purpose, Fog computing as an intermediary layer between the cloud and end user plays its pivotal role for low latency, better visualization, and context awareness [8,19]. Fog computing can also increase the security in the public cloud. A trustworthy cloud provider is necessary but accidents still tend to happen and information gets lost. The exposure risk for information leakage can be limited which is sent to the cloud by initial processing on the fog [20]. Complex security challenges are being faced by big data and cloud computing. The most promising way to deal with these challenges is fog computing [21].

In recent years Cloud and Fog computing have drawn profound attention in e-health related systems and applications. In [24], authors present a cloud computing solution for patient data collection in health care institutions. The proposed system uses sensors attached to medical equipment to collect patient data and sends the data to cloud for providing ubiquitous access. Introducing smart gateways Chen et al. [25] introduce a smart gateway for health care system using wireless sensor network. The proposed gateway acting as a bridge between wireless sensor network and public communication networks has a data decision system. In [26], authors propose a mobile gateway for ubiquitous health care system using ZigBee and Bluetooth. The gateway presents various services such as alarms and analysis of medical data. Yang et al. present a personal health monitoring gateway based on smartphone [27]. The proposed gateway uses a Bluetooth interface to upload gathered data to remote servers. In [28] mobile fog is introduced for future internet applications which will be geographically distributed and are latency sensitive. Mobile fog consists of different devices like smartphones, smart watches, tablets, and even drones. A case study given by Tuan et al. [29] highlights the feasibility of IoT to monitor human health in real-time using ubiquitous health monitoring systems. Their proposed health monitoring system exploit the concept of Fog computing at smart gateways providing advanced techniques and services such as embedded data mining, distributed storage, and notification service at the edge of network.

8 Conclusion and future work

In this paper we have proposed a framework of Health Fog for sharing and processing health-related information based on data acquired from multiple resources. We have used Fog computing features as an intermediary layer between the cloud server and end user to avoid unnecessary flow of information and better control over data privacy and security while processing and sharing the information. In future extension to Health Fog, we will incorporate the social media as an input data resource and also we will build the knowledge reservoir accrued over single instances of Health Fog users. With this addition the feature scale of Health Fog will be equipped with more usability and shareability.

Acknowledgments This work was supported by a grant from the Kyung Hee University in 2013[KHU-20130439].

References

1. FitnessKeeper (2015) Runkeeper everyone every run. http://runkeeper.com/
2. Noom (2015) Noom help you to make a change. https://www.noom.com/
3. Azumio (2015) Argus quantify your day-to-day. http://www.azumio.com/s/argus/index.html
4. myfitnesspal (2015) Lose weight with myfitnesspal. https://www.myfitnesspal.com/
5. Fitocracy (2015) Fitocracy. https://www.fitocracy.com/
6. Digifit (2015) Digifit personal engagement solution. http://www.digifit.com/personal-solutions.html
7. MIOLink (2015) Mio train with heart. http://www.mioglobal.com
8. Bonomi F, Milito R, Zhu J, Addepalli S (2012) Fog computing and its role in the internet of things. In: Proceedings of the first edition of the MCC workshop on Mobile cloud computing, ACM, pp 13–16
9. Firstbrook P, MacDonald N (2015) The growing importance of cloud access security brokers. https://www.gartner.com/doc/2032015/growing-importance-cloud-access-security
10. HP cloud access security protection platform. Business white paper : Case for a cloud access security brokers. hp.com/go/CloudAccessSecurity, (2015)
11. Fischlin M (2001) A cost-effective pay-per-multiplication comparison method for millionaires. In: Topics in Cryptology CT-RSA 2001, Springer. pp 457–471
12. Sweeney Latanya (2002) Achieving k-anonymity privacy protection using generalization and suppression. Int J Uncertain Fuzziness Knowl Based Syst 10(05):571–588
13. Sweeney Latanya (2002) k-anonymity: a model for protecting privacy. Int J Uncertain Fuzziness Knowl Based Syst 10(05):557–570
14. Evdokimov S, Günther O (2007) Encryption techniques for secure database outsourcing. In: Computer Security–ESORICS 2007, Springer, pp 327–342
15. De Capitani S, Di Vimercati FS, Jajodia S, Paraboschi S, Samarati P (2007) A data outsourcing architecture combining cryptography and access control. In: Proceedings of the 2007 ACM workshop on Computer security architecture, ACM, pp 63–69
16. Curino C, Jones EPC, Popa RA, Malviya N, Wu E, Madden S, Balakrishnan H, Zeldovich N (2011) Relational cloud: a database-as-a-service for the cloud
17. Google (2013) Google app engine. https://cloud.google.com/products/app-engine
18. Shi Y, Ding G, Wang H, Roman HE, Lu S (2015) The fog computing service for healthcare. In: 2015 2nd International Symposium on Future Information and Communication Technologies for Ubiquitous HealthCare (Ubi-HealthTech), IEEE, pp 1–5
19. Stojmenovic I, Wen S (2014) The fog computing paradigm: Scenarios and security issues. In: 2014 Federated Conference on Computer Science and Information Systems (FedCSIS), IEEE, pp 1–8
20. Stolfo SJ, Salem MB, Keromytis AD (2012) Fog computing: mitigating insider data theft attacks in the cloud. In: 2012 IEEE Symposium on Security and Privacy Workshops (SPW), IEEE, pp 125–128
21. Yannuzzi M, Milito R, Serral-Gracia R, Montero D, Nemirovsky M (2014) Key ingredients in an iot recipe: fog computing, cloud computing, and more fog computing. In: 2014 IEEE 19th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), pp 325–329
22. Paillier P (2000) Trapdooring discrete logarithms on elliptic curves over rings. In: Okamoto T (ed) Advances in CryptologyASI—ACRYPT 2000. Springer, Berlin, Heidelberg, pp 573–584
23. Paillier P (1999) Public key cryptosystems based on composite degree residuosity classes. In: Proceedings of the 17th international conference on Theory and application of cryptographic techniques, EUROCRYPT'99, Springer-Verlag, Berlin, Heidelberg, pp 223–238

24. Rolim CO, Koch FL, Westphall CB, Werner J, Fracalossi A, Salvador GS (2010) A cloud computing solution for patient's data collection in health care institutions. In: Second International Conference on eHealth, Telemedicine, and Social Medicine, 2010. ETELEMED'10, IEEE, pp. 95–99
25. Chen Y, Shen W, Huo H, Xu Y (2010) A smart gateway for health care system using wireless sensor network. In: 2010 Fourth International Conference on Sensor Technologies and Applications (SENSORCOMM), IEEE, pp 545–550
26. Laine TH, Lee C, Suk H (2014) Mobile gateway for ubiquitous health care system using zigbee and bluetooth. In: 014 Eighth International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS), 2. IEEE, pp 139–145
27. Yang S, Gerla M (2011) Personal gateway in mobile health monitoring. In: 011 IEEE International Conference on Pervasive Computing and Communications Workshops (PERCOM Workshops), 2. IEEE, pp 636–641
28. Hong K, Lillethun D, Ramachandran U, Ottenwälder B, Koldehofe B (2013) Mobile fog: a programming model for large-scale applications on the internet of things. In: Proceedings of the second ACM SIGCOMM workshop on Mobile cloud computing. ACM, pp 15–20
29. Gia TN, Rahmani MJAM, Westerlund T, Liljeberg P, Tenhunen H (2015) Fog computing in healthcare internet-of-things: a case study on ecg feature extraction
30. Hussain J, Khan WA, Afzal M, Hussain M, Kang BH, Lee S (2014) Adaptive user interface and user experience based authoring tool for recommendation systems. In: Ubiquitous Computing and Ambient Intelligence. Personalisation and User Adapted Services. Springer, pp 136–142

Section 2

Mining Minds Feature Evolution

Mining Minds Version 3

Data Curation Layer 3.0
Features Evolution

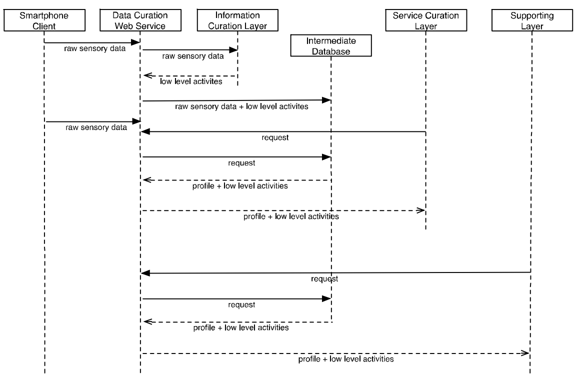
Principal Investigator:
Professor Sungyoung Lee

경희대학교 KYUNG HEE UNIVERSITY KITECH 한국생산기술연구원 POSTECH 포항공과대학교 UTAS 녹십자헬스케어 TAPACROSS U2 system User & Usability system

Data Acquisition and Synchronization

Benefit | Limitation

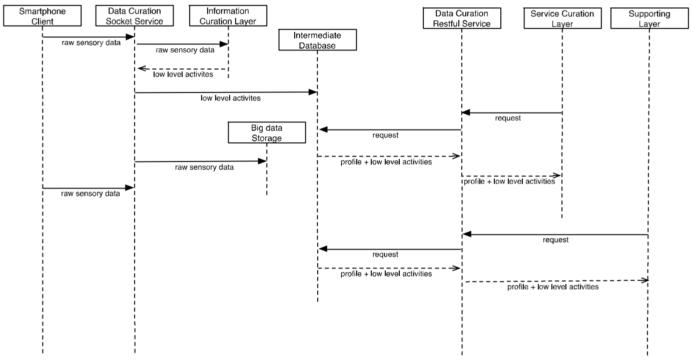
V1. SOAP-based Web services → Faster Implementation → Performance bottleneck



2

Data Acquisition and Synchronization

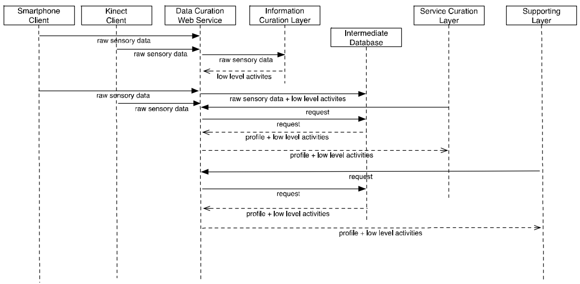
	Benefit	Limitation
V1. SOAP-based Web services	→ Faster Implementation	→ Performance bottleneck
V1.5 Sockets + Restful Services	→ Performance	→ Scalability (Blocking Communication)



3

Data Acquisition and Synchronization

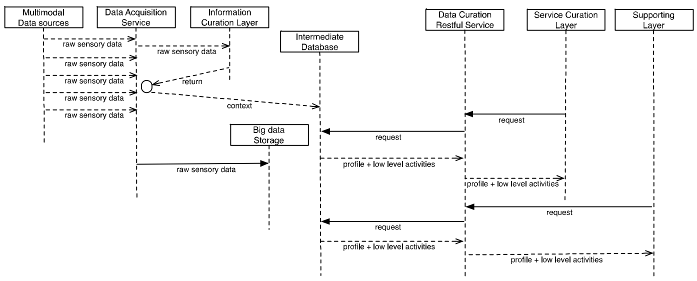
	Benefit	Limitation
V1. SOAP-based Web services	→ Faster Implementation	→ Performance bottleneck
V1.5 Sockets + Restful Services	→ Performance	→ Scalability (Blocking Communication)
V2 Sockets + Restful Services	→ Performance	→ Scalability (Blocking Communication)



4

Data Acquisition and Synchronization

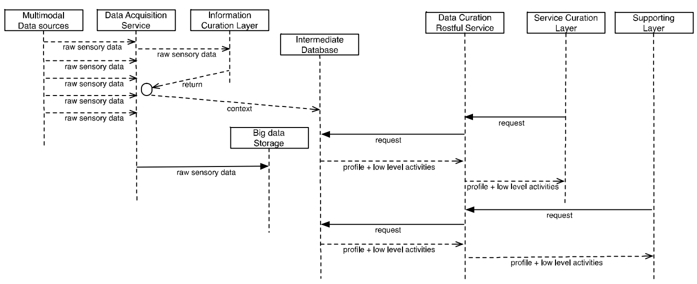
	Benefit	Limitation
V1. SOAP-based Web services →	Faster Implementation →	Performance bottleneck
V1.5 Sockets + Restful Services →	Performance →	Scalability (Blocking Communication)
V2 Sockets + Restful Services →	Performance →	Scalability (Blocking Communication)
V2.5 Node.js + Restful Services →	Performance + Non-blocking Communication →	Limited data sources support



5

Data Acquisition and Synchronization

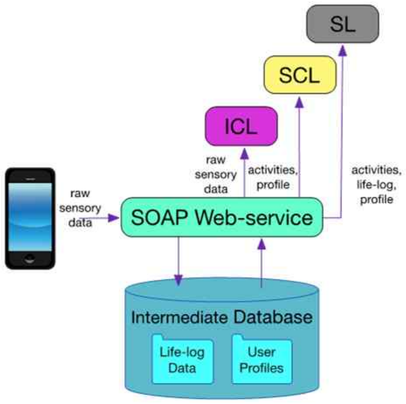
	Benefit	Limitation
V1. SOAP-based Web services →	Faster Implementation →	Performance bottleneck
V1.5 Sockets + Restful Services →	Performance →	Scalability (Blocking Communication)
V2 Sockets + Restful Services →	Performance →	Scalability (Blocking Communication)
V2.5 Node.js + Restful Services →	Performance + Non-blocking Communication →	Limited data sources support
V3.0 Node.js + Restful Services →	Performance + Non-blocking Communication + IoT Support	



6

Big Data and Intermediate Database

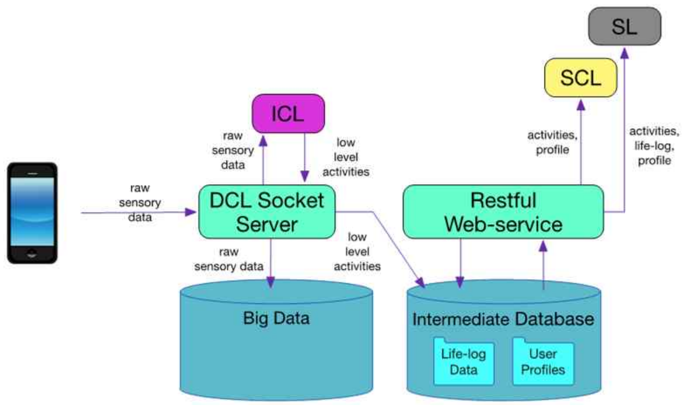
	Benefit		Limitation
V1. Only Intermediate DB →	Faster Implementation	→	Intermediate DB Stress (Performance Bottleneck)



7

Big Data and Intermediate Database

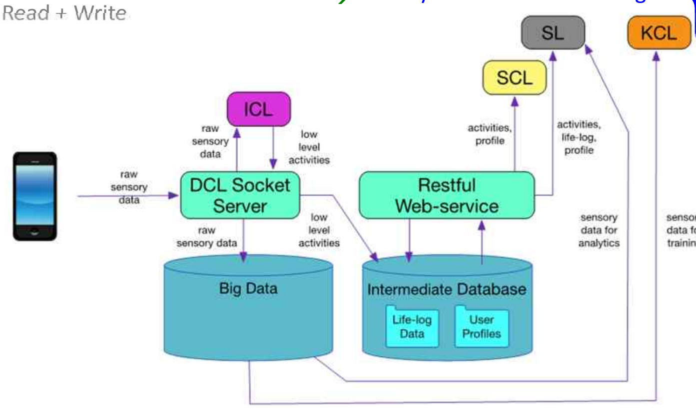
	Benefit		Limitation
V1. Only Intermediate DB →	Faster Implementation	→	Intermediate DB Stress (Performance Bottleneck)
V1.5 Big Data For Sensory Data Write →	Performance	→	No Analytics on Sensory Data + No Data for Training



8

Big Data and Intermediate Database

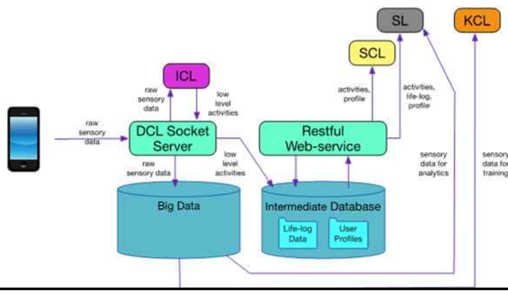
	Benefit	Limitation
V1. Only Intermediate DB →	Faster Implementation →	Intermediate DB Stress (Performance Bottleneck)
V1.5 Big Data For Sensory Data Write →	Performance →	No Analytics on Sensory Data + No Data for Training
V2.x Big Data For Sensory Data Read + Write →	Analytics + Data for Training	1. Upfront Requirement Analysis for Big Data Queries 2. File distribution structure for Big Data Fragmentation 3. Ontology / Model for faster file location indexing



9

Big Data and Intermediate Database

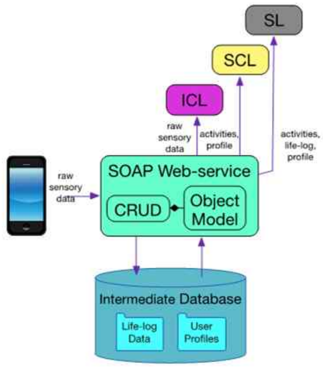
	Benefit	Limitation
V1. Only Intermediate DB →	Faster Implementation →	Intermediate DB Stress (Performance Bottleneck)
V1.5 Big Data For Sensory Data Write →	Performance →	No Analytics on Sensory Data + No Data for Training
V2.x Big Data For Sensory Data Read + Write →	Analytics + Data for Training	1. Upfront Requirement Analysis for Big Data Queries 2. File distribution structure for Big Data Fragmentation 3. Ontology / Model for faster file location indexing
V3 Big Data For Encrypted Sensory Data Read + Write →	Soft Real-Time stream based retrieval for analytics	1. Lower performance on encrypted data 2. Delay in retrieval for read request



10

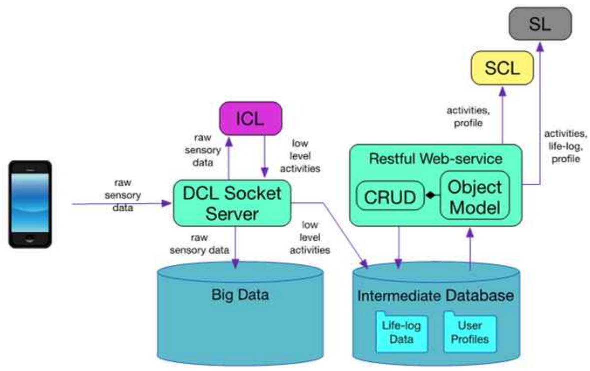
Life Representation and Mapping

	Benefit	Limitation
V1. Object Model	→ Performance	→ No Inference (No High Level Context in ICL)



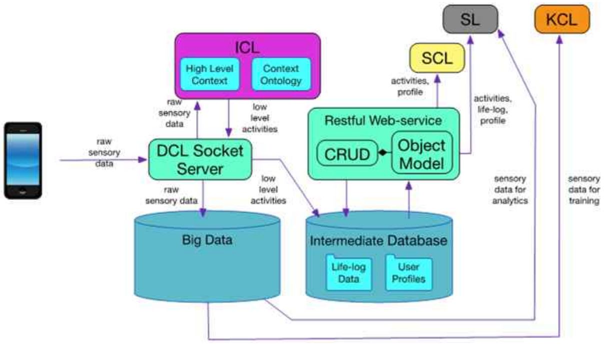
Life Representation and Mapping

	Benefit	Limitation
V1. Object Model	→ Performance	→ No Inference (No High Level Context in ICL)
V1.5 Object Model	→ Performance	→ No Inference (No High Level Context in ICL)



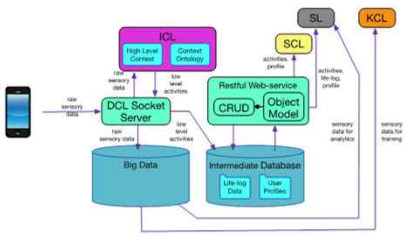
Life Representation and Mapping

	Benefit	Limitation
V1. Object Model	→ Performance	→ No Inference (No High Level Context in ICL)
V1.5 Object Model	→ Performance	→ No Inference (No High Level Context in ICL)
V2.x Object Oriented Life-log + Ontology for High Level Context at ICL	→ Performance + Inference	1. Only ICL requires to Inference , ICL can host an ontology natively 2. Triples will be stored in Intermediate DB; however KCL and SCL will read data via Interface (XML, JSON, etc.)



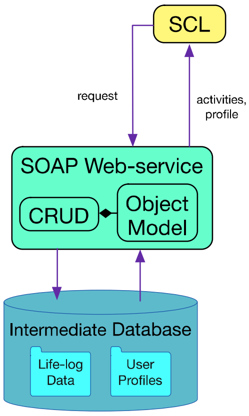
Life Representation and Mapping

	Benefit	Limitation
V1. Object Model	→ Performance	→ No Inference (No High Level Context in ICL)
V1.5 Object Model	→ Performance	→ No Inference (No High Level Context in ICL)
V2.x Object Oriented Life-log + Ontology for High Level Context at ICL	→ Performance + Inference	1. Only ICL requires to Inference , ICL can host an ontology natively 2. Triples will be stored in Intermediate DB; however KCL and SCL will read data via Interface (XML, JSON, etc.)
V3 Object Oriented Life-log + Physical & Nutrition Assessment	→ Performance + Inference + Assessment	1. Critical diseases like hypertension, diabetes assessment are not handled



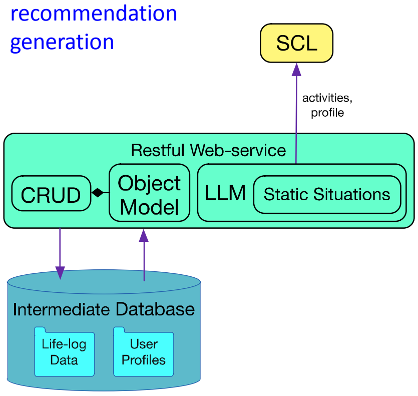
Lifelog Monitoring

		Benefit		Limitation
V1. NA	→	NA	→	No trigger/situation based recommendations



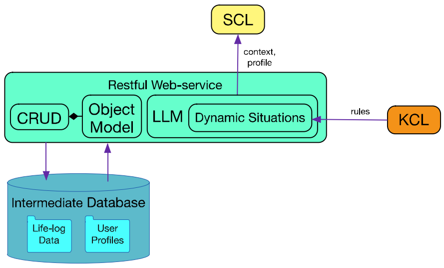
Lifelog Monitoring

		Benefit		Limitation
V1. NA	→	NA	→	No trigger/situation based recommendations
V1.5 Life-log Monitoring (LLM)	→	Trigger-based recommendation generation	→	Design-time static situations



Lifelog Monitoring

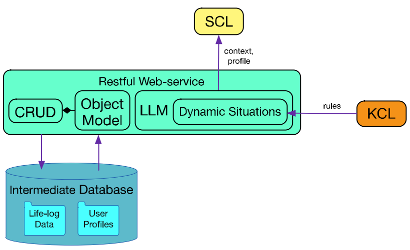
	Benefit	Limitation
V1. NA	→ NA	→ No trigger/situation based recommendations
V1.5 Life-log Monitoring (LLM)	→ Trigger-based recommendation generation	→ Design-time static situations
V2.x Life-log Monitoring (LLM)	→ Dynamic situations with runtime-invocation	1. A configuration utility will be created for situation generation by KCL 2. LLM will load the new situation at runtime for invocation



17

Lifelog Monitoring

	Benefit	Limitation
V1. NA	→ NA	→ No trigger/situation based recommendations
V1.5 Life-log Monitoring (LLM)	→ Trigger-based recommendation generation	→ Design-time static situations
V2.x Life-log Monitoring (LLM)	→ Dynamic situations with runtime-invocation	1. A configuration utility will be created for situation generation by KCL 2. LLM will load the new situation at runtime for invocation
V3 Life-log Monitoring (LLM)	→ Dynamic situations with runtime-invocation on the basis of nutrition intake	1. Critical disease monitoring situation are not handled.



18

Mining Minds Version 3

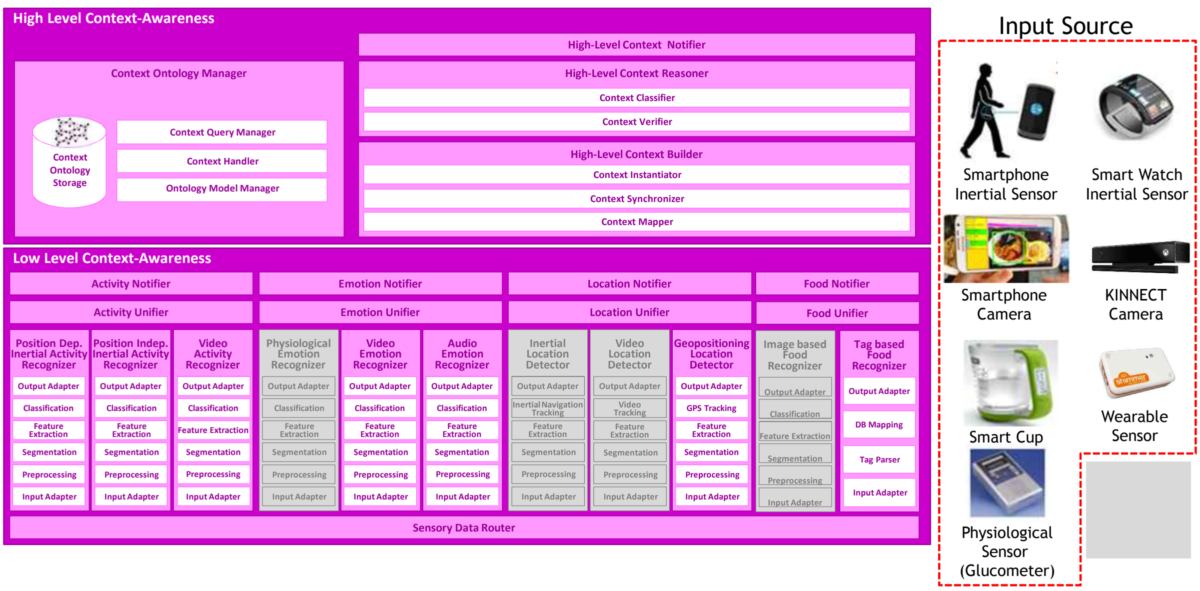
Information Curation Layer 3.0 Features Evolution

Principal Investigator:
Professor Sungyoung Lee

경희대학교 KYUNG HEE UNIVERSITY · KITECH 한국생산기술연구원 · POSTECH 포항공과대학교 · UTAS · 녹십자헬스케어 · TAPACROSS · U2 system User & Usability system

Challenges and solutions (MMV3.0-ICL)

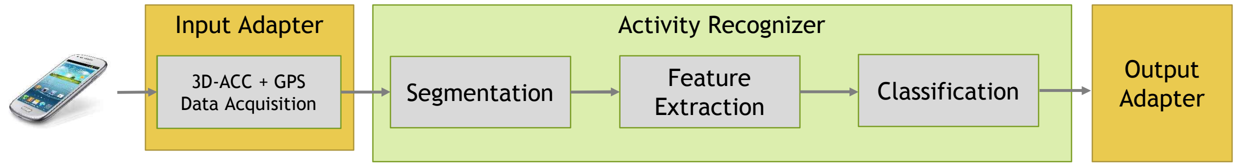
Information Curation Layer of MMV3.0



Challenges and solutions (MMV3.0-ICL)

1. Human activity recognition (AR)

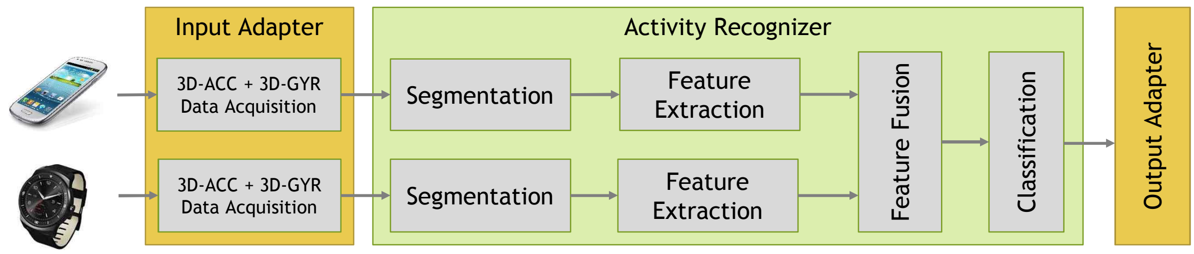
	Benefit	Limitation
V1. Smartphone-based AR	→ High accuracy	→ Only 5 activities



Challenges and solutions (MMV3.0-ICL)

1. Human activity recognition (AR)

	Benefit	Limitation
V1. Smartphone-based AR	→ High accuracy	→ Only 5 activities
V1.5 Smartphone and Wearable-based AR	→ More activities (up to 15)	→ Poor recognition performance

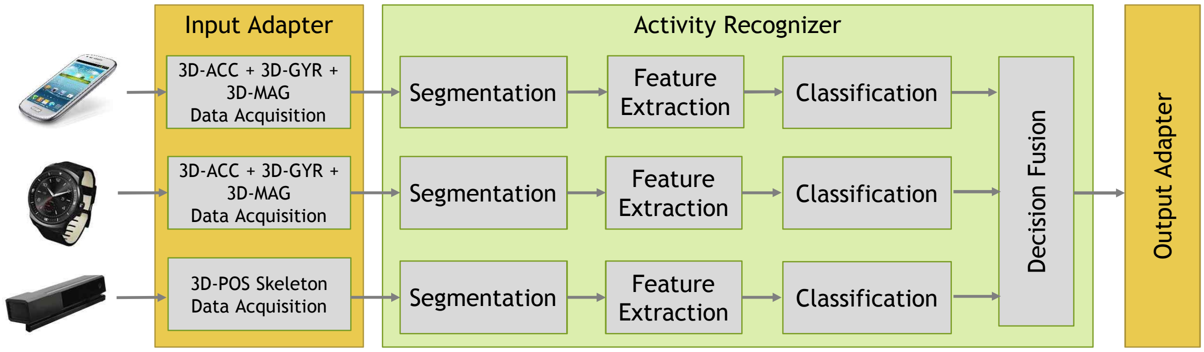


Challenges and solutions (MMV3.0-ICL)

1. Human activity recognition (AR)

	Benefit	Limitation
V1. Smartphone-based AR	→ High accuracy	→ Only 5 activities
V1.5 Smartphone and Wearable-based AR	→ More activities (up to 15)	→ Poor recognition performance
V2 Smartphone, Wearable and Video-based AR (multisensor fusion)	→ High accuracy with the support of different sensor	

Confidence level: Medium

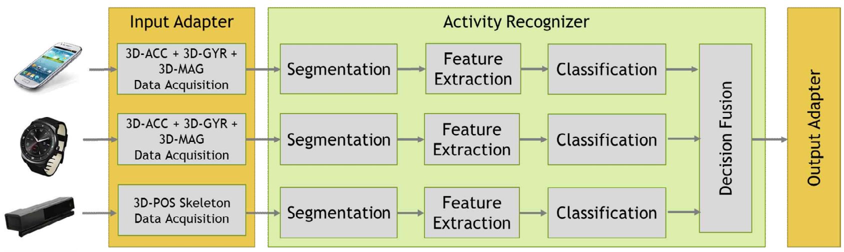


Challenges and solutions (MMV3.0-ICL)

1. Human activity recognition (AR)

	Benefit	Limitation
V1. Smartphone-based AR	→ High accuracy	→ Only 5 activities
V1.5 Smartphone and Wearable-based AR	→ More activities (up to 15)	→ Poor recognition performance
V2 Smartphone, Wearable and Video-based AR (multisensor fusion)	→ High accuracy with the support of different sensor	→ Poor recognition performance for some activities (e.g., "eating")
V2.5 Smartphone, Wearable and Video-based AR (multisensor fusion)	→ Refining AR Model for high accuracy & detailed eating motions	

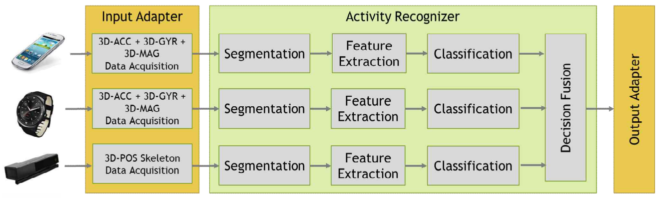
Confidence level: Medium



Challenges and solutions (MMV3.0-ICL)

1. Human activity recognition (AR)

	Benefit	Limitation
V1. Smartphone-based AR	→ High accuracy	→ Only 5 activities
V1.5 Smartphone and Wearable-based AR	→ More activities (up to 15)	→ Poor recognition performance
V2 Smartphone, Wearable and Video-based AR (multisensor fusion)	→ High accuracy with the support of different sensor	→ Poor recognition performance for some activities (e.g., "eating")
V2.5 Smartphone, Wearable and Video-based AR (multisensor fusion)	→ Refining AR Model for high accuracy & detailed eating motions	→ Poor recognition performance for office work and commuting activities (e.g., "reading book", "Bus")
V3.0 Smartphone, Wearable and Video-based AR (multisensor fusion)	→ High accuracy with multiple Activity Recognition Framework For commuting and office work accuracy	



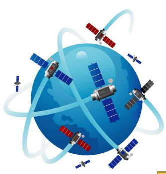
Confidence level: Medium

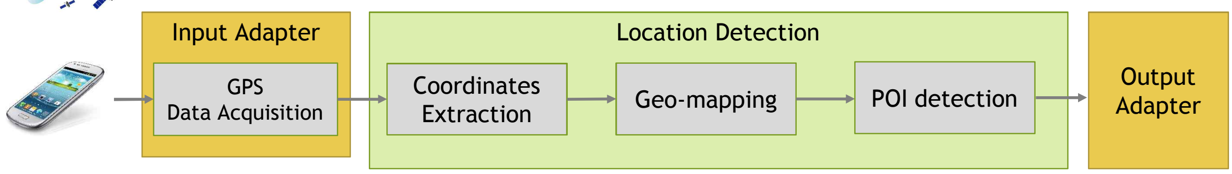
Challenges and solutions (MMV3.0-ICL)

2. Location detection (LD)

	Benefit	Limitation
V1. (Not implemented)	→ ( )	→ ( )
V1.5 (Not implemented)	→ ( )	→ ( )
V2 Outdoor location detection	→ Identification of the user geoposition for behavior tracking	

Confidence level: High



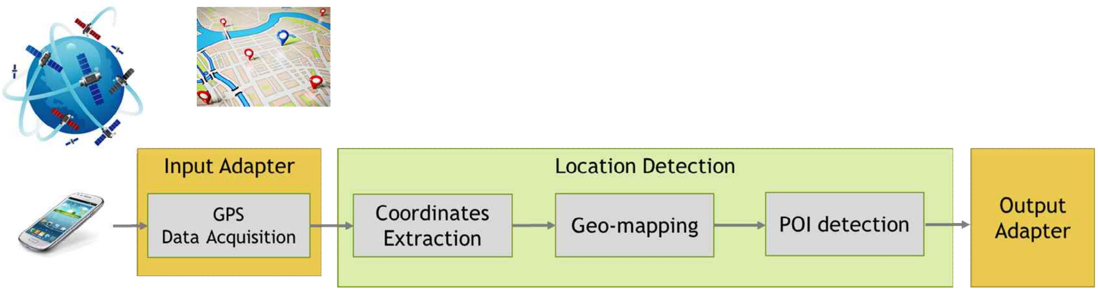


Challenges and solutions (MMV3.0-ICL)

2. Location detection (LD)

	Benefit	Limitation
V1. (Not implemented)	→ ( )	→ ( )
V1.5 (Not implemented)	→ ( )	→ ( )
V2 Outdoor location detection	→ Identification of the user geoposition for behavior tracking	→ Location can only be set at the stage of user registration
V2.5 Outdoor location detection	→ New location can be added anytime	

Confidence level: High

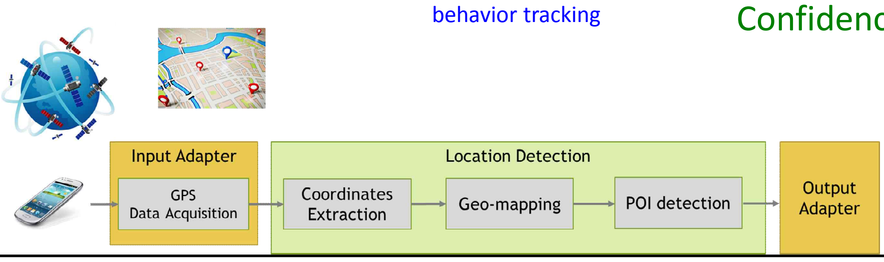


Challenges and solutions (MMV3.0-ICL)

2. Location detection (LD)

	Benefit	Limitation
V1. (Not implemented)	→ ( )	→ ( )
V1.5 (Not implemented)	→ ( )	→ ( )
V2 Outdoor location detection	→ Identification of the user geoposition for behavior tracking	→ Location can only be set at the stage of user registration
V2.5 Outdoor location detection	→ New location can be added anytime	→ User should to register location oneself
V3.0 Outdoor location detection	→ Semi auto location registration with behavior tracking	

Confidence level: High

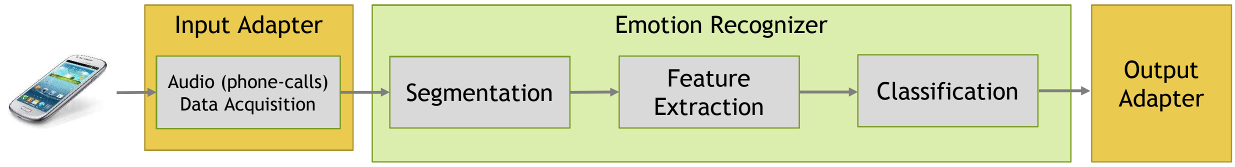


Challenges and solutions (MMV3.0-ICL)

3. Emotion Recognition (ER)

	Benefit	Limitation
V1. (Not implemented)	→ ( )	→ ( )
V1.5 (Not implemented)	→ ( )	→ ( )
V2 Audio-based ER	→ Identification of the user emotional states for a holistic description of the people behavior	→ Poor recognition performance, only 3 emotions

Confidence level: Low-Medium

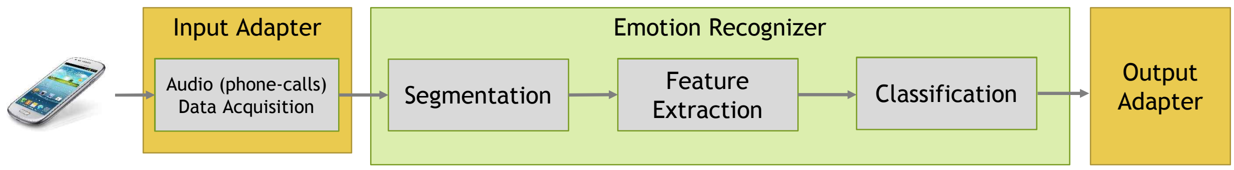


Challenges and solutions (MMV3.0-ICL)

3. Emotion Recognition (ER)

	Benefit	Limitation
V1. (Not implemented)	→ ( )	→ ( )
V1.5 (Not implemented)	→ ( )	→ ( )
V2 Audio-based ER	→ Identification of the user emotional states for a holistic description of the people behavior	→ Poor recognition performance, only 3 emotions
V2.5 Audio-based ER	→ Refining ER Model for high accuracy	→ Poor recognition performance for Neutral emotion

Confidence level: Low-Medium

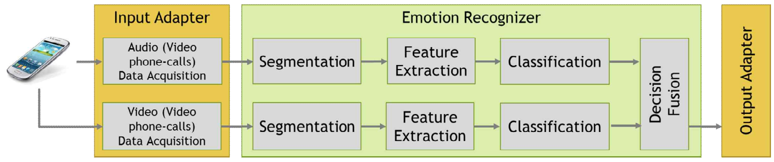


Challenges and solutions (MMV3.0-ICL)

3. Emotion Recognition (ER)

	Benefit	Limitation
V1. (Not implemented)	→ ( )	→ ( )
V1.5 (Not implemented)	→ ( )	→ ( )
V2 Audio-based ER	→ Identification of the user emotional states for a holistic description of the people behavior	→ Poor recognition performance, only 3 emotions
V2.5 Audio-based ER (Phone call)	→ Refining ER Model for high accuracy	→ Poor recognition performance For Neutral emotion
V3 Video and Audio based ER (Video Phone call)	→ More emotions (up to 4)	

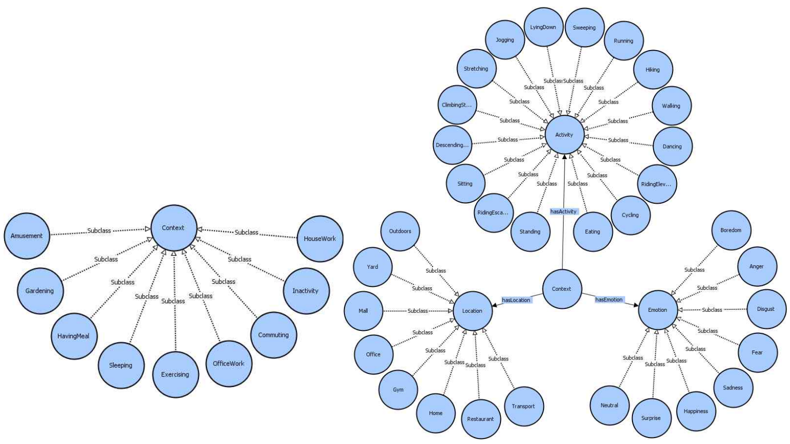
Confidence level: Low-Medium



Challenges and solutions (MMV3.0-ICL)

4. High-Level Context Awareness (HLCA)

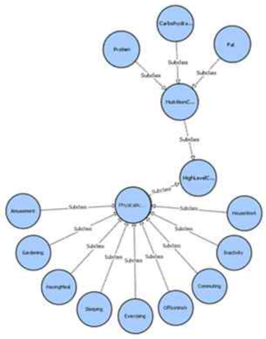
	Benefit	Limitation
V1 & 1.5 (Not considered)	→ ( )	→ ( )
V2 High-Level Context Inference	→ Enrichment of the user behavior description. **Physical Activity based services.**	→ Nutrition, important for Personal Health & Wellness, also directly related with behavior being not considered

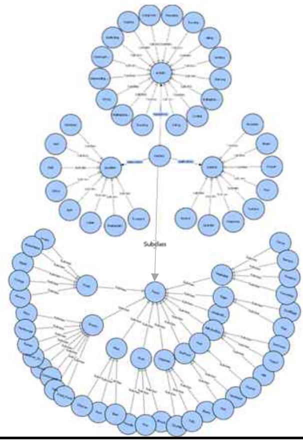


Challenges and solutions (MMV3.0-ICL)

4. High-Level Context Awareness (HLCA)

	Benefit	Limitation
V1 & 1.5 (Not considered)	→ ( )	→ ( )
V2 High-Level Context Inference	→ Enrichment of the user behavior description. **Physical Activity based services**.	→ Nutrition, important for Personal Health & Wellness, also directly related with behavior being not considered
V2.5 High-Level Context Inference	→ Enrichment of the user behavior description. **Nutrient based services**.	→ Lack of clinical information and water intake for context awareness in wellness domain

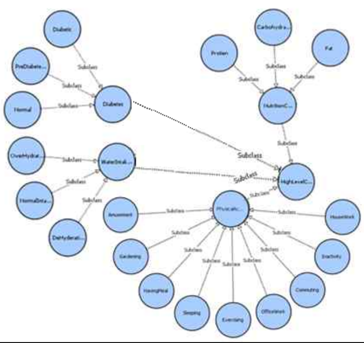


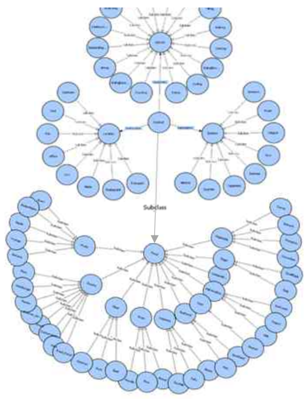


Challenges and solutions (MMV3.0-ICL)

4. High-Level Context Awareness (HLCA)

	Benefit	Limitation
V1 & 1.5 (Not considered)	→ ( )	→ ( )
V2 High-Level Context Inference	→ Enrichment of the user behavior description. **Physical Activity based services**.	→ Nutrition, important for Personal Health & Wellness, also directly related with behavior being not considered
V2.5 High-Level Context Inference	→ Enrichment of the user behavior description. **Nutrient based services**.	→ Lack of clinical information and water intake for context awareness in wellness domain
V3.0 High-Level Context Inference	→ Enrichment of the user behavior description. • **Diabetes Tracking & Water Intake Monitoring Services**. • **Inclusion of SWRL Rules**	Confidence level: • High • Medium





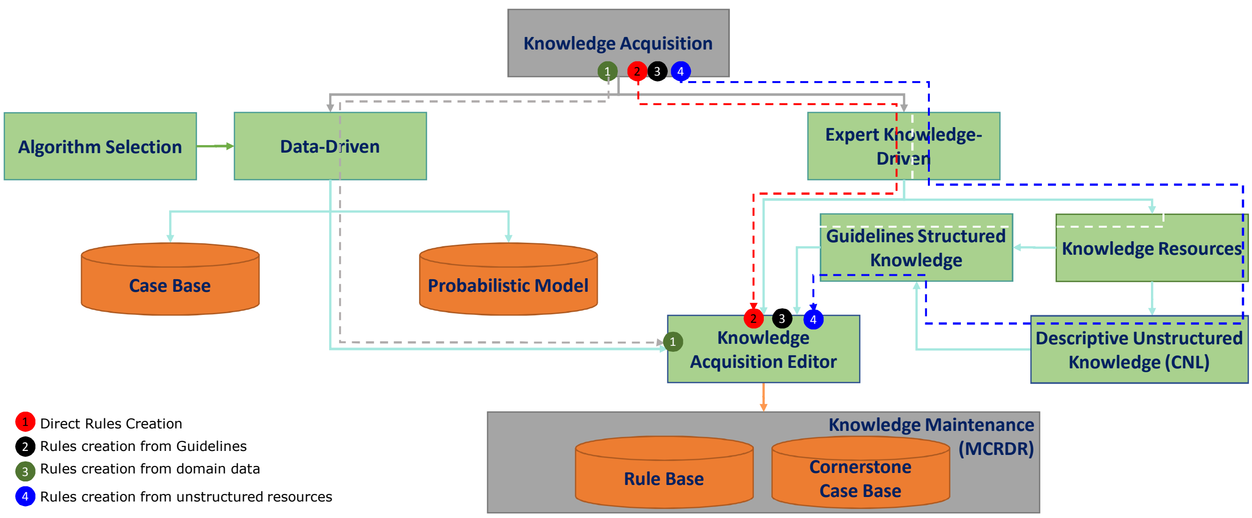
Mining Minds Version 3

Knowledge Curation Layer 3.0 Features Evolution

Principal Investigator:
Professor Sungyoung Lee

경희대학교 KYUNG HEE UNIVERSITY · KITECH 한국생산기술연구원 · POSTECH 포항공과대학교 · UTAS · 녹십자헬스케어 · TAPACROSS · U2 system User & Usability system

Scope of Knowledge Curation Layer: Rule Creation Paths

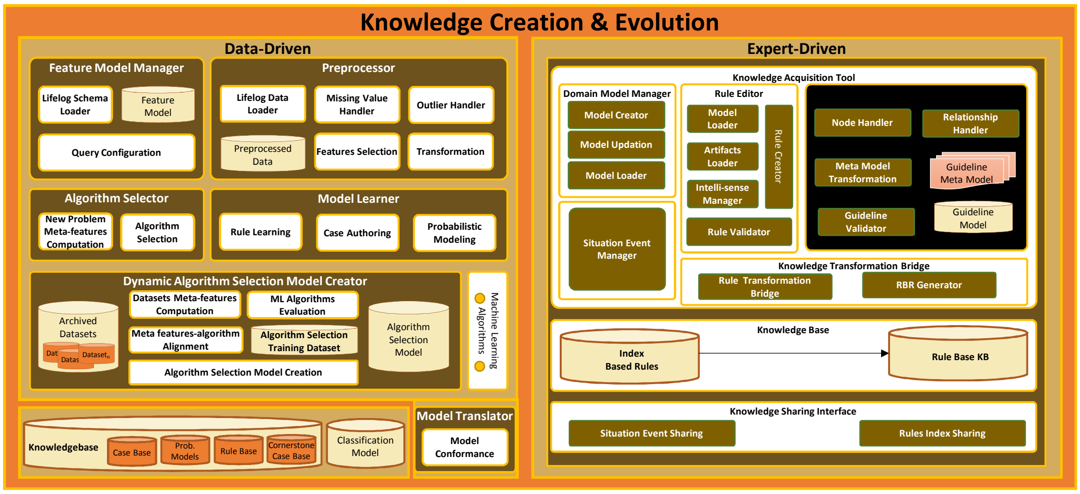


1 Direct Rules Creation
2 Rules creation from Guidelines
3 Rules creation from domain data
4 Rules creation from unstructured resources

경희대학교 KYUNG HEE UNIVERSITY · UTAS

36

Knowledge Curation Layer Architecture

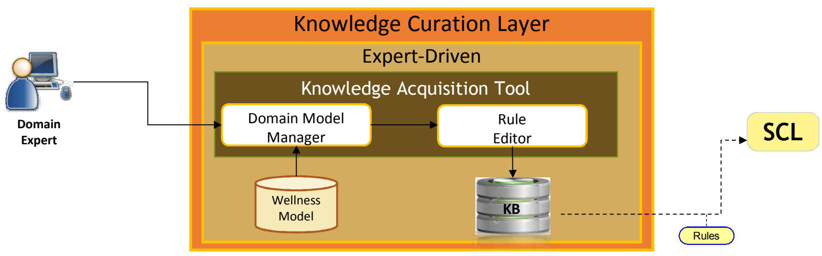


37

Challenges and solutions (MMV1.5-KCL)

1. Expert Driven Knowledge Acquisition

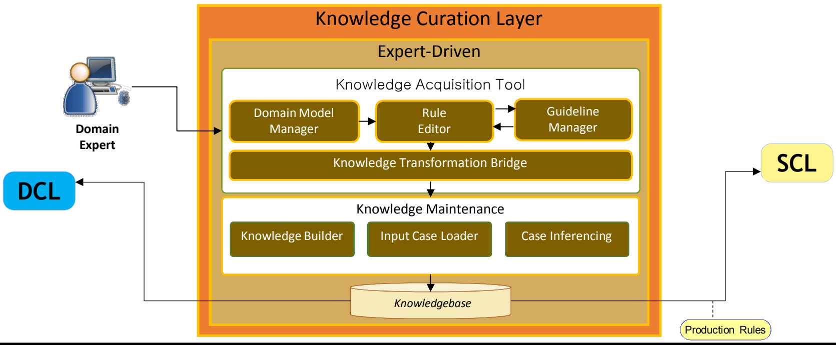
MM Version	Benefits	Limitations
V1. (Not considered)	→ ( )	→ ( )
V1.5: Expert base knowledge (Path-I)	→ Rule base KA	→ • No Validation • No SCL direct Integration



Challenges and solutions (MMV1.5-KCL)

1. Expert Driven Knowledge Acquisition

MM Version	Benefits	Limitations	Challenges
V1. (Not considered)	→ ( )	→ ( )	→ ( )
V1.5: Expert base knowledge (Path-I)	→ Rule base KA	→ • No Validation • No SCL direct Integration	→ ( )
V2.0: Rule Editor(Path-I)	→ • Facilitate expert for knowledge creation	→ ( )	→ • Creation without validation

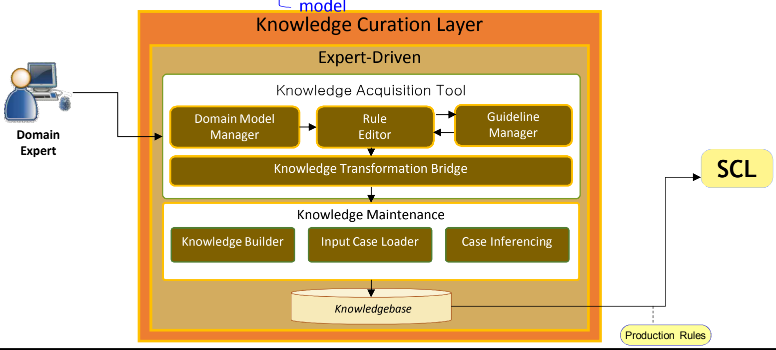


Confidence level: Low-Medium

Challenges and solutions (MMV1.5-KCL)

1. Expert Driven Knowledge Acquisition

MM Version	Benefits	Limitations	Challenges
V1. (Not considered)	→ ( )	→ ( )	→ ( )
V1.5: Expert base knowledge (Path-I)	→ Rule base KA	→ • No Validation • No SCL direct Integration	→ ( )
V2.0: Rule Editor(Path-I)	→ • Facilitate expert for knowledge creation	→ ( )	→ • Creation without validation
V2.5: Rule Editor with (Path-I) Domain Model Manager	→ • Facilitate expert for knowledge creation for nutrition domain • Facilitate expert to create domain model	→ ( )	→ • Domain model Creation without validation

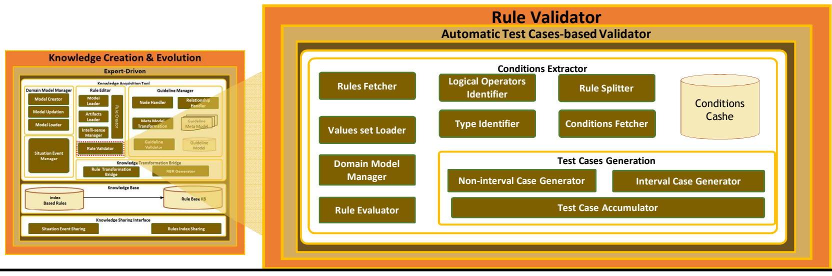


Confidence level: Low-Medium

Challenges and solutions (MMV1.5-KCL)

1. Expert Driven Knowledge Acquisition

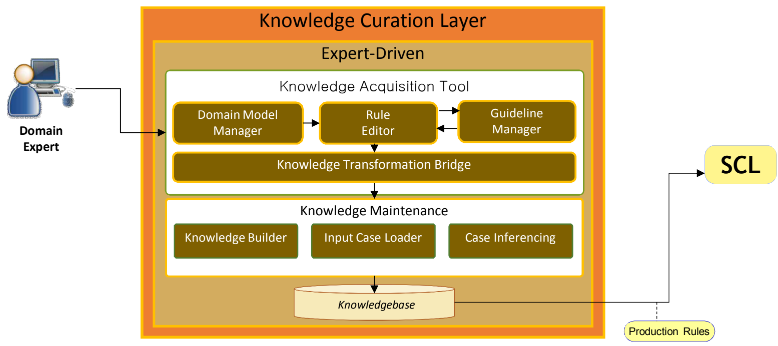
MM Version	Benefits	Limitations	Challenges
V1. (Not considered)	( )	( )	( )
V1.5: Expert base knowledge (Path-I)	Rule base KA	• No Validation • No SCL direct Integration	( )
V2.0: Rule Editor(Path-I)	• Facilitate expert for knowledge creation	( )	• Creation without validation
V2.5: Rule Editor with (Path-I) Domain Model Manager	• Facilitate expert for knowledge creation for nutrition domain • Facilitate expert to create domain model	( )	• Domain model Creation without validation
V3.0: Rule Editor with support of Rules and Situation Validation process (Path-I)	• To ensure consistency of knowledge • Enhance confidence and trust of domain expert on created knowledge	( )	• Achieving validation with optimal generated test case



Challenges and solutions (MMV1.5-KCL)

2. Expert Driven Knowledge Acquisition (Domain Model Manager)

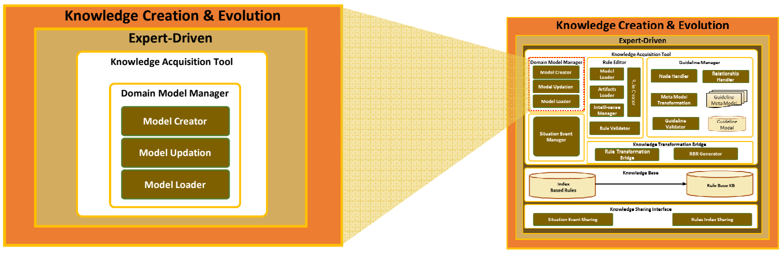
MM Version	Benefits	Limitations	Challenges
V1. (Not considered)	( )	( )	( )
V1.5: (Not considered)	( )	( )	( )
V2.0: (Not considered)	( )	( )	( )
V2.5: Rule Editor with (Path-I) Domain Model Manager	• Facilitate expert for knowledge creation for nutrition domain • Facilitate expert to create domain model	( )	• Domain model Creation without validation



Challenges and solutions (MMV1.5-KCL)

2. Expert Driven Knowledge Acquisition (Domain Model Manager)

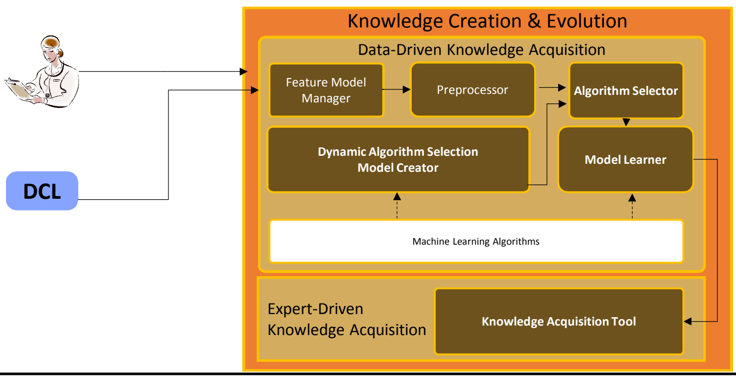
MM Version	Benefits	Limitations	Challenges
V1. (Not considered)	→ ( )	→ ( )	→ ( )
V1.5: (Not considered)	→ ( )	→ ( )	→ ( )
V2.0: (Not considered)	→ ( )	→ ( )	→ ( )
V2.5: Rule Editor with (Path-I) Domain Model Manager	→ • Facilitate expert for knowledge creation for nutrition domain • Facilitate expert to create domain model	→ ( )	→ • Domain model Creation without validation
V3.0: Domain Model Manager Toolkit	→ • Creation, Updation, and maintenance of Wellness model • Supporting knowledge acquisition with validated Wellness model	→ (Wellness model simple GUI support)	→ • Wellness model representation and validation



Challenges and solutions (MMV1.5-KCL)

2. Data Driven Knowledge Acquisition

MM Version	Benefits	Limitations	Challenges
V1: (Not considered)	→ ( )	→ ( )	→ ( )
V1.5: (Not considered)	→ ( )	→ ( )	→ ( )
V2.0: Creating classification model using life log data (Path-III)	→ • DTs from lifelog data	→ ( )	→ • Overall accuracy depends on available data • Big data handling

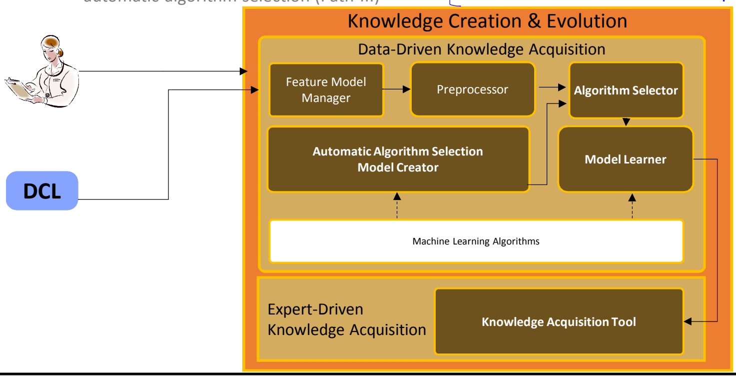


Confidence level:
Low-Medium

Challenges and solutions (MMV2-KCL)

2. Data Driven Knowledge Acquisition

MM Version	Benefits	Limitations	Challenges
V1: (Not considered)	→ ( )	→ ( )	→ ( )
V1.5: (Not considered)	→ ( )	→ ( )	→ ( )
V2.0: Creating classification model using life log data (Path-III)	→ • DTs from lifelog data	→ ( )	→ • Overall accuracy depends on available data • Big data handling
V2.5: Creating classification model using automatic algorithm selection (Path-III)	→ • Accurate Algorithm Selection • DTs conformance with domain expert	→ ( )	→ • Algorithm selection accuracy • Rule extraction from model



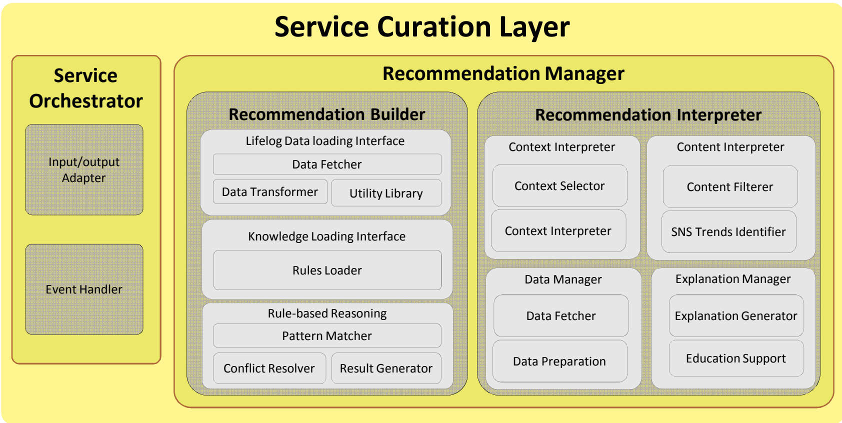
Confidence level:
Low-Medium

Mining Minds Version 3

Service Curation Layer 3.0 Features Evolution

Principal Investigator:
Professor Sungyoung Lee

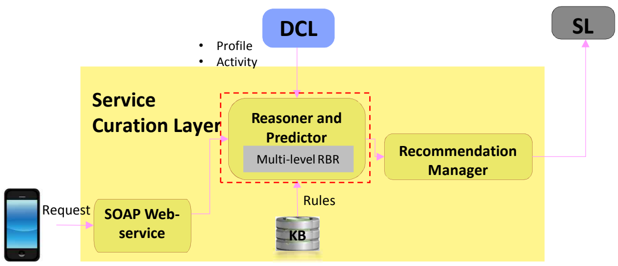
MMV2-SCL Architecture



Challenges and solutions (MMV2-SCL)

1. Reasoning

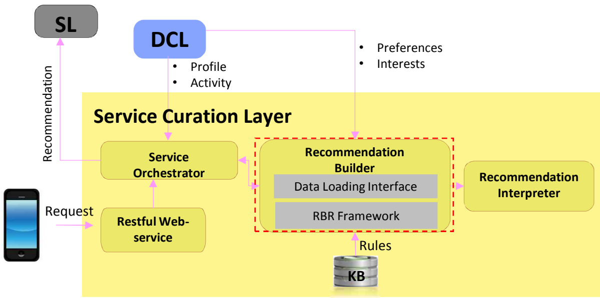
		Benefit		Limitation
V1. Rule-based Reasoning (Reasoner and Predictor)	→	1. Granular personalization	→	1. Less flexible 2. Service and Data Management 3. No Automatic Rules Retrieval 4. Handcrafted Rules (not scalable)



Challenges and solutions (MMV2-SCL)

1. Reasoning

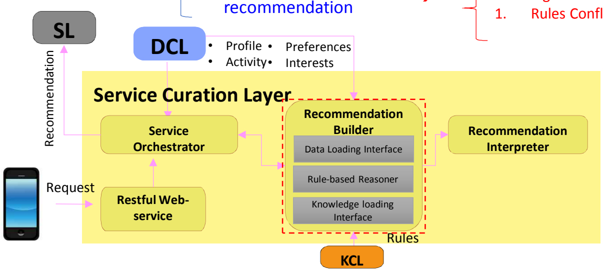
		Benefit		Limitation
V1. Rule-based Reasoning (Reasoner and Predictor)	→	1. Granular personalization	→	1. Less flexible 2. Service and Data Management 3. Handcrafted Rules (not scalable) 4. No Automatic Rules Retrieval
V1.5. Rule-based Reasoning (Recommendation Builder)	→	1. Flexibility 2. Improved data & service communication	→	1. Partial Automatic Rules (authoring tool) 2. No Unified Interface with KCL 3. Low precision



Challenges and solutions (MMV2-SCL)

1. Reasoning

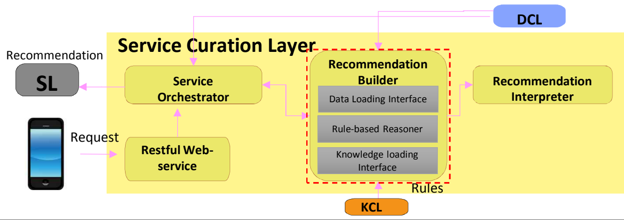
		Benefit		Limitation
V1. Rule-based Reasoning (Reasoner and Predictor)	→	1. Granular personalization	→	1. Less flexible 2. Service and Data Management 3. Handcrafted Rules (not scalable) 4. No Automatic Rules Retrieval
V1.5. Rule-based Reasoning (Recommendation Builder)	→	1. Flexibility 2. Improved data & service communication	→	1. Partial Automatic Rules (authoring tool) 2. No Unified Interface with KCL
V2. RBR (Recommendation Builder)	→	1. Preciseness 2. Scalability 3. Unified knowledge interface	→	Challenges: 1. Interfacing with KCL 2. Design & development of extensible reasoning framework
V2.5 RBR forward chaining & conflict resolution (Recommendation Builder)	→	1. Accuracy 2. Non-conflicting recommendation	→	Challenges: 1. Rules Conflicts resolution



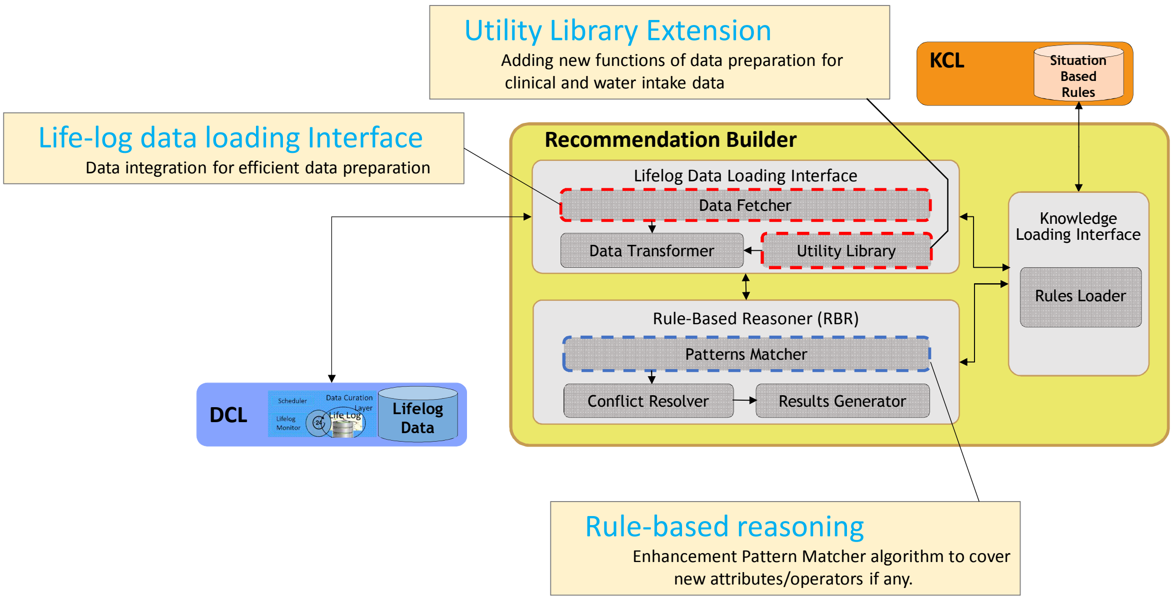
Challenges and solutions (MMV2-SCL)

1. Recommendation Builder: Reasoning

		Benefit		
V1. Rule-based Reasoning (Reasoner and Predictor)	→	1. Granular personalization		
V1.5. Rule-based Reasoning (Recommendation Builder)	→	1. Flexibility 2. Improved data & service communication		
V2. RBR (Recommendation Builder)	→	1. Preciseness 2. Scalability 3. Unified knowledge interface	→	Challenges: 1. Interfacing with KCL 2. Design & development of extensible reasoning framework
V2.5 RBR forward chaining & conflict resolution (Recommendation Builder)	→	1. Accuracy 2. Non-conflicting	→	Challenges: 1. Rules Conflicts resolution
V3.0 RBR forward chaining & conflict resolution enhancement with → Dynamic data integration → Utility Library → Operator parsing (conditional)	→	1. Efficient data preparing 2. Performance gain	→	Challenges: 1. Data elements finding for smooth integration



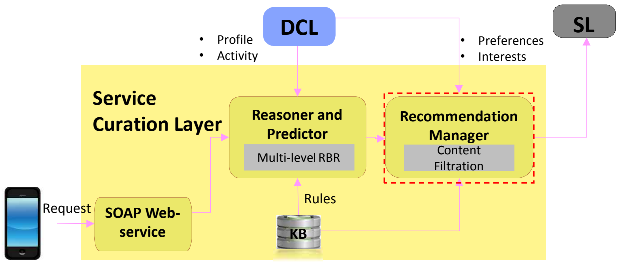
Recommendation Builder V3 Plan



Challenges and solutions (MMV2-SCL)

2. Interpretation and Explanation

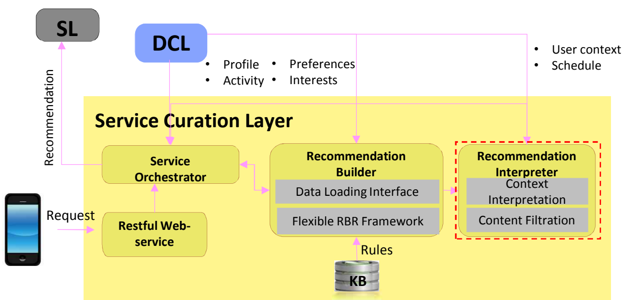
		Benefit		Limitation
V1. Content Filtration (Recommendation Manager)	→	• Preference-based Personalization	→	1. Lack of understandability 2. No context interpretation 3. No explanation



Challenges and solutions (MMV2-SCL)

2. Interpretation and Explanation

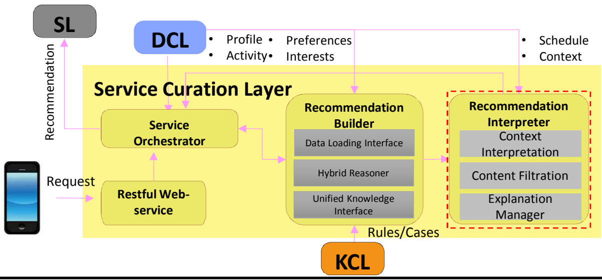
		Benefit		Limitation
V1. Content Filtration (Recommendation Manager)	→	1. Preference-based Personalization	→	1. Lack of understandability 2. No context interpretation 3. No explanation
V1.5. Context Interpretation (Recommendation Interpreter)	→	1. Understandable 2. Contextually (time-based) interpreted reco	→	1. Limited and static explanation



Challenges and solutions (MMV2-SCL)

2. Interpretation and Explanation

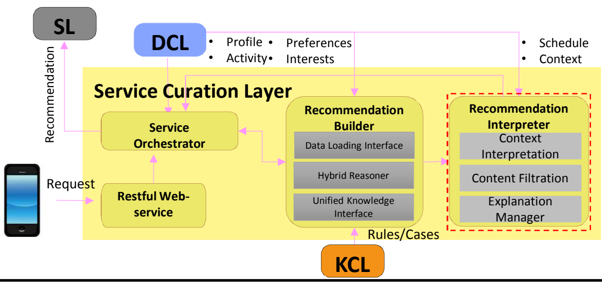
	Benefit	Limitation
V1.0 Content Filtration (Recommendation Manager) →	1. Preference-based Personalization →	1. Lack of understandability 2. No context interpretation 3. No explanation
V1.5. Context Interpretation (Recommendation Interpreter) →	1. Understandable 2. Contextually (time-based) interpreted rec →	1. Limited explanation 2. Static explanation
V2.0 Explanation →	1. Situation-aware Explanation 2. Education Support →	Challenges: 1. Cross-Context Interpretations 2. Dynamic Explanations



Challenges and solutions (MMV2-SCL)

2. Interpretation and Explanation

	Benefit	Limitation
V1.0 Content Filtration (Recommendation Manager) →	1. Preference-based Personalization →	1. Lack of understandability 2. No context interpretation 3. No explanation
V1.5. Context Interpretation (Recommendation Interpreter) →	1. Understandable 2. Contextually (time-based) interpreted rec →	1. Limited explanation 2. Static explanation
V2.0 Explanation V2.5 Social Trends →	1. Situation-aware Explanation 2. Education Support 3. Social Trends →	Challenges: 1. Cross-Context Interpretations 2. Dynamic Explanations



Challenges and solutions (MMV2-SCL)

2. Interpretation and Explanation

Benefit

V1.0 Content Filtration (Recommendation Manager) → 1. Preference-based Personalization

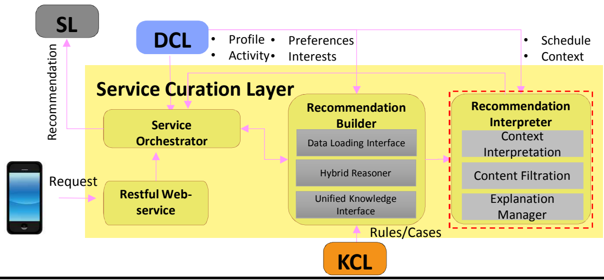
V1.5. Context Interpretation (Recommendation Interpreter) → 1. Understandable 2. Contextually (time-based) interpreted rec

V2.0 Explanation

V2.5 Social Trends → 1. Situation-aware Explanation 2. Social Trends 3. Enrich recommendation

V3.0 Contextual Matrix Updating & Extended explanation with clinical information

→ Challenges:
1. Cross-Context Interpretations
2. Dynamic Explanations
3. Correlation among new contexts with existing contexts



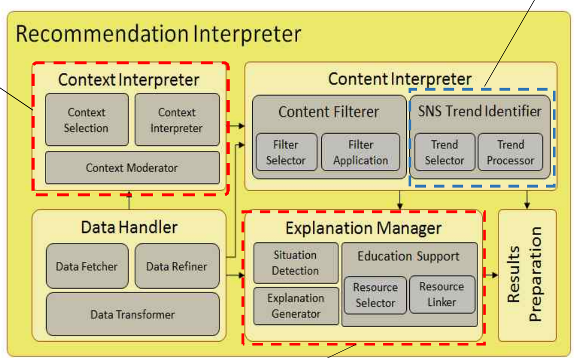
Recommendation Interpreter V3 Plan

Relearning of Contextual Matrix

Conducting survey to acquire preference knowledge for new low and high level contexts.

Social Trends Refinements

- Enhancements regarding filtration of social trends (if changed/added)



Explanation Enhancements

- Recommendation enhancements with additional description of clinical/water information.
- Research for finding more videos

Challenges and solutions (MMV2-SCL)

3. Services Scope and Types

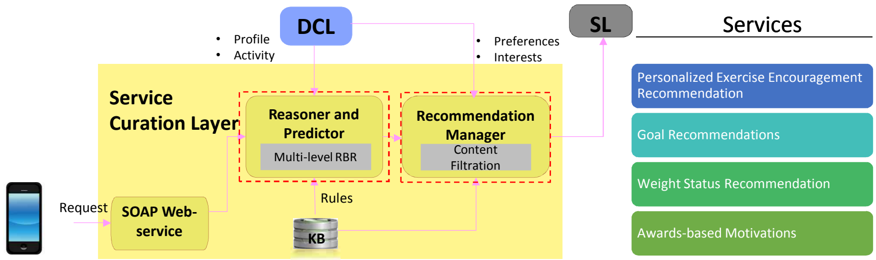
- V1. Physical Activity Recommendations (Calories Based) →

Benefit

1. User encouragement for phy. activity rec. using goal and award-based services →

Limitation

1. Less practical in terms of user understandability



Challenges and solutions (MMV2-SCL)

3. Services Scope and Types

- V1. Physical Activity Recommendations (Calories Based) →
- V1.5 Physical Activity Recommendations and Educational Facts →

Benefit

1. User encouragement for phy. activity rec. using goal and award-based services →

Limitation

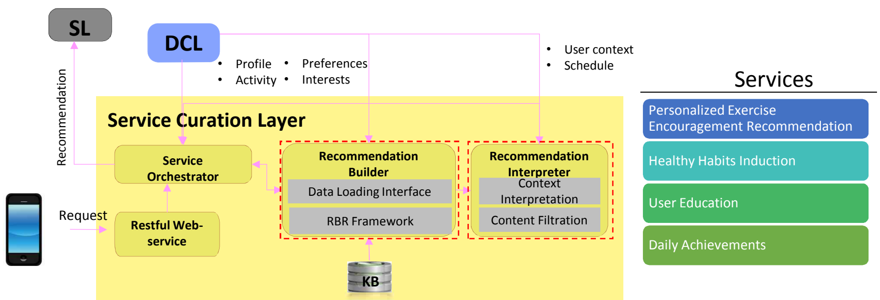
1. Less practical in terms of user understandability

Benefit

1. Physical actv. rec
2. Avoiding abnormal sedentary behaviors →

Limitation

1. Static situation handling
2. Limited context-awareness



Challenges and solutions (MMV2-SCL)

3. Services Scope and Types

- V1. Physical Activity Recommendations (Calories Based)
- V1.5 Physical Activity Recommendations and Educational Facts
- V2.0 Physical Activity Recommendations and Educational Facts
- V2.5 Nutrition Recommendations

Benefit

V1 →
1. User encouragement for phy. activity rec. using goal and award-based services

V1.5 →
1. Avoiding abnormal sedentary behaviors
2. User education

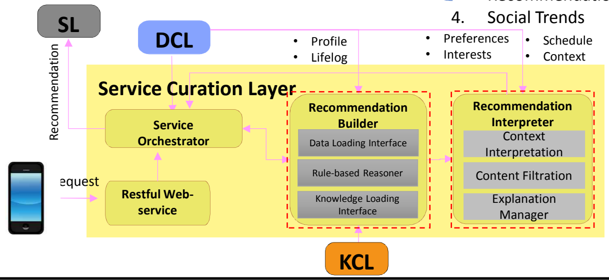
V2.0 / V2.5 →
1. Avoiding abnormal sedentary behaviors
2. User education
3. Food-integrated phy. act. Recommendations
4. Social Trends

Limitation

→ 1. Less practical in terms of user understandability

→ 1. Static situation handling
2. Limited context-awareness

→ Challenges:
1. Increased Service dependencies management
2. Integration of food and physical activity services



Services

- Personalized Exercise Encouragement Recommendation
- Healthy Habits Induction
- User Education
- Nutrition Services with Social Trends

Challenges and solutions (MMV2-SCL)

3. Services Scope and Types

- V1. Physical Activity Recommendations (Calories Based)
- V1.5 Physical Activity Recommendations and Educational Facts
- V2.0 Physical Activity Recommendations and Educational Facts
- V2.5 Nutrition Recommendations
- V3.0 Water intake recommendations

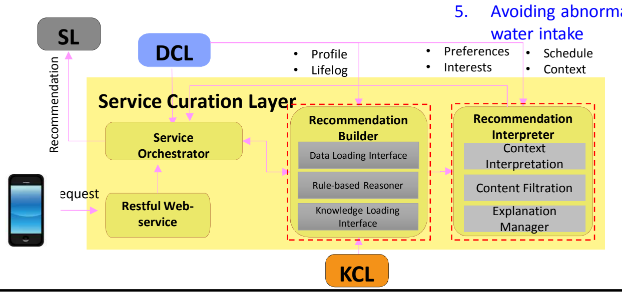
Benefit

V1 →
1. User encouragement for phy. activity rec. using goal and award-based services

V1.5 →
1. Avoiding abnormal sedentary behaviors
2. User education

V2.0 / V2.5 / V3.0 →
1. Avoiding abnormal sedentary behaviors
2. User education
3. Food Recommendations
4. Social Trends
5. Avoiding abnormal patterns in water intake

→ Challenges:
1. Increased Service dependencies management
2. Integration of food and physical activity services
3. Delivery mode and format of services



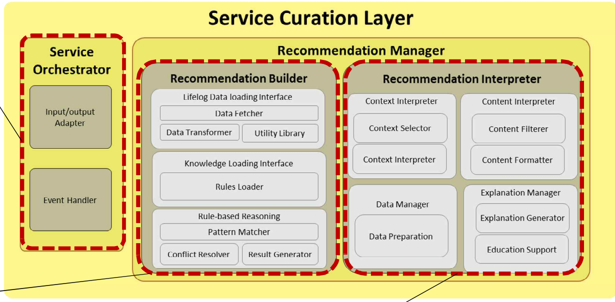
Services

- Personalized Exercise Encouragement Recommendation
- Healthy Habits Induction
- User Education
- Nutrition Services with Social Trends
- Water Intake Services

New Service Requirements Fulfillment Tasks

Service Orchestrator

- New web service method implementation for water intake
- New clients for acquiring clinical data/smart cup from DCL



Recommendation Builder

- Dynamic data integration function implementation
- Data preparation utility enhancements

Recommendation Interpreter

- Identifying special condition relevant to clinical data for recommendation filtration
- Survey conduction for updating contextual matrices.

Mining Minds Version 3

Supporting Layer 3.0 Features Evolution

Principal Investigator:
Professor Sungyoung Lee

경희대학교 KYUNG HEE UNIVERSITY KITECH 한국생산기술연구원 POSTECH 포항공과대학교 UTAS 녹십자헬스케어 TAPACROSS U2 system User & Usability system

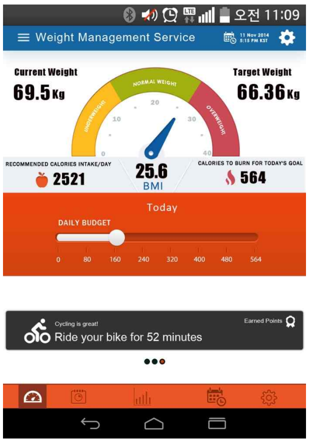
Challenges and solutions (MMV2.5-SL)

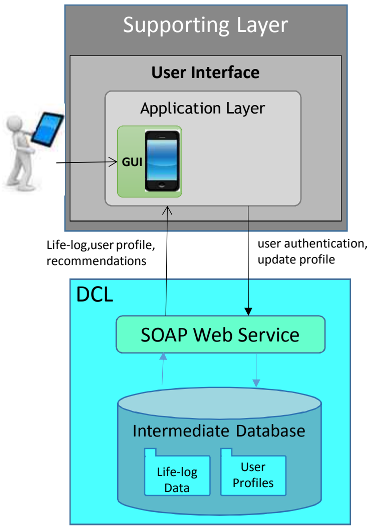
1. User Interface and User Experience

Benefit Limitation

V1. User Interfaces → Services User Interface → Static UI

- User interfaces for Calories based Weight Management Scenario
- Admin view for monitoring the activities of the users and system





Challenges and solutions (MMV2.5-SL)

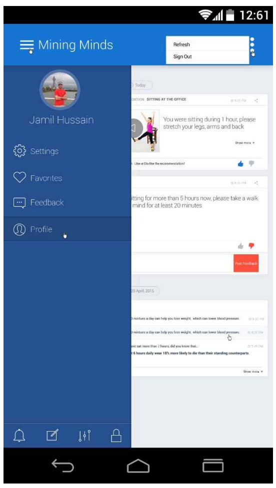
1. User Interface and User Experience

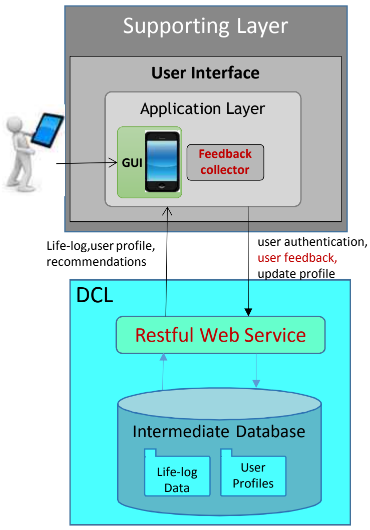
Benefit Limitation

V1. User Interfaces → Services User Interface → Static UI

V1.5 User Interfaces + Feedback → Services User Interface → Static UI

- User interfaces for educational and facts based Weight Management Scenario
- Expert view for monitoring user behavior and feedback provision



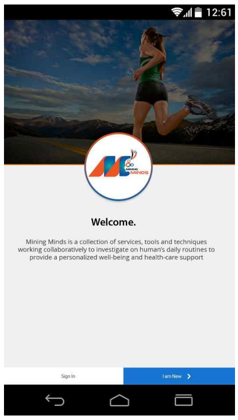


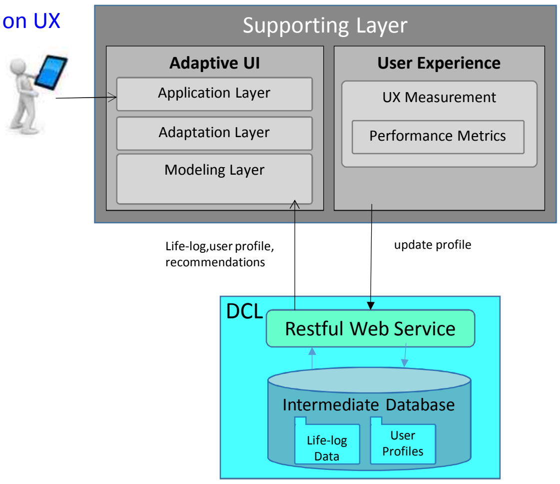
Challenges and solutions (MMV2.5-SL)

1. User Interface and User Experience

	Benefit	Limitation
V1. User Interfaces	→ Services User Interface	→ Static UI
V1.5 User Interfaces + **Feedback**	→ Services User Interface	→ Static UI
V2 Adaptive UI + User Experience	→ Adaptive UI based on UX	

- User interfaces for weight management scenario
- Adaptive User Interface based on user profile, device and



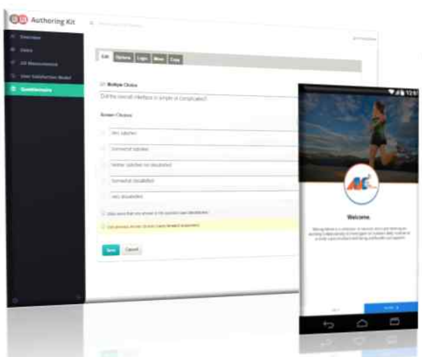


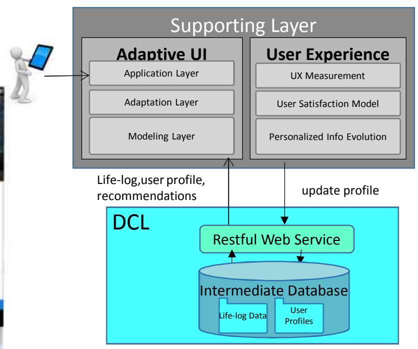
Challenges and solutions (MMV2.5-SL)

1. User Interface and User Experience

	Benefit	Limitation
V1. User Interfaces	→ Services User Interface	→ Static UI
V1.5 User Interfaces + **Feedback**	→ Services User Interface	→ Static UI
V2 Adaptive UI + User behavior analytics	→ Adaptive UI based on UX	→ User Experience Quantification
V2.5 Adaptive UI + User Experience	→ User Experience Quantifications	

- User experience quantification based on user behavior data along with self-reported data
- User feedback collection and its data analysis
- Evaluation of user experience

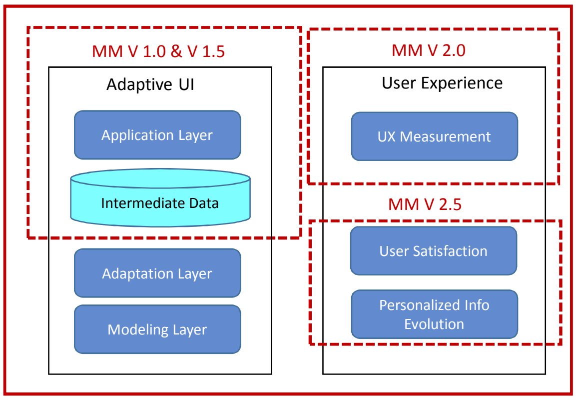


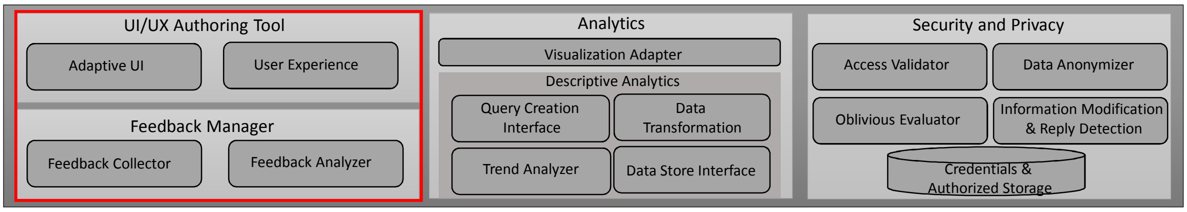


Challenges and solutions (MMV2.5-SL)

1. User Interface and User Experience

		Benefit		Limitation
V1. User Interfaces	→	Services User Interface	→	Static
V1.5 User Interfaces + **Feedback**	→	Services User Interface	→	Static
V2 Adaptive UI + User Experience	→	Adaptive UI based on UX	→	User Experience Quantification
V2.5 Adaptive UI + User Experience	→	User Experience Quantifications		

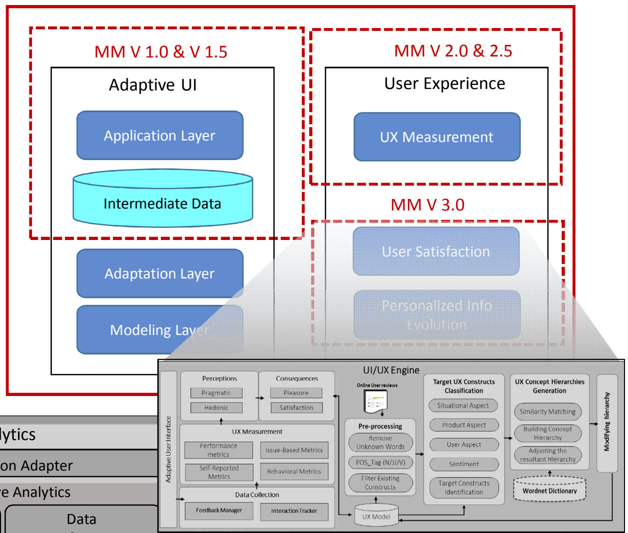


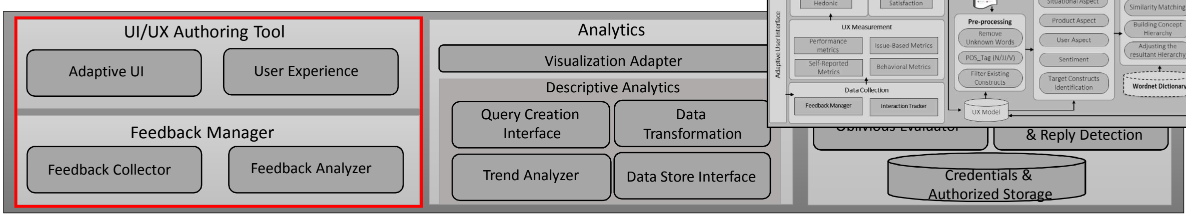


Challenges and solutions (MMV3.0 -SL)

1. User Interface and User Experience

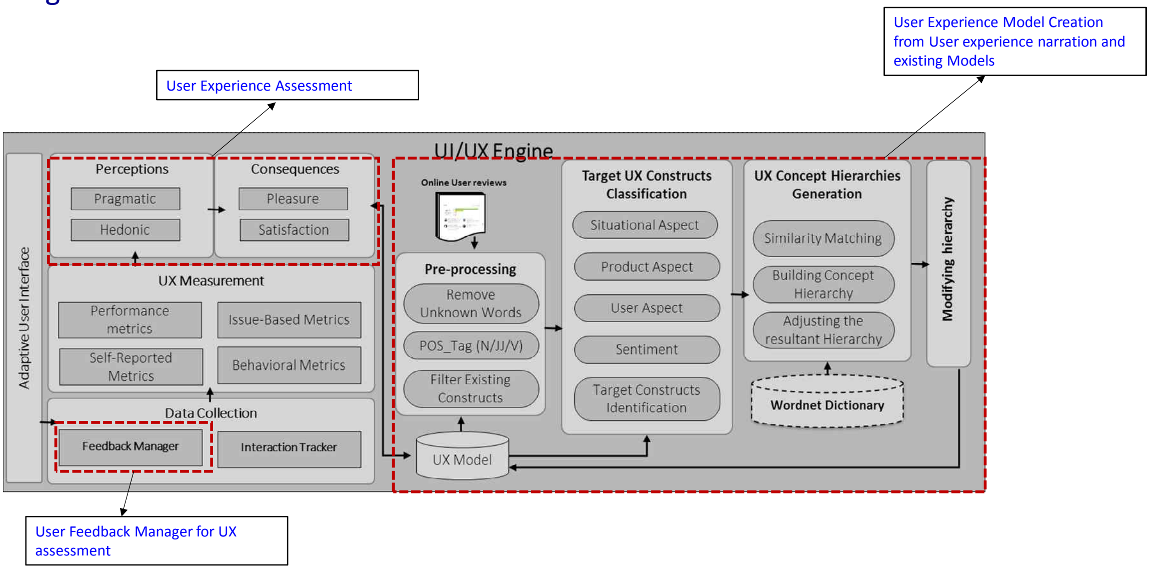
		Benefit		Limitation
V1. User Interfaces	→	Services User Interface	→	Static
V1.5 User Interfaces + **Feedback**	→	Services User Interface	→	Static
V2 Adaptive UI + User Experience	→	Adaptive UI based on UX	→	User Experience Quantification
V2.5 Adaptive UI + User Experience	→	User Experience Quantifications	→	No automation, No Authoring Tool
V3.0 (1) Development of UX engine	→	Automatic User Experience Assessment based on user behavioral data		
(2) UI/UX Authoring Tool	→	To develop the UI/UX Authoring Tool that facilitate the UX expert to write the adaption rules and questions based on user model and interaction data analysis for adaptive user interface		





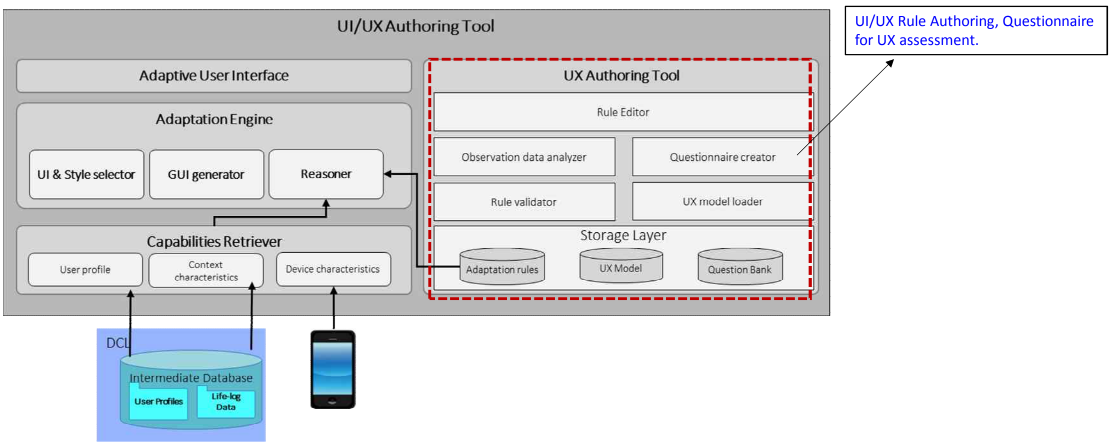
Challenges and solutions (MMV3.0 -SL)

1a. UX Engine



Challenges and solutions (MMV3.0 -SL)

1b. UI/UX Authoring Tool

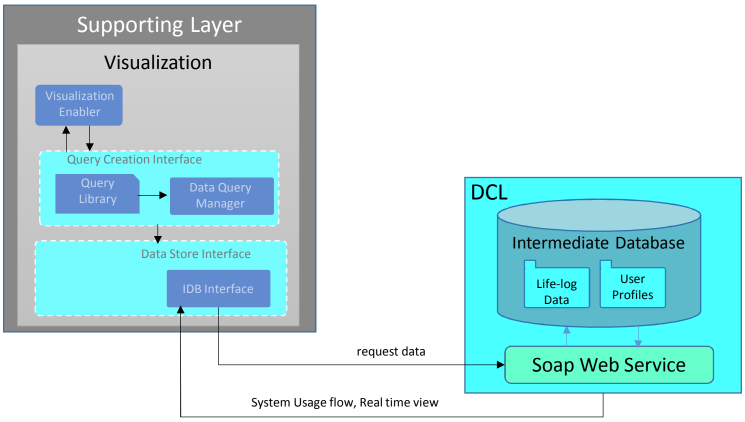


Challenges and solutions (MMV2.5-SL)

2. Analytics

	Benefit	Limitation
V1. Admin View	System usage visualization and data correlation between layers	Not intuitive and performance inefficient due to high volume in Intermediate DB

- Admin view for monitoring work flow and trigger points of the sensory data
- System Usage only from intermediate data base

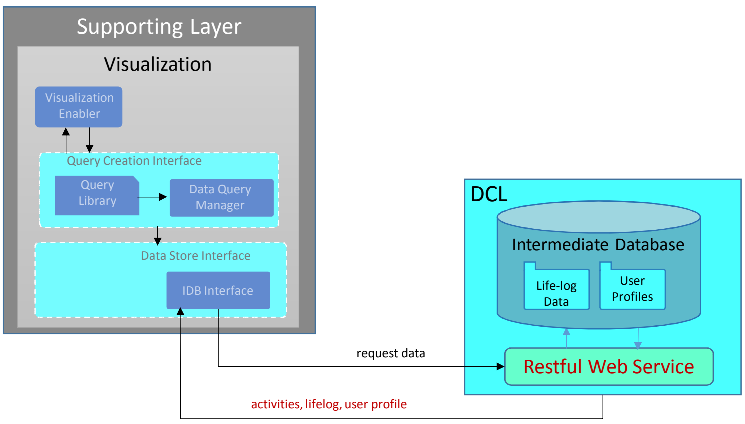


Challenges and solutions (MMV2.5-SL)

2. Analytics

	Benefit	Limitation
V1. Admin View	→ System usage visualization and data correlation between layers	→ Not intuitive and performance inefficient due to high volume in Intermediate DB
V1.5 Expert View	→ Users feedback, activities and system usage visualization	→ Limited features due to intermediate data schema and size

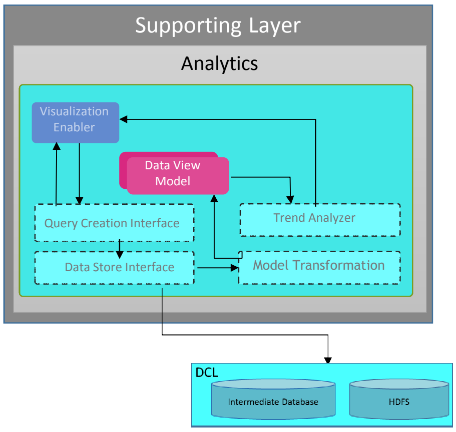
- Expert view for visualizing user activities and feedback for recommendations
- System Usage only from intermediate data base



Challenges and solutions (MMV2.5-SL)

2. Analytics

	Benefit	Limitation
V1. Admin View	→ System usage visualization and data correlation between layers	→ Not intuitive and performance inefficient due to high volume in Intermediate DB
V1.5 Expert View	→ Users feedback, activities and system usage visualization	→ Limited features due to intermediate data schema and size
V2 Big data analytics	→ Users feedback, activities and facts analytics. Insights into big data from sensory data from system usage perspective.	

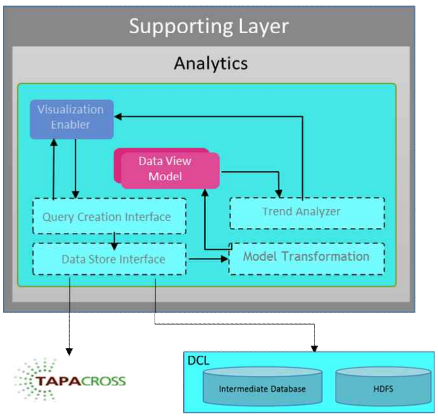


- Analytics based on statistics, clustering and association
- Insights from the big data repository and communication between both repositories
- Query library for the big data repository

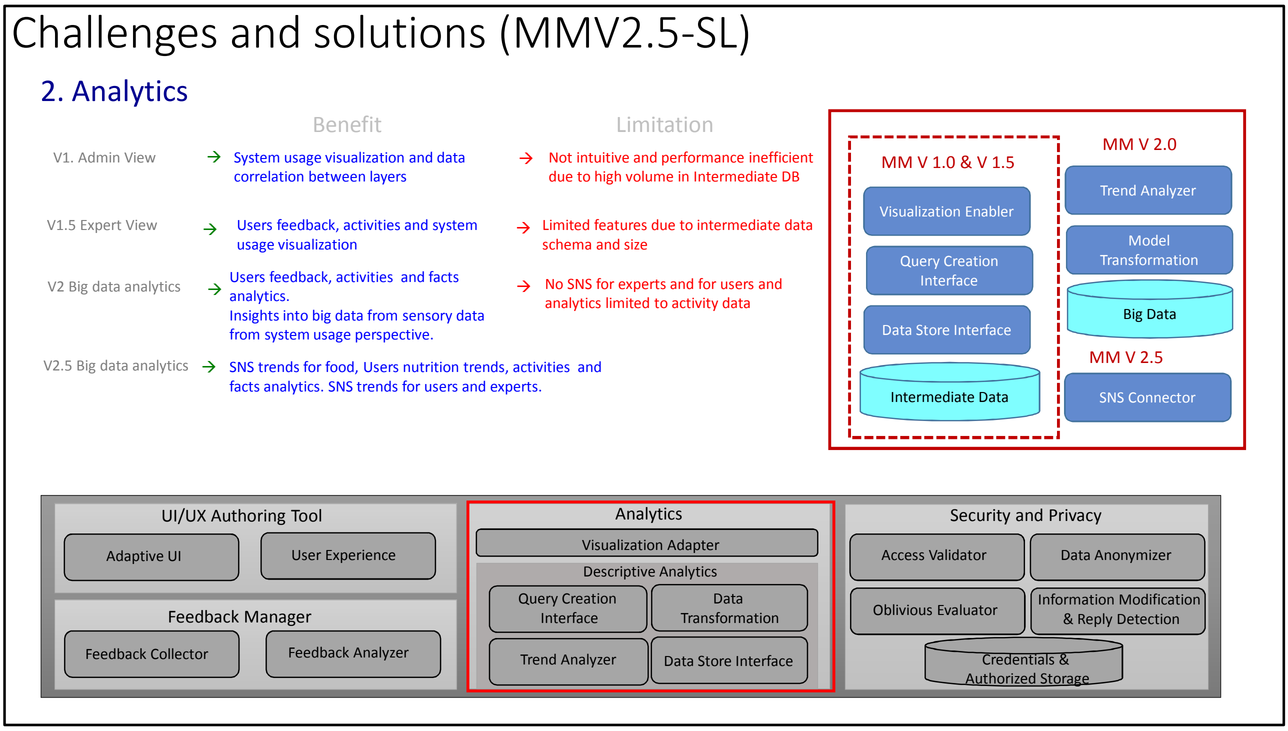
Challenges and solutions (MMV2.5-SL)

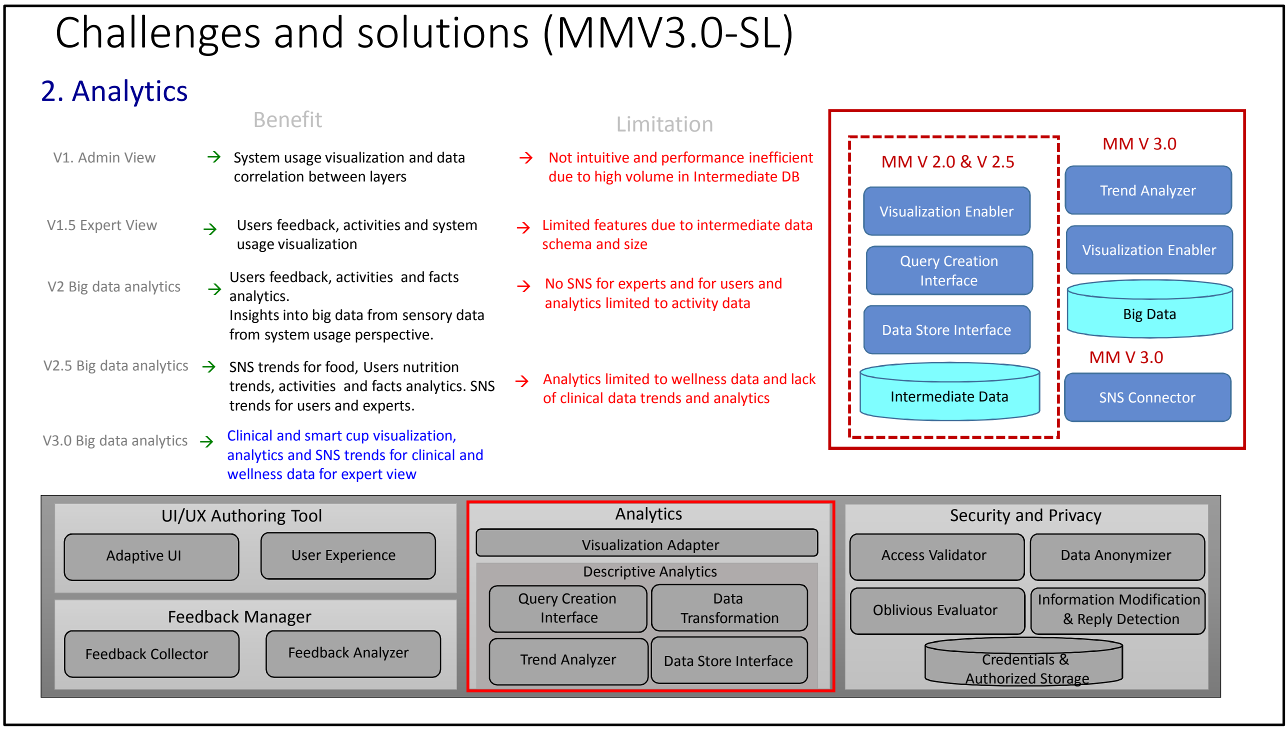
2. Analytics

	Benefit	Limitation
V1. Admin View	→ System usage visualization and data correlation between layers	→ Not intuitive and performance inefficient due to high volume in Intermediate DB
V1.5 Expert View	→ Users feedback, activities and system usage visualization	→ Limited features due to intermediate data schema and size
V2 Big data analytics	→ Users feedback, activities and facts analytics. Insights into big data from sensory data from system usage perspective.	→ No SNS for experts and for users and analytics limited to activity data
V2.5 Big data/SNS analytics	→ SNS trends for food, Users nutrition trends, activities and facts analytics. SNS trends for users and experts.	



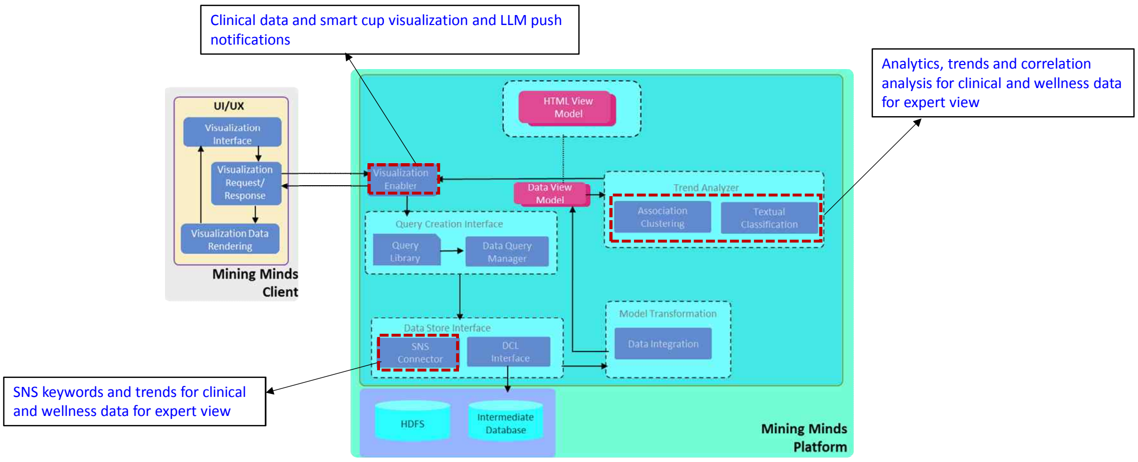
- Nutrition trends and analytics for expert panel and user through service curation layer
- SNS gateway for SNS trends
- Analytics based on statistics, clustering and association
- Insights from the big data repository and communication between both repositories
- Query library for the big data repository


Challenges and solutions (MMV2.5-SL)
2. Analytics
Benefit
Limitation
V1. Admin View
System usage visualization and data correlation between layers
Not intuitive and performance inefficient due to high volume in Intermediate DB
V1.5 Expert View
Users feedback, activities and system usage visualization
Limited features due to intermediate data schema and size
V2 Big data analytics
Users feedback, activities and facts analytics.
Insights into big data from sensory data from system usage perspective.
No SNS for experts and for users and analytics limited to activity data
V2.5 Big data analytics
SNS trends for food, Users nutrition trends, activities and facts analytics. SNS trends for users and experts.
MM V 1.0 & V 1.5
Visualization Enabler
Query Creation Interface
Data Store Interface
Intermediate Data
MM V 2.0
Trend Analyzer
Model Transformation
Big Data
MM V 2.5
SNS Connector
UI/UX Authoring Tool
Adaptive UI
User Experience
Feedback Manager
Feedback Collector
Feedback Analyzer
Analytics
Visualization Adapter
Descriptive Analytics
Query Creation Interface
Data Transformation
Trend Analyzer
Data Store Interface
Security and Privacy
Access Validator
Data Anonymizer
Oblivious Evaluator
Information Modification & Reply Detection
Credentials & Authorized Storage


Challenges and solutions (MMV3.0-SL)
2. Analytics
Benefit
Limitation
V1. Admin View
System usage visualization and data correlation between layers
Not intuitive and performance inefficient due to high volume in Intermediate DB
V1.5 Expert View
Users feedback, activities and system usage visualization
Limited features due to intermediate data schema and size
V2 Big data analytics
Users feedback, activities and facts analytics.
Insights into big data from sensory data from system usage perspective.
No SNS for experts and for users and analytics limited to activity data
V2.5 Big data analytics
SNS trends for food, Users nutrition trends, activities and facts analytics. SNS trends for users and experts.
Analytics limited to wellness data and lack of clinical data trends and analytics
V3.0 Big data analytics
Clinical and smart cup visualization, analytics and SNS trends for clinical and wellness data for expert view
MM V 2.0 & V 2.5
Visualization Enabler
Query Creation Interface
Data Store Interface
Intermediate Data
MM V 3.0
Trend Analyzer
Visualization Enabler
Big Data
MM V 3.0
SNS Connector
UI/UX Authoring Tool
Adaptive UI
User Experience
Feedback Manager
Feedback Collector
Feedback Analyzer
Analytics
Visualization Adapter
Descriptive Analytics
Query Creation Interface
Data Transformation
Trend Analyzer
Data Store Interface
Security and Privacy
Access Validator
Data Anonymizer
Oblivious Evaluator
Information Modification & Reply Detection
Credentials & Authorized Storage

Challenges and solutions (MMV3.0-SL)

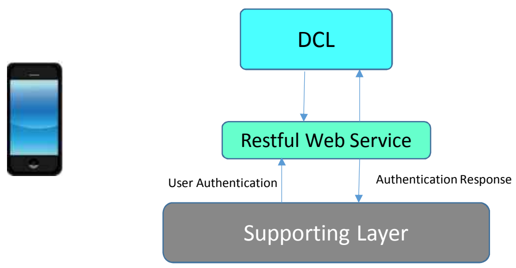
2. Analytics



Challenges and solutions (MMV2-SL)

3. Security

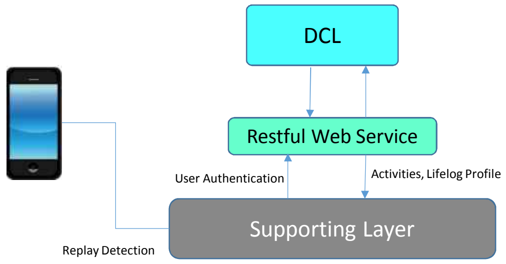
	Benefit	Limitation
V1.5 Expert View Authentication	→ User Authentication based on access layer protocol for expert panel Data encryption and decryption for databases	→ No secure communication between databases and mobile application



Challenges and solutions (MMV2-SL)

3. Security

		Benefit		Limitation
V1.5 Expert View Authentication	→	User Authentication based on access layer protocol for expert panel data encryption and decryption for databases	→	No secure communication between databases and mobile application
V2 Communication between user application and data repositories	→	Replay Detection implementation for secure communication. Data encryption and decryption for databases		



<u>Section 3</u>

Mining Minds Open Source Process

Section 3.1

Open Source Plan and Release Process

Mining Minds v3.0

Open Source Plan and Release process

M. Bilal Amin (PhD)
m.b.amin@ieee.org

경희대학교
KYUNG HEE UNIVERSITY

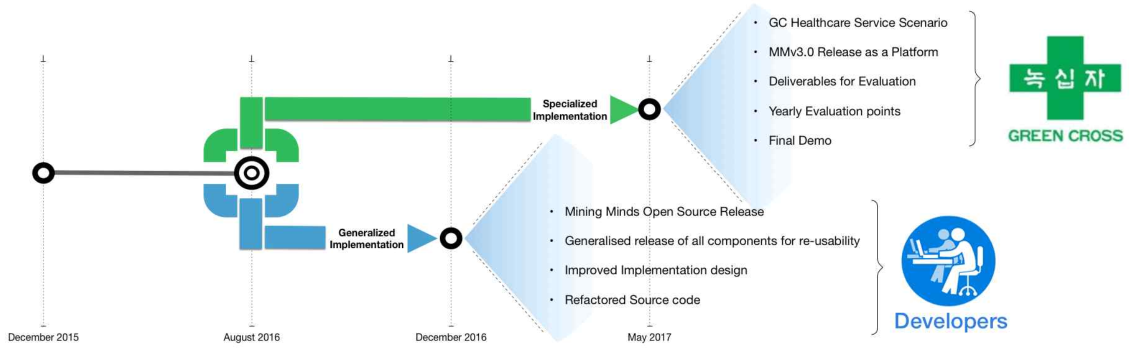
Agenda

2

- Timeline
- Open Source Process
- Timelines
- Implementation Examples
- Build & Release Process
- License
- Benefits

Timelines

3



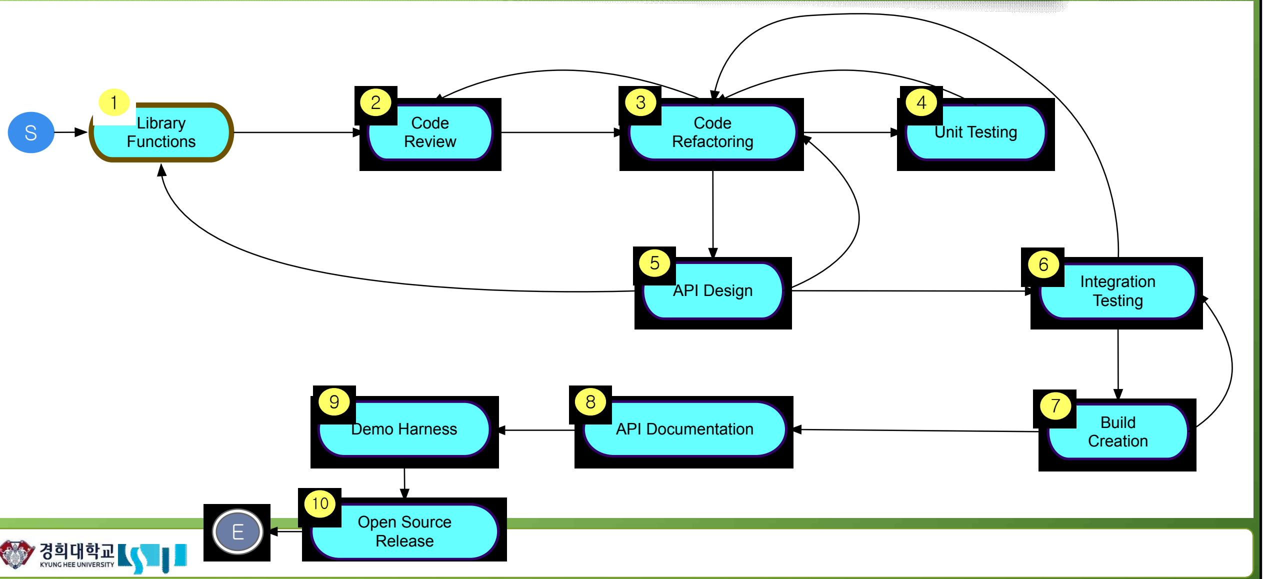
Mining Minds

Open source Process

경희대학교
KYUNG HEE UNIVERSITY

Open Source Release Process

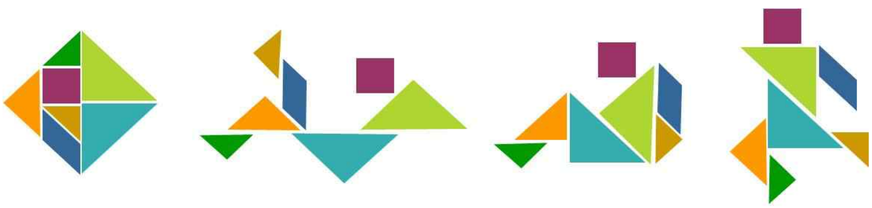
5



Task 1: Library Functions

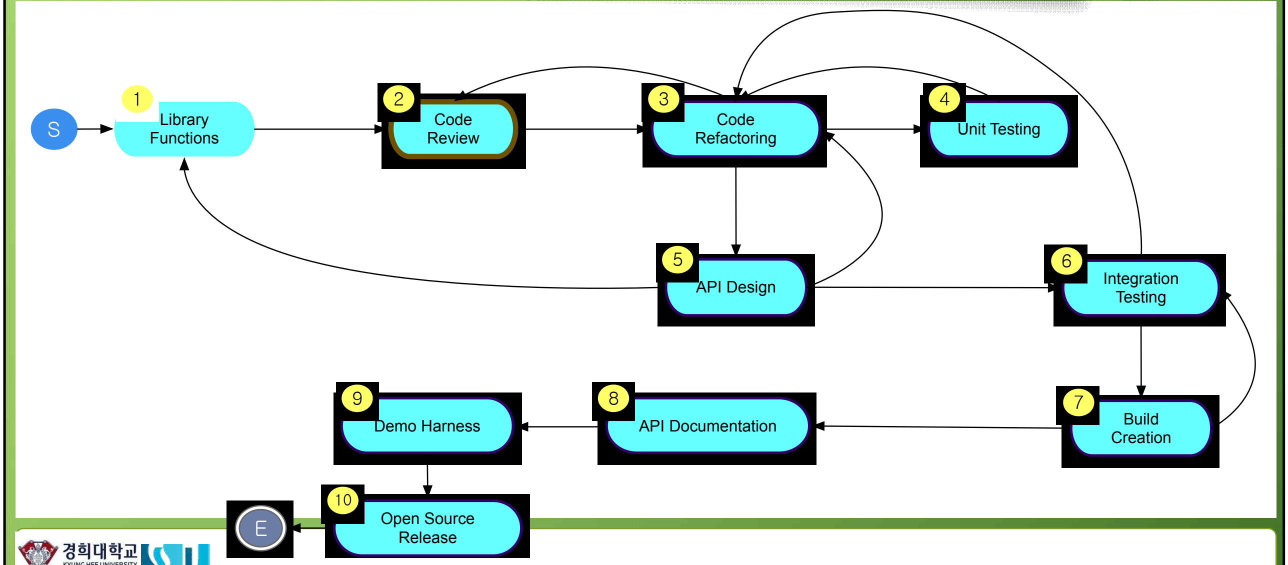
6

- Goal : Identification of Reusable Methods as high-level abstraction for Layer API
- Example: [DCL]
  - Add new Sensory Data Acquisition Device
  - Use LLM on historic data for Behavior Modeling
  - Use ADR for Descriptive and Predictive Analytics



Same Pieces – Different Uses

Open Source Release Process

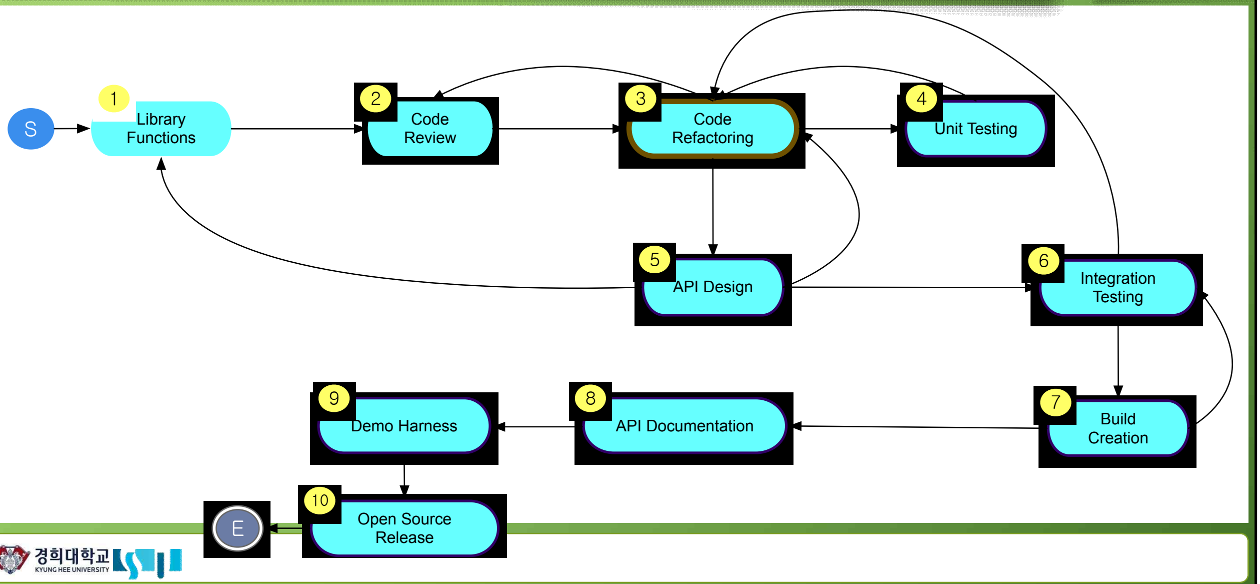


Task 2: Code Review

- Code Review Team: Dr. Bilal, Dr. Ali, Dr. Maqbool, Mr. Taqdir, Component Expert
- Goal: Determine the Quality of the Code Following implementation standards
  - Defect-free, well-documented software
  - Software that complies with enterprise coding standards
  - Teaching and sharing knowledge between team members

Open Source Release Process

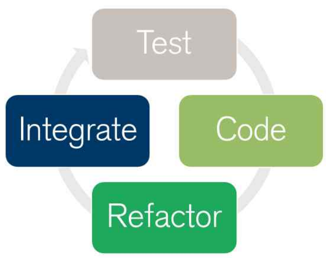
9



Task 3: Code Re-factoring (1/4)

10

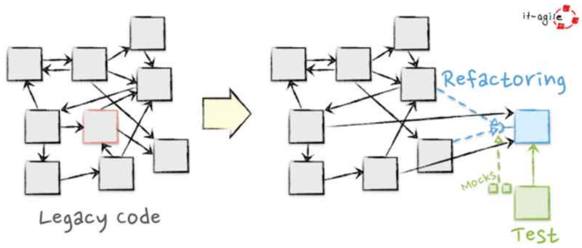
- Goal: Re-organization of code for Increased Re-usability, Extensibility, and Maintainability
- When the System's source code is easily understandable, the system is more maintainable
- Improving the design of existing code without changing its observable behavior
- Coding Standards
  - Java: https://www.securecoding.cert.org/confluence/display/java/Java+Coding+Guidelines
  - Javascript: http://javascript.crockford.com/code.html
  - Commenting Guidelines:
    - http://www.hongkiat.com/blog/source-code-comment-styling-tips/
    - http://www.devtopics.com/13-tips-to-comment-your-code/



Task 3: Code Re-factoring (2/4)

11

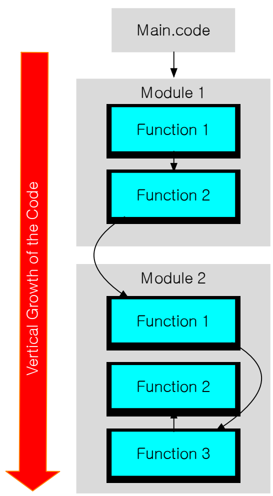
- Refactoring includes:
  - Rename attributes
  - Move Class / Structures
  - Create Abstraction / Extract Interface
  - Extract method for Re-usability
  - Extract Superclass
  - Encapsulation / Properties generation
  - Replace Conditions with Polymorphism
  - http://www.methodsandtools.com/archive/archive.php?id=4

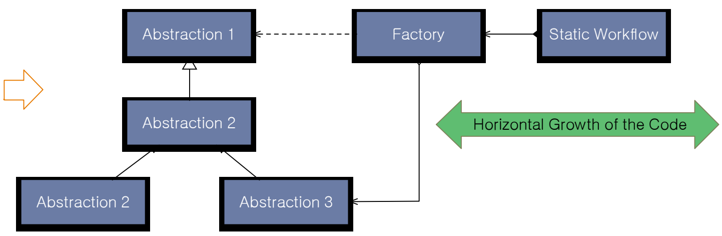


경희대학교 KYUNG HEE UNIVERSITY

Task 3: Code Re-factoring (3/4)

12





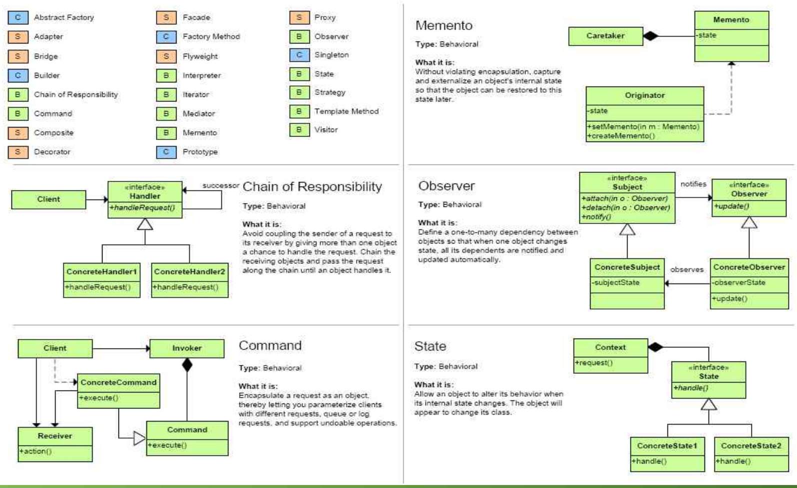
- Inclusion of Object Oriented Design Patterns
  - A way to solve issues related to software development using a proven solution
  - Facilitates the development of highly cohesive modules with minimal coupling
  - Isolate the variability that may exist, making the overall system easier to understand and maintain.
  - Increases communication efficiency between developers
    - Software professionals can immediately picture the high-level design in their heads when they refer the name of the pattern used to solve a particular issue.

경희대학교 KYUNG HEE UNIVERSITY

Task 3 Code Re-factoring (4/4)

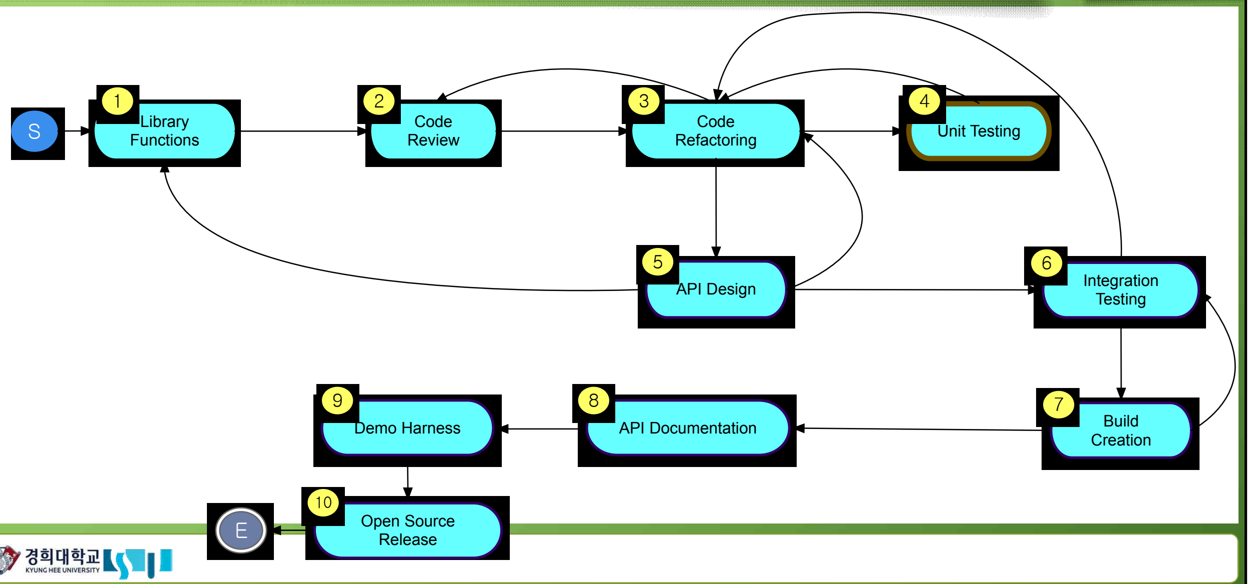
13

Inclusion of Object Oriented Design Patterns



Open Source Release Process

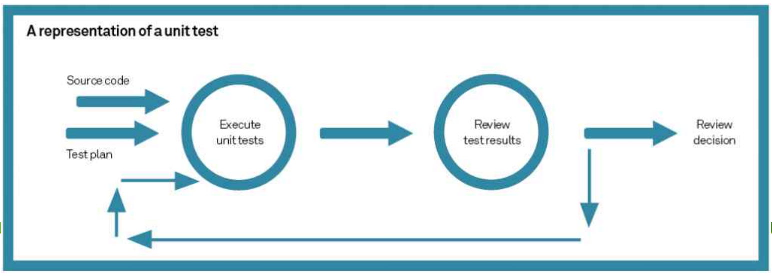
14



Task 5: Unit Testing

15

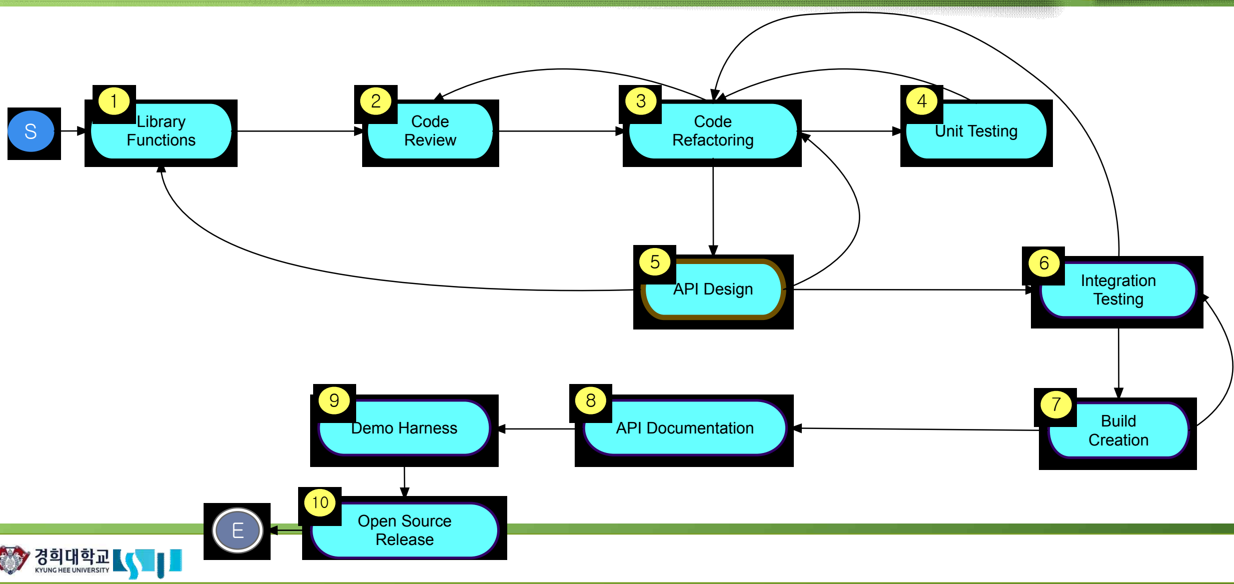
- The smallest testable parts of an application, called units, are individually and independently scrutinized for proper operation.
- Goal: Isolate each part of the program and show that the individual parts are correct
- Automated Unit Testing Frameworks available
  - JUnit is a unit testing framework for the Java programming language. http://junit.org/junit4/
  - QUnit is a JavaSript unit testing framework. https://qunitjs.com/intro/



경희대학교 KYUNG HEE UNIVERSITY

Open Source Release Process

16

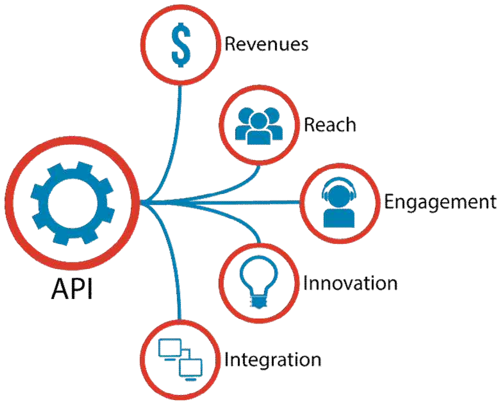


경희대학교 KYUNG HEE UNIVERSITY

Task 5: API Design

17

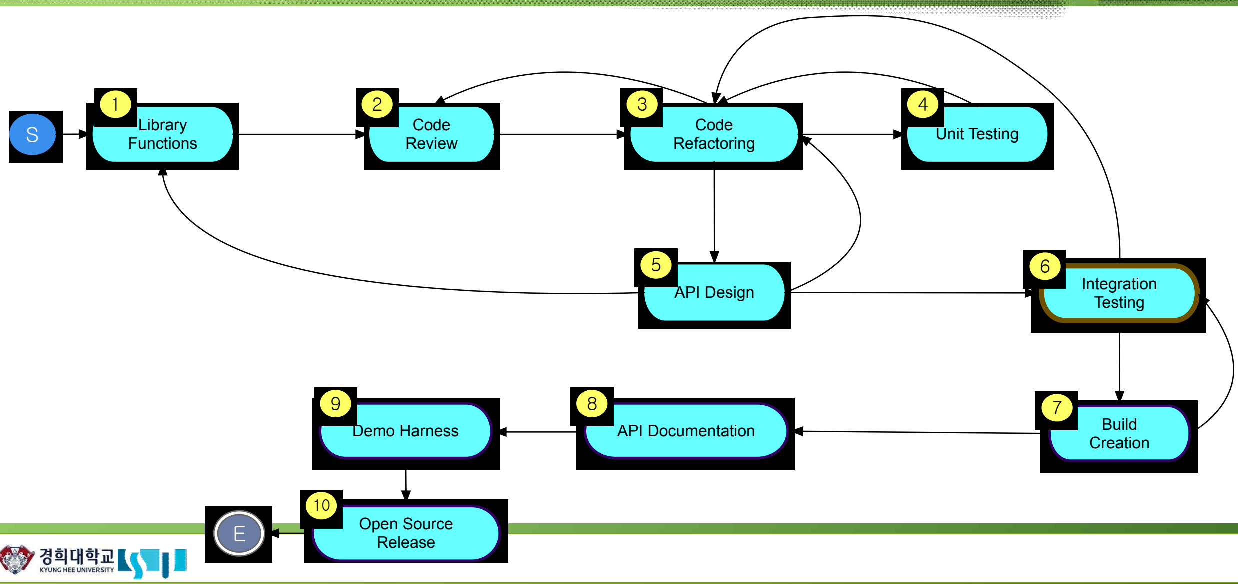
- Goal : Ease of Use for the developer
- API design should reflect:
  - Easy to learn
  - Easy to use, even without documentation
  - Hard to misuse
  - Easy to read and maintain code that uses it
  - Sufficiently powerful to satisfy requirements
  - Easy to extend
  - Appropriate to audience



- http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/32713.pdf

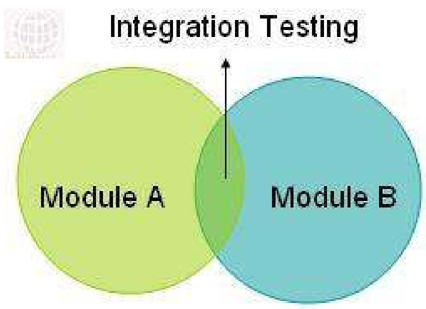
Open Source Release Process

18

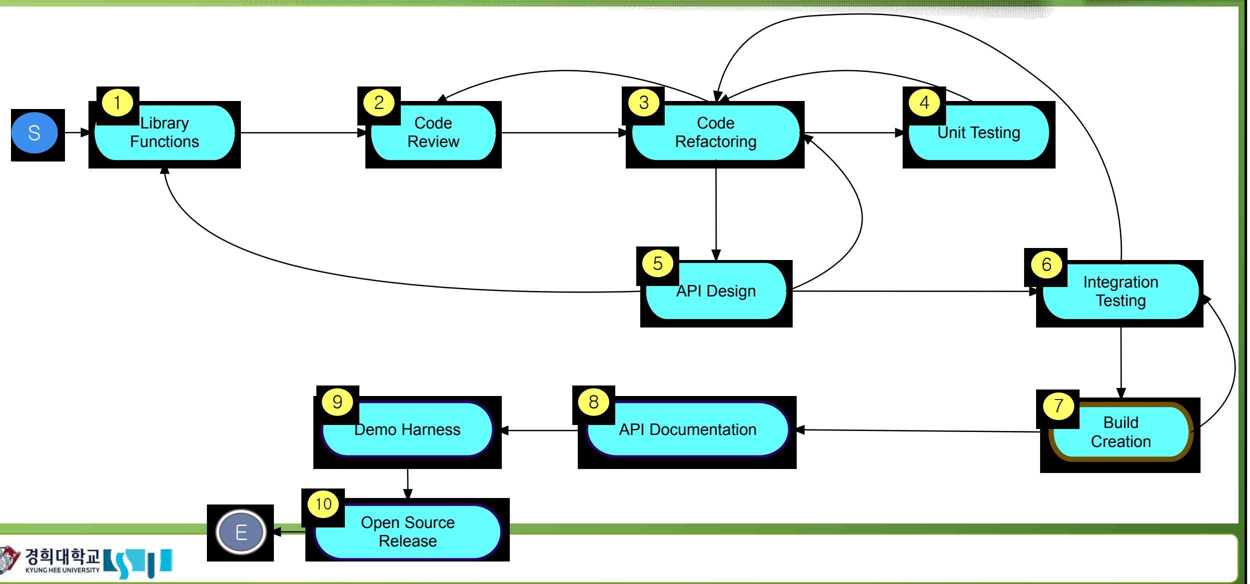


Task 6: Integration Testing

- Goal: Integrated execution of a layer as a composite entity
- Individual software modules are combined and tested as a group.
- It occurs after unit testing and before validation



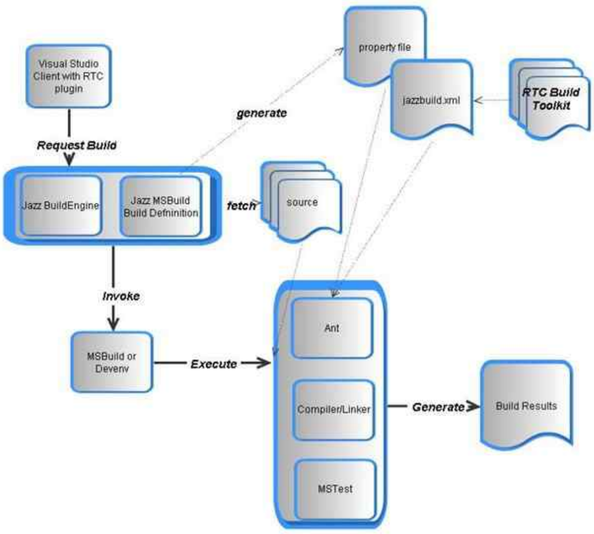
Open Source Release Process



Task 7: Build Creation

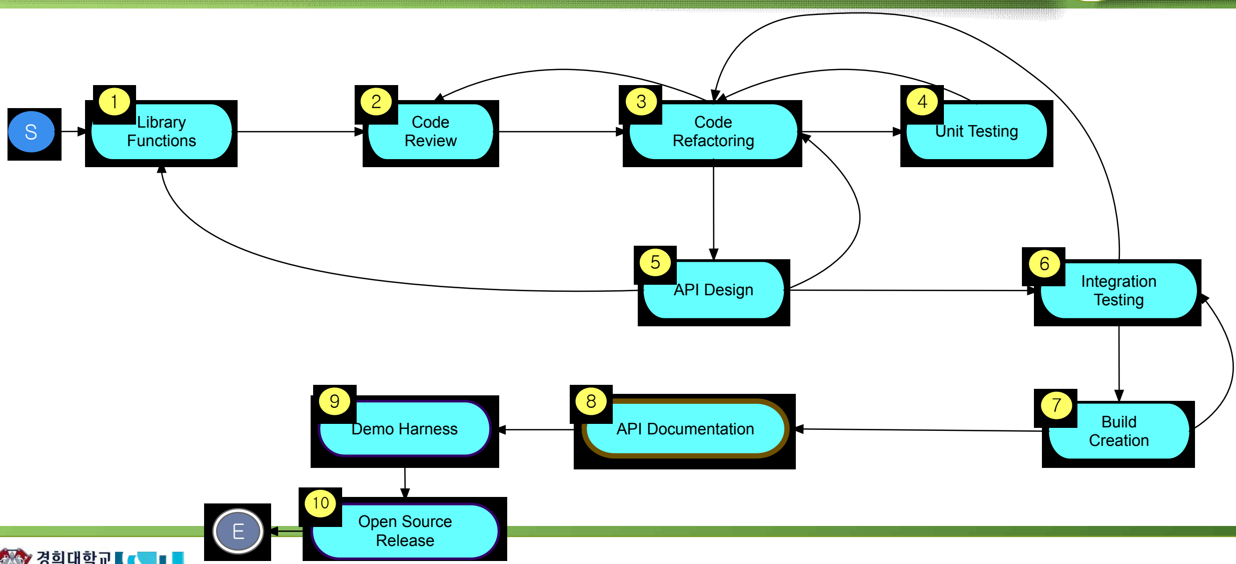
21

- Goal : Integrated compilation of a layer as a composite entity
- Component-wise build and Layer-wise build
- Rule of thumb : Build must never break (Never ever!!!)
- Nightly builds, weekly builds etc.
- ANT : http://ant.apache.org
- Every layer will nominate a Build Engineer. (Mao for DCL)



Open Source Release Process

22



Task 8: API Documentation

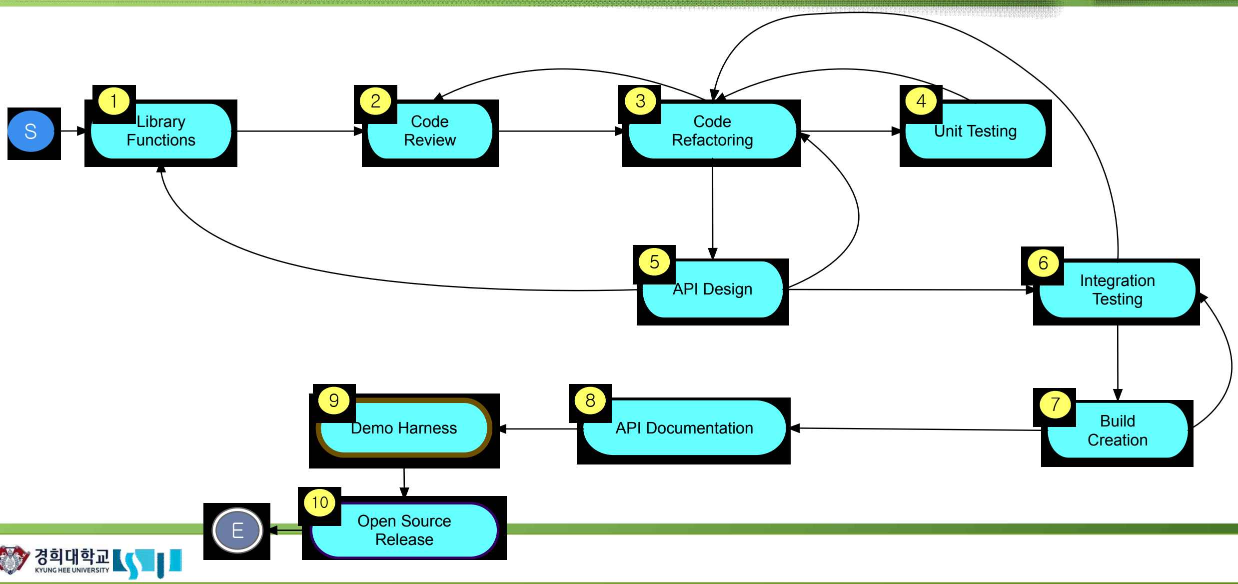
23

- Goal: Documentation of implemented library functions in a standard hierarchical format
  - Easy to understand
  - Available in downloadable and browse-able form
  - Must provide traceability between versions
  - Automated API Document generation tools
- Example : http://api.jquery.com
- You are making a developer focused product, the documentation is as core to the user experience as the endpoints themselves.
- http://blog.parse.com/learn/engineering/designing-great-api-docs/

경희대학교 KYUNG HEE UNIVERSITY

Open Source Release Process

24



경희대학교 KYUNG HEE UNIVERSITY

Task 9: Demo Harness

25

- Goal : Demo to the API, illustrating its usage
  - Realistic problem with solution
  - Mostly utilized as tutorials
  - Must execute from end-to-end to illustrate complete execution
  - Not a trivial one

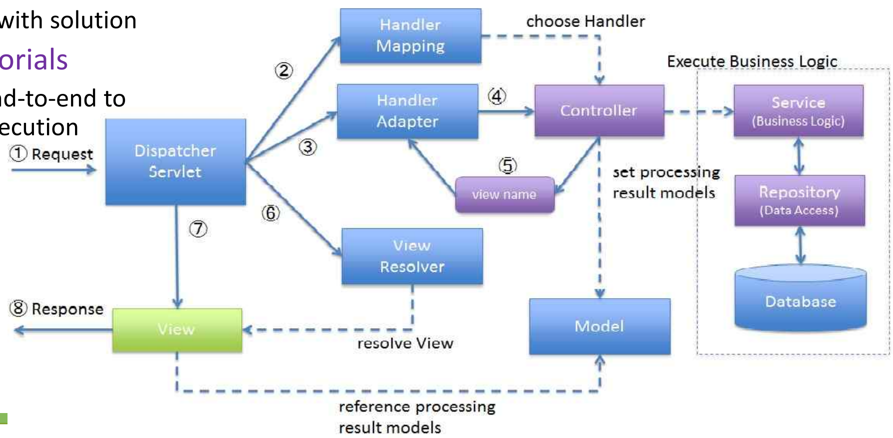
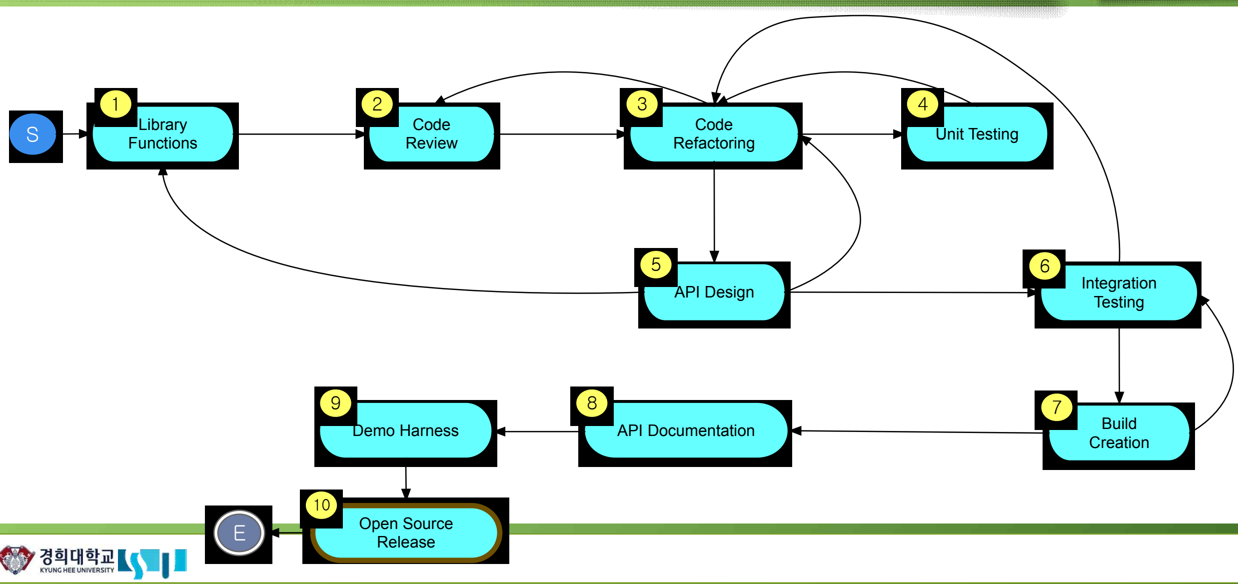


Fig by javawithease

Open Source Release Process

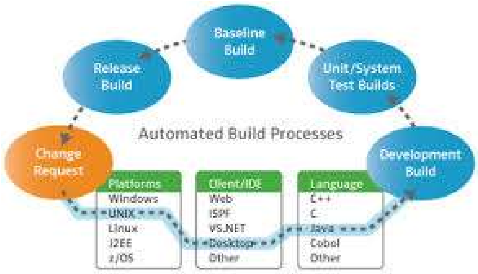
26



Task 10: Open Source Release

27

- Goal : Availability of the Code for Open source community to download and contribute
- Web-based repositories such as github or sourceforge will be used
- After release all the Ming Minds code will be fork or commit from the web-based repository
- Familiarize yourself with code repository terms



경희대학교 KYUNG HEE UNIVERSITY

Mining Minds V3.0

Process Implementation

경희대학교 KYUNG HEE UNIVERSITY

DCL Gantt Chart

29

#	Title		Jul 17 Jul 23	Jul 24 Jul 30	Jul 31 Aug 6	Aug 7 Aug 13	Aug 14 Aug 20	Aug 21 Aug 27	Aug 28 Sep 3	Sep 4 Sep 10	Sep 11 Sep 17	Sep 18 Sep 24	Sep 25 Oct 1	Oct 2 Oct 8	Oct 9 Oct 15	Oct 16 Oct 22	Oct 23 Oct 29	Oct 30 Nov 5	Nov 6 Nov 12	Nov 13 Nov 19	Nov 20 Nov 26	Nov 27 Dec 3	Dec 4 Dec 10
1	DAS & LLM Library Functions Identification (Mao & Bilal Ali)	3w																					
2	LLRM Library Functions Identification (Taqdir)	3w																					
3	ADR & PDR Library Functions Identification (Usman & Cho)	3w																					
4	DAS & LLM Code Review (Mao & Bilal Ali)	1w																					
5	LLRM, ADR, & PDR, Code Review (Taqdir, Usman & Cho)	1w																					
6	Code Refactoring & Unit Testing (Mao, Usman, Bilal Ali, Taqdir, & Cho)	10w																					
7	API Design & Documentation (Mao, Usman, Bilal Ali, Taqdir, & Cho)	3w																					
8	Demo Harness (Mao, Usman, Bilal Ali, Taqdir, & Cho)	4w																					
9	Build Creation (Mao & Usman)	2w																					
10	Open Source Release (TEAM DCL)	0h																					

ICL Gantt Chart

30

No	Title	Effort	Jul 17-Jul 23	Jul 24-Jul 30	Jul 31-Aug 6	Aug 7-Aug 13	Aug 14-Aug 20	Aug 21-Aug 27	Aug 28-Sep 3	Sep 4-Sep 10	Sep 11-Sep 17	Sep 18-Sep2 4	Sep 25-Oct 1	Oct 2-Oct 8	Oct 9-Oct. 15	Oct 16 - Oct 22	Oct 23 - Oct 29	Oct. 30 - Nov 5	Nov 6 - Nov 12	Nov 13 - Nov 19	Nov 20-Nov 26	Nov 27-Dec 3	Dec 4 - Dec 10
1	Smartphone based AR Library Functions Identification -Tae Ho, Dong Uk, Jaehun	3w																					
2	HLCA Library Functions Identification -Asif	3																					
3	Code Review for ICL - Dong Uk, Jaehun	4w																					
4	Code Re-factoring - Dong Uk	3w																					
5	API Design - Jaehun, Asif, Dong Uk	3w																					
6	Unit Testing - Dong Uk, Jaehun, Asif	2w																					
7	Build Creation - Dong Uk, Jaehun, Asif	2w																					
8	API Documentation - Jaehun, Dong Uk, Asif	4w																					
9	Demo Harness - Jaehun, Asif	1w																					
10	Open Source Release -TEAM ICL	0h																					

KCL Gantt Chart

31

#	Title		Jul 31 Aug 6	Aug 7 Aug 13	Aug 14 Aug 20	Aug 21 Aug 27	Aug 28 Sep 3	Sep 4 Sep 10	Sep 11 Sep 17	Sep 18 Sep 24	Sep 25 Oct 1	Oct 2 Oct 8	Oct 9 Oct 15	Oct 16 Oct 22	Oct 23 Oct 29	Oct 30 Nov 5	Nov 6 Nov 12	Nov 13 Nov 19	Nov 20 Nov 26	Nov 27 Dec 3	Dec 4 Dec 10
1	**Rules Meta Information API** • Function Identification • API Design • Code Refactoring (Taqdir)	4w																			
2	**Rules Fetching API** • Function Identification • API Design • Code Refactoring (Taqdir)	4w																			
3	**Rules Persistence API** • Function Identification • API Design • Code Refactoring (Taqdir)	4w																			
4	**Situation Sharing API** • Function Identification • API Design • Code Refactoring (Dr. Maqbool)	3w																			
5	**Rule Sharing API** • Function Identification • API Design • Code Refactoring (Dr. Maqbool)	3w																			
6	**I-KAT Toolkit** • Rules Dashboard GUI • Rule Editor GUI (Taqdir)	4w																			
7	**Code Review** • Unit testing & documentation • Creating Build (Dr. Maqbool, Taqdir)	5w																			

SCL Gantt Chart

1

#	Title		SEP 18 OCT 3	OCT 4 OCT 9	OCT 10 OCT 12	OCT 13 OCT 19	OCT 20 OCT 28	OCT 29 NOV 3	NOV 4 NOV 10	NOV 11 NOV 17	NOV 18 NOV 30	DEC 1 DEC 6	DEC 7 DEC 7
1	Rebuilding RI on Design Pattern (Imran)	16d											
2	Library Functions Identification (Afzal, Imran & Sadiq)	7d											
3	Code Review (Afzal, Imran & Sadiq)	3d											
4	Code Refactoring (Afzal, Imran & Sadiq)	7d											
5	Unit Testing (Afzal, Imran & Sadiq)	9d											
6	API Design (Afzal, Imran & Sadiq)	5d											
7	Integration Testing (Imran & Sadiq)	7d											
8	Demo (Sadiq & Imran)	7d											
9	API Documentation (Imran & Sadiq)	17d											
10	Build Creation (Sadiq)	6d											
11	Open Source Release (SCL Team)	1d											

SL Gantt Chart

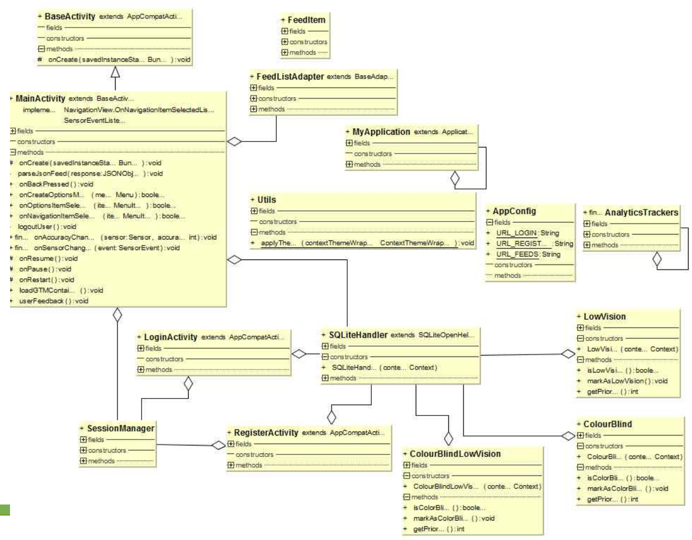
33

#	Title		Oct 2 – Oct 8	Oct 9 – Oct 15	Oct 16 – Oct 22	Oct 23 – Oct 29	Oct 30 – Nov 5	Nov 6 – Nov 12	Nov 13 – Nov 19	Nov 20 – Nov 26	Nov 27 – Dec 3	Dec 4 – Dec 10
1	Descriptive Analytics Library Functions Identification (Shujaat Hussain)	2w										
2	UI/UX Functions Identification (Jamil Hussain)	2w										
4	Descriptive Analytics Code Review (Shujaat Hussain)	1w										
5	UI/UX Code Review (Jamil Hussain)	1w										
6	Code Refactoring & Unit Testing (Shujaat and Jamil)	6w										
7	API Design & Documentation (Shujaat and Jamil)	4w										
8	Demo Harness (Shujaat and Jamil)	3w										
9	Build Creation (Shujaat and Jamil)	3w										
10	Open Source Release (Shujaat and Jamil)	2w										

Code Redesign : Object Oriented Design Patterns

34

- **For complex rules handling**
  - **composite design pattern:** A composite rule is triggered if all conditions of its composing rules are satisfied.
- **Scheduling a rules engine**
  - **Java scheduler Quartz :** This will schedule the rules engine to start based on specified time-based
- **Handling runtime event condition handling**
  - **Observer Design pattern**: It detect the runtime change in any event condition and change the adaptation rule engine behavior after that change

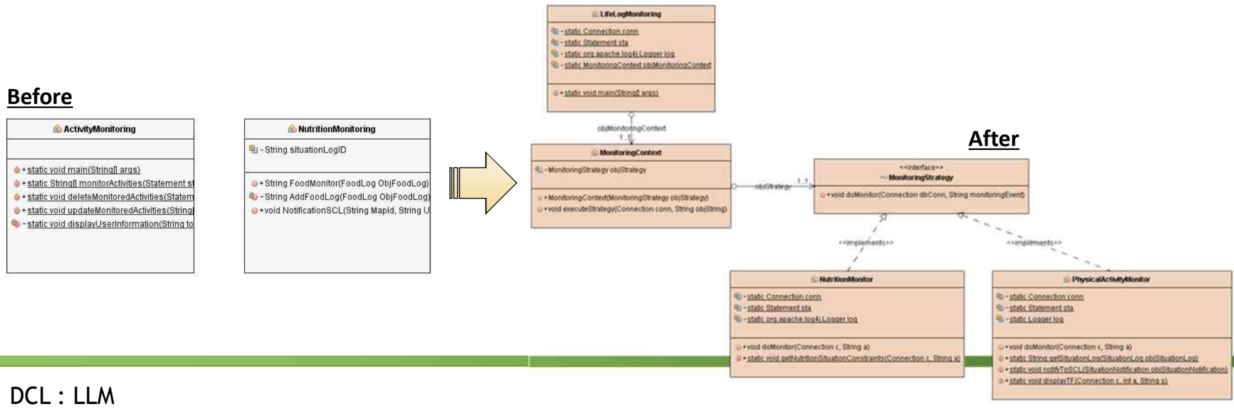


SL : UI/UX

Object Oriented Design Patterns

35

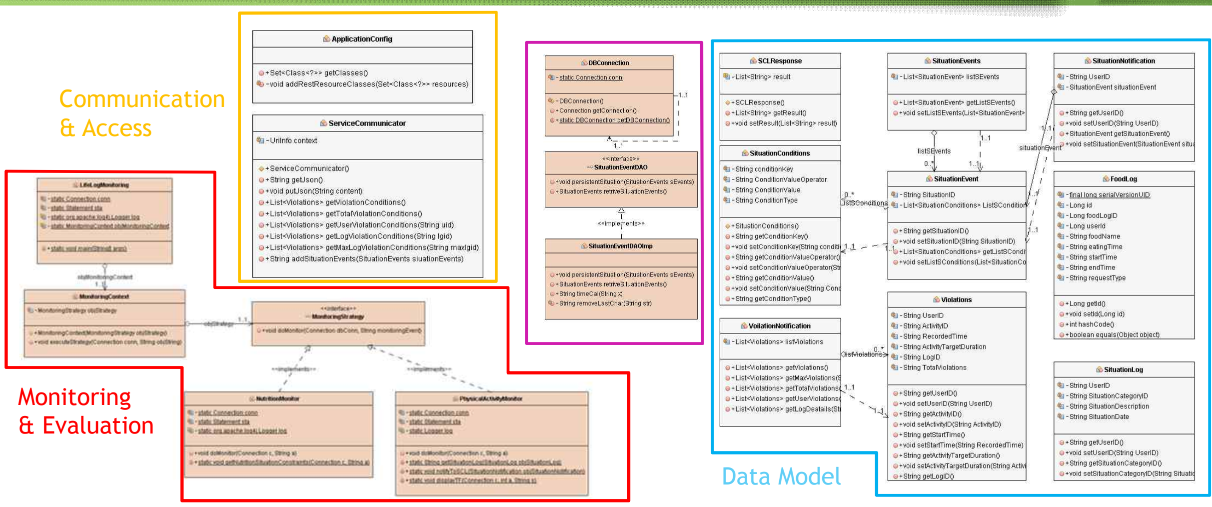
Strategy pattern gives flexibility to change monitoring algorithm at run time.
The **behavior pattern** to represent various monitoring strategies that changes the executing algorithm of the context object.



DCL : LLM

Object Oriented Design Patterns

36

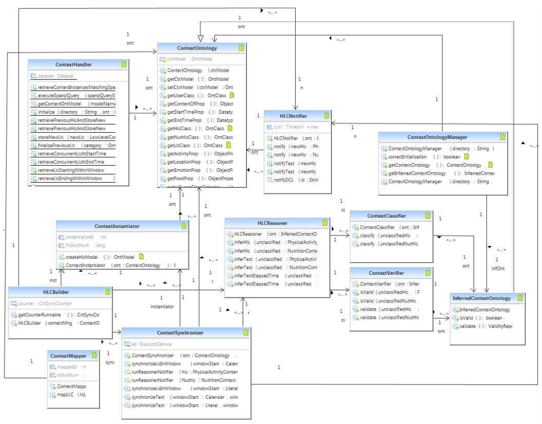


DCL : LLM

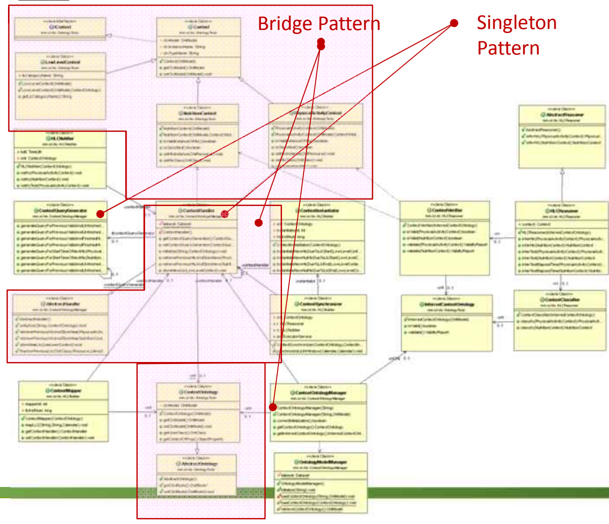
Object Oriented Design Patterns

37

Before



After



ICL : High Level Context awareness

Code Refactoring Examples (Code Commenting)

38

Function Level Comments

```
        try
        {
            Session session = this.sessionFactory.openSession();
            Rule objRule = (Rule) session.createQuery("from Rule r left outer join fetch r.c
            logger.info("Rule loaded successfully, Rule details="+objRule);
            session.close();
            return objRule;
        }
        catch(Exception  ex )
        {
            logger.info("Error occurred in rule loading "+ ex.getMessage());
            return null;
        }
    }
```

```
        try
        {
             Session session = this.sessionFactory.openSession();
             Rule objRule = (Rule) session.load(Rule.class, new Integer(id));
                if(null != objRule){
                    session.delete(objRule);
                }
                session.close();
                logger.info("Rule deleted successfully, Rule details="+objRule);
        }
        catch(Exception  ex )
        {
            logger.info("Error occurred in rule deleting "+ ex.getMessage());
        }

    }
```

Class Level Comments

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Repository;
import org.uclab.mm.kcl.edkat.datamodel.Conclusion;
import org.uclab.mm.kcl.edkat.datamodel.Rule;

```

```
@Repository
public class RuleDAOImpl implements RuleDAO {

    /*
     * This creation of static object of logger for Rule
    */
    private static final Logger logger = LoggerFactory.getLogger(RuleDAOImpl.class);

    /*
     * This creation of Session for factory
    */
     private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sf){
        this.sessionFactory = sf;
    }

    /*
     * This function is the implementation for add new rule
    */
    public Rule addRule(Rule objRule) {
        try
        {
```

KCL : Expert-Driven Knowledge Acquisition

Refactoring Examples (Commenting & Logging)

39

Before Refactoring

```
@Override
public List<UserRiskFactors> RetriveData(Object objUserRiskFactors) {
    UserRiskFactors objOuterUserRiskFactors = new UserRiskFactors();
    List<UserRiskFactors> objListInnerUserRiskFactors = new ArrayList<UserRiskFactors>();
    objOuterUserRiskFactors =  (UserRiskFactors) objUserRiskFactors;

    try
    {
```

Code comments for understandability

```
@GET
@Produces("application/json")
@Consumes("application/json")
@Path("UserDevice/{UserDeviceID}")
public List<UserDevice> GetUserDevice(@PathParam ("UserDeviceID") String UserDeviceID) {
```

After Refactoring

Before Refactoring

```
catch (Exception e)
{
    e.printStackTrace();
    objDbResponse.add("Error");
    System.err.println("Problem in registration user!");
}
return objDbResponse;
```

Logging Exception

```
try
{
    Session session = this.sessionFactory.openSession();
            Rule objRule = (Rule) session.createQuery("from Rule r left outer join fetch r
    logger.info("Rule loaded successfully, Rule details="+objRule);
    session.close();
    return objRule;
}
catch(Exception  ex )
{
    logger.info("Error occurred in rule loading "+ ex.getMessage());
    return null;
}
```

After Refactoring

DCL : Lifelog Mapping & Representation

Refactoring Examples (Logging)

40

Before Refactoring

```
System.out.println(
        e.printStackTrace();
}
```

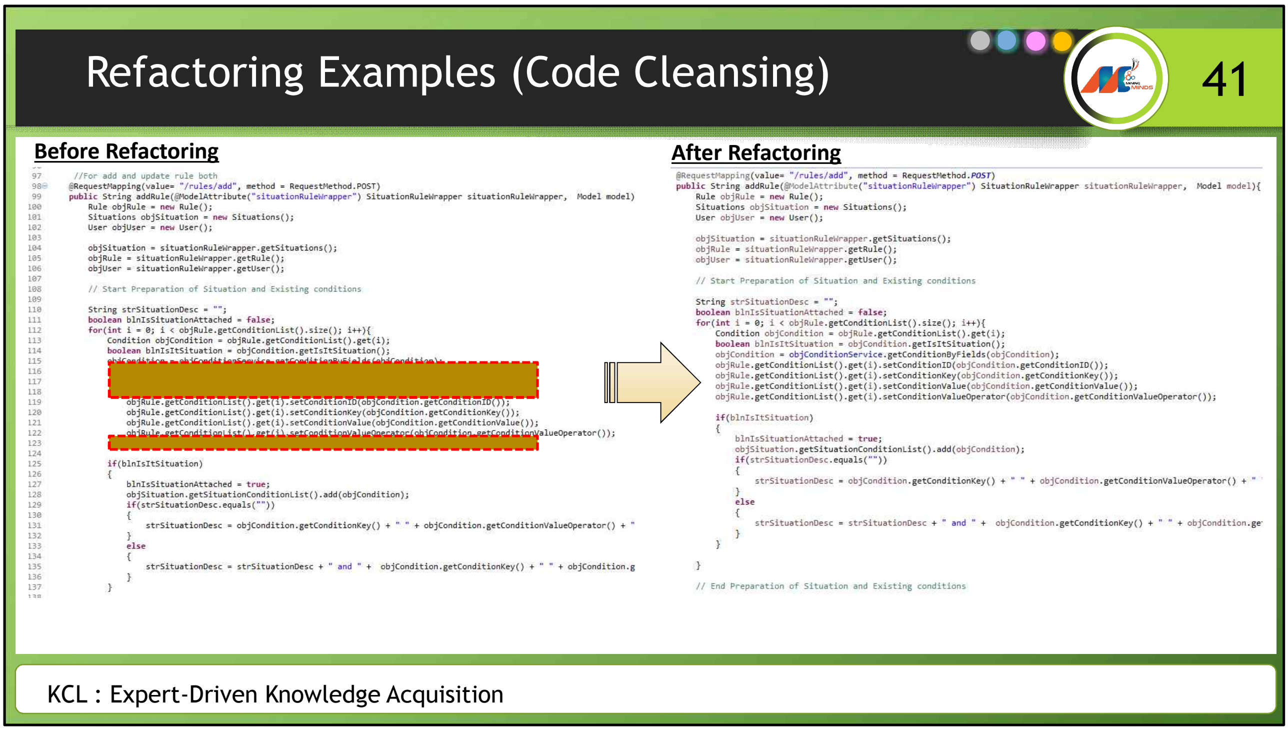
After Refactoring

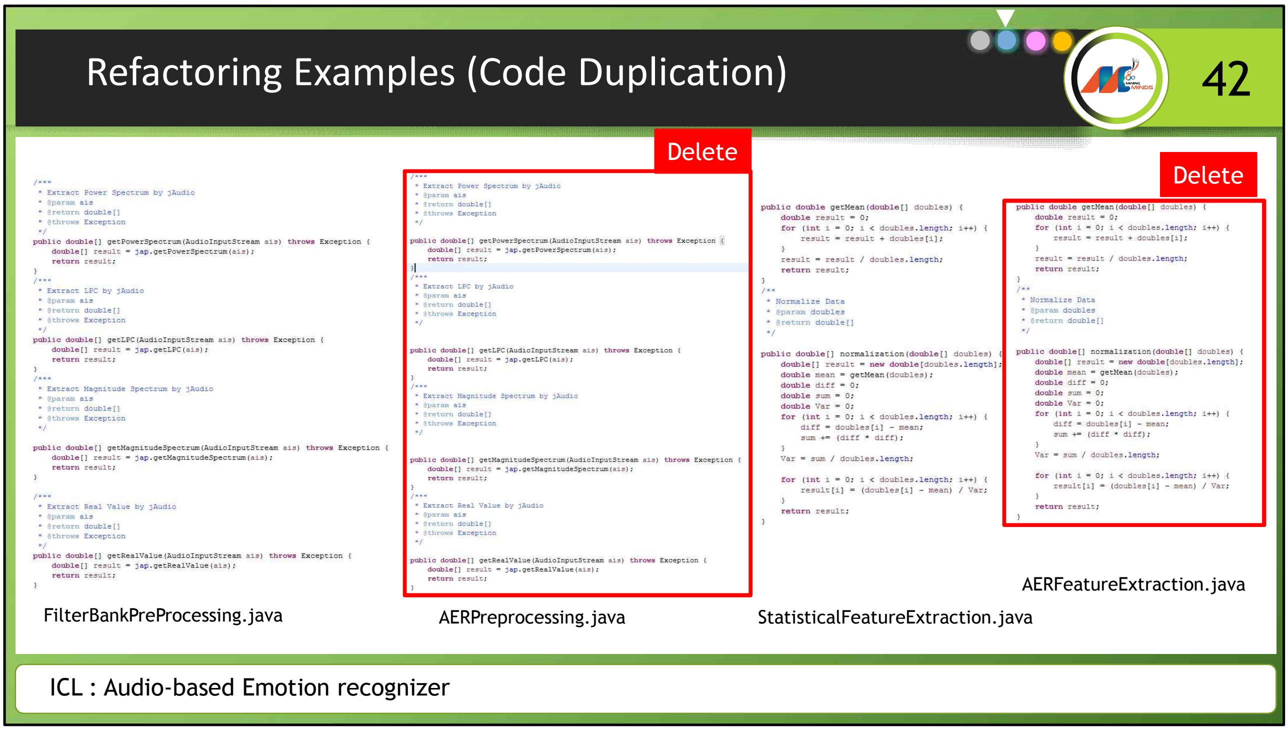
```
String timestampS = tutil.parseCal(dateStart);
Log.info("[HLC Notifier] DCL Notification Message: " + id + ", " + label + ", " + userID + ", " + timestampS);
try {

} catch (Exception e) {
    Log.error("Issue with HLC Notifier");
}
}
```

```
2016-11-19 02:48:53 INFO  HLCNotifier:238 - ********--------------------*******************************------------------*******************
2016-11-19 02:48:53 INFO  HLCNotifier:239 - UnidentifiedHLC
2016-11-19 02:48:53 INFO  HLCNotifier:240 - rest hlc of label UnidentifiedHLC ["39754","No Error"]
2016-11-19 02:48:53 INFO  HLCNotifier:241 - ********--------------------*******************************------------------*******************
2016-11-19 02:48:54 INFO  HLCNotifier:142 - [HLC Notifier: PhysicalActivityContext: ] New instance: hlc_0966046156_0000000000000000004 - Inactivity.
Previous Instance: none. Notify DCL.
2016-11-19 02:48:54 INFO  HLCNotifier:235 - [HLC Notifier] DCL Notification Message: hlc_0966046156_0000000000000000004, Inactivity, 9735, 2016 04 1
10:11:17
2016-11-19 02:48:54 INFO  HLCNotifier:238 - ********--------------------*******************************------------------*******************
2016-11-19 02:48:54 INFO  HLCNotifier:239 - Inactivity
2016-11-19 02:48:54 INFO  HLCNotifier:240 - rest hlc of label Inactivity ["39755","No Error"]
2016-11-19 02:48:54 INFO  HLCNotifier:241 - ********--------------------*******************************------------------*******************
```

ALL	All levels including custom levels.
DEBUG	Designates fine-grained informational events that are most useful to debug an application.
ERROR	Designates error events that might still allow the application to continue running.
FATAL	Designates very severe error events that will presumably lead the application to abort.
INFO	Designates informational messages that highlight the progress of the application at coarse-grained level.
OFF	The highest possible rank and is intended to turn off logging.
TRACE	Designates finer-grained informational events than the DEBUG.
WARN	Designates potentially harmful situations.

ICL : High Level Context awareness


Refactoring Examples (Code Cleansing)
41
Before Refactoring
After Refactoring
KCL : Expert-Driven Knowledge Acquisition


Refactoring Examples (Code Duplication)
42
Delete
Delete
FilterBankPreProcessing.java
AERPreprocessing.java
StatisticalFeatureExtraction.java
AERFeatureExtraction.java
ICL : Audio-based Emotion recognizer

Refactoring Examples (Exception Handling)

43

Before Refactoring

```java
    public Rule addRule(Rule objRule) {
        Session session = this.sessionFactory.openSession();
        Transaction tx = session.beginTransaction();
        objRule = (Rule) session.merge(objRule);
        tx.commit();
        session.close();
        logger.info("Rule saved successfully, Rule Details="+objRule);
        return objRule;
    }

    public Rule updateRule(Rule objRule) {

        Session session = this.sessionFactory.openSession();
        Transaction tx = session.beginTransaction();
        session.update(objRule); // Replaced with updated rule
        tx.commit();
        session.close();
        logger.info("Rule updated successfully, Rule Details="+objRule);
        return objRule;
    }
```

After Refactoring

```java
    public Rule addRule(Rule objRule) {

    }

    public Rule updateRule(Rule objRule) {

    }
```

KCL : Expert-Driven Knowledge Acquisition

Refactoring Examples (Polymorphism - Dynamic Typing)

44

Utilized Polymorphism + Enum for Plug & play recognizers

As-is: Hard-coded recognition process
Every time new recognizer introduced
-> need to modify below code

```java
VideoActivityRecognizer vr = new VideoActivityRecognizer(userID);
GPSLocationRecognizer gr = new GPSLocationRecognizer(userID);
AudioEmotionRecognizer emr = new AudioEmotionRecognizer(ext, null,

vr.recognize(input, userID, timeStamp);
gr.recognize(input, userID, timeStamp)
emr.recognize(segmented, userID, timeStamp)
```

Too add recognizer, just
1. inherit the interface,
2. implement abstract functions, and
3. register the recognizer
No need to modify below main process

Current: Registration based, plug & play

```java
for(RecognizerType rec: RecognizerType.values()){ //for each registered recognizers
    recognizers.get(rec.getValue()).ConsumeData(it, data); //consume the current sensor data

    if(recognizers.get(rec.getValue()).isThreasholdReached()) //if threashold reached, call the recognizer
    {
        es.execute(recognizers.get(rec.getValue()).getRunnable(userID, curTime, unifier));
    }
}
```

ICL : Router

Refactoring Examples (Inline Queries → Stored Procedures)

45

Before Refactoring

```java
char count = 'n';

if (tok.length > 0) {
    for (int j = 0; j < tok.length; j++) {

        }
    }
}
```

After Refactoring

```java
SituationConditions objSituationConditionsMeasuring = new SituationConditions();
SituationConditions objSituationConditionsConstraints = new SituationConditions();
SituationLog objSituationLog = new SituationLog();
String situationLogID = null;

    while (rs.next()) {

        String stActivity = rs.getString("Activity");
        String stActivityOperator = rs.getString("ActivityOperator");
        String stActivityValue = rs.getString("Activityvalue");
        String stActivityDataType = rs.getString("ActivityDataType");

        String stMeasuringMetric = rs.getString("MeasuringMetric");
        String stMeasuringOperator = rs.getString("MeasuringOperator");
        String stMeasuringTarget = rs.getString("MeasuringTargetValue");
        int scltime = Integer.parseInt(stMeasuringTarget);
        scltime = scltime / 60;
        String stMeasuringTargetValue = scltime + "h";
```

DCL : Lifelog Monitoring

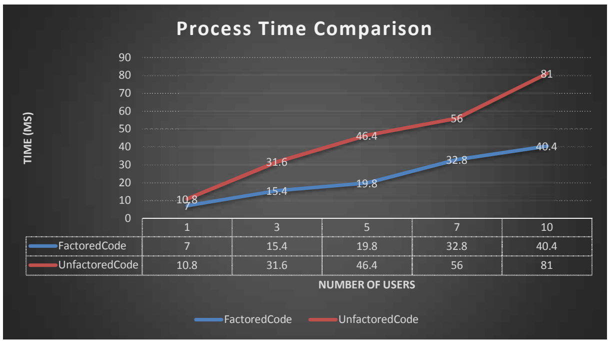
Refactoring Examples (Inline Queries → Stored Procedures)

46

After Refactoring

```java
SituationConditions objSituationConditionsMeasuring = new SituationConditions();
SituationConditions objSituationConditionsConstraints = new SituationConditions();
SituationLog objSituationLog = new SituationLog();
String situationLogID = null;

    while (rs.next()) {

        String stActivity = rs.getString("Activity");
        String stActivityOperator = rs.getString("ActivityOperator");
        String stActivityValue = rs.getString("Activityvalue");
        String stActivityDataType = rs.getString("ActivityDataType");

        String stMeasuringMetric = rs.getString("MeasuringMetric");
        String stMeasuringOperator = rs.getString("MeasuringOperator");
        String stMeasuringTarget = rs.getString("MeasuringTargetValue");
        int scltime = Integer.parseInt(stMeasuringTarget);
        scltime = scltime / 60;
        String stMeasuringTargetValue = scltime + "h";
```

Performance Improvement



- Multiple Time Execute
- No. of users group : 5
- Draw Average processing time
- **Average Improvement: 48.89%**

DCL : Lifelog Monitoring

Unit Testing Examples

47

```java
@Test
public void testCompareString2(){
  String v1 = "setting";
  String v2 = "standing";
  String op = "=";

}
```

```java
@Test
public void testCompareBoolean3(){
  boolean v1 = true;
  boolean v2 = false;
  String op = "!=";

}
```

```java
@Test
public void conflictResolverTest1() {
  String jsonFact = "{\"Current Activity\":\"LyingDown\",\"Activity Duration\":\"1h\",\"Acti

  List<FiredRule> firedRules = null;
```

```java
@Test
public void fireRuleTest2() {
  String jsonFact = "{\"Current Activity\":\"LyingDown\",\"Activity Duration\":\"1h\"}";


  String expected = "19";
```

SCL : Recommendation Builder

Mining Minds v3.0

Open source License

경희대학교
KYUNG HEE UNIVERSITY

Copyright VS Copyleft

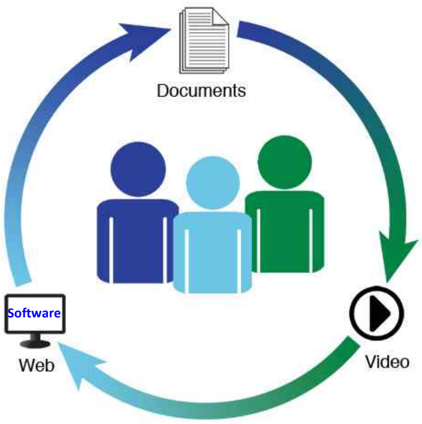
49

- Protect the authors software from unauthorized copying or selling of their work
- The authors permission is needed to take place these activities

- Provides a method for software to be modified, distributed back to community
- Gives rights to every one to use, modify, and redistribute the code but the distribution terms remain unchanged

Open Source

50



- The term "open source" refers to something that can be modified because its design is publicly accessible.
- Open source software is software whose source code is available for modification or enhancement by anyone.

open source™

Mining Minds as Open Source

51

Capabilities (Without Application Licensing Restriction)	GPL (Linux)	Dual-GPL (MySQL)	LGPL/MPL (OpenOffice, Firefox)	
1) Download	✔	✔	✔	✔
2) Evaluate	✔	✔	✔	✔
3) Deploy	✔	✔	✔	✔
4) Redistribute	🛇 [1]	✔ [3]	✔	✔
5) Modify	🛇 [2]	🛇 [2]	🛇 [2]	✔ [4]

1) Application needs to be licensed under GPL if redistributed with the GPL asset.
2) Library code modifications need to be licensed under the same license as the originating asset.
3) Usually requires a commercial license from the copyright holder.
4) Although much more permissive than an OSI license, some BSD based licenses, such as Apache V2, still have some copyleft materials.

- MM will open source code using standard licenses on OSI organization
- For open source of Mining Minds the Apache license is selected
- Each component of MM should develop as an separate API
- Each component should granted separate license

open source initiative

Mining Minds v3.0

Open source Build & Release

경희대학교 KYUNG HEE UNIVERSITY

Build & Release Background

53

Code Compilation

- Performed by the individual developer
- Code is compiled from high-level to Intermediate- and binary level
- Current high level languages use hybrid compilation
- Contributing S/W: Compiler

Build Creation

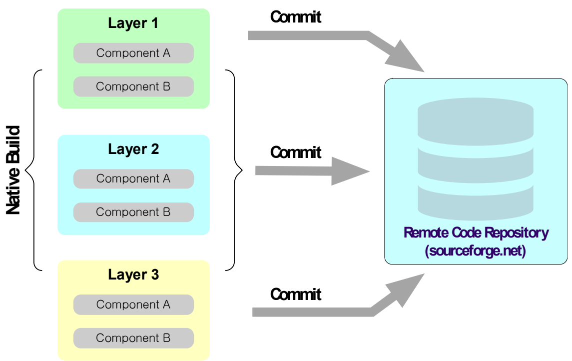
- Integrated compilation of several code components into deployable artifacts
- Mainly performed by writing scripts and shell files
- Contributing S/W: Ant, Maven, Cons etc.

Project Release

- Availability of integrated build as a product to consumers
- Public availability over web
- Open source release over open source communities
- Often complimented with API-class documentation for evolution of the product

Build Strategy I – Single step release

54



Pros

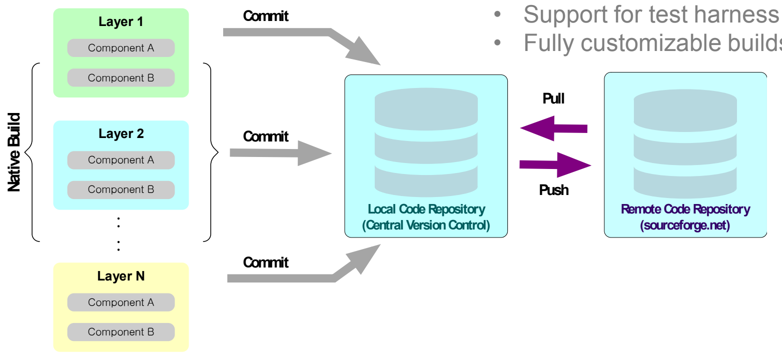
- Faster commit with most recent code availability
- Efficient strategy for smaller projects and smaller teams
- No dedicated build engineers required
- Support for prototyping and quick to release projects

Cons

- Always periodic compilation
- Community dependent integration
- No local compilation
- No support for test harness VM
- Lesser control on build customization and formalization

Build Strategy II – Dual step release

55



Pros

- Support for Large-scale heterogeneous codebase systems
- Supports continuous and periodic build generation
- Community Independent integration with a local copy of most latest code commit
- Support for test harness VM
- Fully customizable builds with formalization

Cons

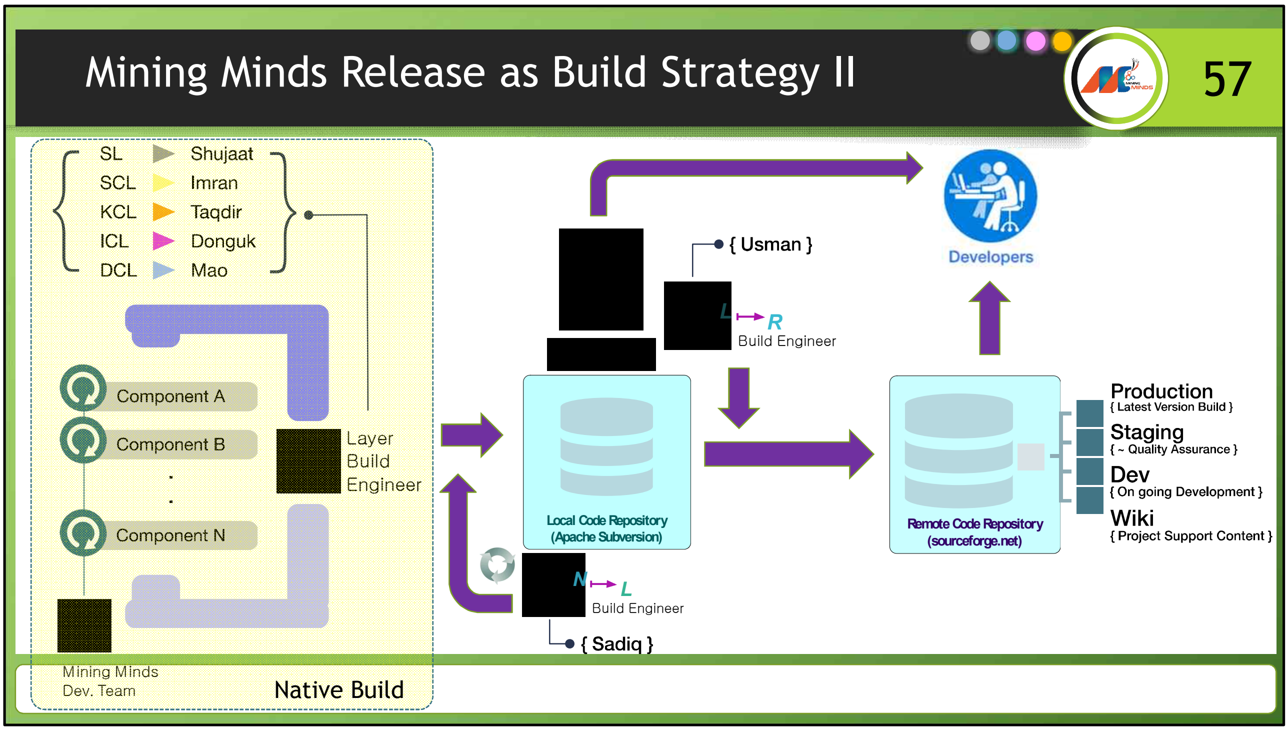
- Slower commit with most non-recent code availability
- Requires dedicated build engineers required

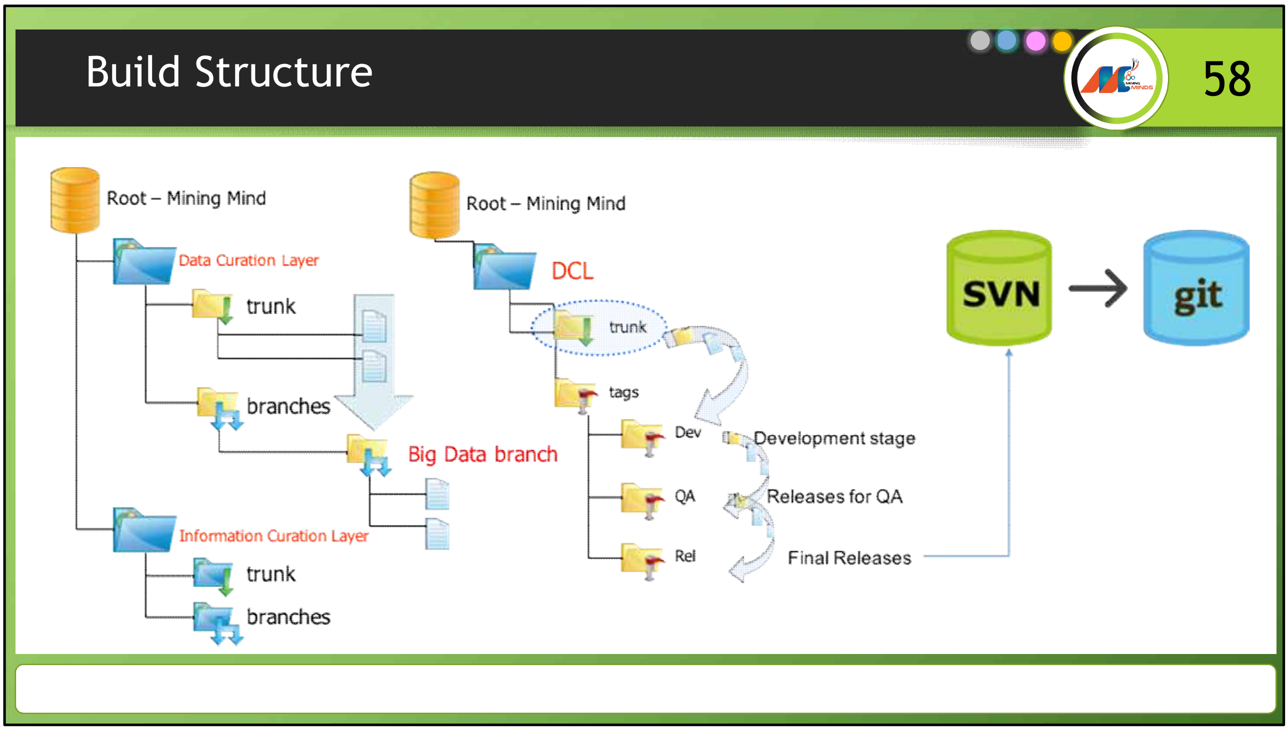
Mining Minds Build Requirements

56

- Mining Minds is arguably a medium to large scale project with not only research incentive but also Product realization
- Development of Mining Minds is a continuous activity requiring continuous local builds; however, due to twice-a-year release cycle it requires period build support
- Mining Minds is a large development team with control over code and build
- For Mining Minds demo purposes, team regularly generates scenario-based Demo Harnesses
- For the evolution of Mining Minds platform, customizable builds are required for deployment in other application areas

"Build Strategy II with Dual Step release satisfies the priority requirements of Mining Minds platform"


Mining Minds Release as Build Strategy II
57
SL Shujaat
SCL Imran
KCL Taqdir
ICL Donguk
DCL Mao
Component A
Component B
Component N
Layer Build Engineer
Mining Minds Dev. Team
Native Build
{ Usman }
L R
Build Engineer
Developers
Local Code Repository (Apache Subversion)
Remote Code Repository (sourceforge.net)
Production
{ Latest Version Build }
Staging
{ ~ Quality Assurance }
Dev
{ On going Development }
Wiki
{ Project Support Content }
N L
Build Engineer
{ Sadiq }


Build Structure
58
Root – Mining Mind
Data Curation Layer
trunk
branches
Big Data branch
Information Curation Layer
trunk
branches
Root – Mining Mind
DCL
trunk
tags
Dev
QA
Rel
Development stage
Releases for QA
Final Releases
SVN
git

Roles & Responsibilities

59

Team Members

- Generation of source code
- Compilation of source code
- Unit testing of source code
- Documentation of their respective components

Layer Build Engineers

- Assurance of reliable build
- Aggregation of compiled source code
- Implementation of local build strategy

N→L Build Engineer

- Design of local build strategy
- Execution of continuous integration process
- Administration of local build server

L→R Build Engineer

- Periodic mirroring of local build server to remote open source repository
- Administration of remote open source repository
- Creation and delivery of demo harness Virtual Machine

Team Leads

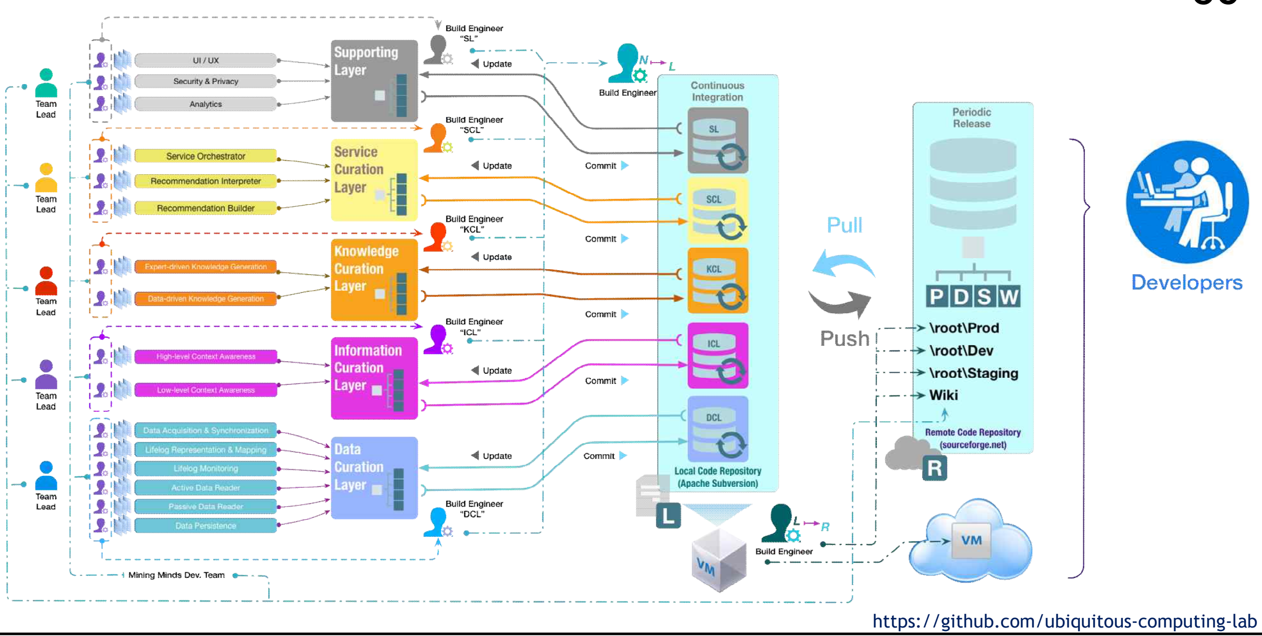
- Creation and compilation of Wiki content including API Specification
- Compilation of specification documents
- Management of Mining Minds Wiki

Project Lead

- Overall execution of project release
- Reliable and quality production build
- Delivery of the project as per timeline

Mining Minds V2.5 Open source Release Map

60



https://github.com/ubiquitous-computing-lab

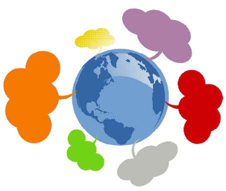
Mining Minds V3.0

Benefits of Open Source Release

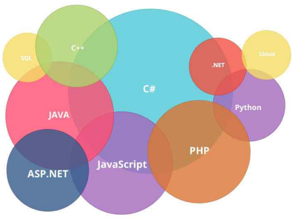
경희대학교
KYUNG HEE UNIVERSITY

Benefits of Open Source Release

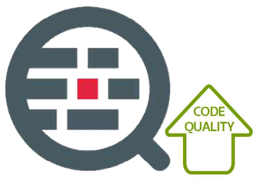
1. Implementation Benefits



Global Community
of developers
project continuation



Coding Skill set



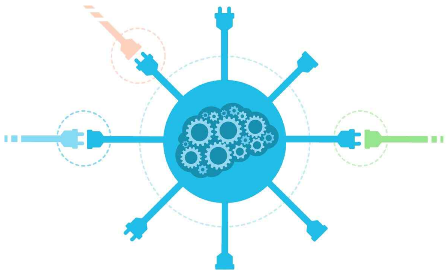
Continuous Feedback
(Peer reviews)

Software Engineering
Experience

Benefits of Open Source Release

63

2. Research Benefits



Code Availability
for Feedback
&
Evaluation

Code Re-use
for other
research
domains

Collaboration Initiatives

Benefits of Open Source Release

64

3. Community Benefits

Free High-quality
software distribution
among researchers
and developers

Support for Software
commercialization

Support for Startups
with high quality off
the shelf software
components

Section 3.2

Mining Minds Version 3.0 Layers Open Source Detail

Mining Minds

DCL

Open Source V3.0

Mining Minds

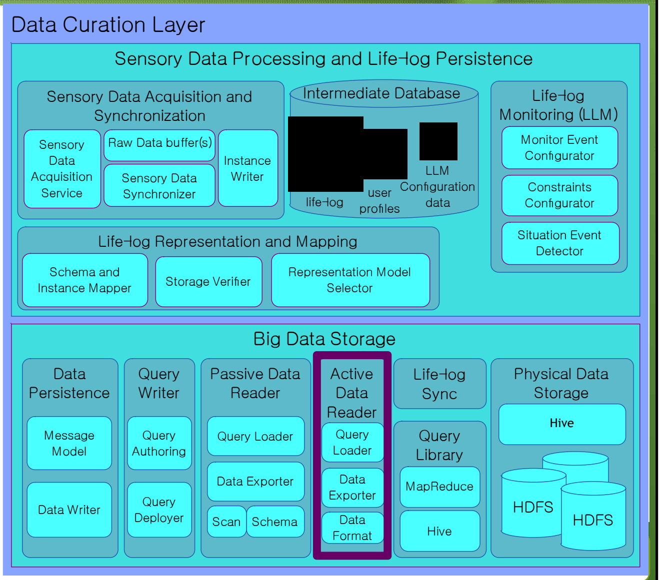
Active Data Reader

Open Source MMV3.0

Introduction – Open Source Detail (ADR)

3

- Active Data Reader (ADR):
  - For Analytics and Visualization, SL required real-time data read based on provided parameters.
  - Support for Visualization and Analytics

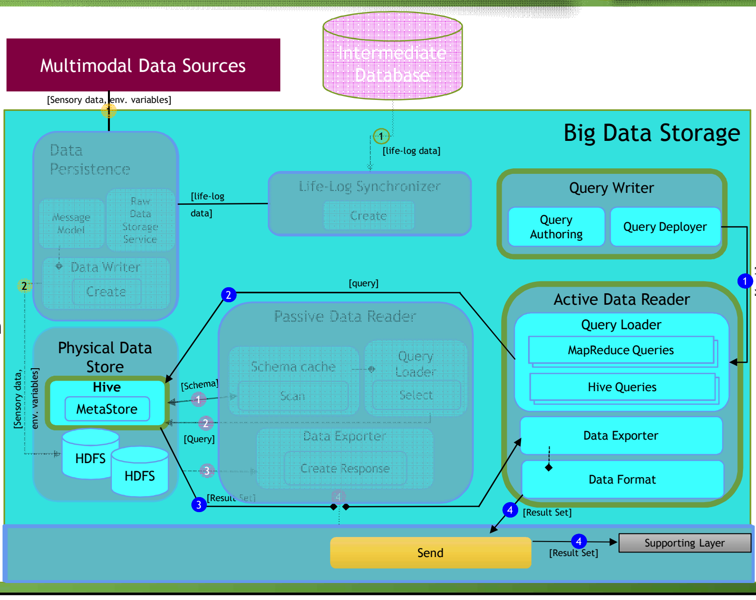


Active Data Reader (ADR) – Conceptual View

4

<u>Online Data Request for Data Visualization and Analytics</u>

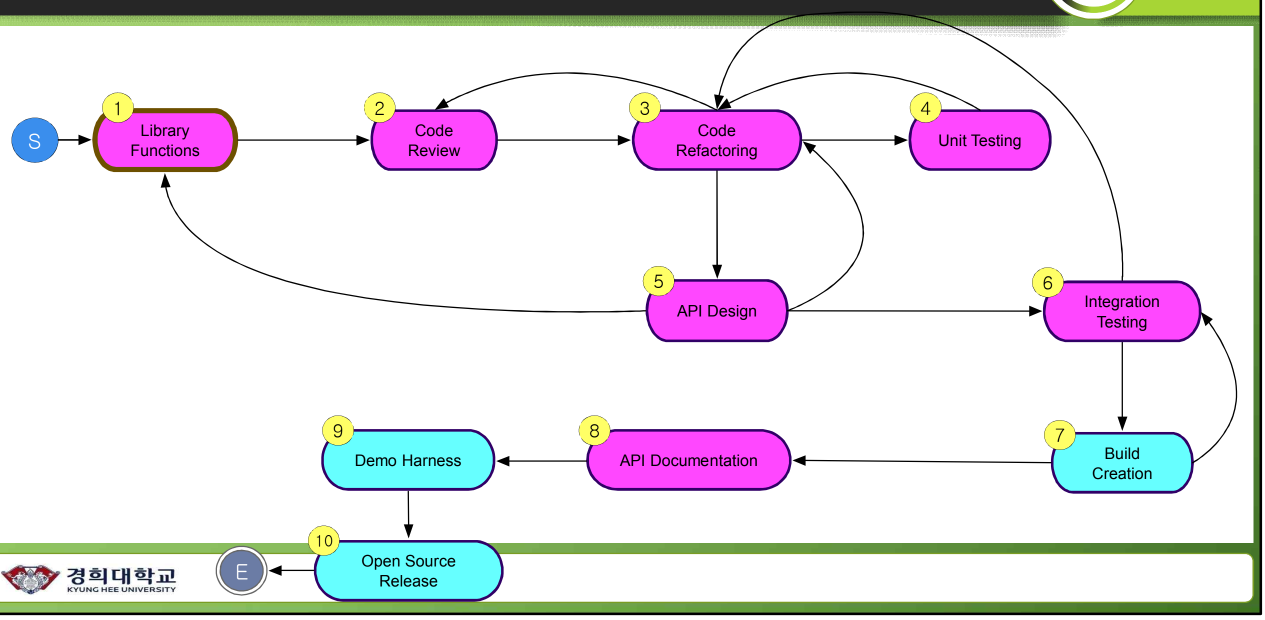
1. Data read request is generated by Visualization and Analytics Components of SL
2. Active data reader selects the particular query depending upon the query parameters
3. Selected Query is sent to Hive for execution
4. Required data is returned as a result set to Data exporter
5. Result-set is converted into data message per defined data format



Outline

- Open Source Roadmap
  - Library Function
  - Class Diagram
  - Refactoring
  - Unit Testing
  - Integration Testing
  - API Design
  - Build Creation
  - API Documentation
  - Demo Harness
  - Open Source Release

Open Source Release Process - Roadmap



Task :1 Open Source - Libraries Summary

7

Categorization of Libraries in our Component – Active Data Reader (ADR)

경희대학교
KYUNG HEE UNIVERSITY

Task :1 Open Source Libraries - Detail

8

Library	API Details	License
Hive Thrift Server API	Provide a JDBC based Connections to connect the Hive and Hadoop	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
Gson API	Used to convert Java Objects into their JSON representation and JSON to Java Objects	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
Spring MVC	Used as a Library to Provider a REST based services on the top of the Hadoop.	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
log4j	Track the logging to the Hive Connection	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
Apache Hive	Provides the SQL based operations on the top of the Hadoop	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
Hadoop Common Logging	Provide the logging support to the Apache Hive.	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
Junit API	JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks	Eclipse Public License 1.0 http://www.eclipse.org/legal/epl-v10.html
REST ASSURED	Provide the REST based end points testing.	Eclipse Public License - v 1.0

경희대학교
KYUNG HEE UNIVERSITY

Task :1 Open Source Libraries – Features

9

Comparison	Apache v 2.0	Eclipse Public License
Distribute: Must distribute licence with binary or source	√ ☺	√ ☺
Modify: There has to be A **notification** to changed files	√ ☺	√ ☺
Source Code: Any change must distributed in source form	√ ☺	√ ☺
Sublicense: All derivate work must be under the same license	√ ☺	√ ☺
Private use: Must show license when run from command line	√ ☺	√ ☺
Open Source: Non derivate works can have different license	√ ☺	√ ☺
Any Country: May exclude countries where there is A contradiction with **patent** in that country	√ ☺	√ ☺

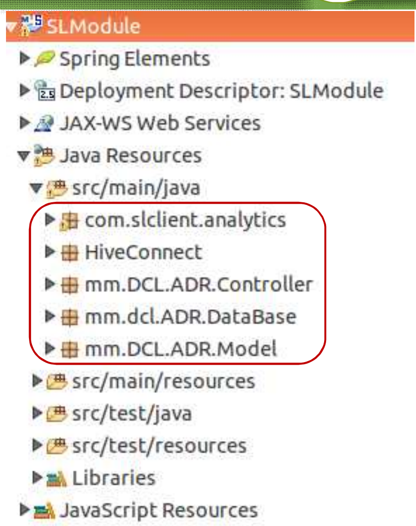
☺ **For the Eclipse / Apache Public License there is no restriction to use the Junit. But the restriction applies while using it commercial purposes.**

Task :1 Open Source - Library Function

10

- Library Function Identification
  - Packages Categorization
    - mm.DCL.ADR.Controller
      - Code to accept the request from the REST
    - mm.DCL.ADR.Model
      - Create a pattern with respect to the tables
    - mm.DCL.ADR.Database
      - Connect to the Apache Hive

Identification of Library Functions

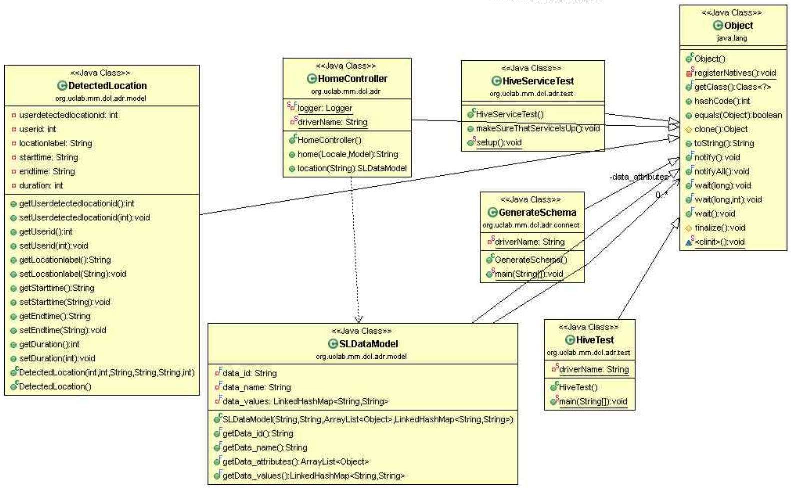


경희대학교
KYUNG HEE UNIVERSITY

Task :2 Code Review- Active Data Reader (ADR)

11

- **Class Diagram**
  - **Packages Categorization**
    - mm.DCL.ADR.Controller
      - Code to accept the request from the REST
    - mm.DCL.ADR.Model
      - Create a pattern with respect to the tables
    - mm.DCL.ADR.Database
      - Connect to the Apache Hive
  - **Some modification in Code pattern needed.**
    - Spring (Model – View – Controller)



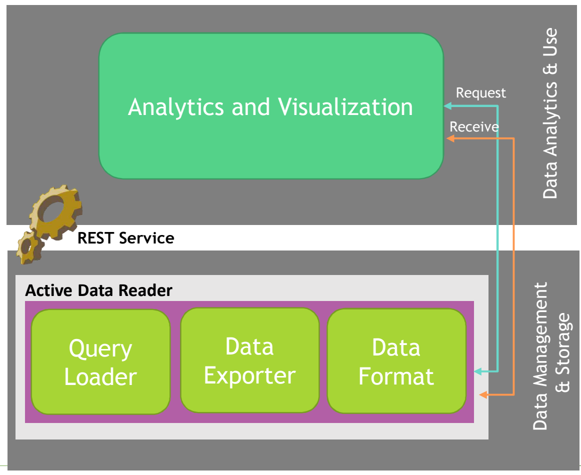
경희대학교
KYUNG HEE UNIVERSITY

Task :3 Open Source - Code Refactoring

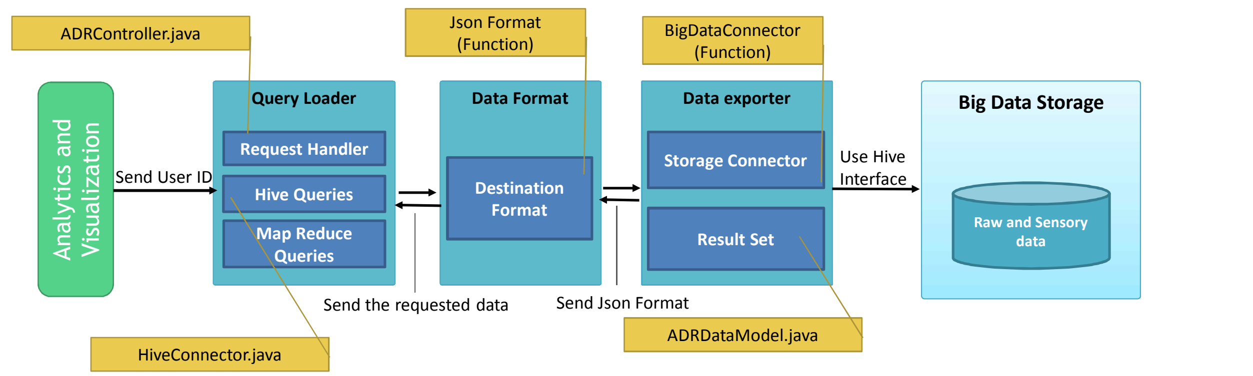
12

<u>Code Refactoring</u>

1. Implementation of Spring Model View Controller (MVC)
2. Code Comments
3. Rest Services Testing



Task :3 Active Data Reader - Architecture



Component: Active data Reader

GitHub Source Code Available at: https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

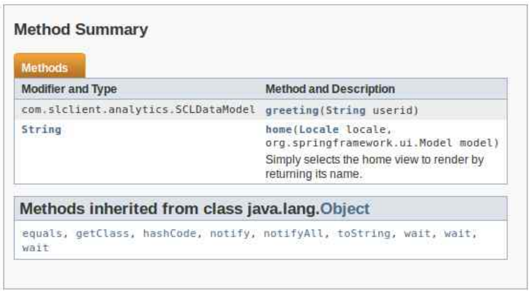
Task :3 Refactored Model

Package mm.DCL.ADR.Model

Create a pattern with respect to the Hive.

Class Summary

Class	Description
DataModel	This class will handles the request from the rest services
CreateTables	This class will create a tables and stored raw sensory data.


Method Summary

Methods

Modifier and Type	Method and Description
com.slclient.analytics.SCLDataModel	greeting(String userid)
String	home(Locale locale, org.springframework.ui.Model model) Simply selects the home view to render by returning its name.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Example Code

```
/**
 * Get the userid and return the data from Hive Table
 */
public DataModel(String data_id, String data_name, ArrayList<Object> data_attributes,
                LinkedHashMap<String, String> data_values

 String line = "incoming userid = " + userid;

            System.out.println(line);

            return new SCLDataModel(data_id, data_name, data_attributes,
data_
```

경희대학교 KYUNG HEE UNIVERSITY

Task :3 Refactored Model

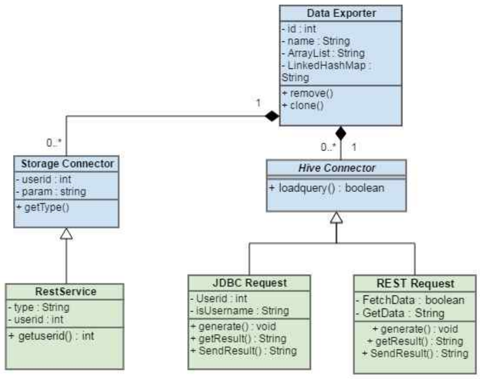
15

Package mm.DCL.ADR.Model

Create a pattern with respect to the Hive.

Class Summary

Class	Description
DataModel	This class will handles the request from the rest services
CreateTables	This class will create a tables and stored raw sensory data.



Example Code

```
/**
 * Get the userid and return the data from Hive Table
 */
public DataModel(String data_id, String data_name, ArrayList<Object> data_attributes,
                    LinkedHashMap<String, String> data_values

 String line = "incoming userid = " + userid;

                    System.out.println(line);

                    return new SCLDataModel(data_id, data_name, data_attributes,
data_
```

경희대학교
KYUNG HEE UNIVERSITY

Task :3 Refactored Controller

16

Package mm.DCL.ADR.Controller

Handles the request coming from the REST Services

Class Summary

Class	Description
ADRController	This class will handles the request from the rest services
Hive Connector	This class will connect to the Hive to query

Example Code

```
/**
 * Handles requests for data Visualization.
 */
@Controller

public class ADRController {

private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
private static String driverName = "org.apache.hive.jdbc.HiveDriver";
```

Method Summary

Methods

Modifier and Type	Method and Description
static void	main(java.lang.String[] args)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Summary

Methods

Modifier and Type	Method and Description
com.slclient.analytics.SCLDataModel	greeting(String userid)
String	home(Locale locale, org.springframework.ui.Model model) Simply selects the home view to render by returning its name.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

경희대학교
KYUNG HEE UNIVERSITY

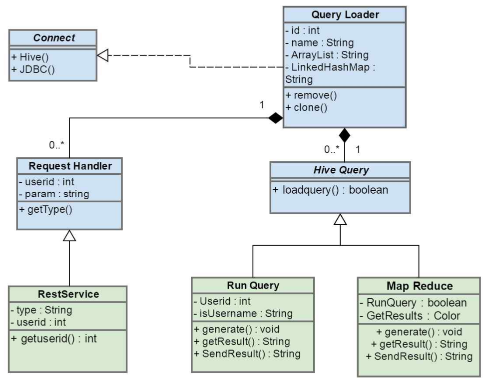
Task :3 Refactored Controller

17

Package mm.DCL.ADR.Controller

Handles the request coming from the REST Services

Class Summary	
Class	Description
ADRController	This class will handles the request from the rest services
Hive Connector	This class will connect to the Hive to query



Example Code

```
/**
 * Handles requests for data Visualization.
 */
@Controller

public class ADRController {

private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
private static String driverName = "org.apache.hive.jdbc.HiveDriver";
```

경희대학교 KYUNG HEE UNIVERSITY

Task :4 Unit Testing – Active Data Reader

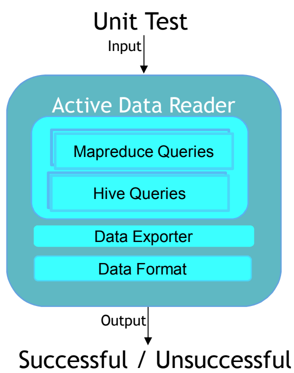
18

- **Unit Test**
  - Assured REST API Communication

- **Unit Test Goal:**
  - The aim of the test is to check the **REST communication** between the **Supporting Layer (SL)** and **Active Data Reader (ADR).**
  - **Example of the Unit Test Code:**

```
Getting and parsing a response body:
// Example with JsonPath String json =
get("/ActiveDataReader").asString() List<String> UserIds =
from(json).get("getuserId"); // Example with XmlPath String xml =
post("/Analytics").andReturn().body().asString() Node category =
from(json).get("getAllUserID[0]");
```

  - Result: Test Pass



REST-assured
Open Source Test Suite
https://github.com/rest-assured/rest-assured

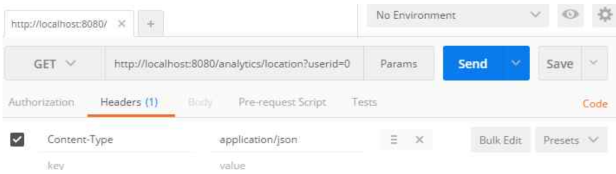
경희대학교 KYUNG HEE UNIVERSITY

Task :5 API Design

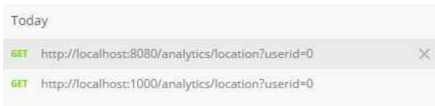
19

1. Api Address: http://localhost:8080/analytics/

2. Use HTTP Rest Client to send the Request towards Active Data Reader (ADR)



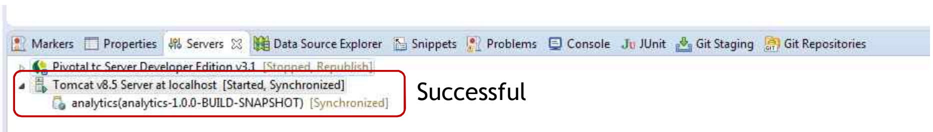
3. Successfully send the json response



경희대학교 KYUNG HEE UNIVERSITY

Task :6 Integration Testing

20



Successful

Successful

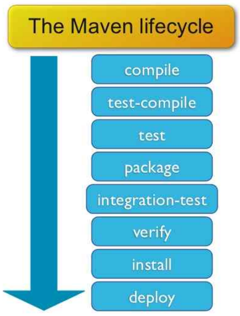
Upon completion of unit testing, the units or modules are integrated which gives raise to integration testing. The purpose of this, is to verify the functional, performance, and reliability between the modules that are integrated.

경희대학교 KYUNG HEE UNIVERSITY

Task :7 Active Data Reader - Build Creation

21

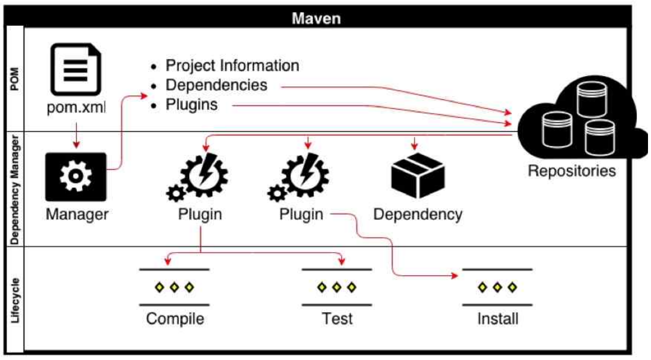
A Build Lifecycle is Made Up of Phases



Resources

Build Creation: https://maven.apache.org
Rest API: https://jax-rs-spec.java.net/
Version: https://subversion.apache.org/

- **Component:** Active Data Reader
  - Environment: Maven & Jax-Rs

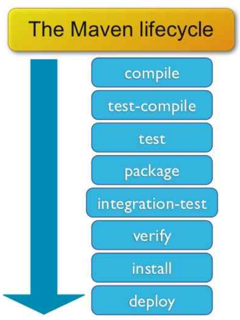


GitHub Source Code Available at: https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

Task :7 Active Data Reader - Build Creation

22

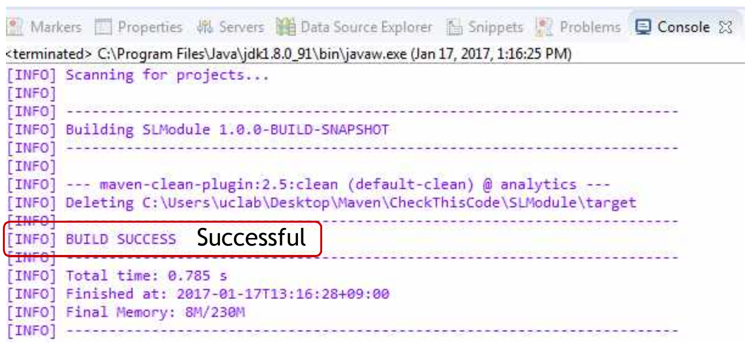
A Build Lifecycle is Made Up of Phases



Resources

Build Creation: https://maven.apache.org
Rest API: https://jax-rs-spec.java.net/
Version: https://subversion.apache.org/

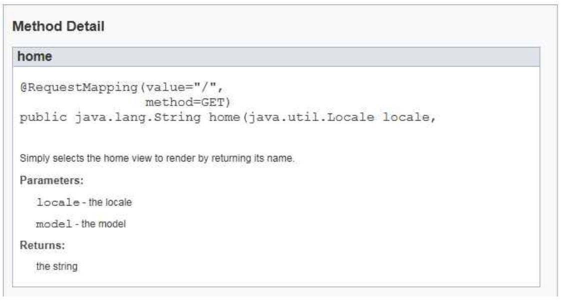
- **Component:** Active Data Reader
  - Environment: Maven & Jax-Rs

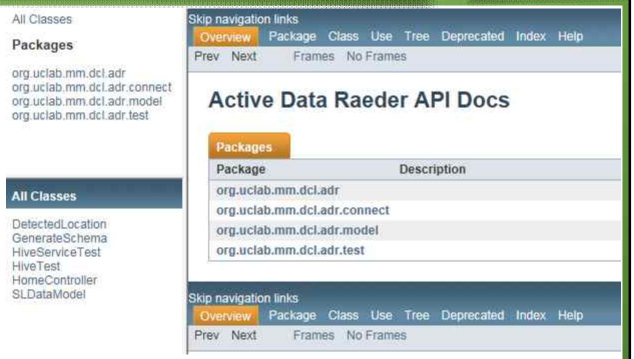


GitHub Source Code Available at: https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

Task :8 API Documentation

23





The Java Platform API Specification is defined by the documentation comments in the source code and any documents marked as specifications reachable from those comments.

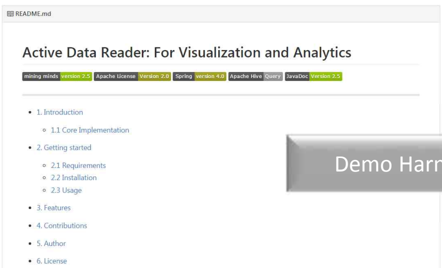
Javadoc Available at: https://ubiquitous-computing-lab.github.io/Mining-Minds/doc/dcl-doc/ative-data-reader/doc/index.html

경희대학교 KYUNG HEE UNIVERSITY

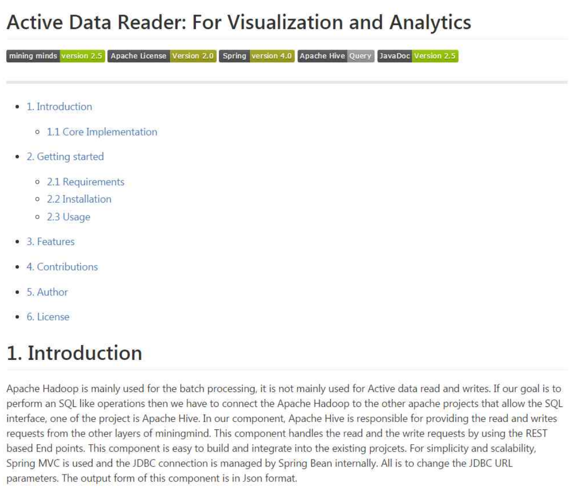
Task :9 Demo Harness

24

The installation and usage information available as demo harness on GitHub.



Demo Harness

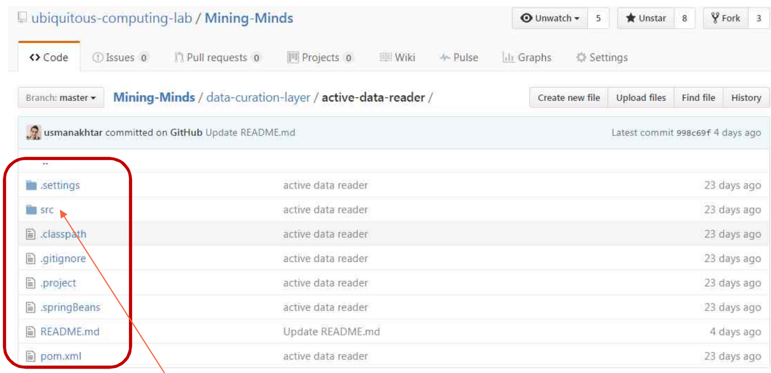


Demo Available at: https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

경희대학교 KYUNG HEE UNIVERSITY

Task :10 Open Source Release

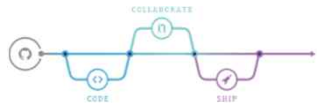
25



Source Code Folder Available at GitHub

The source code is released under the following License:

- The Apache License, Version 2.0 (the "License")
- The copy of License may be obtained at http://www.apache.org/licenses/LICENSE-2.0



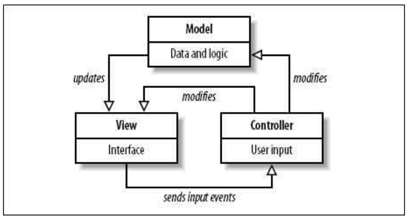
GitHub

경희대학교 KYUNG HEE UNIVERSITY

Code Refactoring - Summary

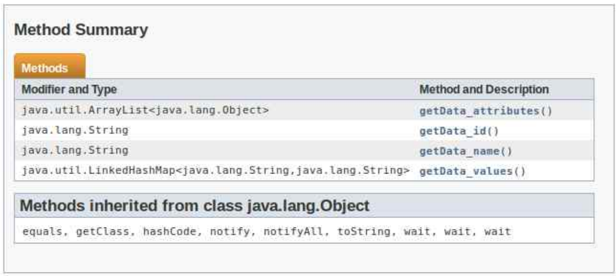
26

- **Code Refactoring Summary**
  - Packages Categorization
    - Code style: Follows Google Code Style https://google.github.io/styleguide/javaguide.html
    - Modified the methods and the classes
  - Design Pattern: Model View Controller (MVC)



  - Maven for Dependencies

Apache maven



Method Summary

Methods

Modifier and Type	Method and Description
java.util.ArrayList<java.lang.Object>	getData_attributes()
java.lang.String	getData_id()
java.lang.String	getData_name()
java.util.LinkedHashMap<java.lang.String,java.lang.String>	getData_values()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Active Data Reader API:

Java API for RESTful Web Services (**JAX-RS**)

JAX-RS

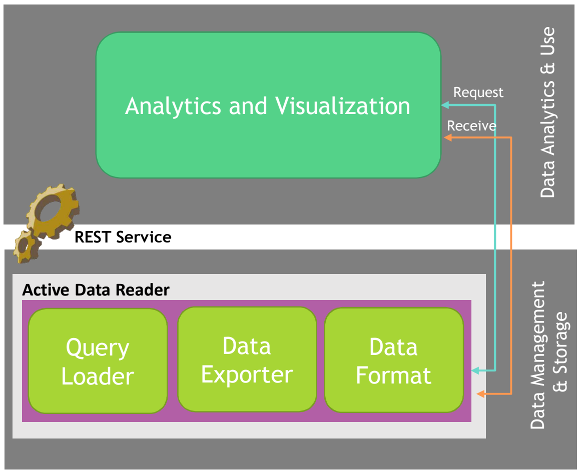
경희대학교 KYUNG HEE UNIVERSITY

Active Data Reader Implementation - Summary

27

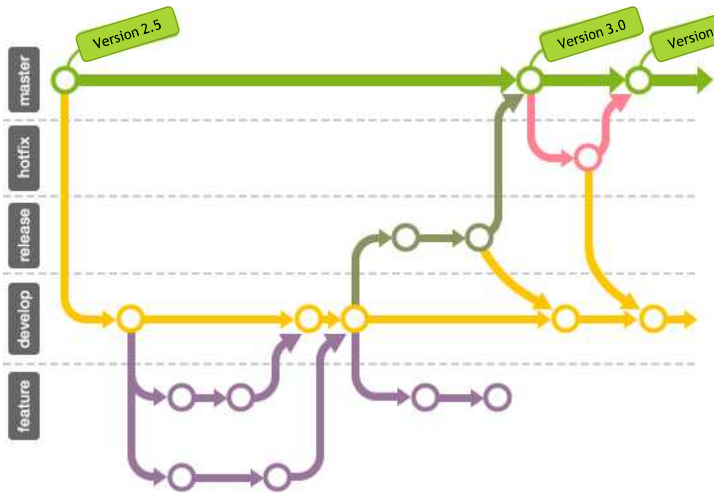
Provide Data for Visualization

1. Built for REST Services (request handle).
2. Support for Visualization and Analytics
3. Highly scalable and interactive



Active Data Reader – Version Control.

28



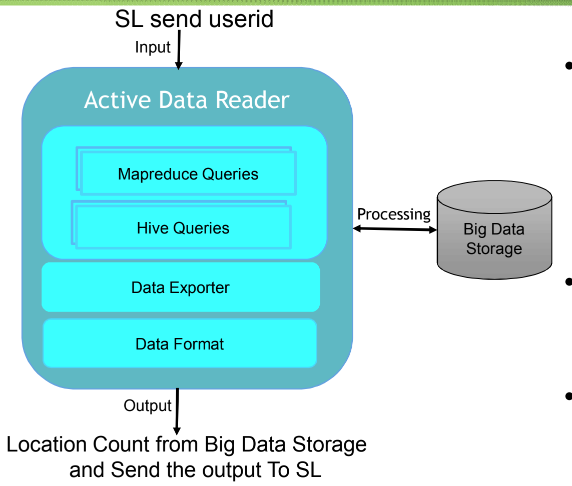
Master
Codebase residing in the master branch is considered to be production-ready.

Develop
Codebase residing in the develop branch is used in your team's day to day development.

Source Code Available at: https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

MMv3.0 Active Data Reader – Summary

29



- **Output**
  - ✓ User Location from the Big data (Give the SL on label counts after searching the records the Big Storage)
    - ✓ Using REST SERVICES
- **Input**
  - ✓ User ID to calculate the Location
- **Implementation Module**
  - ✓ Active Data Reader

경희대학교 KYUNG HEE UNIVERSITY

Mining Minds

Life-log Monitoring

Open Source V3.0

경희대학교 KYUNG HEE UNIVERSITY KITECH 한국생산기술연구원 POSTECH 포항공과대학교 UNIVERSITY of TASMANIA 녹십자헬스케어 TAPACROSS U2 system User & Usability system KOCSA 한국컨설팅서비스협회

Agenda

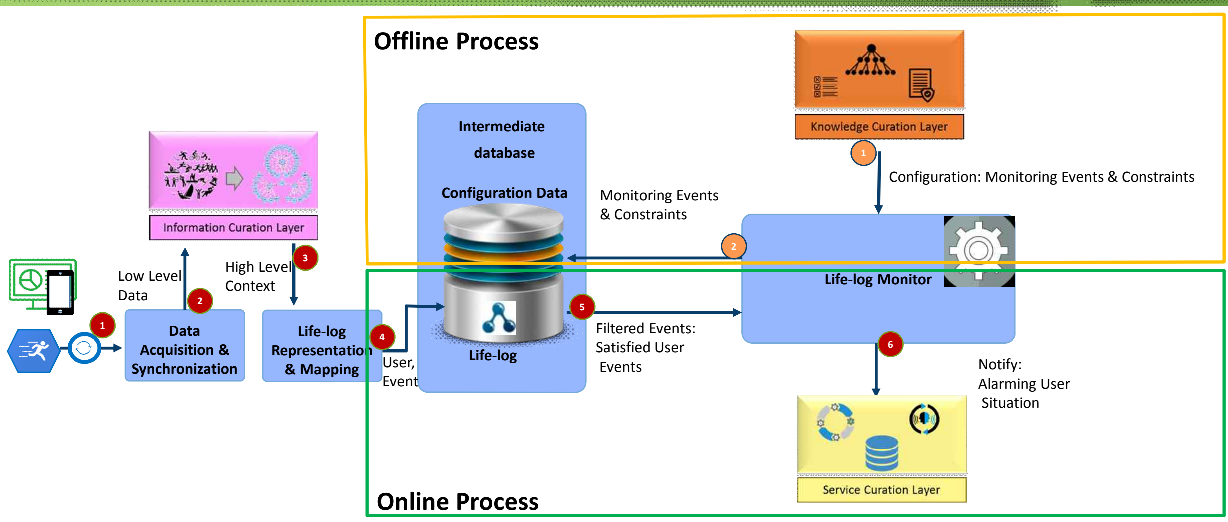
31

- Conceptual View
- Architecture
- Open Source
  - Library Function
  - Class Diagram
  - Refactoring
    - Code Refactoring
    - Inline Query Conversion
    - Log Maintenance
  - Refactoring Impact
  - Unit Testing

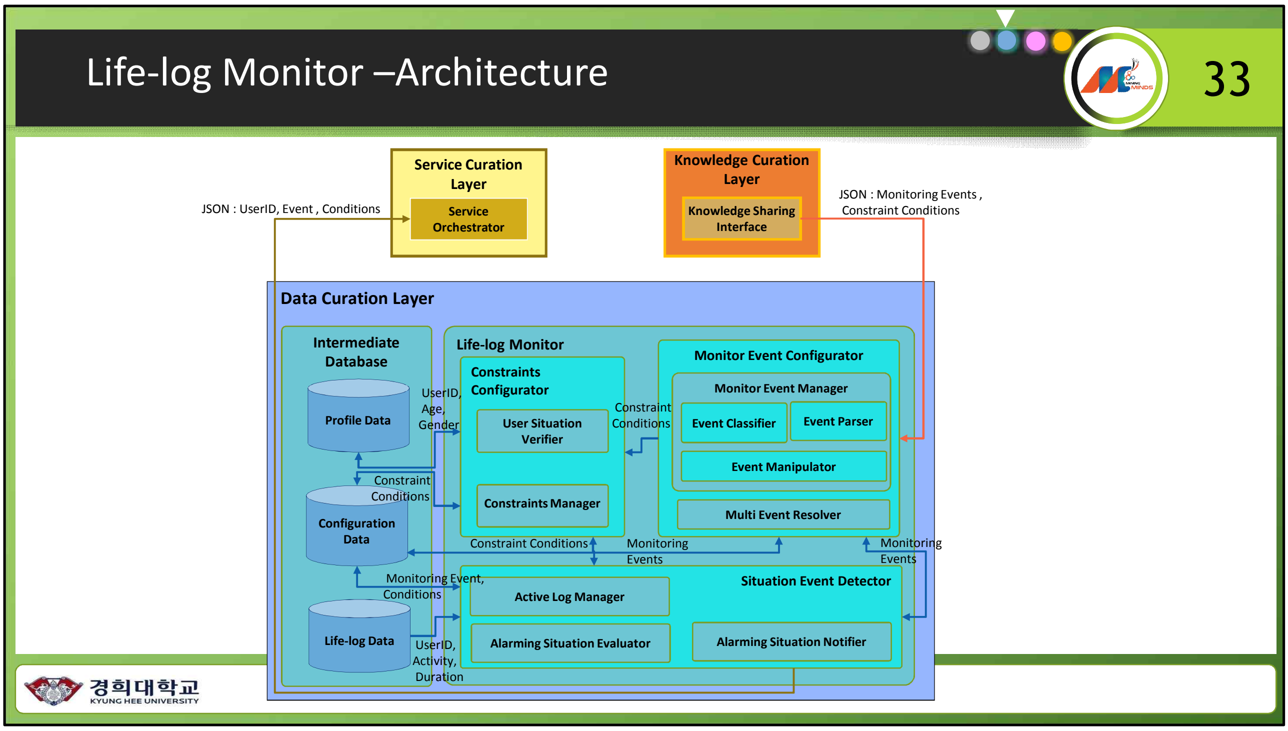
경희대학교 KYUNG HEE UNIVERSITY

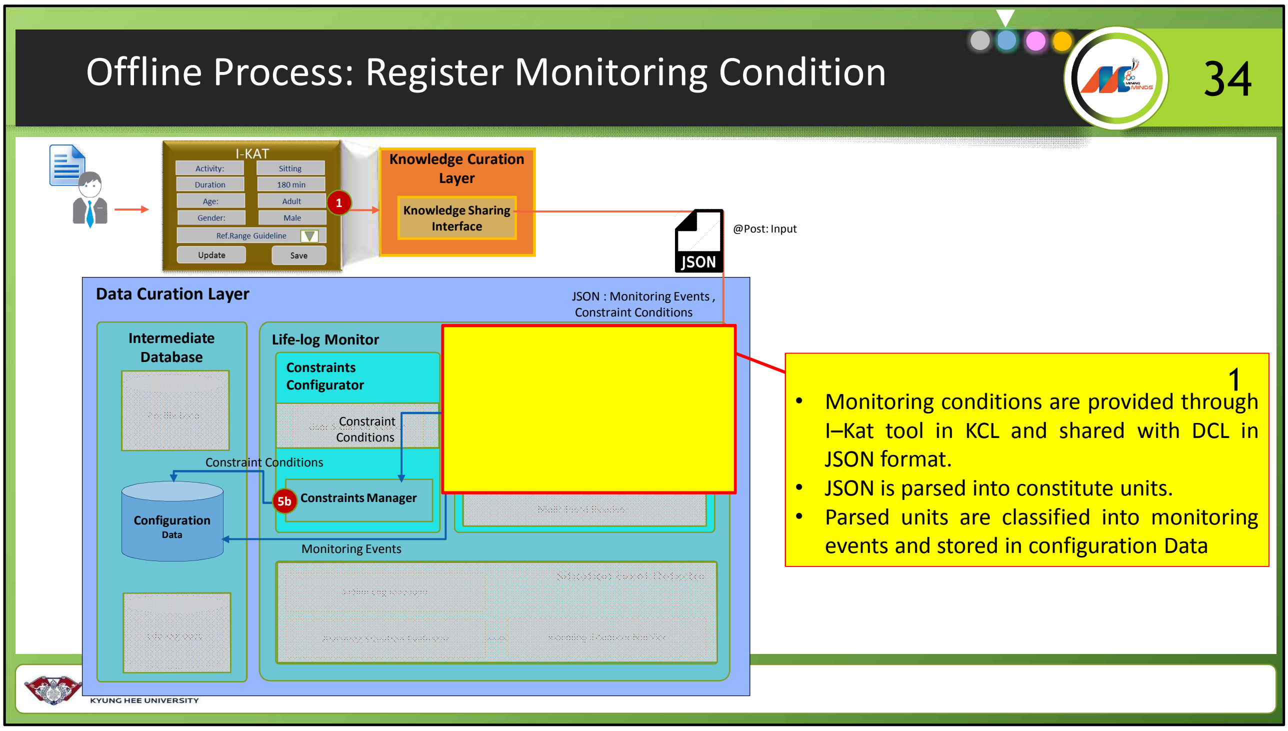
Life-log Monitoring – Conceptual View

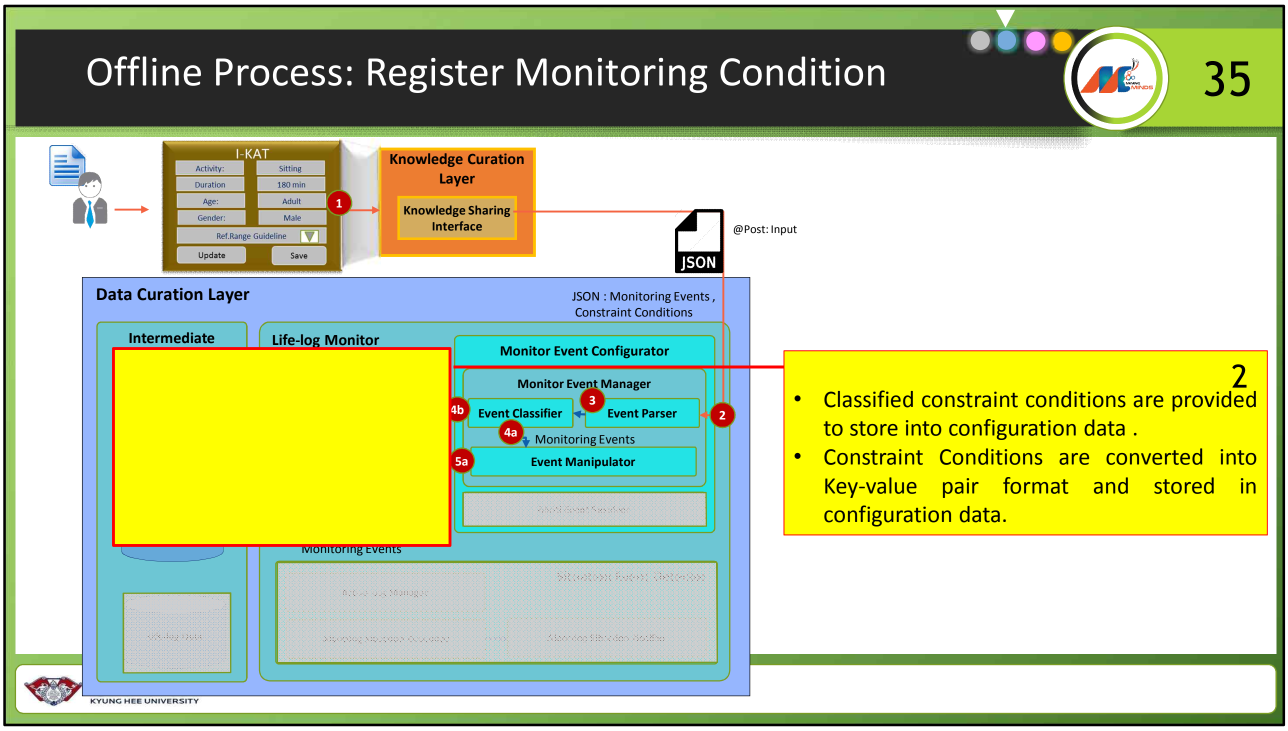
32

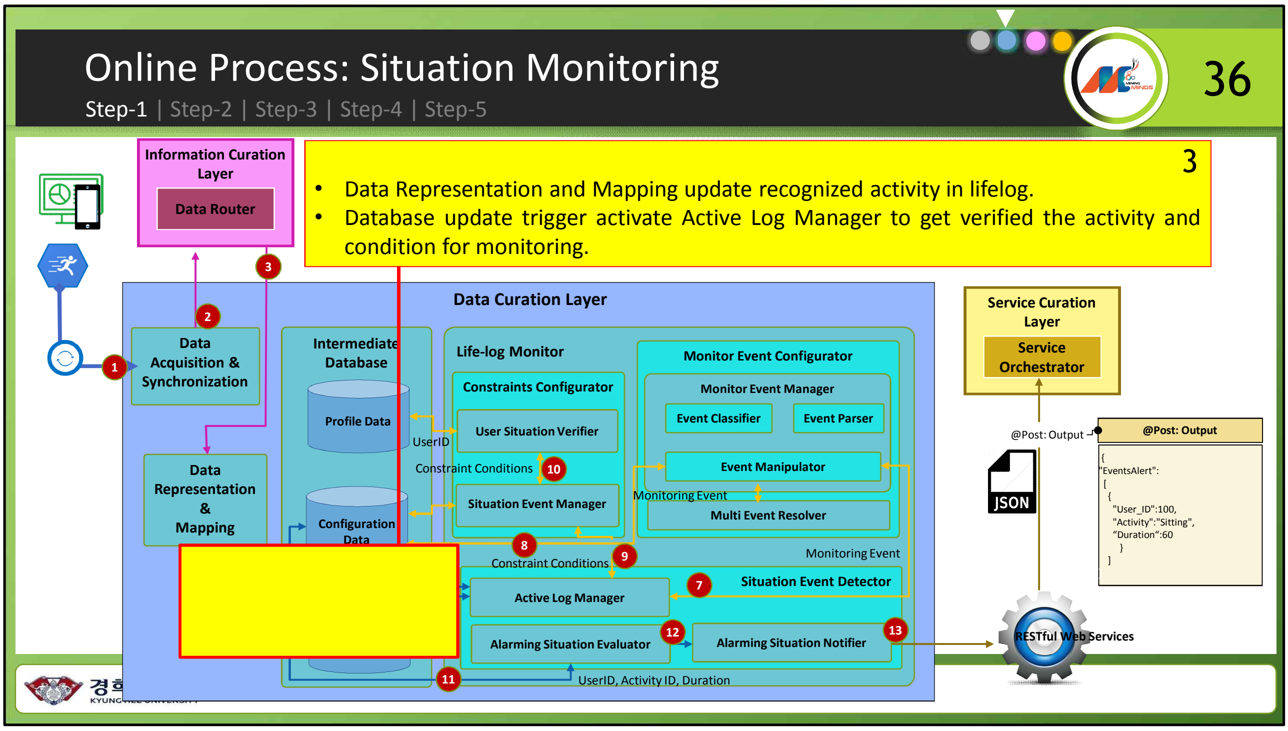


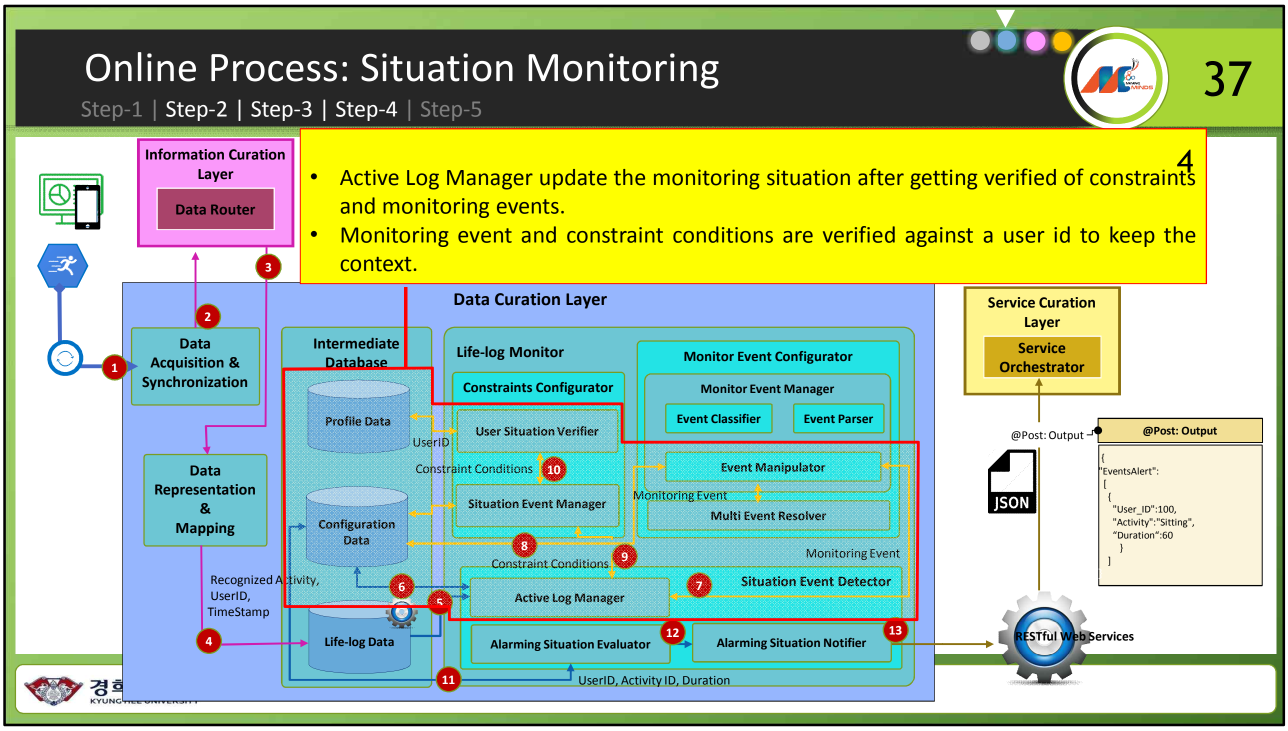
경희대학교 KYUNG HEE UNIVERSITY

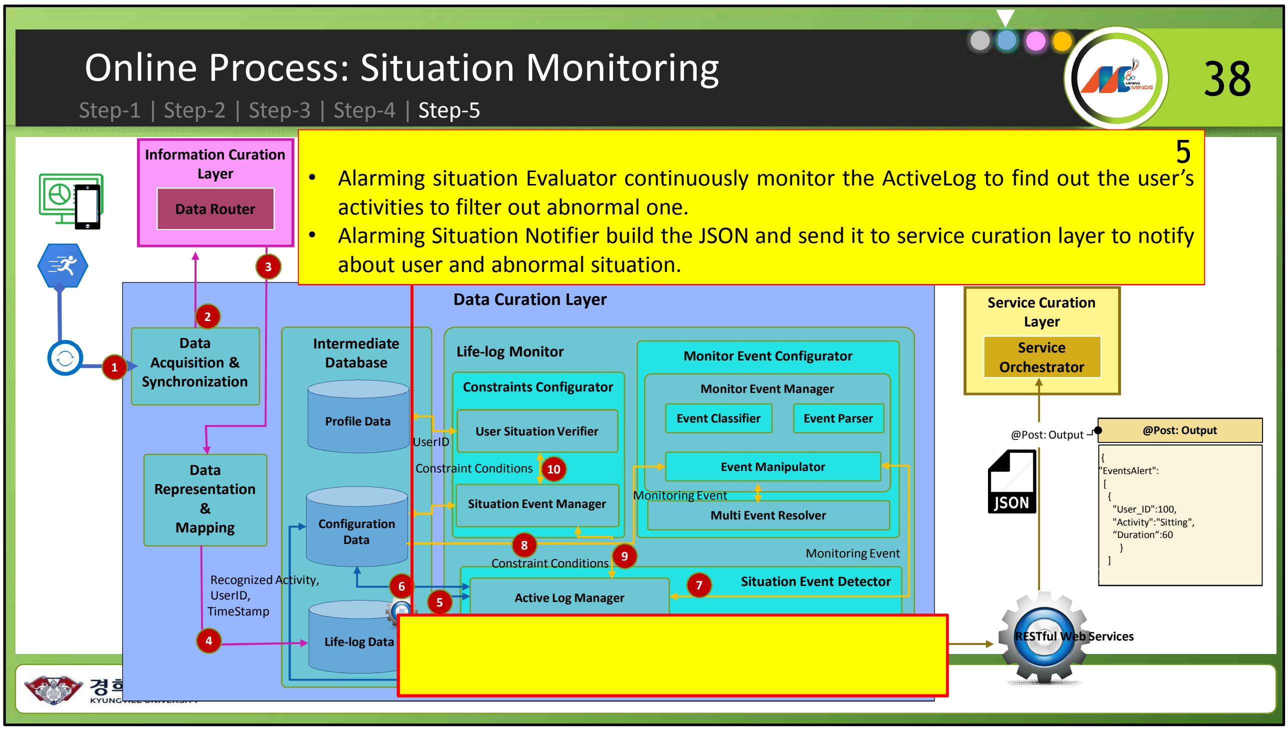

Life-log Monitor –Architecture
33
Service Curation Layer
Service Orchestrator
Knowledge Curation Layer
Knowledge Sharing Interface
JSON : UserID, Event , Conditions
JSON : Monitoring Events , Constraint Conditions
Data Curation Layer
Intermediate Database
Profile Data
Configuration Data
Life-log Data
UserID, Age, Gender
Constraint Conditions
Monitoring Event, Conditions
UserID, Activity, Duration
Life-log Monitor
Constraints Configurator
User Situation Verifier
Constraints Manager
Constraint Conditions
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Event Manipulator
Multi Event Resolver
Constraint Conditions
Monitoring Events
Monitoring Events
Situation Event Detector
Active Log Manager
Alarming Situation Evaluator
Alarming Situation Notifier
경희대학교
KYUNG HEE UNIVERSITY


Offline Process: Register Monitoring Condition
34
I-KAT
Activity: Sitting
Duration 180 min
Age: Adult
Gender: Male
Ref.Range Guideline
Update
Save
1
Knowledge Curation Layer
Knowledge Sharing Interface
JSON
@Post: Input
Data Curation Layer
JSON : Monitoring Events , Constraint Conditions
Intermediate Database
Configuration Data
Life-log Monitor
Constraints Configurator
Constraint Conditions
Constraint Conditions
5b
Constraints Manager
Monitoring Events
1
• Monitoring conditions are provided through I–Kat tool in KCL and shared with DCL in JSON format.
• JSON is parsed into constitute units.
• Parsed units are classified into monitoring events and stored in configuration Data
KYUNG HEE UNIVERSITY


Offline Process: Register Monitoring Condition
35
I-KAT
Activity: Sitting
Duration 180 min
Age: Adult
Gender: Male
Ref.Range Guideline
Update
Save
Knowledge Curation Layer
Knowledge Sharing Interface
JSON
@Post: Input
Data Curation Layer
JSON : Monitoring Events , Constraint Conditions
Intermediate
Life-log Monitor
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Monitoring Events
Event Manipulator
Monitoring Events
2
Classified constraint conditions are provided to store into configuration data .
Constraint Conditions are converted into Key-value pair format and stored in configuration data.
KYUNG HEE UNIVERSITY


Online Process: Situation Monitoring
Step-1 | Step-2 | Step-3 | Step-4 | Step-5
36
Information Curation Layer
Data Router
3
Data Representation and Mapping update recognized activity in lifelog.
Database update trigger activate Active Log Manager to get verified the activity and condition for monitoring.
Data Curation Layer
Data Acquisition & Synchronization
Data Representation & Mapping
Intermediate Database
Profile Data
Configuration Data
Life-log Monitor
Constraints Configurator
User Situation Verifier
Situation Event Manager
UserID
Constraint Conditions
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Event Manipulator
Monitoring Event
Multi Event Resolver
Constraint Conditions
Monitoring Event
Situation Event Detector
Active Log Manager
Alarming Situation Evaluator
Alarming Situation Notifier
UserID, Activity ID, Duration
Service Curation Layer
Service Orchestrator
@Post: Output
JSON
RESTful Web Services


Online Process: Situation Monitoring
Step-1 | Step-2 | Step-3 | Step-4 | Step-5
37
4
Active Log Manager update the monitoring situation after getting verified of constraints and monitoring events.
Monitoring event and constraint conditions are verified against a user id to keep the context.
Information Curation Layer
Data Router
Data Curation Layer
Data Acquisition & Synchronization
Data Representation & Mapping
Intermediate Database
Profile Data
Configuration Data
Life-log Data
Life-log Monitor
Constraints Configurator
User Situation Verifier
Situation Event Manager
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Event Manipulator
Multi Event Resolver
Situation Event Detector
Active Log Manager
Alarming Situation Evaluator
Alarming Situation Notifier
UserID
Constraint Conditions
Monitoring Event
Recognized Activity, UserID, TimeStamp
UserID, Activity ID, Duration
Service Curation Layer
Service Orchestrator
@Post: Output
JSON
RESTful Web Services


Online Process: Situation Monitoring
Step-1 | Step-2 | Step-3 | Step-4 | Step-5
38
5
Alarming situation Evaluator continuously monitor the ActiveLog to find out the user's activities to filter out abnormal one.
Alarming Situation Notifier build the JSON and send it to service curation layer to notify about user and abnormal situation.
Information Curation Layer
Data Router
Data Curation Layer
Data Acquisition & Synchronization
Data Representation & Mapping
Intermediate Database
Profile Data
Configuration Data
Life-log Data
Life-log Monitor
Constraints Configurator
User Situation Verifier
Situation Event Manager
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Event Manipulator
Multi Event Resolver
Situation Event Detector
Active Log Manager
UserID
Constraint Conditions
Monitoring Event
Recognized Activity, UserID, TimeStamp
Service Curation Layer
Service Orchestrator
@Post: Output
JSON
RESTful Web Services

Open Source Task 1- Library Function

39

MM V3.0_DCL_OpenSource_V0.2_20160815_Bilal

Identification of Library Functions

- LifelogMonitoring
  - Web Pages
  - Source Packages
  - Libraries
  - RESTful Web Services
    - ServiceCommunicator [testweb]
      - HTTP Methods
        - addSituationEvents() : String
        - getJson() : String
        - getLogViolationConditions() : List<org.uclab.dcl.llm.objectmodel.Violations>
        - getMaxLogViolationConditions() : List<org.uclab.dcl.llm.objectmodel.Violations>
        - getTotalViolationConditions() : List<org.uclab.dcl.llm.objectmodel.Violations>
        - getUserViolationConditions() : List<org.uclab.dcl.llm.objectmodel.Violations>
        - getViolationConditions() : List<org.uclab.dcl.llm.objectmodel.Violations>
        - putJson() : void
      - Subresource Locators
  - Configuration Files

RESTful-API

REST Web Service

Constructor Summary

Constructors

Constructor and Description
ServiceCommunicator() Creates a new instance of testweb

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.String	addSituationEvents(SituationEvents siuationEvents) Accept the json of the constraints and monitoring conditions required for monitoring and assessing the conditions.
java.lang.String	getJson() Retrieves representation of an instance of ServiceCommunicator
java.util.List<Violations>	getLogViolationConditions(java.lang.String lgid) Retrieves violation details from the log against a log information in term of ID.
java.util.List<Violations>	getMaxLogViolationConditions(java.lang.String maxlgid) Retrieves violation with sequence grater then the provided id from the log.
java.util.List<Violations>	getTotalViolationConditions() Retrieves total count of the violation from the log.
java.util.List<Violations>	getUserViolationConditions(java.lang.String uid) Retrieves total violation from the log against a user information in term of ID.
java.util.List<Violations>	getViolationConditions() Retrieves violation from the log.
void	putJson(java.lang.String content) PUT method for updating or creating an instance of ServiceCommunicator

경희대학교 KYUNG HEE UNIVERSITY

Code Review- Lifelog Monitoring

40

Packages: 04

All Classes

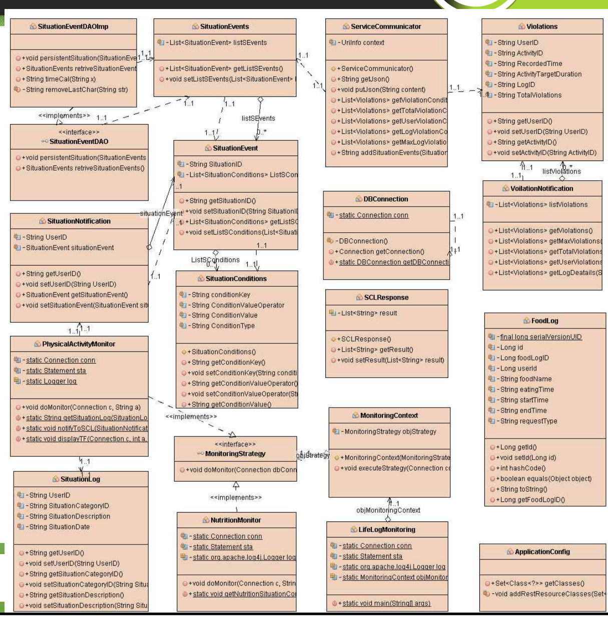
Packages

- org.uclab.dcl.llm.communication
- org.uclab.dcl.llm.dataaccess
- org.uclab.dcl.llm.Monitoring
- org.uclab.dcl.llm.objectmodel

Classes: 19

All Classes

- ApplicationConfig
- DBConnection
- FoodLog
- LifeLogMonitoring
- MonitoringContext
- *MonitoringStrategy*
- NutritionMonitor
- PhysicalActivityMonitor
- SCLResponse
- ServiceCommunicator
- SituationConditions
- SituationEvent
- *SituationEventDAO*
- SituationEventDAOImp
- SituationEvents
- SituationLog
- SituationNotification
- Violations
- VoilationNotification



경희대학교 KYUNG HEE UNIVERSITY

Open Source Task 6- Code Review

41

MM V3.0_DCL_OpenSource_V0.2_20160905_Bilal

- Code Issues
  - Vertical Expansion of code
  - Inline query
    - Change in data query required change in code
  - Absence of design Pattern
    - Code Management and alteration
  - Absence of Log Management
  - Absence of proper comments

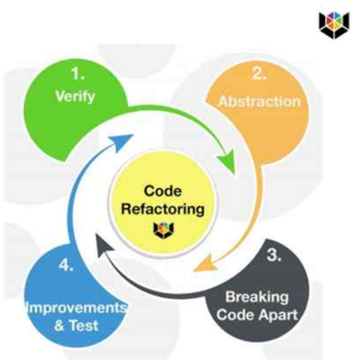
경희대학교
KYUNG HEE UNIVERSITY

Open Source Task 8- Code Refactoring

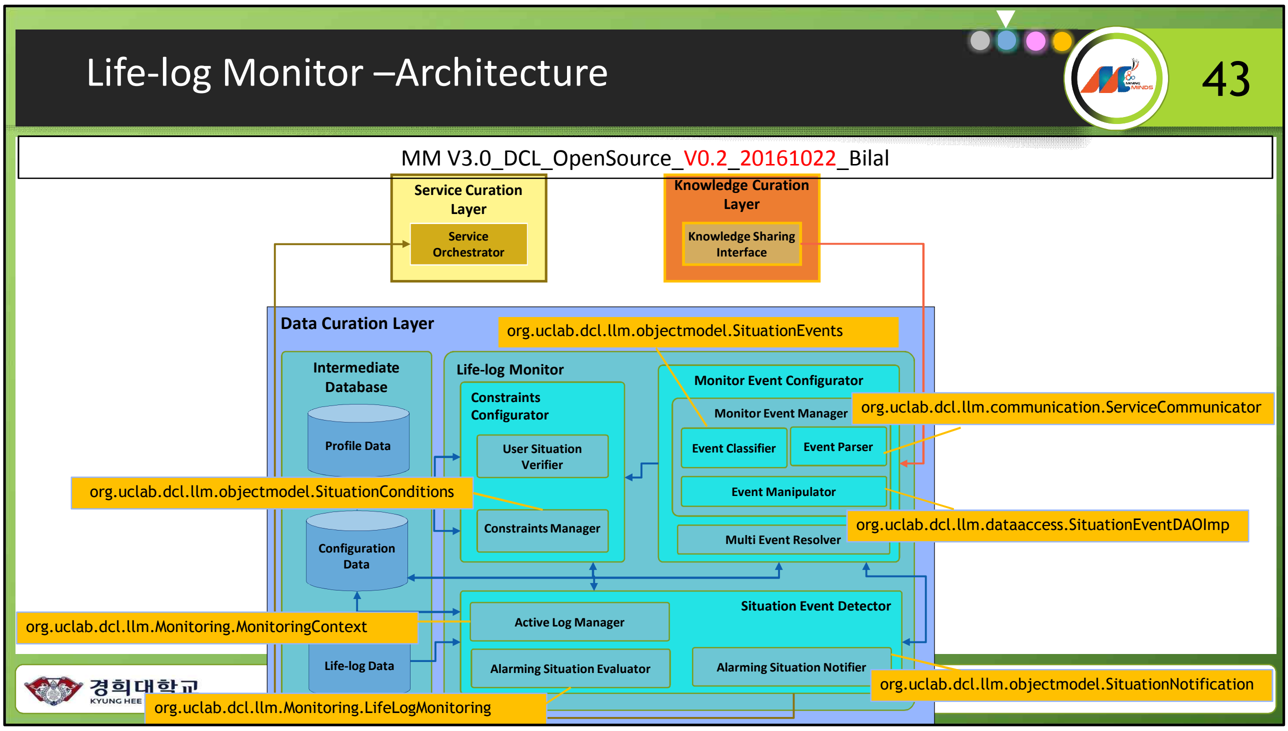
42

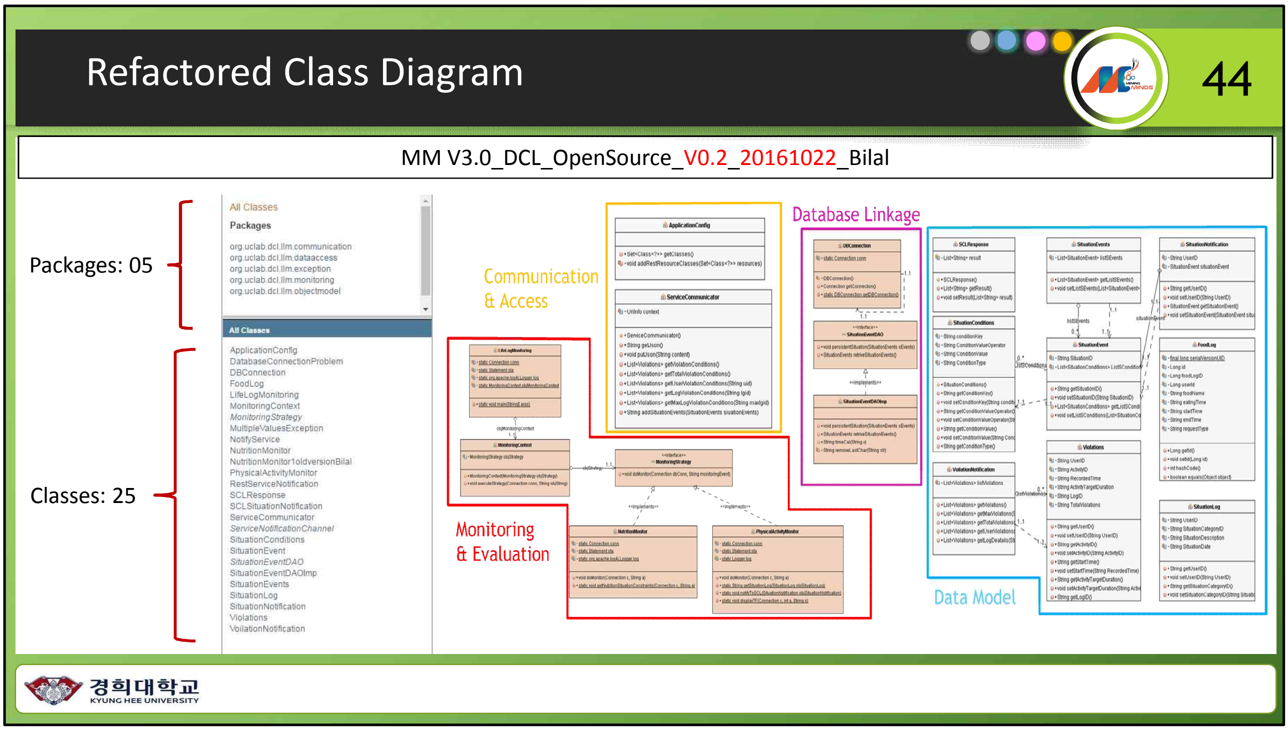
MM V3.0_DCL_OpenSource_V0.2_20161022_Bilal

- Code Refactoring
  - Implementation of Design Pattern
  - Inline query conversion to Database Procedure
  - Code comments
  - Log Maintenance
  - Evaluation



경희대학교
KYUNG HEE UNIVERSITY

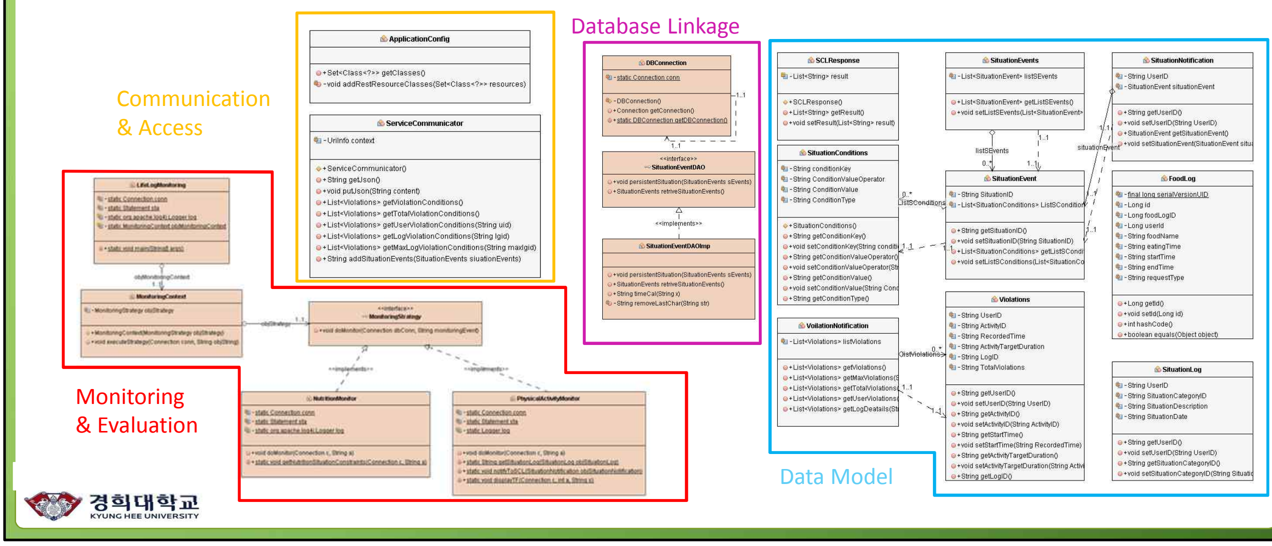

Life-log Monitor –Architecture
43
MM V3.0_DCL_OpenSource_V0.2_20161022_Bilal
Service Curation Layer
Service Orchestrator
Knowledge Curation Layer
Knowledge Sharing Interface
Data Curation Layer
org.uclab.dcl.llm.objectmodel.SituationEvents
Intermediate Database
Profile Data
Life-log Monitor
Constraints Configurator
User Situation Verifier
Constraints Manager
Monitor Event Configurator
Monitor Event Manager
org.uclab.dcl.llm.communication.ServiceCommunicator
Event Classifier
Event Parser
Event Manipulator
org.uclab.dcl.llm.objectmodel.SituationConditions
org.uclab.dcl.llm.dataaccess.SituationEventDAOImp
Configuration Data
Multi Event Resolver
Situation Event Detector
org.uclab.dcl.llm.Monitoring.MonitoringContext
Active Log Manager
Life-log Data
Alarming Situation Evaluator
Alarming Situation Notifier
org.uclab.dcl.llm.objectmodel.SituationNotification
경희대학교
KYUNG HEE
org.uclab.dcl.llm.Monitoring.LifeLogMonitoring


Refactored Class Diagram
44
MM V3.0_DCL_OpenSource_V0.2_20161022_Bilal
Packages: 05
All Classes
Packages
org.uclab.dcl.llm.communication
org.uclab.dcl.llm.dataaccess
org.uclab.dcl.llm.exception
org.uclab.dcl.llm.monitoring
org.uclab.dcl.llm.objectmodel
Classes: 25
All Classes
ApplicationConfig
DatabaseConnectionProblem
DBConnection
FoodLog
LifeLogMonitoring
MonitoringContext
MonitoringStrategy
MultipleValuesException
NotifyService
NutritionMonitor
NutritionMonitor1oldversionBilal
PhysicalActivityMonitor
RestServiceNotification
SCLResponse
SCLSituationNotification
ServiceCommunicator
ServiceNotificationChannel
SituationConditions
SituationEvent
SituationEventDAO
SituationEventDAOImp
SituationEvents
SituationLog
SituationNotification
Violations
VoilationNotification
Communication & Access
Database Linkage
Monitoring & Evaluation
Data Model
경희대학교
KYUNG HEE UNIVERSITY

Refactored Class Diagram

45

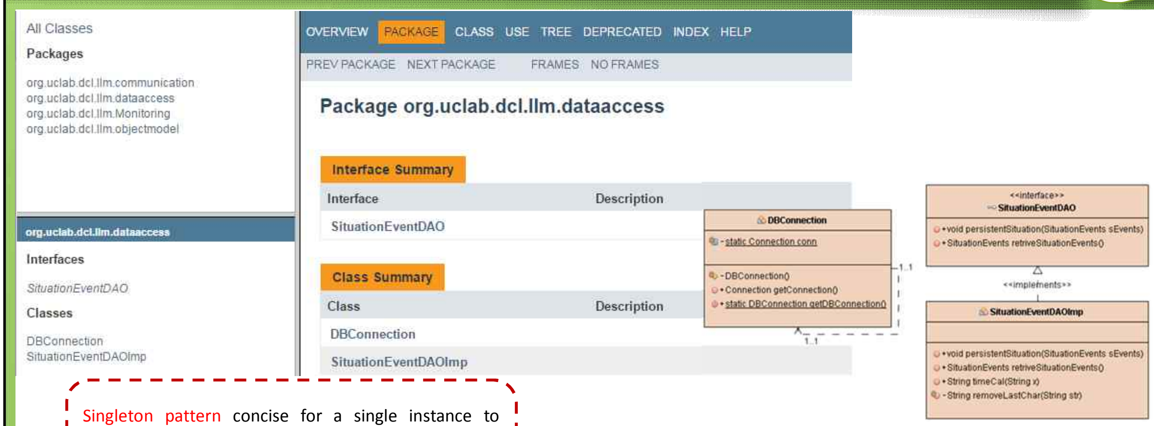
MM V3.0_DCL_OpenSource_V0.2_20161022_Bilal



경희대학교 KYUNG HEE UNIVERSITY

Singleton Pattern – Database Access

46

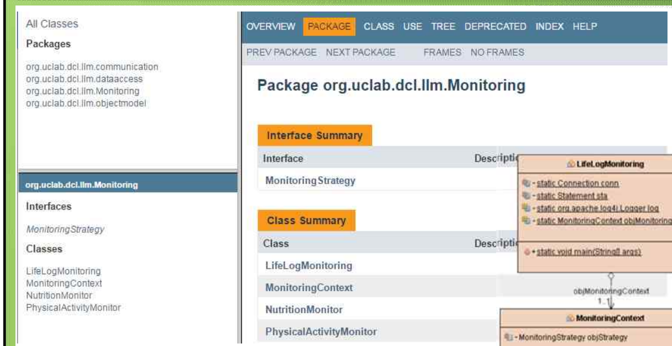


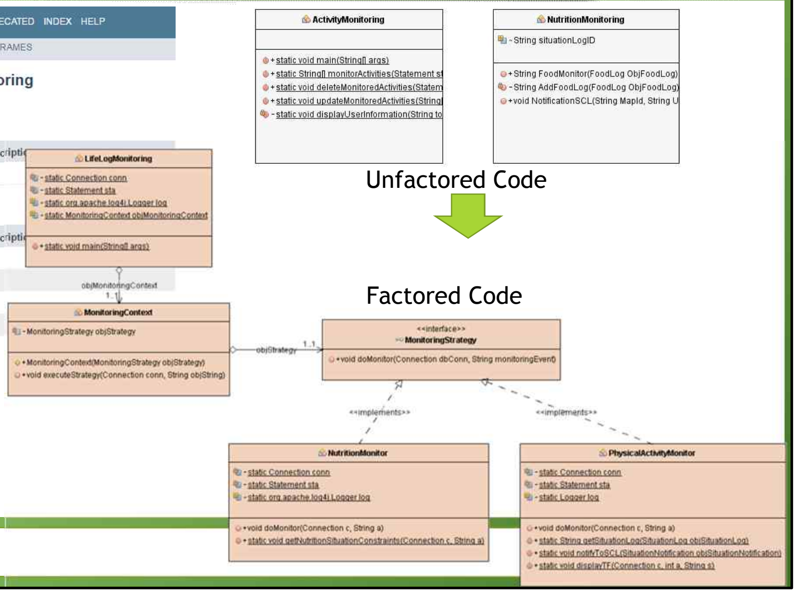
Singleton pattern concise for a single instance to access the database. Avoid the burden of making multiple connection with database and handling them properly for efficient use.

경희대학교 KYUNG HEE UNIVERSITY

Strategy Pattern – Monitoring & Evaluation

47



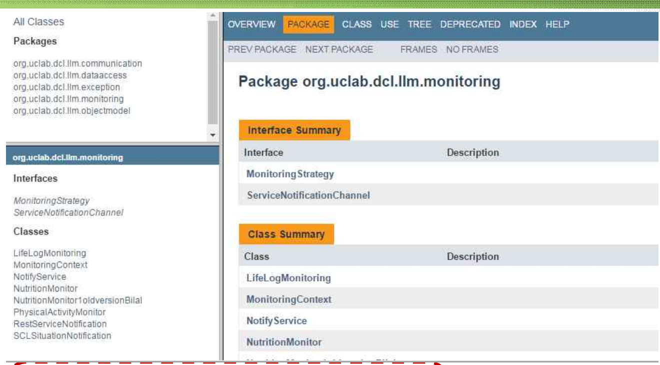


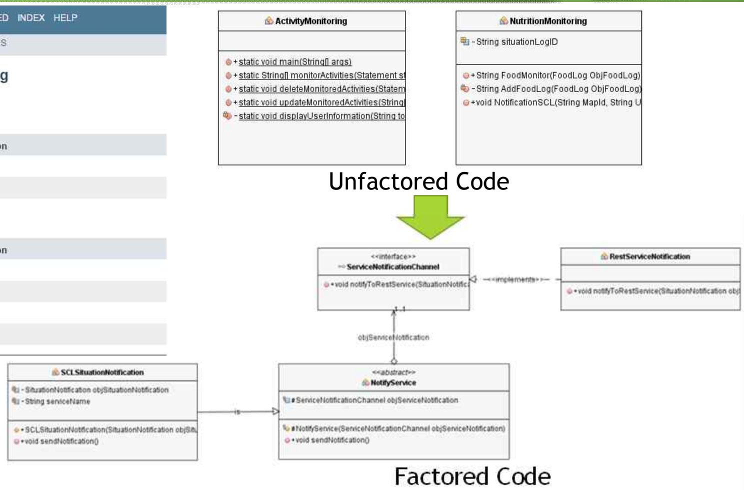
Strategy pattern gives flexibility to change monitoring algorithm at run time. The behavior pattern to represent various monitoring strategies that changes the executing algorithm of the context object.

경희대학교 KYUNG HEE UNIVERSITY

Bridge Pattern – Monitoring Notification

48



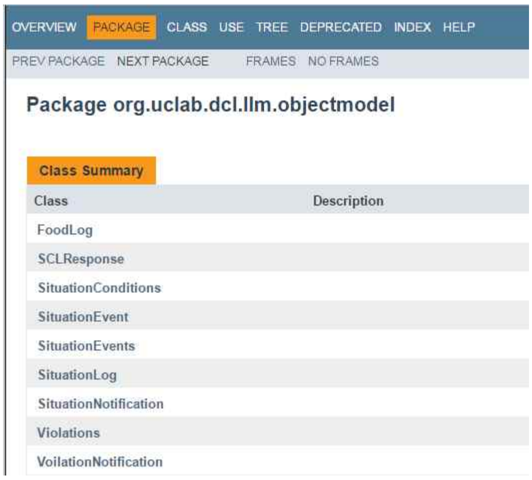


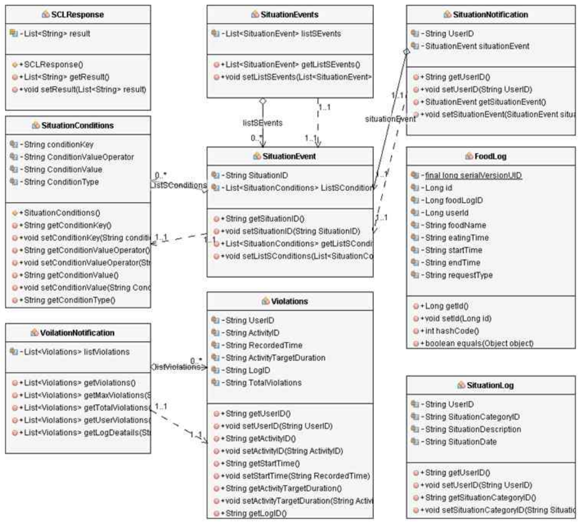
Bridge pattern gives flexibility to change communication way of notification according to the requirements. It decouples the abstraction of communication from its real technique implementation so that the both can vary independently.

경희대학교 KYUNG HEE UNIVERSITY

Object Model

49





경희대학교 KYUNG HEE UNIVERSITY

Code Refactoring- Inline Query to Stored Procedure

50

MM V3.0_DCL_OpenSource_V0.2_20161022_Bilal

- **Conversion of Inline queries to stored procedures**

Unfactored Code

Factored Code

경희대학교 KYUNG HEE UNIVERSITY

Code Refactoring- Maintaining Log

51

MM V3.0_DCL_OpenSource_V0.2_20161022_Bilal

- **Apache Log4j for Errors and Debugging**
  - Convenient and flexible logging mechanism as well as fast performance.

Unfactored Code

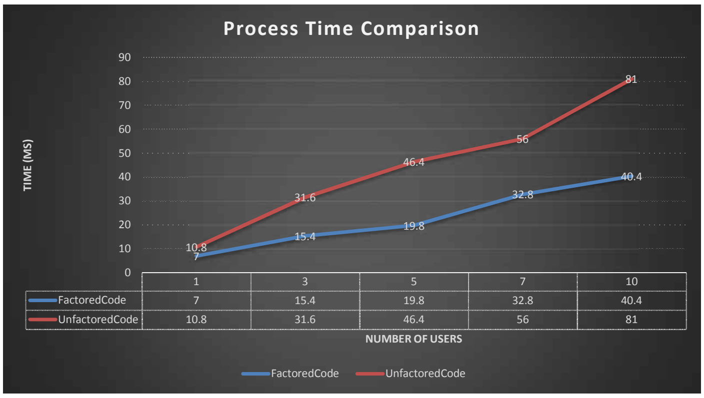
Factored Code

LOG4J TM

Impact of Code Refactoring

52

MM V3.0_DCL_OpenSource_V0.2_20161022_Bilal



	1	3	5	7	10
FactoredCode	7	15.4	19.8	32.8	40.4
UnfactoredCode	10.8	31.6	46.4	56	81

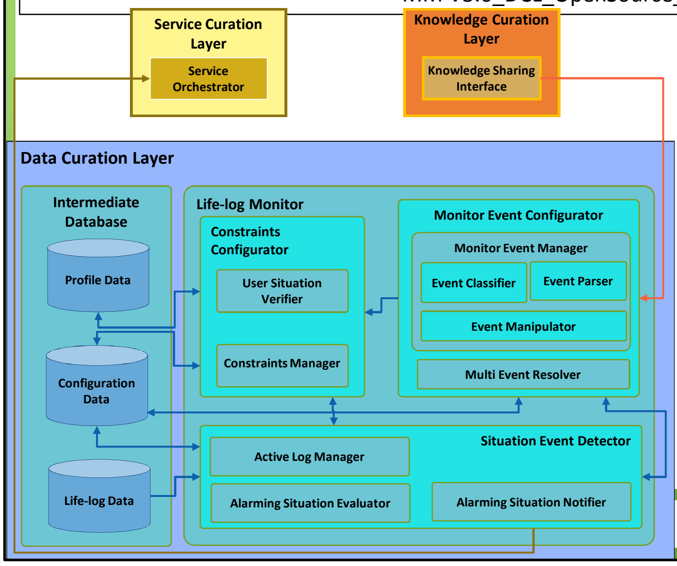
- Multiple Time Execute: 500 times
- No. of users group : 5
- Draw Average processing time
- Comparison : Factored vs Unfactored
- **Average Improvement: 48.89%**

경희대학교 KYUNG HEE UNIVERSITY

MMv2.5 Lifelog Monitoring

53

MM V3.0_DCL_OpenSource_V0.2_20161022_Bilal



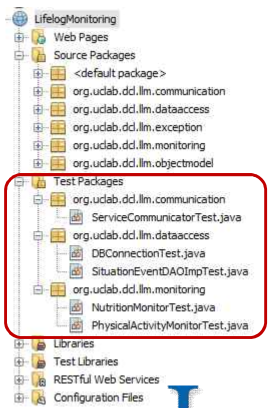
- **Output**
  - ✓ Physical Activity Monitoring
  - ✓ Nutrition Monitoring
  - ✓ Combination of Physical and Nutrition Monitoring
    - ✓ Sedentary behavior + Nutrition
- **Input**
  - ✓ Recognized Activities
  - ✓ Recognized Food
- **Implementation Module**
  - ✓ Monitoring Situation parsing
  - ✓ Alarming Situation Evaluator
  - ✓ Alarming Situation Notifier

Open Source-Current Tasks: Unit Testing

54

MM V3.0_DCL_OpenSource_V0.2_20161104_Bilal

- Unit Testing
- JUnit
- Testing for
  - Public Functions
    - Data Range Values {Test Classes}
  - Exceptions



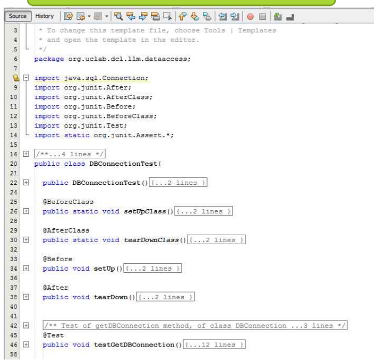
JUnit
Testing Framework

Connection for Data Manipulation

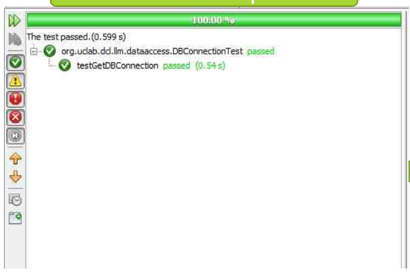
55

MM V3.0_DCL_OpenSource_V0.2_20161104_Bilal

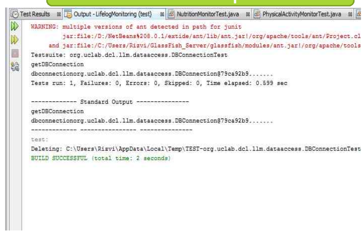
Junit Test Class



Junit Test Report



Junit Output



Database Connection:

- Test is performed for the connection development
- To assure that existed connection should be used

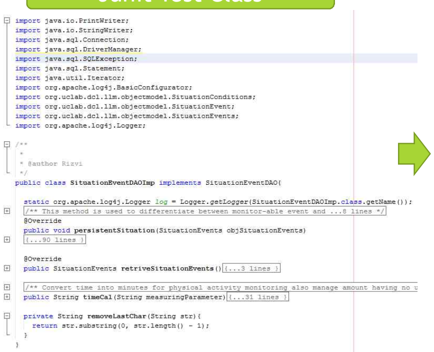
경희대학교 KYUNG HEE UNIVERSITY

Data Retrieval & Persistence

56

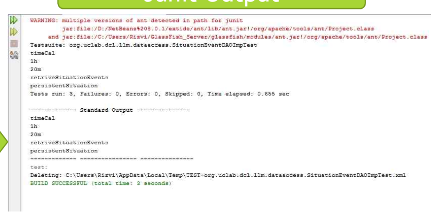
MM V3.0_DCL_OpenSource_V0.2_20161104_Bilal

Junit Test Class



Junit Test Report

Junit Output



Database Retrieval & Persistence :

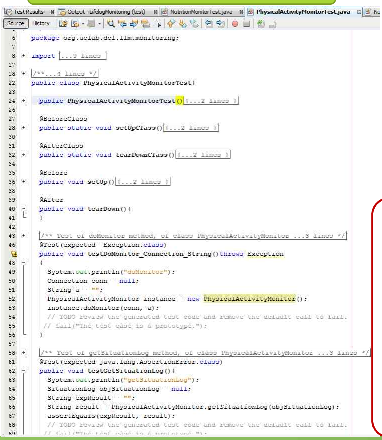
- Test is performed to validate the data retrieved from service
- Test is performed to persist the data into data base.
- Test is performed to validate the exception handling for invalid data.

경희대학교 KYUNG HEE UNIVERSITY

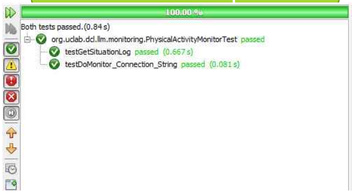
Physical Activities Monitoring

MM V3.0_DCL_OpenSource_V0.2_20161104_Bilal

Junit Test Class



Junit Test Report



Junit Output

Physical Activity Monitoring:

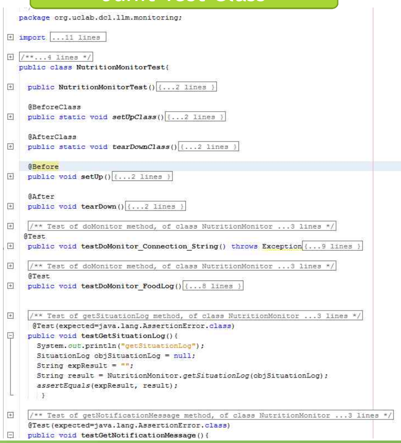
- Test is performed to identify the valid condition based on duration.
- Test is performed with multiple users with multiple situation.
- Test is performed to validate the exception handling for service availability and data connection.

경희대학교 KYUNG HEE UNIVERSITY

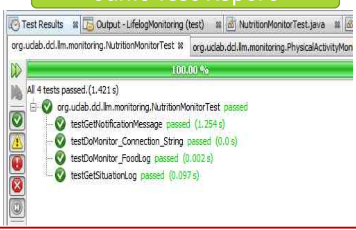
Nutrition Monitoring

MM V3.0_DCL_OpenSource_V0.2_20161104_Bilal

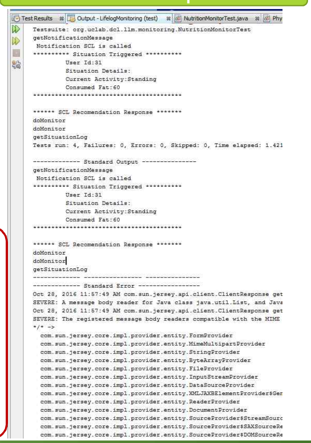
Junit Test Class



Junit Test Report



Junit Output



Nutrition Monitoring:

- Test is performed to generate intimation on valid value of Fats consumption from different foods/ duration.
- Test is performed to validate the exception handling.

경희대학교 KYUNG HEE UNIVERSITY

Mining Minds

Lifelog Mapping and Representation (LLMR)

Data Curation Layer MMV-3.0

경희대학교 KYUNG HEE UNIVERSITY · KITECH 한국생산기술연구원 · POSTECH 포항공과대학교 · UNIVERSITY of TASMANIA · 녹십자헬스케어 · TAPACROSS · U2 system User & Usability system · KOCSA 한국컨설팅서비스협회

Agenda

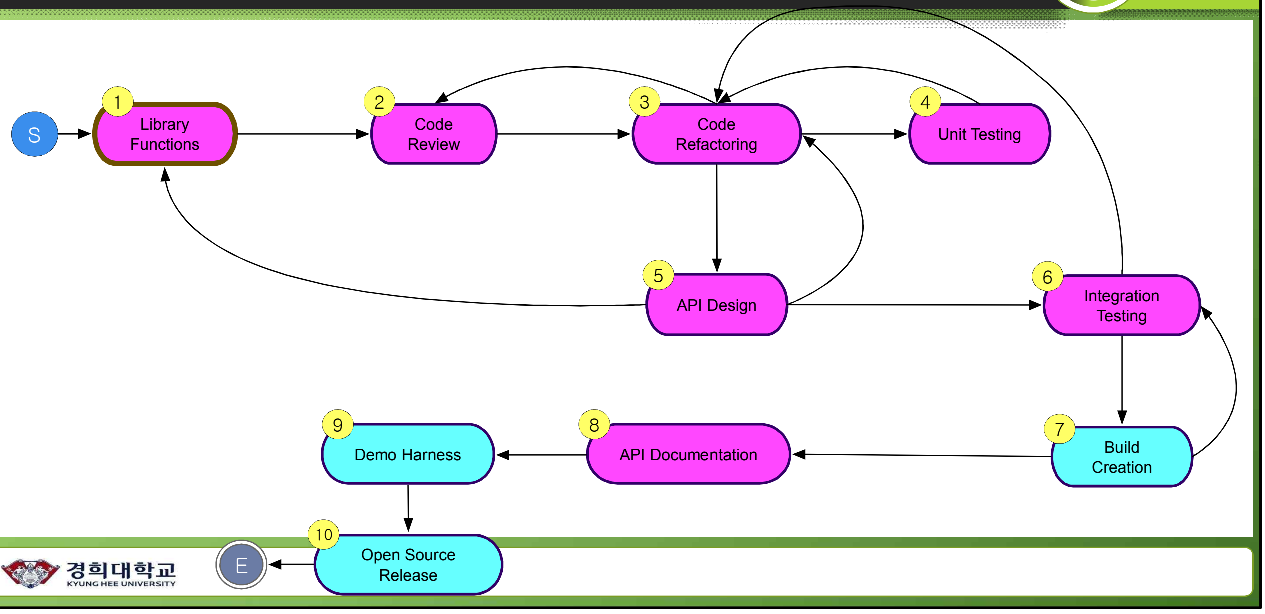
60

- Open Source Roadmap
  - Library Function
  - Class Diagram
  - Refactoring
    - Code Refactoring
    - Inline Query Conversion
    - Refactoring Impact
  - Unit Testing
  - Integration Testing
  - API Design
  - Build Creation
  - API Documentation
  - Demo Harness
  - Open Source Release

경희대학교 KYUNG HEE UNIVERSITY

Open Source Release Process - Roadmap

61



경희대학교 KYUNG HEE UNIVERSITY

LLMR- Conceptual View

62

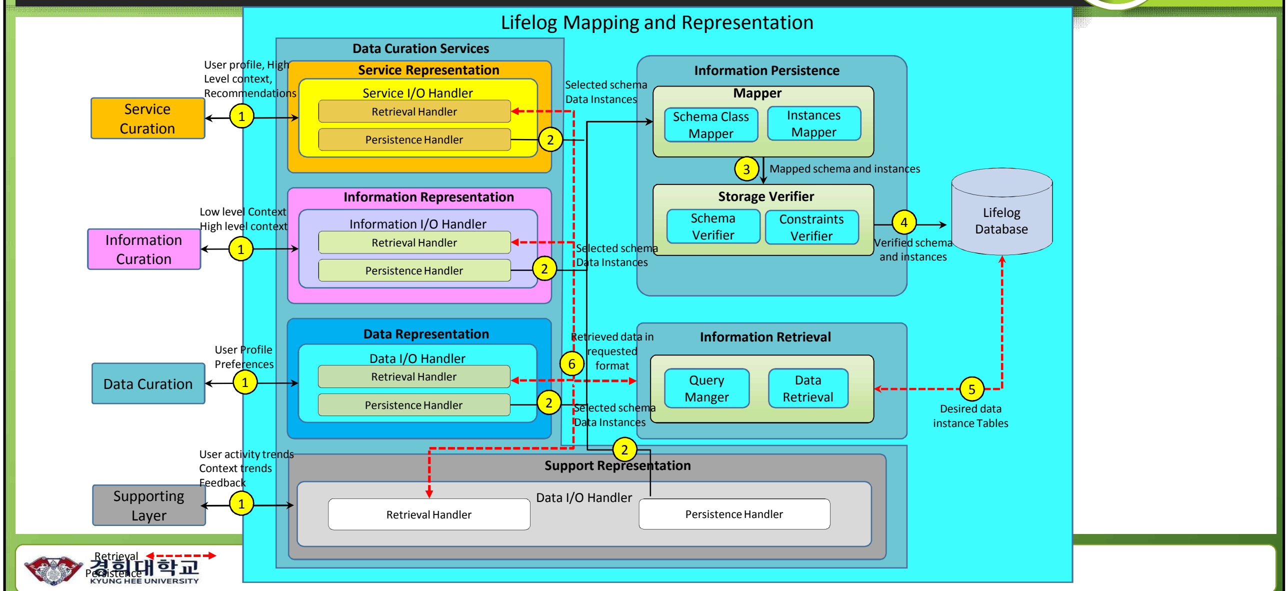
- **Representation of Lifelog:** Daily Activities, Goals, Recommendations and feedback have Continuous Semantics.

Total 9 representation models in literature

경희대학교 KYUNG HEE UNIVERSITY

LLMR - Architecture

63



Task-1 : Library Functions

64

LLMR: Function Identification

Life-log Mapping and Representation

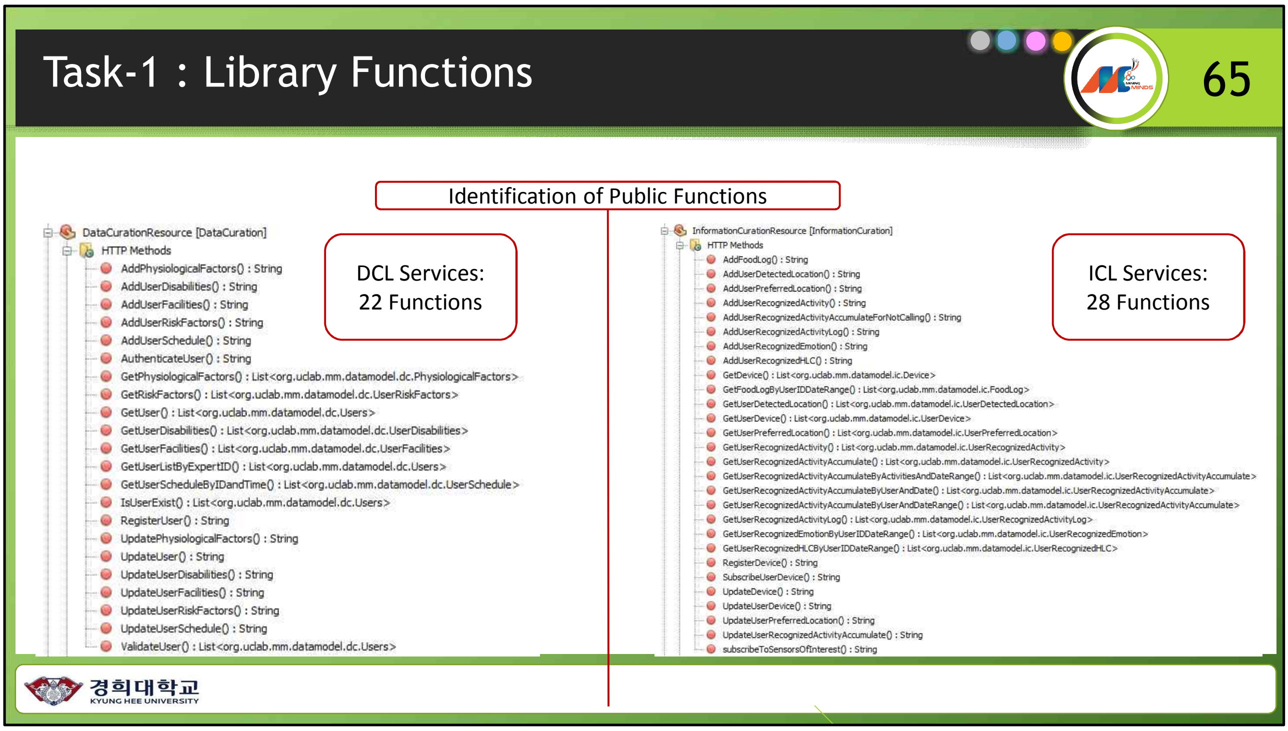
DCL Services
- Packages = 3
- Number of classes = 17
- Public Functions = 22
- Private Function = 44

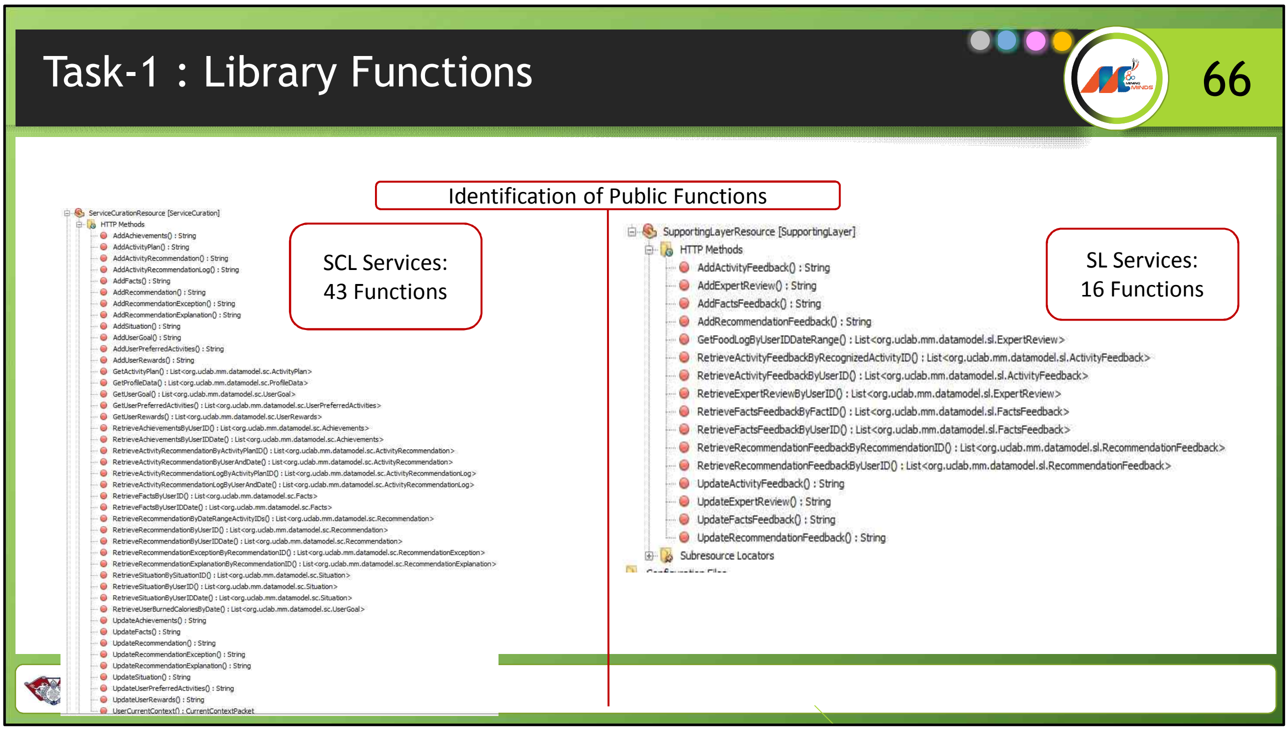
ICL Services
- Packages = 3
- Number of classes = 25
- Public Functions = 28
- Private Functions = 56

SCL Services
- Packages = 3
- Number of classes = 29
- Public Functions = 43
- Private Functions = 86

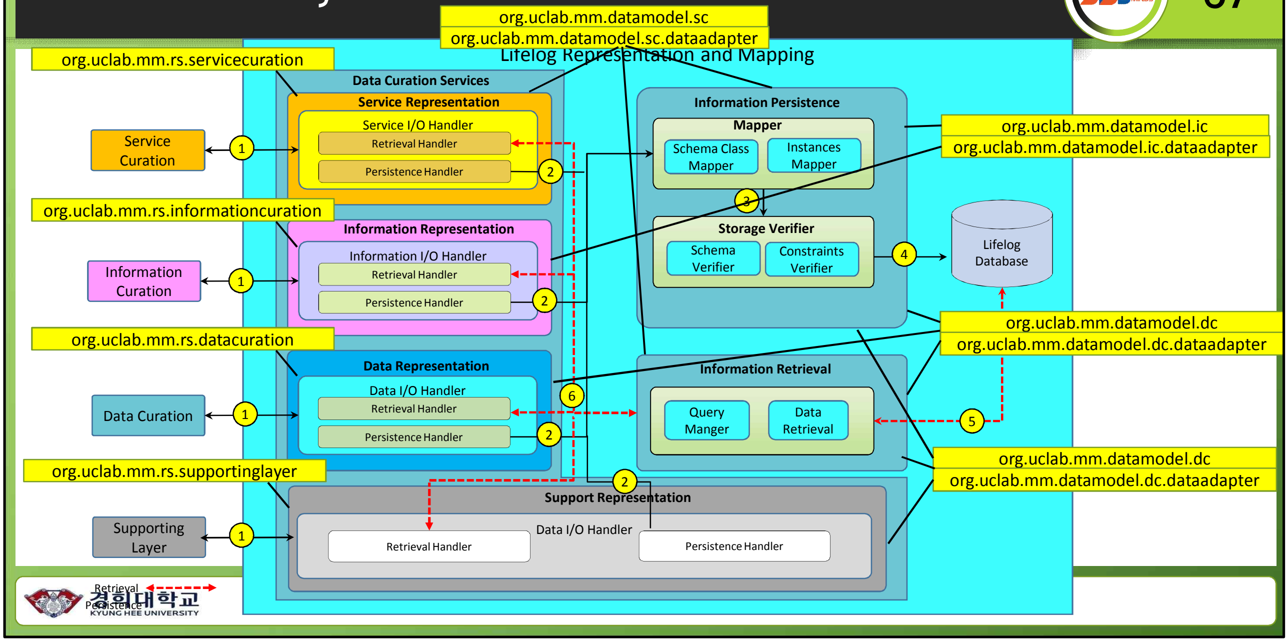
SL Services
- Packages = 3
- Number of classes = 15
- Public Functions = 16
- Private Functions = 32

- Total Packages = 12
- Total classes = 86
- Public Functions = 109
- Private Functions = 218


Task-1 : Library Functions
65
Identification of Public Functions
DataCurationResource [DataCuration]
HTTP Methods
AddPhysiologicalFactors() : String
AddUserDisabilities() : String
AddUserFacilities() : String
AddUserRiskFactors() : String
AddUserSchedule() : String
AuthenticateUser() : String
GetPhysiologicalFactors() : List<org.uclab.mm.datamodel.dc.PhysiologicalFactors>
GetRiskFactors() : List<org.uclab.mm.datamodel.dc.UserRiskFactors>
GetUser() : List<org.uclab.mm.datamodel.dc.Users>
GetUserDisabilities() : List<org.uclab.mm.datamodel.dc.UserDisabilities>
GetUserFacilities() : List<org.uclab.mm.datamodel.dc.UserFacilities>
GetUserListByExpertID() : List<org.uclab.mm.datamodel.dc.Users>
GetUserScheduleByIDandTime() : List<org.uclab.mm.datamodel.dc.UserSchedule>
IsUserExist() : List<org.uclab.mm.datamodel.dc.Users>
RegisterUser() : String
UpdatePhysiologicalFactors() : String
UpdateUser() : String
UpdateUserDisabilities() : String
UpdateUserFacilities() : String
UpdateUserRiskFactors() : String
UpdateUserSchedule() : String
ValidateUser() : List<org.uclab.mm.datamodel.dc.Users>
DCL Services:
22 Functions
InformationCurationResource [InformationCuration]
HTTP Methods
AddFoodLog() : String
AddUserDetectedLocation() : String
AddUserPreferredLocation() : String
AddUserRecognizedActivity() : String
AddUserRecognizedActivityAccumulateForNotCalling() : String
AddUserRecognizedActivityLog() : String
AddUserRecognizedEmotion() : String
AddUserRecognizedHLC() : String
GetDevice() : List<org.uclab.mm.datamodel.ic.Device>
GetFoodLogByUserIDDateRange() : List<org.uclab.mm.datamodel.ic.FoodLog>
GetUserDetectedLocation() : List<org.uclab.mm.datamodel.ic.UserDetectedLocation>
GetUserDevice() : List<org.uclab.mm.datamodel.ic.UserDevice>
GetUserPreferredLocation() : List<org.uclab.mm.datamodel.ic.UserPreferredLocation>
GetUserRecognizedActivity() : List<org.uclab.mm.datamodel.ic.UserRecognizedActivity>
GetUserRecognizedActivityAccumulate() : List<org.uclab.mm.datamodel.ic.UserRecognizedActivity>
GetUserRecognizedActivityAccumulateByActivitiesAndDateRange() : List<org.uclab.mm.datamodel.ic.UserRecognizedActivityAccumulate>
GetUserRecognizedActivityAccumulateByUserAndDate() : List<org.uclab.mm.datamodel.ic.UserRecognizedActivityAccumulate>
GetUserRecognizedActivityAccumulateByUserAndDateRange() : List<org.uclab.mm.datamodel.ic.UserRecognizedActivityAccumulate>
GetUserRecognizedActivityLog() : List<org.uclab.mm.datamodel.ic.UserRecognizedActivityLog>
GetUserRecognizedEmotionByUserIDDateRange() : List<org.uclab.mm.datamodel.ic.UserRecognizedEmotion>
GetUserRecognizedHLCByUserIDDateRange() : List<org.uclab.mm.datamodel.ic.UserRecognizedHLC>
RegisterDevice() : String
SubscribeUserDevice() : String
UpdateDevice() : String
UpdateUserDevice() : String
UpdateUserPreferredLocation() : String
UpdateUserRecognizedActivityAccumulate() : String
subscribeToSensorsOfInterest() : String
ICL Services:
28 Functions
경희대학교
KYUNG HEE UNIVERSITY


Task-1 : Library Functions
66
Identification of Public Functions
ServiceCurationResource [ServiceCuration]
HTTP Methods
AddAchievements() : String
AddActivityPlan() : String
AddActivityRecommendation() : String
AddActivityRecommendationLog() : String
AddFacts() : String
AddRecommendation() : String
AddRecommendationException() : String
AddRecommendationExplanation() : String
AddSituation() : String
AddUserGoal() : String
AddUserPreferredActivities() : String
AddUserRewards() : String
GetActivityPlan() : List<org.uclab.mm.datamodel.sc.ActivityPlan>
GetProfileData() : List<org.uclab.mm.datamodel.sc.ProfileData>
GetUserGoal() : List<org.uclab.mm.datamodel.sc.UserGoal>
GetUserPreferredActivities() : List<org.uclab.mm.datamodel.sc.UserPreferredActivities>
GetUserRewards() : List<org.uclab.mm.datamodel.sc.UserRewards>
RetrieveAchievementsByUserID() : List<org.uclab.mm.datamodel.sc.Achievements>
RetrieveAchievementsByUserIDDate() : List<org.uclab.mm.datamodel.sc.Achievements>
RetrieveActivityRecommendationByActivityPlanID() : List<org.uclab.mm.datamodel.sc.ActivityRecommendation>
RetrieveActivityRecommendationByUserAndDate() : List<org.uclab.mm.datamodel.sc.ActivityRecommendation>
RetrieveActivityRecommendationLogByActivityPlanID() : List<org.uclab.mm.datamodel.sc.ActivityRecommendationLog>
RetrieveActivityRecommendationLogByUserAndDate() : List<org.uclab.mm.datamodel.sc.ActivityRecommendationLog>
RetrieveFactsByUserID() : List<org.uclab.mm.datamodel.sc.Facts>
RetrieveFactsByUserIDDate() : List<org.uclab.mm.datamodel.sc.Facts>
RetrieveRecommendationByDateRangeActivityIDs() : List<org.uclab.mm.datamodel.sc.Recommendation>
RetrieveRecommendationByUserID() : List<org.uclab.mm.datamodel.sc.Recommendation>
RetrieveRecommendationByUserIDDate() : List<org.uclab.mm.datamodel.sc.Recommendation>
RetrieveRecommendationExceptionByRecommendationID() : List<org.uclab.mm.datamodel.sc.RecommendationException>
RetrieveRecommendationExplanationByRecommendationID() : List<org.uclab.mm.datamodel.sc.RecommendationExplanation>
RetrieveSituationBySituationID() : List<org.uclab.mm.datamodel.sc.Situation>
RetrieveSituationByUserID() : List<org.uclab.mm.datamodel.sc.Situation>
RetrieveSituationByUserIDDate() : List<org.uclab.mm.datamodel.sc.Situation>
RetrieveUserBurnedCaloriesByDate() : List<org.uclab.mm.datamodel.sc.UserGoal>
UpdateAchievements() : String
UpdateFacts() : String
UpdateRecommendation() : String
UpdateRecommendationException() : String
UpdateRecommendationExplanation() : String
UpdateSituation() : String
UpdateUserPreferredActivities() : String
UpdateUserRewards() : String
UserCurrentContext() : CurrentContextPacket
SCL Services:
43 Functions
SupportingLayerResource [SupportingLayer]
HTTP Methods
AddActivityFeedback() : String
AddExpertReview() : String
AddFactsFeedback() : String
AddRecommendationFeedback() : String
GetFoodLogByUserIDDateRange() : List<org.uclab.mm.datamodel.sl.ExpertReview>
RetrieveActivityFeedbackByRecognizedActivityID() : List<org.uclab.mm.datamodel.sl.ActivityFeedback>
RetrieveActivityFeedbackByUserID() : List<org.uclab.mm.datamodel.sl.ActivityFeedback>
RetrieveExpertReviewByUserID() : List<org.uclab.mm.datamodel.sl.ExpertReview>
RetrieveFactsFeedbackByFactID() : List<org.uclab.mm.datamodel.sl.FactsFeedback>
RetrieveFactsFeedbackByUserID() : List<org.uclab.mm.datamodel.sl.FactsFeedback>
RetrieveRecommendationFeedbackByRecommendationID() : List<org.uclab.mm.datamodel.sl.RecommendationFeedback>
RetrieveRecommendationFeedbackByUserID() : List<org.uclab.mm.datamodel.sl.RecommendationFeedback>
UpdateActivityFeedback() : String
UpdateExpertReview() : String
UpdateFactsFeedback() : String
UpdateRecommendationFeedback() : String
Subresource Locators
SL Services:
16 Functions

Task-1 : Library Functions



Task-2 : Code Review

LLMR: Design Pattern

Façade Design Pattern

- Provides a unified interface to multiple inner interfaces
- Software Library easy to use, understand and test
- Library become more readable
- Reduce the dependencies of outer code on inner library classes
- Encapsulate the inner objects and methods of library
- Wrap a complicated sub systems with unified interface

Bridge Design Pattern

- Scalability for data storage
- Decouple the abstraction from its implementations
- Divide the responsibilities among different classes
- Hide details from client

Task-2 : Code Review

69

LLMR: Design Pattern

Data Access Object Design Pattern

- Separates low level data accessing API or operations from high level business services
- The goal is to abstract and encapsulate all access to the data and provide an interface.
- The DAO is usually able to create an instance of a data object ("to read data") and also to persist data ("to save data") to the data source

Wrapper Design Pattern

- A wrapper converts the interface of a class into another interface clients expect
- Wrappers let classes work together that couldn't otherwise because of incompatible interfaces
- Provides multiple objects of multiple classes in a single glance

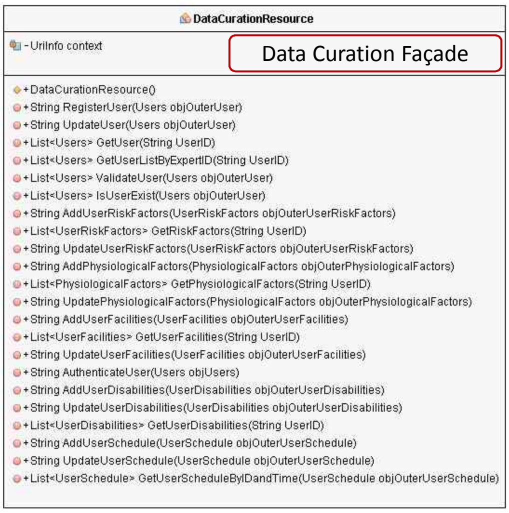
Task-2 : Code Review

70

LLMR: DCL façade

Model Coverage

- Users
- Users' schedules
- Users' physiological Factors
- Users' Address
- Users' Disabilities
- Users' Facilities
- Users' Risk Factors



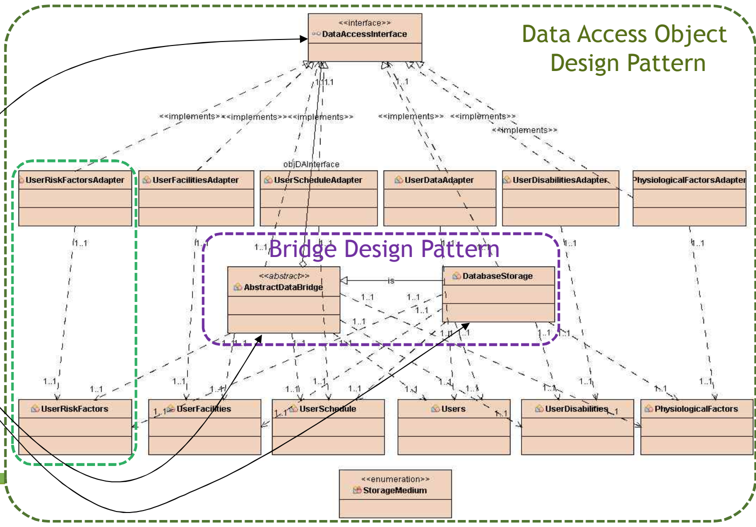
Task-2 : Code Review

71

LLMR: DCL Representation

Data Curation Façade

```
DataCurationResource
- UriInfo context
+ DataCurationResource()
+ String RegisterUser(Users objOuterUser)
+ String UpdateUser(Users objOuterUser)
+ List<Users> GetUser(String UserID)
+ List<Users> GetUserListByExpertID(String UserID)
+ List<Users> ValidateUser(Users objOuterUser)
+ List<Users> IsUserExist(Users objOuterUser)
+ String AddUserRiskFactors(UserRiskFactors objOuterUserRiskFactors)
+ List<UserRiskFactors> GetRiskFactors(String UserID)
+ String UpdateUserRiskFactors(UserRiskFactors objOuterUserRiskFactors)
+ String AddPhysiologicalFactors(PhysiologicalFactors objOuterPhysiologicalFactors)
+ List<PhysiologicalFactors> GetPhysiologicalFactors(String UserID)
+ String UpdatePhysiologicalFactors(PhysiologicalFactors objOuterPhysiologicalFactors)
+ String AddUserFacilities(UserFacilities objOuterUserFacilities)
+ List<UserFacilities> GetUserFacilities(String UserID)
+ String UpdateUserFacilities(UserFacilities objOuterUserFacilities)
+ String AuthenticateUser(Users objUsers)
+ String AddUserDisabilities(UserDisabilities objOuterUserDisabilities)
+ String UpdateUserDisabilities(UserDisabilities objOuterUserDisabilities)
+ List<UserDisabilities> GetUserDisabilities(String UserID)
+ String AddUserSchedule(UserSchedule objOuterUserSchedule)
+ String UpdateUserSchedule(UserSchedule objOuterUserSchedule)
+ List<UserSchedule> GetUserScheduleByIDandTime(UserSchedule objOuterUserSchedule)
```



Task-2 : Code Review

72

LLMR: ICL façade

Model Coverage

- Device
- User Device
- User Recognized Activity
- User Detected Location
- User Preferred Location
- User Recognized Activity
- User Recognized Emotion
- User Recognized HLC
- Food Log
- User GPS Data
- User Accelerometer Data

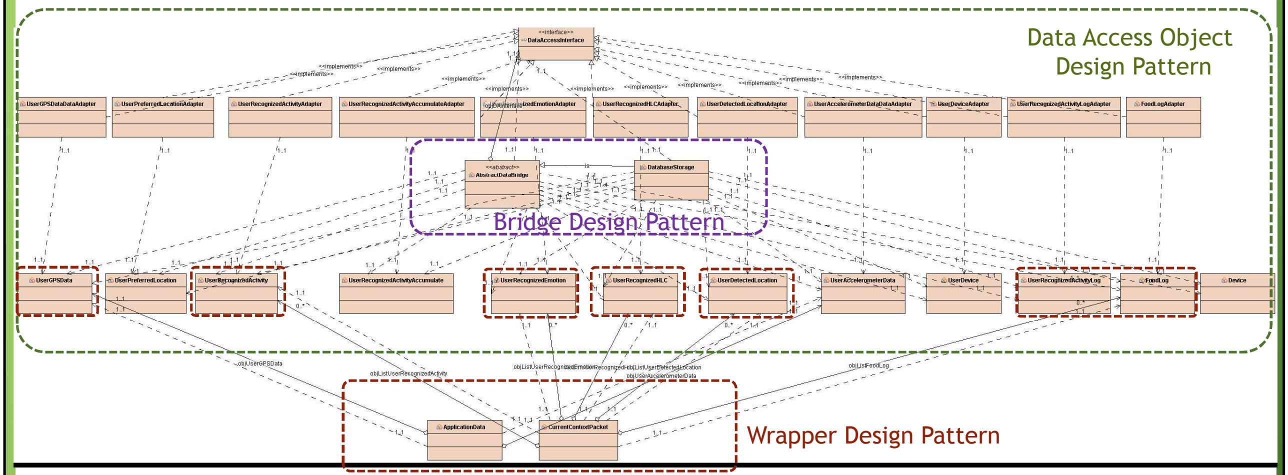
Information Curation Façade

```
InformationCurationResource
- UriInfo context
+ InformationCurationResource()
+ String RegisterDevice(Device objOuterDevice)
+ String UpdateDevice(Device objOuterDevice)
+ List<Device> GetDevice(String DeviceID)
+ String SubscribeUserDevice(UserDevice objOuterUserDevice)
+ String subscribeToSensorsOfInterest(List<UserDevice> objListUserDevice)
+ String UpdateUserDevice(UserDevice objOuterUserDevice)
+ List<UserDevice> GetUserDevice(String UserDeviceID)
+ String AddUserRecognizedActivity(UserRecognizedActivity objOuterUserRecognizedActivity)
+ List<UserRecognizedActivity> GetUserRecognizedActivity(UserRecognizedActivity objOuterUserRecognizedActivity)
+ List<UserRecognizedActivity> GetUserRecognizedActivityAccumulate(UserRecognizedActivity objOuterUserRecognizedActivity)
+ String AddUserDetectedLocation(UserDetectedLocation objOuterUserDetectedLocation)
+ List<UserDetectedLocation> GetUserDetectedLocation(UserDetectedLocation objOuterUserDetectedLocation)
+ String AddUserRecognizedActivityLog(UserRecognizedActivityLog objOuterUserRecognizedActivityLog)
+ List<UserRecognizedActivityLog> GetUserRecognizedActivityLog(UserRecognizedActivityLog objOuterUserRecognizedActivityLog)
+ String AddUserRecognizedActivityAccumulateForNotCalling(UserRecognizedActivityAccumulate objOuterUserRecognizedActivityAccumulate)
+ String UpdateUserRecognizedActivityAccumulate(UserRecognizedActivityAccumulate objOuterUserRecognizedActivityAccumulate)
+ List<UserRecognizedActivityAccumulate> GetUserRecognizedActivityAccumulateByUserAndDate(UserRecognizedActivityAccumulate objOuterUserRecognizedActivityAccumulate)
+ List<UserRecognizedActivityAccumulate> GetUserRecognizedActivityAccumulateByUserAndDateRange(UserRecognizedActivityAccumulate objOuterUserRecognizedActivityAccumulate)
+ List<UserRecognizedActivityAccumulate> GetUserRecognizedActivityAccumulateByActivitiesAndDateRange(UserRecognizedActivityAccumulate objOuterUserRecognizedActivityAccumulate)
+ String AddUserRecognizedEmotion(UserRecognizedEmotion objOuterUserRecognizedEmotion)
+ List<UserRecognizedEmotion> GetUserRecognizedEmotionByUserIDDateRange(UserRecognizedEmotion objOuterUserRecognizedEmotion)
+ String AddUserRecognizedHLC(UserRecognizedHLC objOuterUserRecognizedHLC)
+ List<UserRecognizedHLC> GetUserRecognizedHLCByUserIDDateRange(UserRecognizedHLC objOuterUserRecognizedHLC)
+ String AddUserPreferredLocation(UserPreferredLocation objOuterUserPreferredLocation)
+ String UpdateUserPreferredLocation(UserPreferredLocation objOuterUserPreferredLocation)
+ List<UserPreferredLocation> GetUserPreferredLocation(String UserID)
+ List<FoodLog> GetFoodLogByUserIDDateRange(FoodLog objOuterFoodLog)
+ String AddFoodLog(FoodLog objOuterFoodLog)
```

경희대학교 KYUNG HEE UNIVERSITY

Task-2 : Code Review

LLMR: ICL Representation



Task-2 : Code Review

LLMR: SCL façade

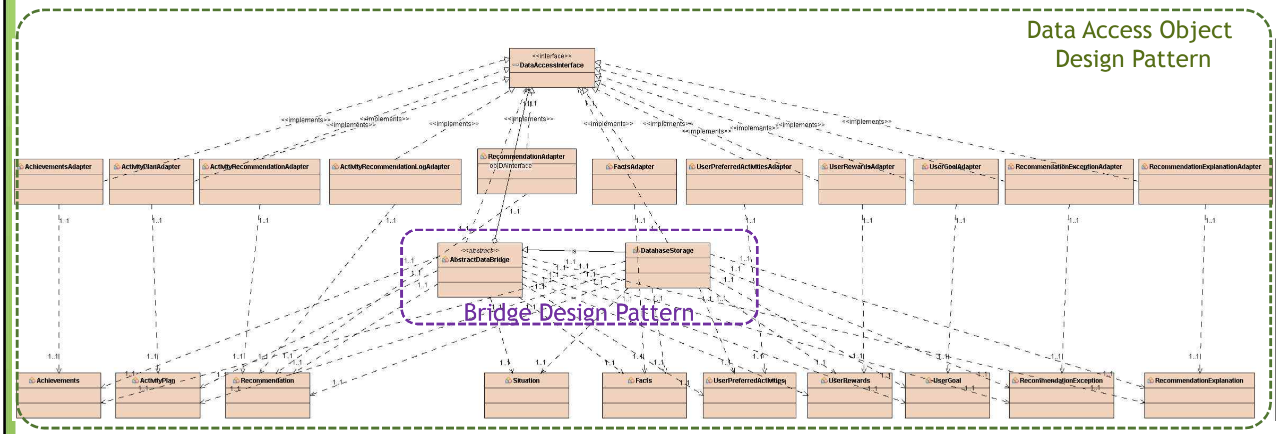
Model Coverage

- Users' Achievement
- Activity Plan
- Recommendations
- Facts
- Recommendation Exception
- Recommendation Explanation
- Users' goal
- Users' Preferred Activities
- Users' Rewards

Task-2 : Code Review

75

LLMR: SCL Representation



경희대학교
KYUNG HEE UNIVERSITY

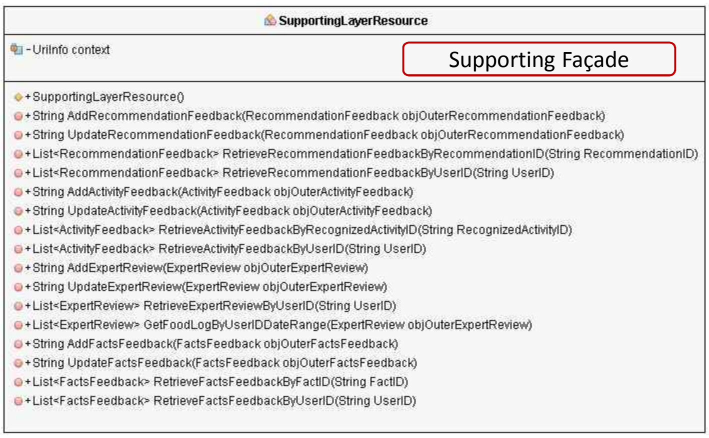
Task-2 : Code Review

76

LLMR: SL façade

Model Coverage

- Activity Feedback
- Expert Review
- Facts Feedback
- Recommendation Feedback
- Accelerometer Data Visualization
- GPS Data Visualization
- Users' Active Session

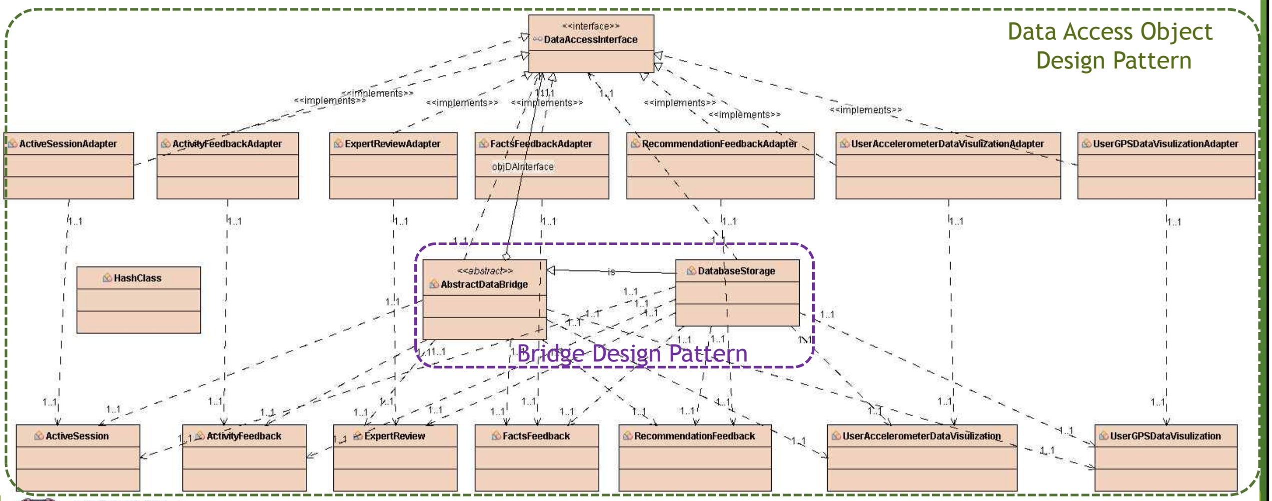


경희대학교
KYUNG HEE UNIVERSITY

Task-2 : Code Review

77

LLMR: SL Representation



경희대학교
KYUNG HEE UNIVERSITY

Task-2 : Code Review

78

I want to Save Risk Factors to Database

DCL Façade

```java
/**
 * Add user risk factors
 * @param objOuterUserRiskFactors
 * @return an instance of java.lang.String
 */
@POST
@Produces("application/json")
@Consumes("application/json")
@Path("AddUserRiskFactors")
public String AddUserRiskFactors(UserRiskFactors objOuterUserRiskFactors) {

    List<String> response = new ArrayList<String>();
    try
    {
        DataAccessInterface objDAInterface = new UserRiskFactorsAdapter();
        AbstractDataBridge objADBridge = new DatabaseStorage(objDAInterface);
        response = objADBridge.SaveUserRiskFactors(objOuterUserRiskFactors);
    }
    catch(Exception ex)
    {}
    Gson objJson = new Gson();
    String strResponse = objJson.toJson(response);
    return strResponse;
}
```

What to operate? Ans. Risk Factor
Where to/from operate? Ans. To Database
What is operation? Ans. Save

I want to Retrieve Recognized HLC from Database

ICL Façade

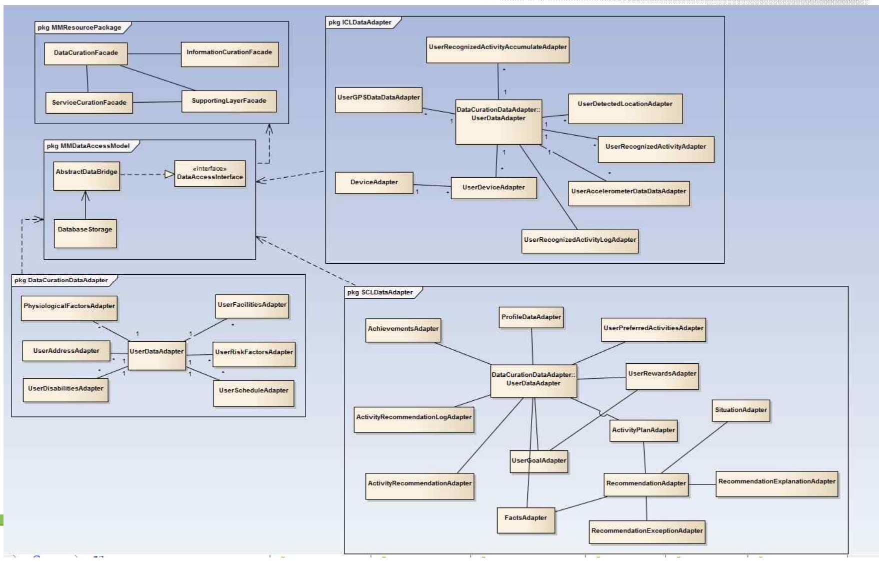
```java
/**
 * Retrieves User Recognized HLC by userid, start time and end time
 * @param objOuterUserRecognizedHLC
 * @return an instance of list of UserRecognizedHLC
 */
@POST
@Produces("application/json")
@Consumes("application/json")
@Path("GetUserRecognizedHLCByUserIDDateRange")
public List<UserRecognizedHLC> GetUserRecognizedHLCByUserIDDateRange(UserRecognizedHLC objOuterUser

    List<UserRecognizedHLC> objListUserRecognizedHLC = new ArrayList<UserRecognizedHLC>();
    String strResponse = "";
    objOuterUserRecognizedHLC.setRequestType("UserDateRange");
    try
    {
        DataAccessInterface objDAInterface = new UserRecognizedHLCAdapter();
        AbstractDataBridge objADBridge = new DatabaseStorage(objDAInterface);
        objListUserRecognizedHLC = objADBridge.RetriveUserRecognizedHLC(objOuterUserRecognizedHLC);
    }
    catch(Exception ex)
    {}

    return objListUserRecognizedHLC;
}
```

What to operate? Ans. Recognized HLC
Where to/from operate? Ans. From Database
What is operation? Ans. Retrieve

경희대학교
KYUNG HEE UNIVERSITY

Task-2 : Code Review

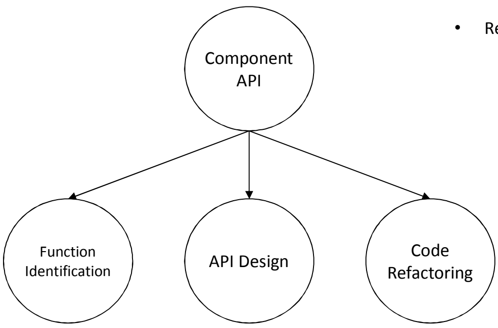
79



경희대학교 KYUNG HEE UNIVERSITY

Task-3 : Code Refactoring

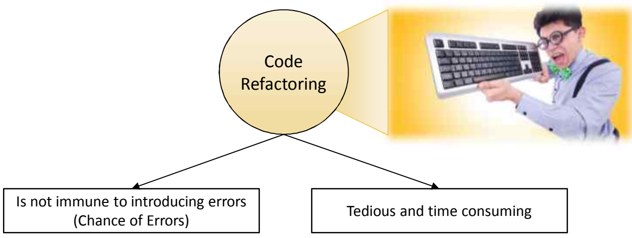
80



- Refactoring, as originally defined by Martin Fowler and Kent Beck

  "A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior… It is a disciplined way to clean up code that minimizes the chances of introducing bugs"

An emerging software development activity



https://www.javacodegeeks.com/2012/04/what-refactoring-is-and-what-it-isnt.html

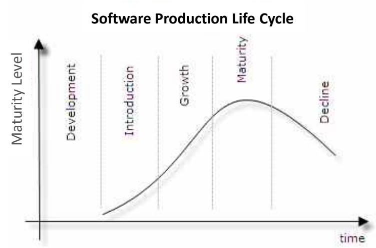
경희대학교 KYUNG HEE UNIVERSITY

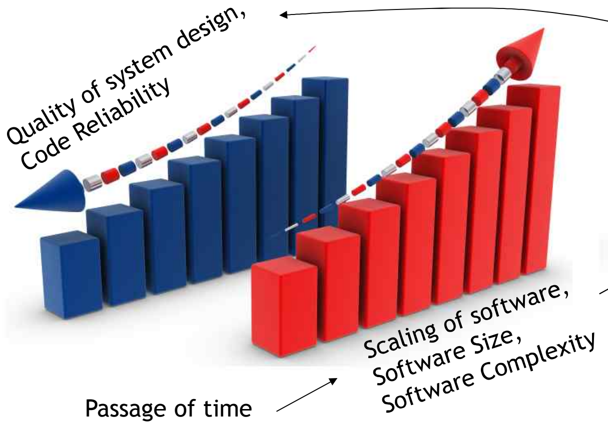
Task-3 : Code Refactoring

81

Who needs refactoring?

To avoid





Number of Errors (Bugs creep in)

Passage of time

Task-3 : Code Refactoring

82

DCL: Code Refactoring

Refactoring Examples

- **Rename:** Changing name of misleading and confusing method, variable, class, or any other item
- **Scalable and flexible Design Pattern:** Using a design pattern for scalability and flexibility.
- **Move Class:** Moving a class from wrong package to better fit package
- **Extract Methods:** A long method needs to be broken up to enhance readability and maintainability
- **Extract Supper Class:** The common behavior is "pulled up" into new parent abstract class.
- **Replace conditional with polymorphism:** Replace multiple conditional operators (If, Switch) with polymorphism to reduce extensive change in code.
- **Exception Handling:** Handle exception at each level of code.

Already Implemented maximum good practices of Software Engineering (100% Implemented)

90% Implemented

Identified Problems

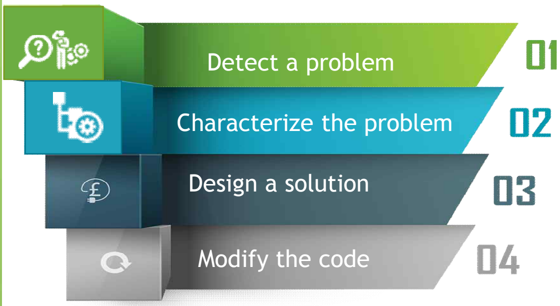
1 Logging Exception

2 Code comments for understandability

DCL: Code Refactoring

1

Logging Exception



Is there a problem? Yes.
What is the problem? Exceptions are handled but not logged.

What are the benefits? It will provide exception details in a logged file when it occurs.
Are there any risk? Yes code crash risk is high.

What should be the "goal state" of the code? Log all types of exceptions in a log file.
Which code transformation(s) will move the code towards the desired state? Update the code of exception handling.

Steps that will carry out the code transformation(s) that leave the code functioning the same way as it did before? Put the logger code in try-catch, it will not disturb the external behavior.

DCL: Code Refactoring

```
catch (Exception e)
{
    e.printStackTrace();
    objDbResponse.add("Error");
    System.err.println("Problem in registration user!");
}
return objDbResponse;
```

1

Logging Exception

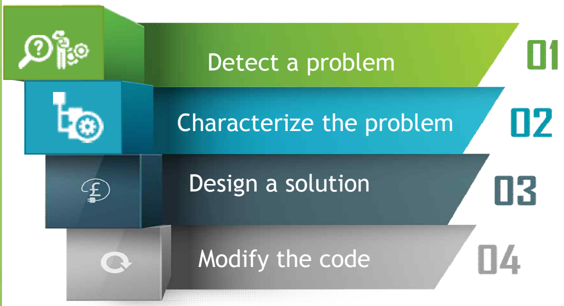
```
try
{
    Session session = this.sessionFactory.openSession();
            Rule objRule = (Rule) session.createQuery("from Rule r left outer join fetch r
    logger.info("Rule loaded successfully, Rule details="+objRule);
    session.close();
    return objRule;
}
catch(Exception  ex )
{
    logger.info("Error occurred in rule loading "+ ex.getMessage());
    return null;
}
```

Life-log Mapping and Representation: Open Source Progress

DCL: Code Refactoring

2

Code comments for understandability



Is there a problem? Yes.
What is the problem? Some functional comments are needed on each function.

What are the benefits? Enhance understandability of the code.
Are there any risk? No, there is no risk.

What should be the “goal state” of the code? To easily understand the code by other developers.
Which code transformation(s) will move the code towards the desired state? All functions of each class need comments.

Steps that will carry out the code transformation(s) that leave the code functioning the same way as it did before? We will put the functional comments on each function and it will not disturb the behavior of the functions.

Task-3 : Code Refactoring

DCL: Code Refactoring

```
@Override
public List<UserRiskFactors> RetriveData(Object objUserRiskFactors) {
    UserRiskFactors objOuterUserRiskFactors = new UserRiskFactors();
    List<UserRiskFactors> objListInnerUserRiskFactors = new ArrayList<UserRiskFactors>();
    objOuterUserRiskFactors =  (UserRiskFactors) objUserRiskFactors;

    try
    {
```

2

Code comments for understandability

```
/**
 * Retrieves User Device by UserDeviceID
 * @param UserDeviceID
 * @return an instance of list of UserDevice
 */
@GET
@Produces("application/json")
@Consumes("application/json")
@Path("UserDevice/{UserDeviceID}")
public List<UserDevice> GetUserDevice(@PathParam ("UserDeviceID") String UserDeviceID) {
```

Task-4 : Unit Testing

87

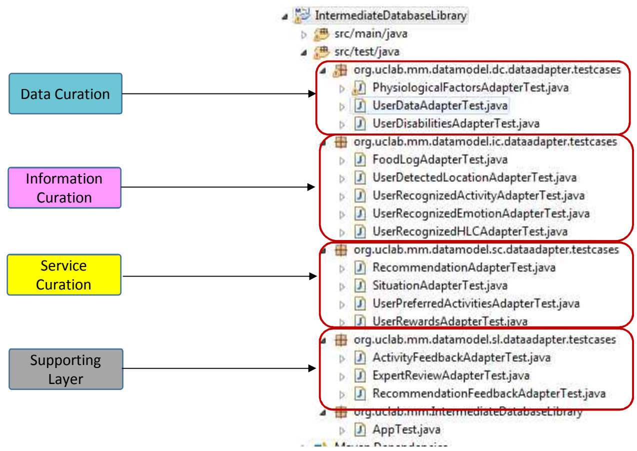
Junit Testing

Using Fiddler	Using JUnit Testing
❑ Web debugging ❑ Web session manipulation ❑ Performance Testing ❑ HTTP/HTTPS Traffic Recording ❑ Security Testing	❑ significantly faster than human resources ❑ Less investment in human resources ❑ Precise and reliable ❑ Programmable
Tested all functions by fiddler during development	Will test all functions by JUnit Testing for precision and reliability

Task-4 : Unit Testing

88

Junit Testing

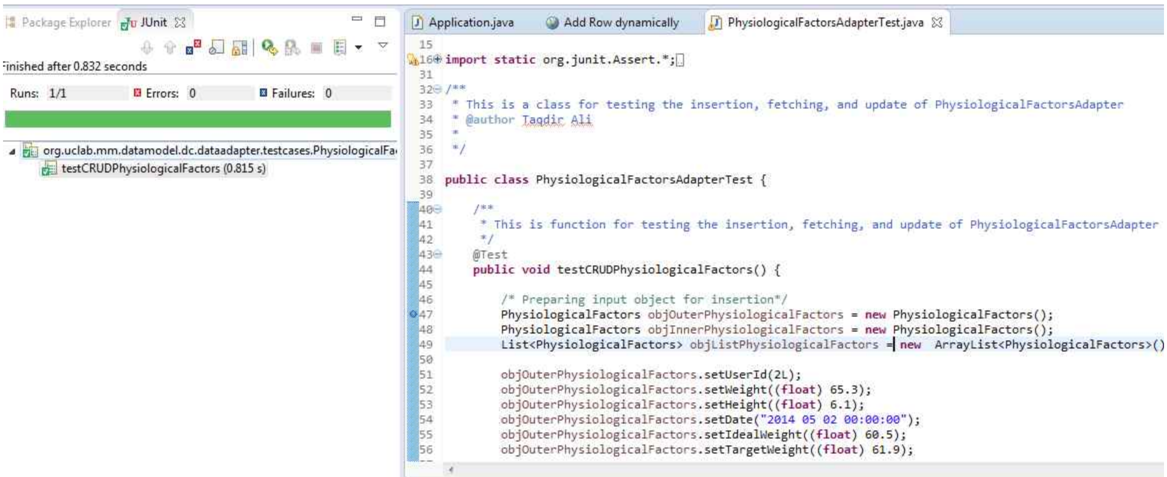


경희대학교 KYUNG HEE UNIVERSITY

Task-4 : Unit Testing

89

Junit Testing for Physiological Factor

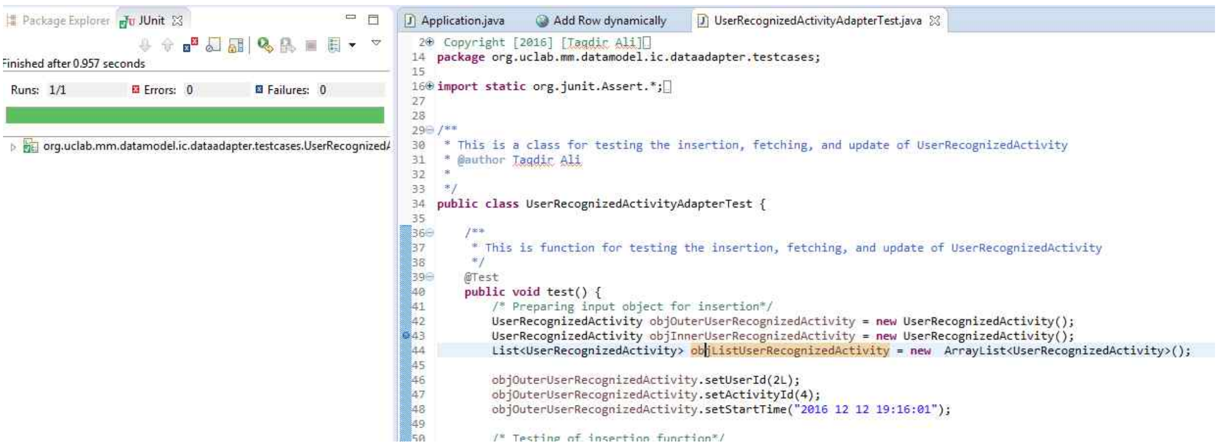


경희대학교
KYUNG HEE UNIVERSITY

Task-4 : Unit Testing

90

Junit Testing Recognized Activity Adapter

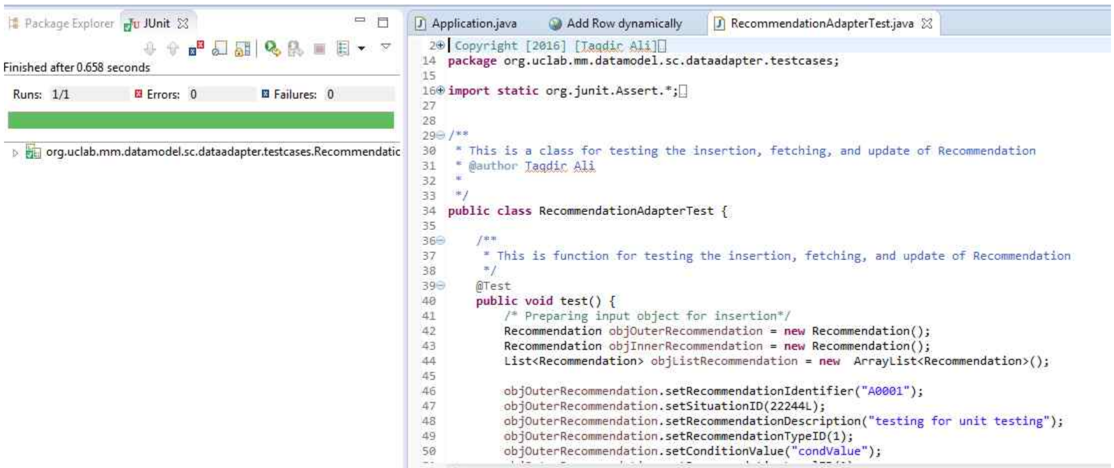


경희대학교
KYUNG HEE UNIVERSITY

Task-4 : Unit Testing

91

Junit Testing for Recommendation Adapter

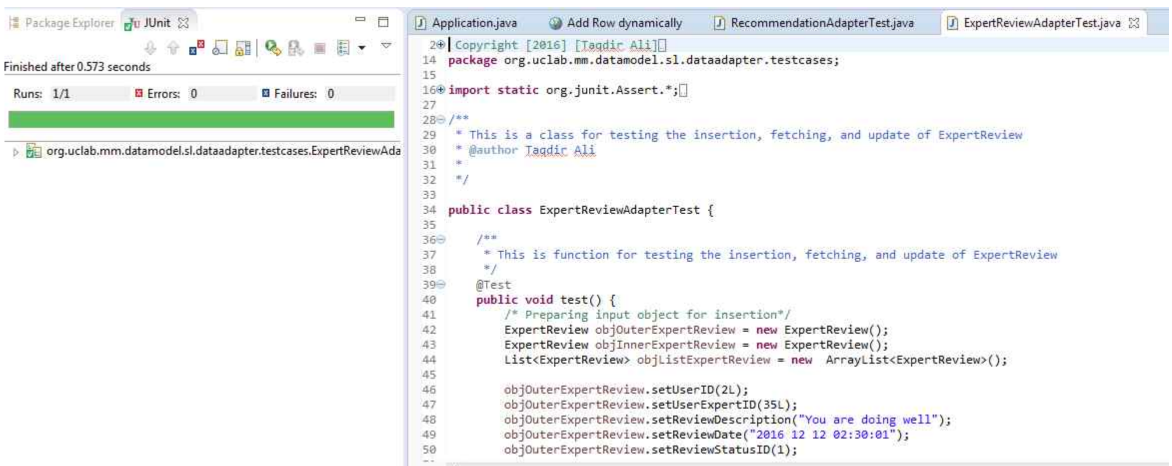


경희대학교
KYUNG HEE UNIVERSITY

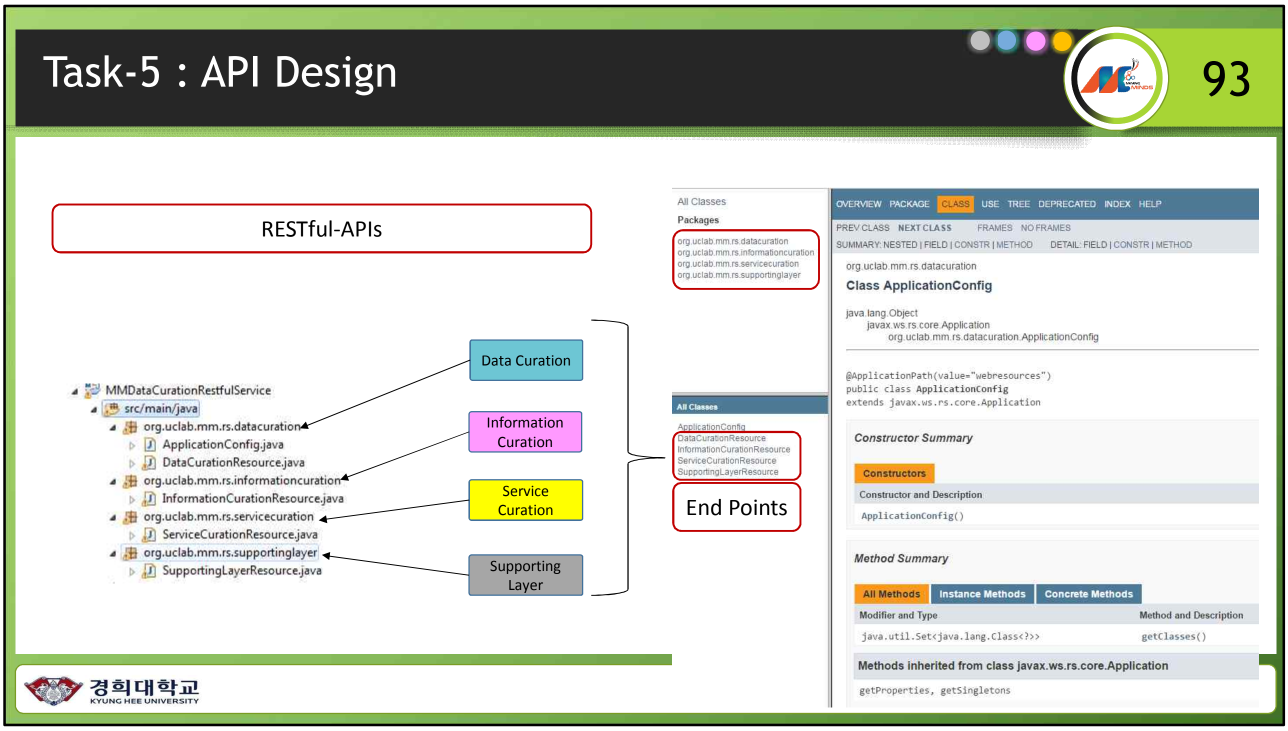
Task-4 : Unit Testing

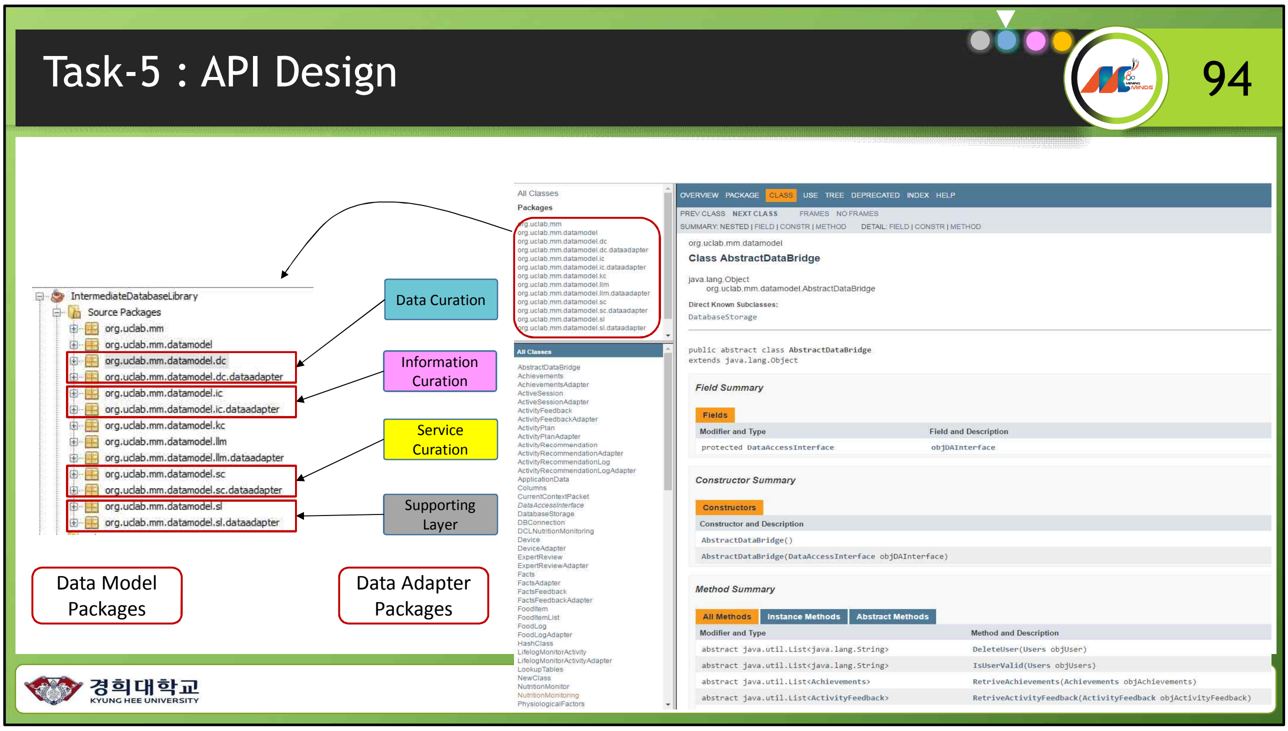
92

Junit Testing for Expert Review Adapter



경희대학교
KYUNG HEE UNIVERSITY

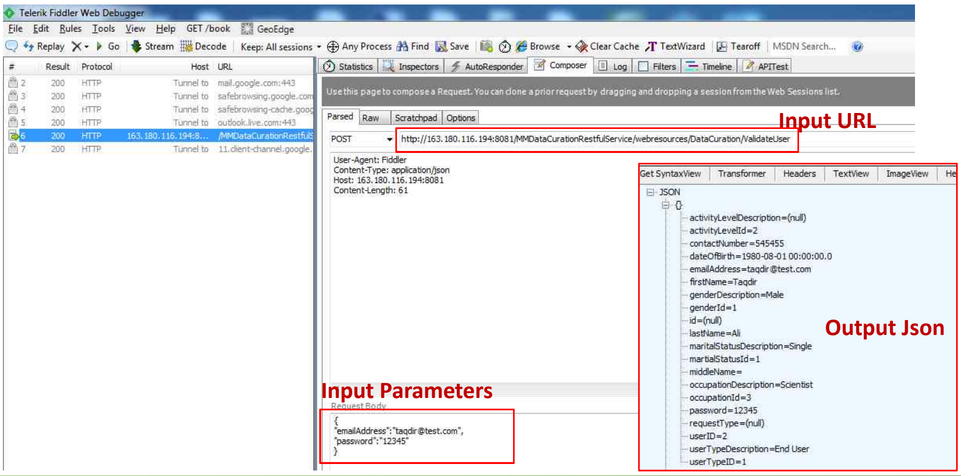

Task-5 : API Design
93
RESTful-APIs
MMDataCurationRestfulService
src/main/java
org.uclab.mm.rs.datacuration
ApplicationConfig.java
DataCurationResource.java
org.uclab.mm.rs.informationcuration
InformationCurationResource.java
org.uclab.mm.rs.servicecuration
ServiceCurationResource.java
org.uclab.mm.rs.supportinglayer
SupportingLayerResource.java
Data Curation
Information Curation
Service Curation
Supporting Layer
All Classes
Packages
org.uclab.mm.rs.datacuration
org.uclab.mm.rs.informationcuration
org.uclab.mm.rs.servicecuration
org.uclab.mm.rs.supportinglayer
All Classes
ApplicationConfig
DataCurationResource
InformationCurationResource
ServiceCurationResource
SupportingLayerResource
End Points
OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
PREV CLASS NEXT CLASS FRAMES NO FRAMES
SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD
org.uclab.mm.rs.datacuration
Class ApplicationConfig
java.lang.Object
javax.ws.rs.core.Application
org.uclab.mm.rs.datacuration.ApplicationConfig
@ApplicationPath(value="webresources")
public class ApplicationConfig
extends javax.ws.rs.core.Application
Constructor Summary
Constructors
Constructor and Description
ApplicationConfig()
Method Summary
All Methods Instance Methods Concrete Methods
Modifier and Type Method and Description
java.util.Set<java.lang.Class<?>> getClasses()
Methods inherited from class javax.ws.rs.core.Application
getProperties, getSingletons
경희대학교
KYUNG HEE UNIVERSITY


Task-5 : API Design
94
IntermediateDatabaseLibrary
Source Packages
org.uclab.mm
org.uclab.mm.datamodel
org.uclab.mm.datamodel.dc
org.uclab.mm.datamodel.dc.dataadapter
org.uclab.mm.datamodel.ic
org.uclab.mm.datamodel.ic.dataadapter
org.uclab.mm.datamodel.kc
org.uclab.mm.datamodel.llm
org.uclab.mm.datamodel.llm.dataadapter
org.uclab.mm.datamodel.sc
org.uclab.mm.datamodel.sc.dataadapter
org.uclab.mm.datamodel.sl
org.uclab.mm.datamodel.sl.dataadapter
Data Curation
Information Curation
Service Curation
Supporting Layer
Data Model Packages
Data Adapter Packages
All Classes
Packages
All Classes
OVERVIEW PACKAGE CLASS USE TREE DEPRECATED INDEX HELP
org.uclab.mm.datamodel
Class AbstractDataBridge
java.lang.Object
org.uclab.mm.datamodel.AbstractDataBridge
Direct Known Subclasses:
DatabaseStorage
public abstract class AbstractDataBridge
extends java.lang.Object
Field Summary
Fields
Modifier and Type Field and Description
protected DataAccessInterface objDAInterface
Constructor Summary
Constructors
Constructor and Description
AbstractDataBridge()
AbstractDataBridge(DataAccessInterface objDAInterface)
Method Summary
All Methods Instance Methods Abstract Methods
Modifier and Type Method and Description
abstract java.util.List<java.lang.String> DeleteUser(Users objUser)
abstract java.util.List<java.lang.String> IsUserValid(Users objUsers)
abstract java.util.List<Achievements> RetriveAchievements(Achievements objAchievements)
abstract java.util.List<ActivityFeedback> RetriveActivityFeedback(ActivityFeedback objActivityFeedback)
경희대학교
KYUNG HEE UNIVERSITY

Task-6 : Integration Test

95

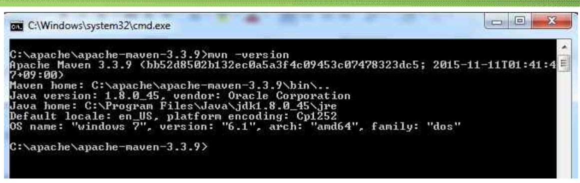
Integration testing Using Fiddler



경희대학교 KYUNG HEE UNIVERSITY

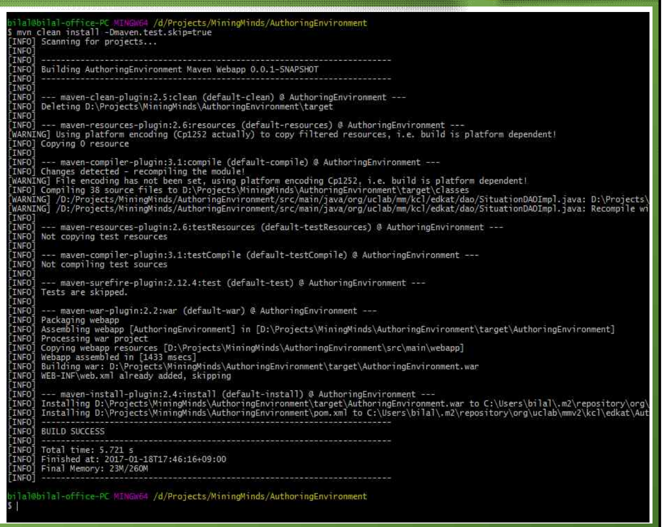
Task-7 : Build Creation

96



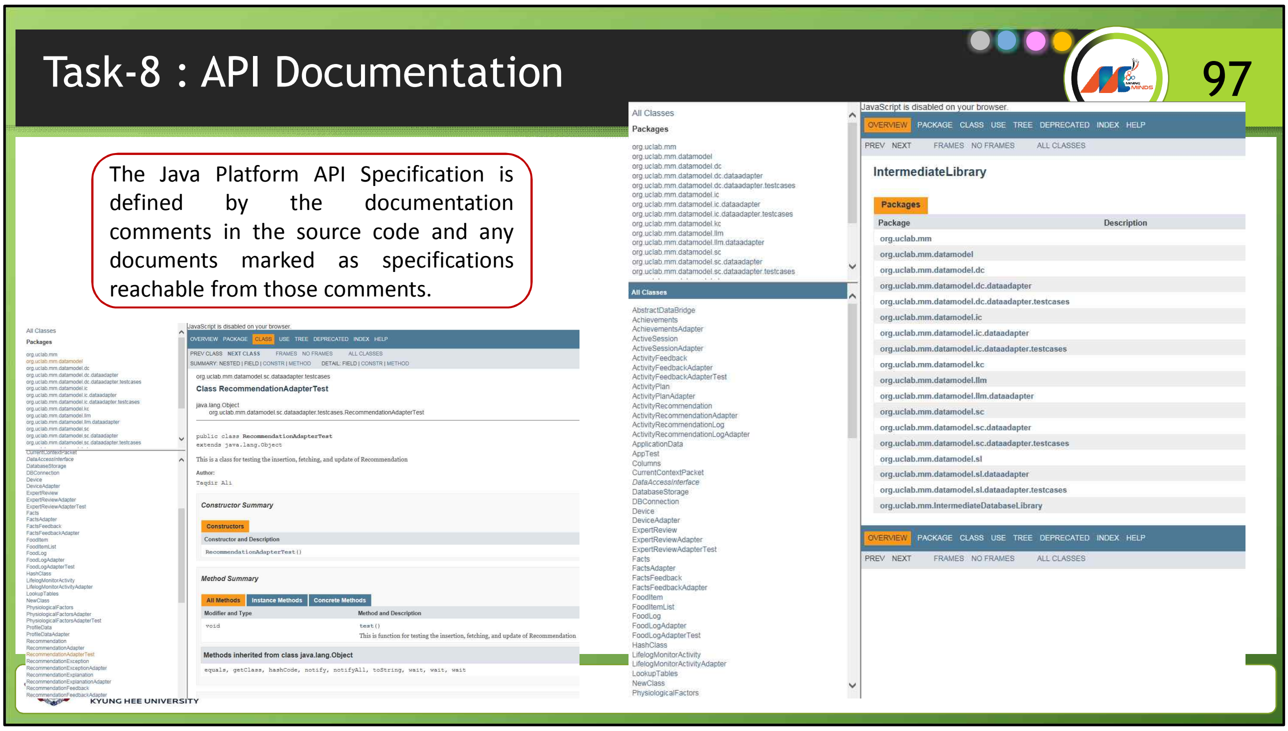
Maven install and start

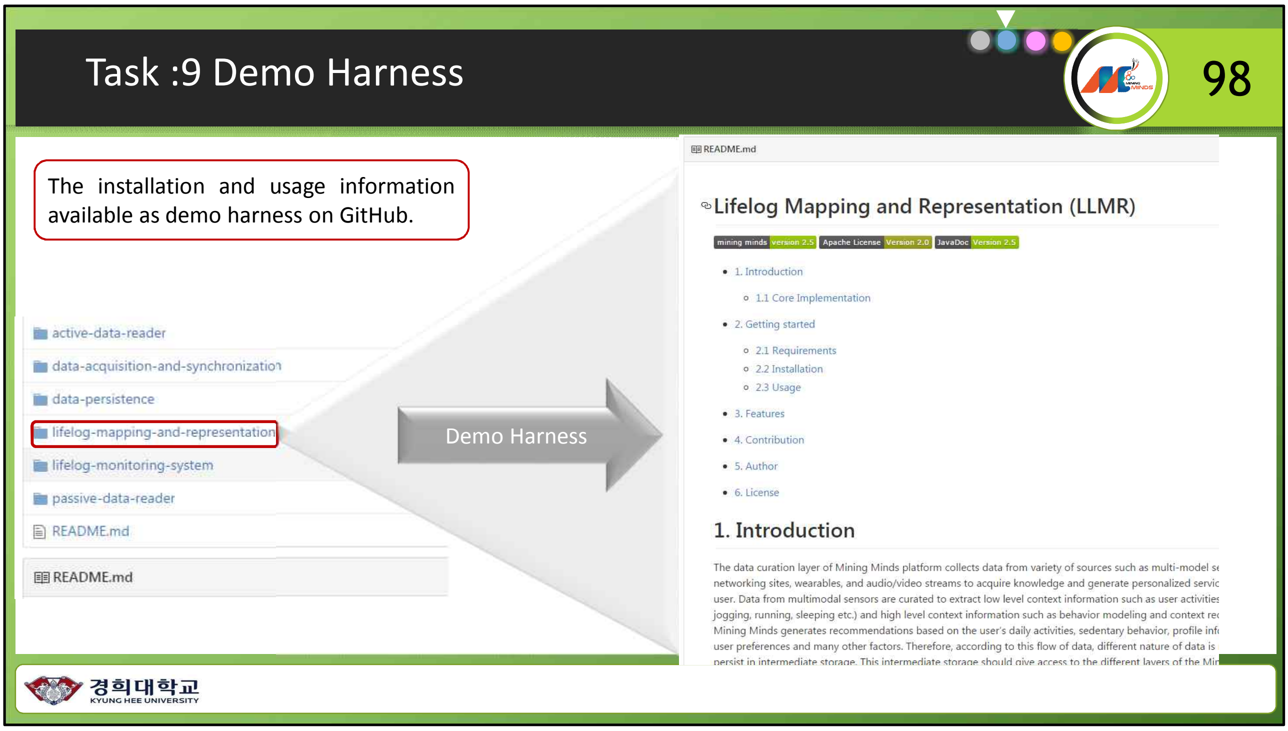
- Maven lets us get our package dependencies easily
- Maven forces us to have a standard directory structure



Build creation through Maven

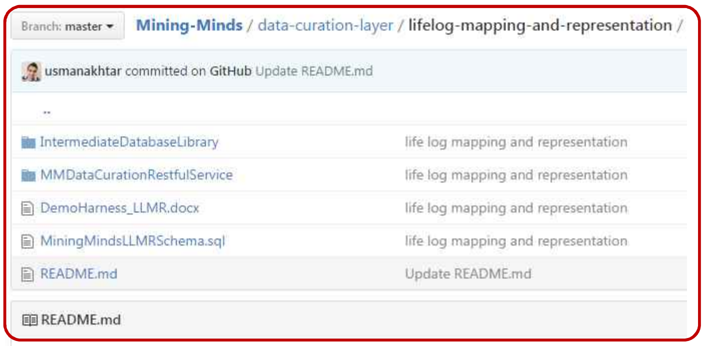
경희대학교 KYUNG HEE UNIVERSITY


Task-8 : API Documentation
97
The Java Platform API Specification is defined by the documentation comments in the source code and any documents marked as specifications reachable from those comments.
IntermediateLibrary
Packages
Class RecommendationAdapterTest
KYUNG HEE UNIVERSITY


Task :9 Demo Harness
98
The installation and usage information available as demo harness on GitHub.
active-data-reader
data-acquisition-and-synchronization
data-persistence
lifelog-mapping-and-representation
lifelog-monitoring-system
passive-data-reader
README.md
README.md
Demo Harness
README.md
Lifelog Mapping and Representation (LLMR)
1. Introduction
1.1 Core Implementation
2. Getting started
2.1 Requirements
2.2 Installation
2.3 Usage
3. Features
4. Contribution
5. Author
6. License
1. Introduction
경희대학교
KYUNG HEE UNIVERSITY

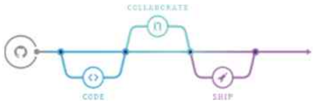
Task :10 Open Source Release

99



The source code is released under the following License:

- The Apache License, Version 2.0 (the "License")
- The copy of License may be obtained at http://www.apache.org/licenses/LICENSE-2.0



GitHub

경희대학교 KYUNG HEE UNIVERSITY

Mining Minds

Data Synchronization Open source

경희대학교 KYUNG HEE UNIVERSITY · KITECH 한국생산기술연구원 · POSTECH 포항공과대학교 · UNIVERSITY of TASMANIA · 녹십자헬스케어 · TAPACROSS · U2 system User & Usability system · KOCSA 한국컨설팅서비스협회

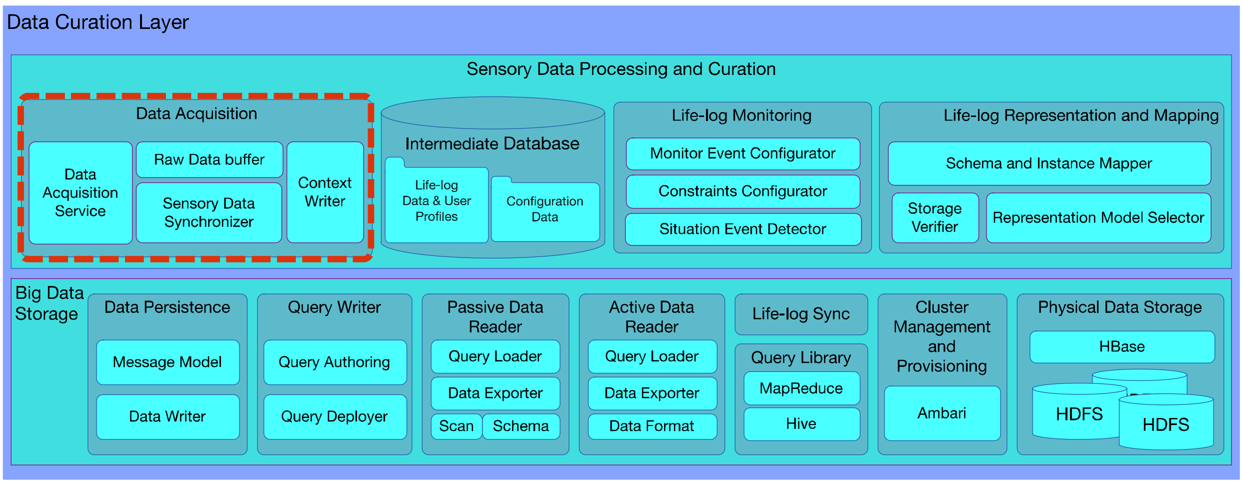
Agenda

101

- Sync component
- Sync architecture
- Open Source release process – roadmap
- Task 1 - Library function
- Task 2 – Code review
- Task 3 – Code refactoring
- Task 4 – Unit testing
- Task 5 – API design pattern
- Task 6+7 – Integration testing + Build creation
- Task 8 – API Documentation
- Task 9 – Demo Harness
- Task 10 – Open Source release

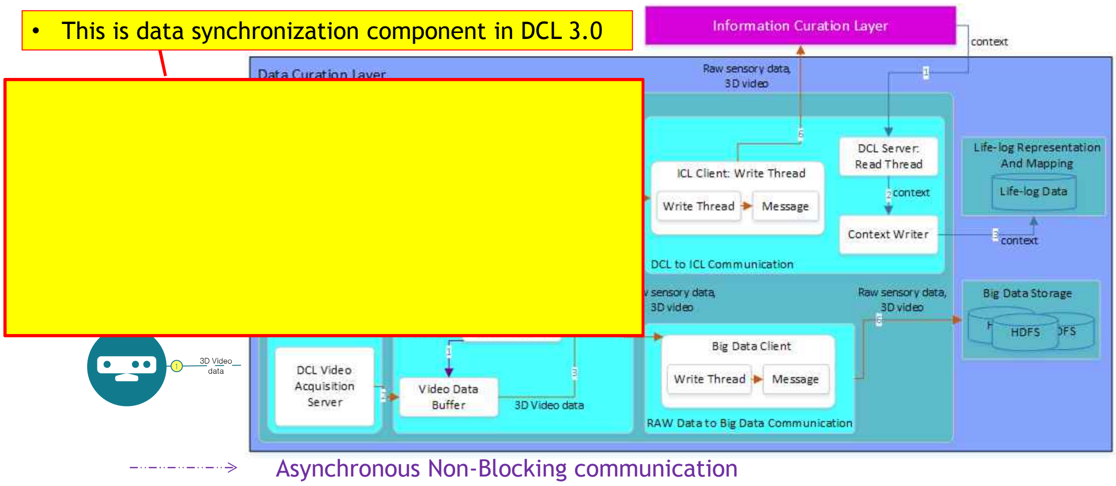
Synchronization component

102

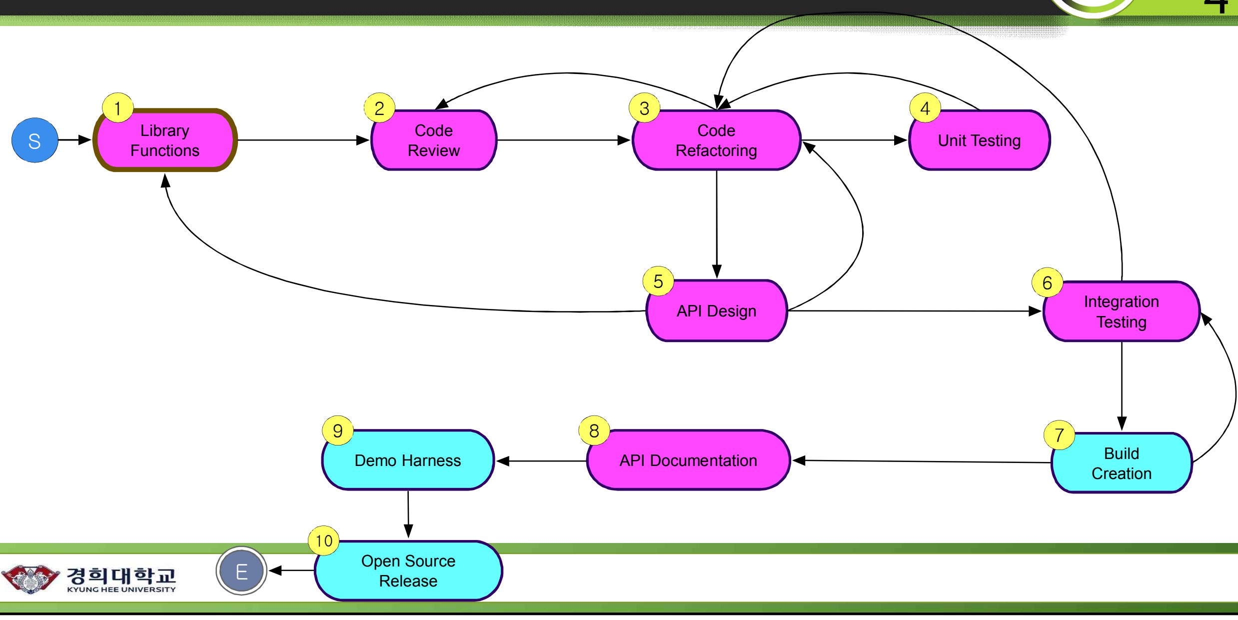


Synchronization architecture

- This is data synchronization component in DCL 3.0



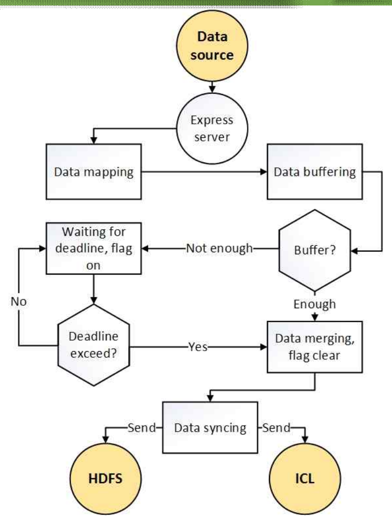
Open Source Release Process - Roadmap



Task: 1 - Open Source - Library function

105

- Identification of Library Functions: how to write API for the synchronization?
  - Thinking: what users do?
    - Add/remove sensors.
    - Change the content of data sent.
    - Change the destination of data sent.
    - Change the deadline for waiting.
    - Some common functions
  - Thinking: what user don't?
    - The working mechanism of the source code



경희대학교 KYUNG HEE UNIVERSITY

Task: 2 - Code Review

106

- Code Issues
  - Vertical Expansion of code.
  - Standardize the code according to well-known convention.
    - Change in data query required change in code.
  - Absence of design pattern.
    - Code Management and alteration .
  - Absence of exception handling.
  - Absence of proper comments.

경희대학교 KYUNG HEE UNIVERSITY

Task: 3 - Code refactoring

107

- Code refactoring progress: 100%.
  - Code style: follow Google JS code style (https://google.github.io/styleguide/javascriptguide.xml).
  - Code convention: follow JS code convention (http://javascript.crockford.com/code.html).
  - Decoupling the configuration and the running code.

```
/*****************************************************************
Author: Dinh-Mao Bui
Affiliation: UC Lab
Project: Mining Mind
Created date: 2016-03-23
Revised date: 2016-09-08
Functionality: do the synchronization for unlimited data source.
*****************************************************************

/*Required libraries*********************************************
var express     =require('express');
var loki        =require('lokijs');
var bodyParser  =require('body-parser');
var requestify  =require('requestify');
var fs = require('fs');
var util = require('util');
var nodeProcess = require('process');
var config = require('./config.json');
var apiHelper=require('./api.js');
/*Required libraries*********************************************
```

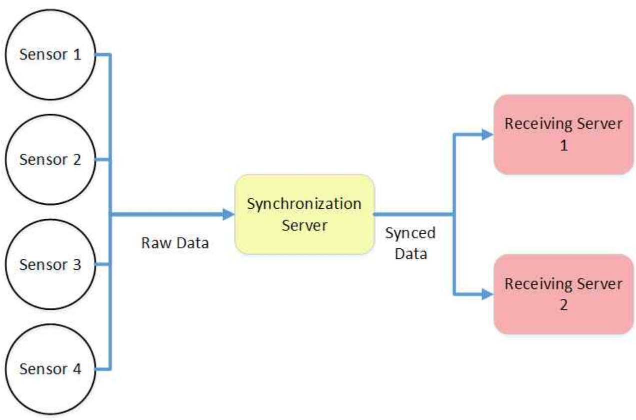
```
/*Load configuration*****************************************************************
var nDataSource=config.nDataSource; // number of data source.
                                    //Theoretically, this source code can
                                    //handle unlimited number of data source.
var deadline =config.deadline;//deadline to issue the forced (bad) data sync
var destUrl=config.destUrl;
/*Load configuration*****************************************************************

/*Global variables*******************************************************************
var tempDate=new Date("2000/01/01");
var summaryDuration=tempDate.getTime();
var nSync=0;
/*Global variables*******************************************************************

/*Server initialization**************************************************************
var app =express();
    app.use(bodyParser.json());
var urlencodedParser = bodyParser.urlencoded({ extended: true })
var db =new loki('sync_temp');
var buffer =db.addCollection('buffer');//main collection for data sync
var historyData =db.addCollection('history');//most previous data for covering the missing value
/*Server initialization**************************************************************
```

경희대학교
KYUNG HEE UNIVERSITY

Task: 4 - Unit Test

108

- Scenario
  - 4 sensors sending data to Synchronization Server
    - Initial sending period: randomly inside a window size.
    - Window size: 3 seconds.
    - Data size: randomly.
    - File attached: yes, 1KB.
  - Procedure
    - Synchronization Server syncs data before sending to 2 different receiving servers.
  - Receiving server
    - Implemented in nodeJS.
    - Receive synced json data, then parse it.
- Purspose: check that whether the synchronization works properly after refractoring or not.

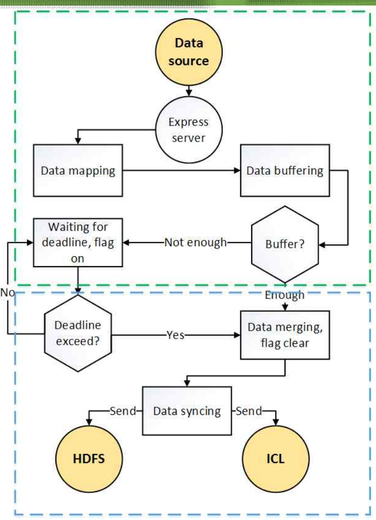


경희대학교
KYUNG HEE UNIVERSITY

Task 5 - API design pattern

10
9

- Module design pattern
  - Divide source code into three modules
    - Sync core: responsible of doing mandatory part of synchronization
    - Configuration: keeping all configuration records.
    - API: let user adjust sync mechanism.
  - Purpose
    - Make the source code easier to maintain.
    - Increase the transparency.
    - Enable plug 'n' play feature.
- Singleton design pattern
  - Make Nodejs server unique based on process id.
  - Increase the stability by preventing the server conflict.
- Progress: 100%



경희대학교
KYUNG HEE UNIVERSITY

Task 6+7: Integration testing + Build creation

11
0

- DAS component is successfully integrated with DCL 2.5 layer.
- The integration testing and benchmarking have been done according to the published paper [1].
- The built creation has been uploaded into Github [2].

[1] Amin, Muhammad Bilal, Oresti Banos, Wajahat Ali Khan, Hafiz Syed Muhammad Bilal, Jinhyuk Gong, Dinh-Mao Bui, Soung Ho Cho et al. "On curating multimodal sensory data for health and wellness platforms." *Sensors* 16, no. 7 (2016): 980.
[2] https://github.com/usmanakhtar/MiningMinds/tree/master/DCL/Data_Acquisition_and_Synchronization

경희대학교
KYUNG HEE UNIVERSITY

Task 8 - API implementation

```
var nodeProcess = require('process');
var jsonFile = require('jsonfile');
function isJson(item) {
    item = typeof item !== "string"
        ? JSON.stringify(item)
        : item;

    try {
        item = JSON.parse(item);
    } catch (e) {
        return false;
    }

    if (typeof item === "object" && item !== null) {
        return true;
    }

    return false;
}

function isURL(str) {
  var pattern = new RegExp('^(https?:\\/\\/)?'+ // protocol
  '((([a-z\\d]([a-z\\d-]*[a-z\\d])*)\\.?)+[a-z]{2,}|'+ // domain name
  '((\\d{1,3}\\.){3}\\d{1,3}))'+ // OR ip (v4) address
  '(\\:\\d+)?(\\/[-a-z\\d%_.~+]*)*'+ // port and path
  '(\\?[;&a-z\\d%_.~+=-]*)?'+ // query string
  '(\\#[-a-z\\d_]*)?$','i'); // fragment locator
  return pattern.test(str);
}
```

```
var destAdjust= function(jsonUrl, jsonConfig) {
    try {
        if(!isJson(jsonUrl)) throw "the input should be in JSON format. Adjustment is cancelled.";
        for(var i=0;i<jsonUrl.length;i++)
        {
            if(!isURL(jsonUrl[i]["url"])) throw "the input should be URL. Adjustment is cancelled.";
        }
        if(JSON.stringify(jsonConfig.destUrl)!=JSON.stringify(jsonUrl))
        {
            jsonConfig.destUrl=jsonUrl;
            jsonFile.writeFileSync("./config.json",jsonConfig);
            console.log("Notice: adjustment completed.");
            nodeProcess.exit(1);
        }else{
            console.log("Notice: The input value is the same as the last one. Adjustment rejected.");
        }
    }catch(err)
    {
        console.log("Error: "+err);
    }
};
exports.sensorQtyAdjust = sensorQtyAdjust;
exports.deadlineAdjust = deadlineAdjust;
exports.destAdjust = destAdjust;
```

Progress
- API implementation: 100%
- Exception handling: 100%
- Module export: 100%

Task 8 – API Documentation

API name	Explanation	Input	Output
sensorQtyAdjust (sensorNumber, jsonConfig)	This function is used to adjust the number of sensors that the system has to handle.	int sensorNumber: number of sensors would be add to the synchronization process. json jsonConfig: current configuration.	The result of sensors adjustment execution.
deadlineAdjust (dueDate, jsonConfig)	This function is used to adjust the deadline of synchronization window.	int dueDate: synchronization window. json jsonConfig: current configuration.	The result of deadline adjustment execution.
destAdjust (jsonUrl, jsonConfig)	This function is used to adjust the list of url destinations. The synced data would be sent to every member of this list.	json jsonUrl: list of url destinations in json format that the synced data would be sent to. json jsonConfig: current configuration.	The result of destination adjustment execution.
coreSync (objLoki, mUserId, history, nDataSource, jsonConfig, badMerge, historyRecord, syncType)	This function is used sync the data.	lokijs objLoki: the reference to the buffer collection. int mUserId: the id of the user. lokijs history: the reference to the history collection. int nDataSource: number of current sensors. json jsonConfig: current configuration. int badMerge: flag for syncing data when exceeding the deadline. associated_array historyRecord: the record that is used to cover the missing data. int syncType: type of sync.	The result of synchronization execution.

Task 8 – API Documentation (cont'd)

113

API name	Explanation	Input	Output
serverPortAdjust (port, jsonConfig)	This function is used to adjust the port for starting the synchronization service.	int port: the port number to start the synchronization service. json jsonConfig: current configuration.	The result of port adjustment execution.
boolean isJson (item)	This function is used to verify a string is json or not.	string item: the item that needs to check for the json format.	True/false.
boolean isURL (str)	This function is used to verify a string is URL or not.	string str: the item that needs to check for the URL validation.	True/false.
syncStart (mergeDataBody, jsonConfig)	This function is used to send the synced data to the list of predefined destinations (URLs).	associated_array mergeDataBody: synced data. json jsonConfig: current configuration.	The result of sending synced data.

경희대학교 KYUNG HEE UNIVERSITY

Task 9 – Demo Harness

114

- **Installation Guide**
- Install Node JS (minimum version: 5.8.0)
- Install following packages using npm command (npm install):
  - 'express' (minimum version: 4.14)
  - 'lokijs' (minimum version: 1.4.1)
  - 'body-parser' (minimum version: 1.15.2)
  - 'requestify' (minimum version: 0.2.3)
  - 'fs' (minimum version: 0.0.1)
  - 'util' (minimum version: 0.10.3)
  - 'process' (minimum version: 0.11.9)
  - 'jsonfile' (minimum version: 2.3.1)
  - 'request' (minimum version: 2.76.0)
  - 'multer' (minimum version: 1.2.0)
- Put the source files all together in the same folder.
- Start synchronization server by using this command: node sync_code.js

경희대학교 KYUNG HEE UNIVERSITY

Task 9 – Demo Harness (cont'd)

11
5

- **Working Test**
- There are two kinds of tests for demonstration. The first one is the normal test for raw sensors data. The procedure for this test is described as follows:
- For testing purpose, start two more sink services, namely 'rev_server.js' and 'rev_server2.js'. These services are used to receive the synced data from synchronization services. The received data would be print out for confirmation purpose.
- Start the 'simulation.html' from your browser. This html file sends a number of simulated sensors data to the synchronization service (sync_code.js).
- If everything goes right, you can see the synced data coming to sink services ('rev_server.js' and 'rev_server2.js').
-
- The second test is multimedia test. This test is used to check whether the synchronization services can well handle the image/audio/video resources or not. The procedure for this test is described as follows:
- Create 'uploads' folder in the same level with the file 'sync_code.js'. This step can be skipped if the aforementioned folder is existed.
- Start 'fileUpload.html' from your browser.
- In 'fileUpload.html' page, choose an arbitrary image/audio/video to upload.
- If everything goes right, you can see the synced data coming to sink services ('rev_server.js' and 'rev_server2.js').

경희대학교
KYUNG HEE UNIVERSITY

Task :10 - Open Source Release - License

11
6

❑ Component and utilized open source library
- 'express' (minimum version: 4.14): for receiving requests and creating routes.
- 'lokijs' (minimum version: 1.4.1): for storing/buffering the data and maintaining the history records.
- 'body-parser' (minimum version: 1.15.2): for getting the parameter in each request.
- 'fs' (minimum version: 0.0.1): for dealing with file system.
- 'util' (minimum version: 0.10.3): for inspecting the structure of complex variables/arrays.
- 'process' (minimum version: 0.11.9): for checking the existence of process.
- 'jsonfile' (minimum version: 2.3.1): for dealing with json file.
- 'request' (minimum version: 2.76.0): for sending data to the destinations.
- 'multer' (minimum version: 1.2.0): for receiving the video/audio/image.

❑ Utilization of library
- All of open source libraries are kept as external source code.
- No need to update the library as long as satisfying the minimum version.

경희대학교
KYUNG HEE UNIVERSITY

Task :10 Open Source Release - License

117

❑ Open source license of the library, with key features of the license

- MIT
  - Feature: the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the software.
  - Affected libraries: Node JS, 'express', 'lokijs', 'body-parser', 'util', 'process', 'jsonfile', 'multer'.
- ISC
  - Feature: similar to MIT, but without "Berne convention".
  - Affected library: 'fs'.
- Apache 2.0
  - Feature: in every licensed file changed, a notification must be added stating that changes have been made to that file.
  - Affected library: 'request'.

경희대학교
KYUNG HEE UNIVERSITY

118

Question and issue report
please send to mao.bui@khu.ac.kr.

경희대학교
KYUNG HEE UNIVERSITY

Mining Minds

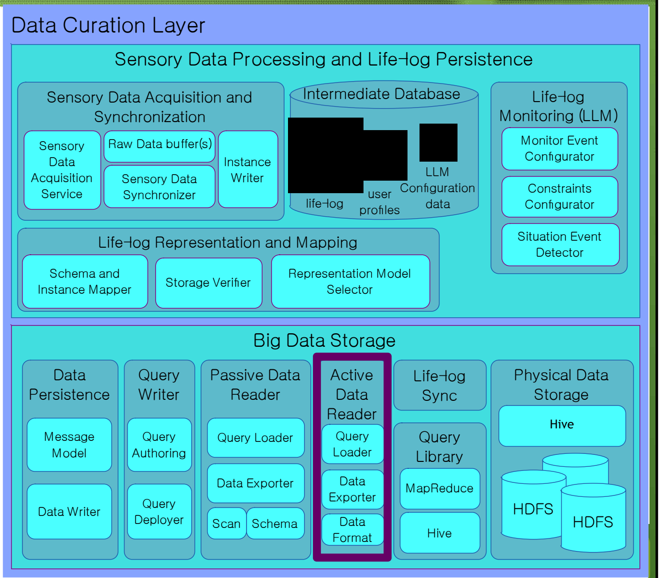
Active Data Reader

Open Source MMV3.0

Introduction – Open Source Detail (ADR)

12
0

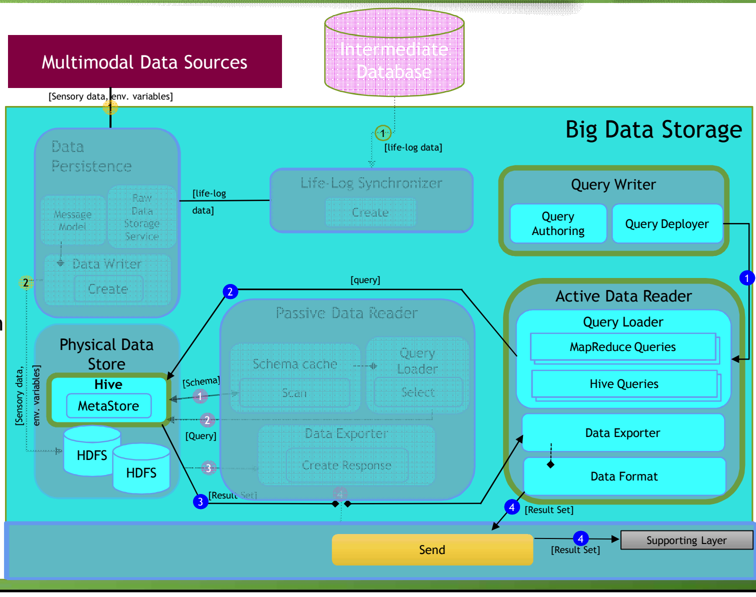
- Active Data Reader (ADR):
  - For Analytics and Visualization, SL required real-time data read based on provided parameters.
  - Support for Visualization and Analytics



Active Data Reader (ADR) – Conceptual View

Online Data Request for Data Visualization and Analytics

1. Data read request is generated by Visualization and Analytics Components of SL
2. Active data reader selects the particular query depending upon the query parameters
3. Selected Query is sent to Hive for execution
4. Required data is returned as a result set to Data exporter
5. Result-set is converted into data message per defined data format

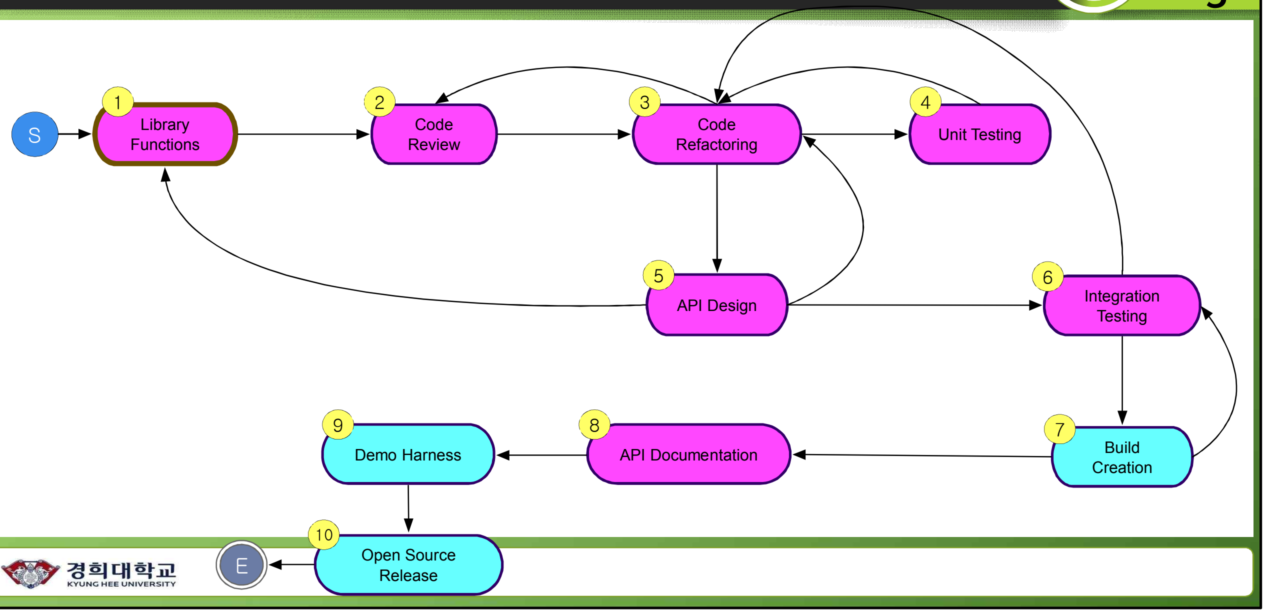


Outline

- Open Source Roadmap
  - Library Function
  - Class Diagram
  - Refactoring
  - Unit Testing
  - Integration Testing
  - API Design
  - Build Creation
  - API Documentation
  - Demo Harness
  - Open Source Release

Open Source Release Process - Roadmap

123



경희대학교 KYUNG HEE UNIVERSITY

Task :1 Open Source - Libraries Summary

124

Categorization of Libraries in our Component – Active Data Reader (ADR)

경희대학교 KYUNG HEE UNIVERSITY

Task :1 Open Source Libraries - Detail

125

Library	API Details	License
Hive Thrift Server API	Provide a JDBC based Connections to connect the Hive and Hadoop	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
Gson API	Used to convert Java Objects into their JSON representation and JSON to Java Objects	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
Spring MVC	Used as a Library to Provider a REST based services on the top of the Hadoop.	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
log4j	Track the logging to the Hive Connection	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
Apache Hive	Provides the SQL based operations on the top of the Hadoop	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
Hadoop Common Logging	Provide the logging support to the Apache Hive.	Apache 2.0 http://www.apache.org/licenses/LICENSE-2.0.txt
Junit API	JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks	Eclipse Public License 1.0 http://www.eclipse.org/legal/epl-v10.html
REST ASSURED	Provide the REST based end points testing.	Eclipse Public License - v 1.0

경희대학교 KYUNG HEE UNIVERSITY

Task :1 Open Source Libraries – Features

126

Comparison	Apache v 2.0	Eclipse Public License
Distribute: Must distribute licence with binary or source	√ ☺	√ ☺
Modify: There has to be A **notification** to changed files	√ ☺	√ ☺
Source Code: Any change must distributed in source form	√ ☺	√ ☺
Sublicense: All derivate work must be under the same license	√ ☺	√ ☺
Private use: Must show license when run from command line	√ ☺	√ ☺
Open Source: Non derivate works can have different license	√ ☺	√ ☺
Any Country: May exclude countries where there is A contradiction with **patent** in that country	√ ☺	√ ☺

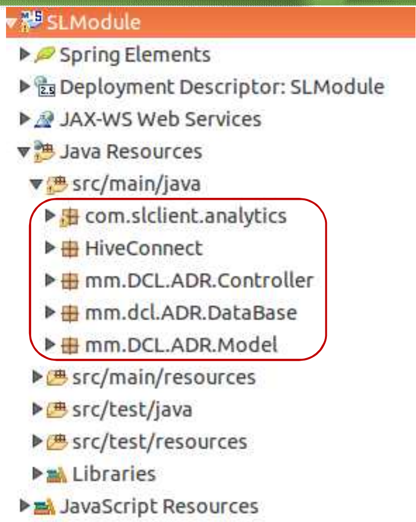
☺ For the Eclipse / Apache Public License there is no restriction to use the Junit. But the restriction applies while using it commercial purposes.

Task :1 Open Source - Library Function

127

- Library Function Identification
  - Packages Categorization
    - mm.DCL.ADR.Controller
      - Code to accept the request from the REST
    - mm.DCL.ADR.Model
      - Create a pattern with respect to the tables
    - mm.DCL.ADR.Database
      - Connect to the Apache Hive

Identification of Library Functions

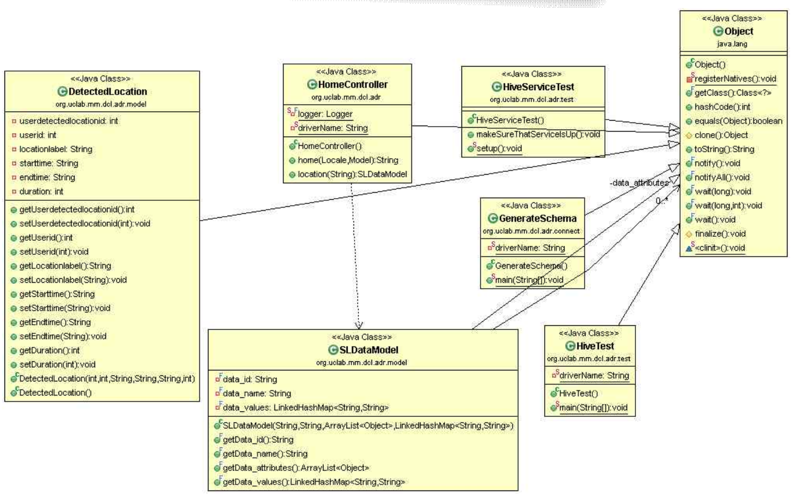


경희대학교 KYUNG HEE UNIVERSITY

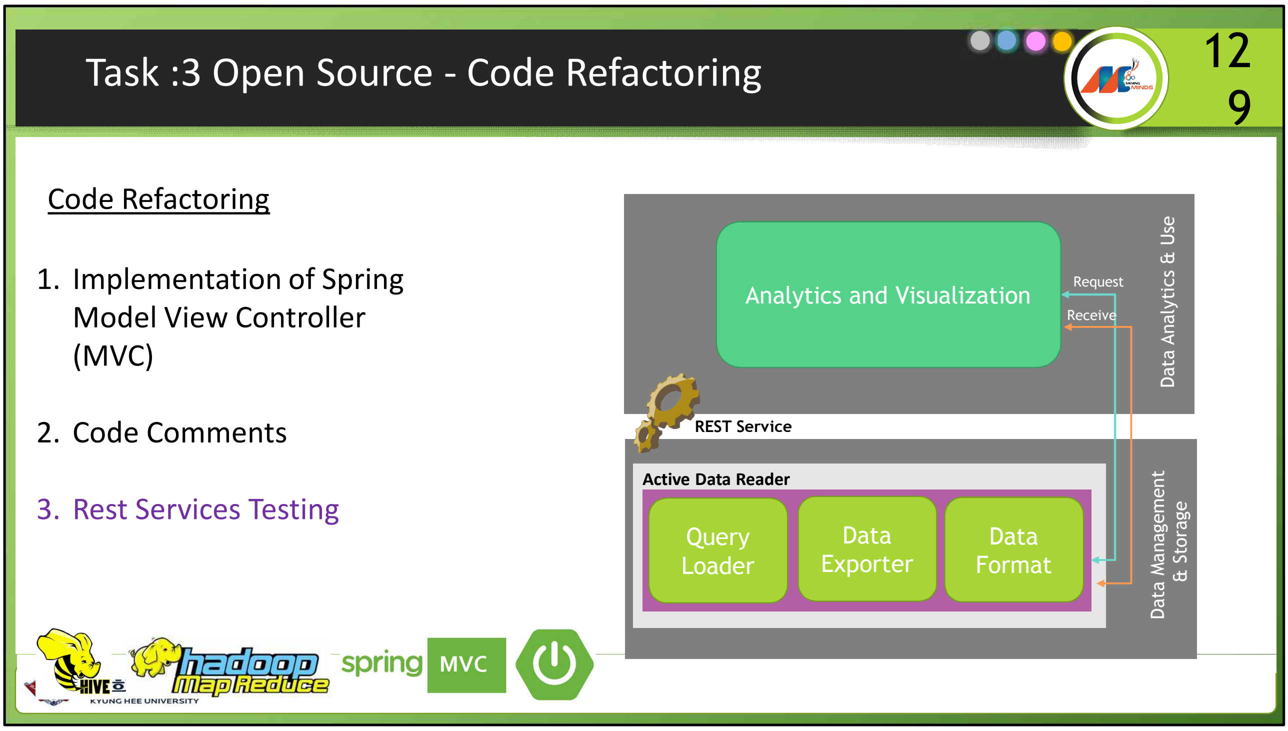
Task :2 Code Review- Active Data Reader (ADR)

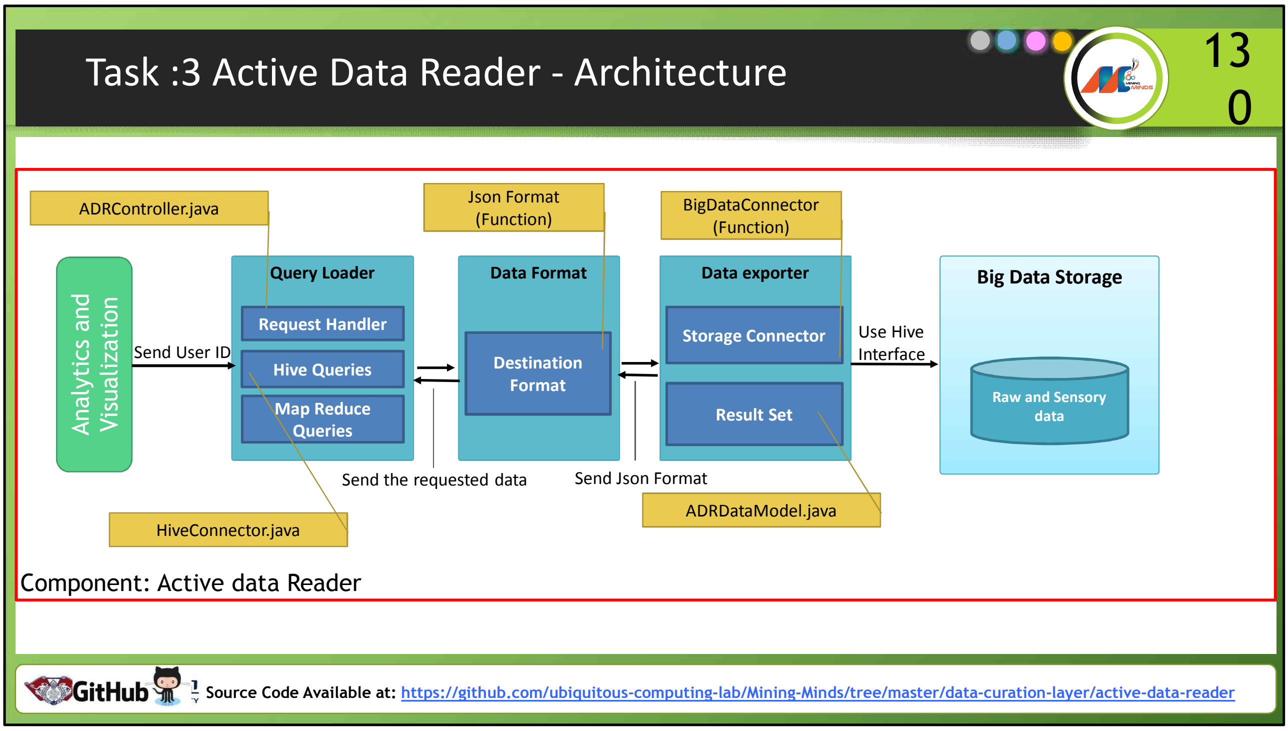
128

- **Class Diagram**
  - **Packages Categorization**
    - mm.DCL.ADR.Controller
      - Code to accept the request from the REST
    - mm.DCL.ADR.Model
      - Create a pattern with respect to the tables
    - mm.DCL.ADR.Database
      - Connect to the Apache Hive
  - **Some modification in Code pattern needed.**
    - Spring (Model – View – Controller)



경희대학교 KYUNG HEE UNIVERSITY


Task :3 Open Source - Code Refactoring
12
9
Code Refactoring
1. Implementation of Spring Model View Controller (MVC)
2. Code Comments
3. Rest Services Testing
Analytics and Visualization
Request
Receive
Data Analytics & Use
REST Service
Active Data Reader
Query Loader
Data Exporter
Data Format
Data Management & Storage
HIVE
hadoop MapReduce
spring
MVC
KYUNG HEE UNIVERSITY


Task :3 Active Data Reader - Architecture
13
0
ADRController.java
Json Format (Function)
BigDataConnector (Function)
Analytics and Visualization
Send User ID
Query Loader
Request Handler
Hive Queries
Map Reduce Queries
Data Format
Destination Format
Data exporter
Storage Connector
Result Set
Use Hive Interface
Big Data Storage
Raw and Sensory data
Send the requested data
Send Json Format
ADRDataModel.java
HiveConnector.java
Component: Active data Reader
GitHub
Source Code Available at: https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

Task :3 Refactored Model

13
1

Package mm.DCL.ADR.Model

Create a pattern with respect to the Hive.

Class Summary

Class	Description
DataModel	This class will handles the request from the rest services
CreateTables	This class will create a tables and stored raw sensory data.

Method Summary

Methods

Modifier and Type	Method and Description
com.slclient.analytics.SCLDataModel	greeting(String userid)
String	home(Locale locale, org.springframework.ui.Model model) Simply selects the home view to render by returning its name.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Example Code

```
/**
 * Get the userid and return the data from Hive Table
 */
public DataModel(String data_id, String data_name, ArrayList<Object> data_attributes,
                 LinkedHashMap<String, String> data_values

 String line = "incoming userid = " + userid;

             System.out.println(line);

             return new SCLDataModel(data_id, data_name, data_attributes,
```

data_ 경희대학교 KYUNG HEE UNIVERSITY

Task :3 Refactored Model

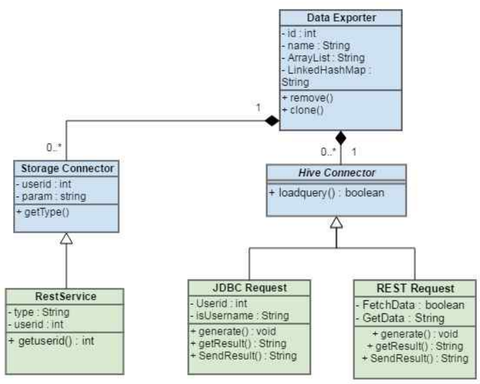
13
2

Package mm.DCL.ADR.Model

Create a pattern with respect to the Hive.

Class Summary

Class	Description
DataModel	This class will handles the request from the rest services
CreateTables	This class will create a tables and stored raw sensory data.



Example Code

```
/**
 * Get the userid and return the data from Hive Table
 */
public DataModel(String data_id, String data_name, ArrayList<Object> data_attributes,
                 LinkedHashMap<String, String> data_values

 String line = "incoming userid = " + userid;

             System.out.println(line);

             return new SCLDataModel(data_id, data_name, data_attributes,
```

data_ 경희대학교 KYUNG HEE UNIVERSITY

Task :3 Refactored Controller

133

Package mm.DCL.ADR.Controller

Handles the request coming from the REST Services

Class Summary

Class	Description
ADRController	This class will handles the request from the rest services
Hive Connector	This class will connect to the Hive to query

Example Code

```
/**
 * Handles requests for data Visualization.
 */
@Controller

public class ADRController {

private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
private static String driverName = "org.apache.hive.jdbc.HiveDriver";
```

Method Summary

Methods

Modifier and Type	Method and Description
static void	main(java.lang.String[] args)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Method Summary

Methods

Modifier and Type	Method and Description
com.slclient.analytics.SCLDataModel	greeting(String userid)
String	home(Locale locale, org.springframework.ui.Model model) Simply selects the home view to render by returning its name.

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

경희대학교
KYUNG HEE UNIVERSITY

Task :3 Refactored Controller

134

Package mm.DCL.ADR.Controller

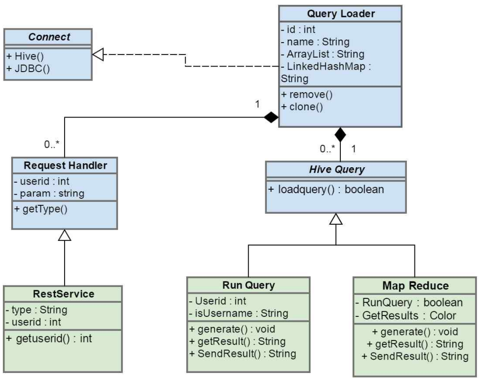
Handles the request coming from the REST Services

Class Summary

Class	Description
ADRController	This class will handles the request from the rest services
Hive Connector	This class will connect to the Hive to query

Example Code

```
/**
 * Handles requests for data Visualization.
 */
@Controller

public class ADRController {

private static final Logger logger = LoggerFactory.getLogger(HomeController.class);
private static String driverName = "org.apache.hive.jdbc.HiveDriver";
```



경희대학교
KYUNG HEE UNIVERSITY

Task :4 Unit Testing – Active Data Reader

13
5

- **Unit Test**
  - Assured REST API Communication

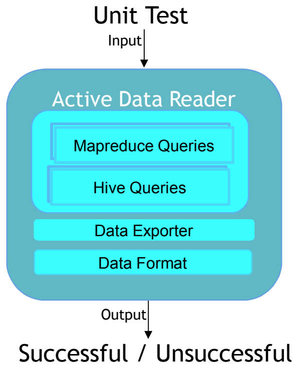
- **Unit Test Goal:**
  - The aim of the test is to check the **REST communication** between the **Supporting Layer** (SL) and **Active Data Reader (ADR).**

  - **Example of the Unit Test Code:**

Getting and parsing a response body:

```
// Example with JsonPath String json =
get("/ActiveDataReader").asString() List<String> UserIds =
from(json).get("getuserId"); // Example with XmlPath String xml =
post("/Analytics").andReturn().body().asString() Node category =
from(json).get("getAllUserID[0]");
```

  - Result: Test Pass



REST-assured
Open Source Test Suite
https://github.com/rest-assured/rest-assured

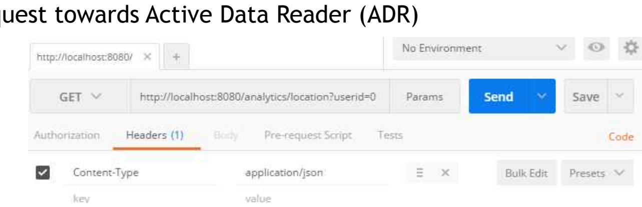
경희대학교 KYUNG HEE UNIVERSITY

Task :5 API Design

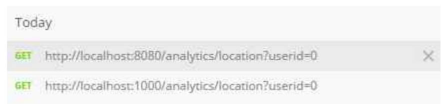
13
6

1. Api Address: http://localhost:8080/analytics/

2. Use HTTP Rest Client to send the Request towards Active Data Reader (ADR)



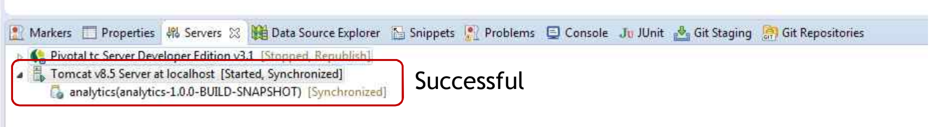
3. Successfully send the json response



경희대학교 KYUNG HEE UNIVERSITY

Task :6 Integration Testing

13
7



Successful

Upon completion of unit testing, the units or modules are integrated which gives raise to integration testing. The purpose of this, is to verify the functional, performance, and reliability between the modules that are integrated.

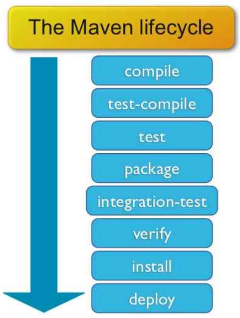
Successful

경희대학교
KYUNG HEE UNIVERSITY

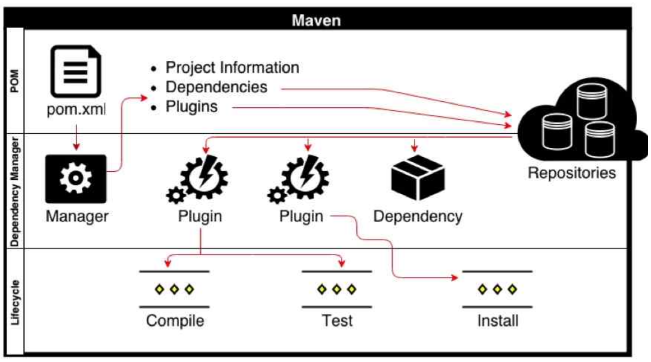
Task :7 Active Data Reader - Build Creation

13
8

A Build Lifecycle is Made Up of Phases



- **Component:** Active Data Reader
  - Environment: Maven & Jax-Rs



Resources

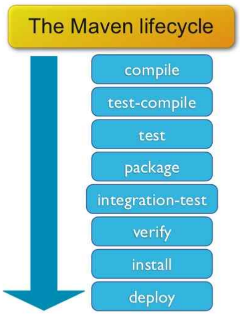
Build Creation: https://maven.apache.org
Rest API: https://jax-rs-spec.java.net/
Version: https://subversion.apache.org/

GitHub **Source Code Available at:** https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

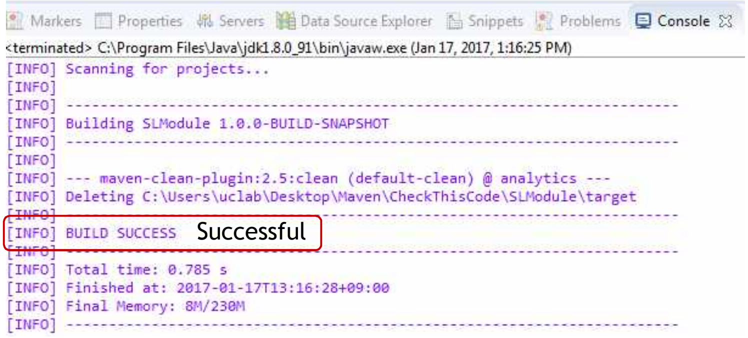
Task :7 Active Data Reader - Build Creation

13
9

A Build Lifecycle is Made Up of Phases



- **Component:** Active Data Reader
  - Environment: Maven & Jax-Rs



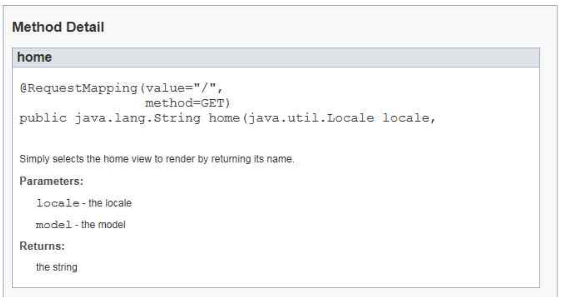
Resources

Build Creation: https://maven.apache.org
Rest API: https://jax-rs-spec.java.net/
Version: https://subversion.apache.org/

GitHub Source Code Available at: https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

Task :8 API Documentation

14
0



The Java Platform API Specification is defined by the documentation comments in the source code and any documents marked as specifications reachable from those comments.

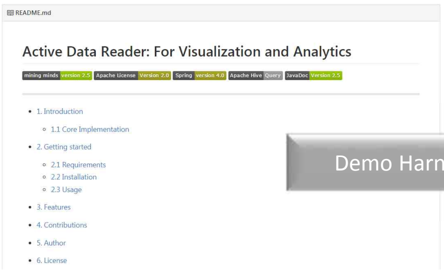
Javadoc Available at: https://ubiquitous-computing-lab.github.io/Mining-Minds/doc/dcl-doc/ative-data-reader/doc/index.html

KYUNG HEE UNIVERSITY

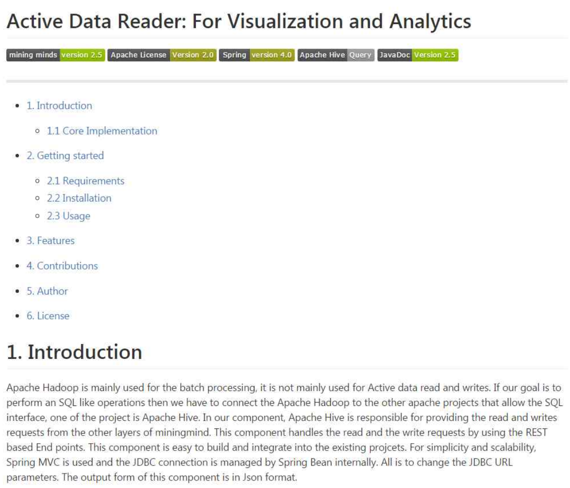
Task :9 Demo Harness

141

The installation and usage information available as demo harness on GitHub.



Demo Harness

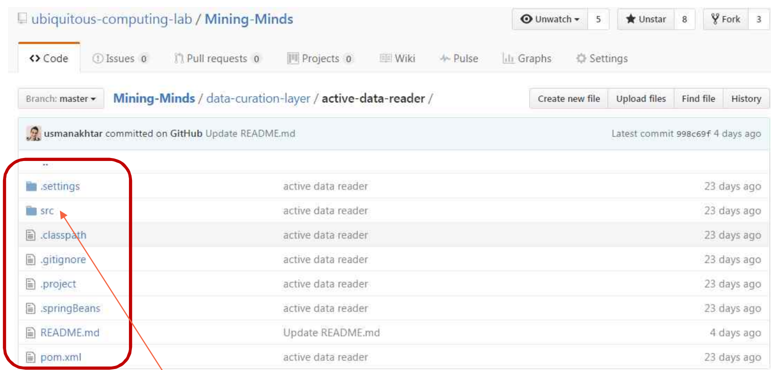


경희대학교 KYUNG HEE UNIVERSITY

Demo Link: https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

Task :10 Open Source Release

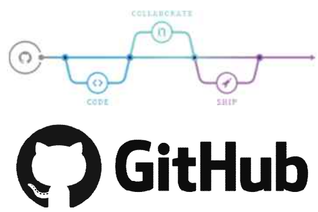
142



Source Code Folder Available at GitHub

The source code is released under the following License:

- The Apache License, Version 2.0 (the "License")
- The copy of License may be obtained at http://www.apache.org/licenses/LICENSE-2.0

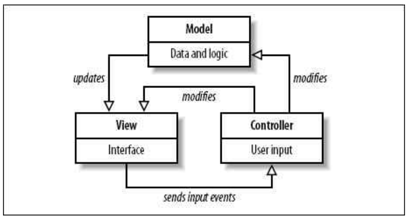


경희대학교 KYUNG HEE UNIVERSITY

Code Refactoring - Summary

143

- **Code Refactoring Summary**
  - Packages Categorization
    - Code style: Follows Google Code Style https://google.github.io/styleguide/javaguide.html
    - Modified the methods and the classes
  - Design Pattern: Model View Controller (MVC)



  - Maven for Dependencies

Apache maven

Method Summary

Methods

Modifier and Type	Method and Description
java.util.ArrayList<java.lang.Object>	getData_attributes()
java.lang.String	getData_id()
java.lang.String	getData_name()
java.util.LinkedHashMap<java.lang.String,java.lang.String>	getData_values()

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Active Data Reader API:

Java API for RESTful Web Services (**JAX-RS**)

JAX-RS

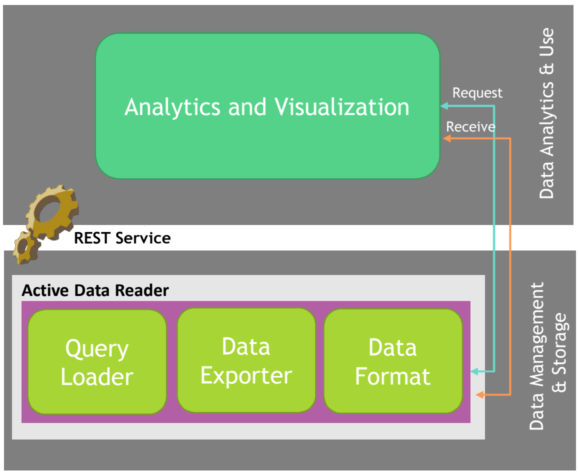
경희대학교 KYUNG HEE UNIVERSITY

Active Data Reader Implementation - Summary

144

Provide Data for Visualization

1. Built for REST Services (request handle).
2. Support for Visualization and Analytics
3. Highly scalable and interactive

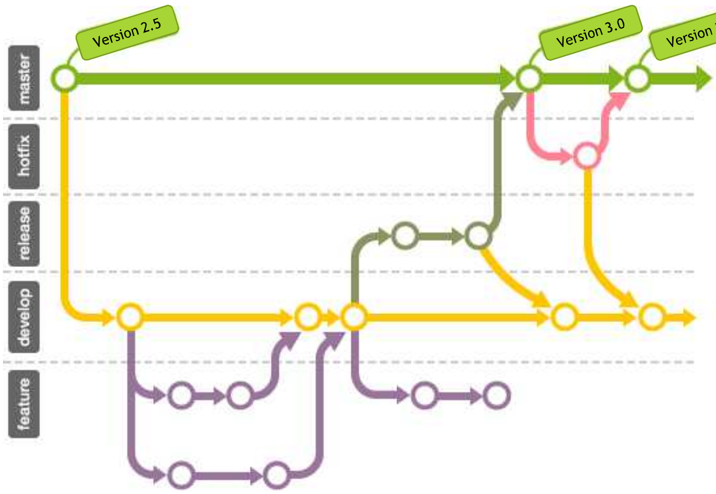


HIVE hadoop MapReduce

KYUNG

Active Data Reader – Version Control.

14
5



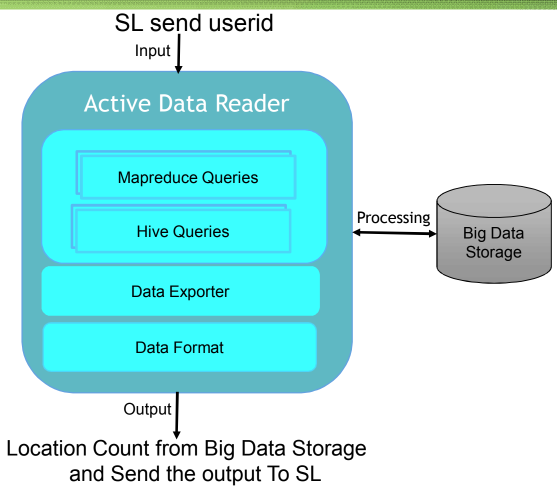
Master
Codebase residing in the master branch is considered to be production-ready.

Develop
Codebase residing in the develop branch is used in your team's day to day development.

GitHub Source Code Available at: https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

MMv3.0 Active Data Reader – Summary

14
6



- **Output**
  - ✓ User Location from the Big data (Give the SL on label counts after searching the records the Big Storage)
    - ✓ Using REST SERVICES
- **Input**
  - ✓ User ID to calculate the Location
- **Implementation Module**
  - ✓ Active Data Reader

경희대학교 KYUNG HEE UNIVERSITY

Mining Minds

ICL

Open Source MMV3.0

경희대학교 KYUNG HEE UNIVERSITY KITECH 한국생산기술연구원 POSTECH 포항공과대학교 UNIVERSITY of TASMANIA 녹십자헬스케어 TAPACROSS U2 system User & Usability system KOCSA 한국컨설팅서비스협회

Mining Minds

LLCA Open Source

Information Curation Layer MMV-3.0

경희대학교 KYUNG HEE UNIVERSITY KITECH 한국생산기술연구원 POSTECH 포항공과대학교 UNIVERSITY of TASMANIA 녹십자헬스케어 TAPACROSS U2 system User & Usability system KOCSA 한국컨설팅서비스협회

LLCA Libraries

2

LLCA libraries

- 10 libraries

List of libraries
- Javax.servlet, com.sun.jersey, org.json, Apache Commons, Hamcrest, RestAssured, Log4j, slf4jAPI, Weka, Xerces2

List of licences
- CDDL, GPL 2.0, BSD, EPL, Apach 2.0, MIT

Compatibility issues against Apache 2.0

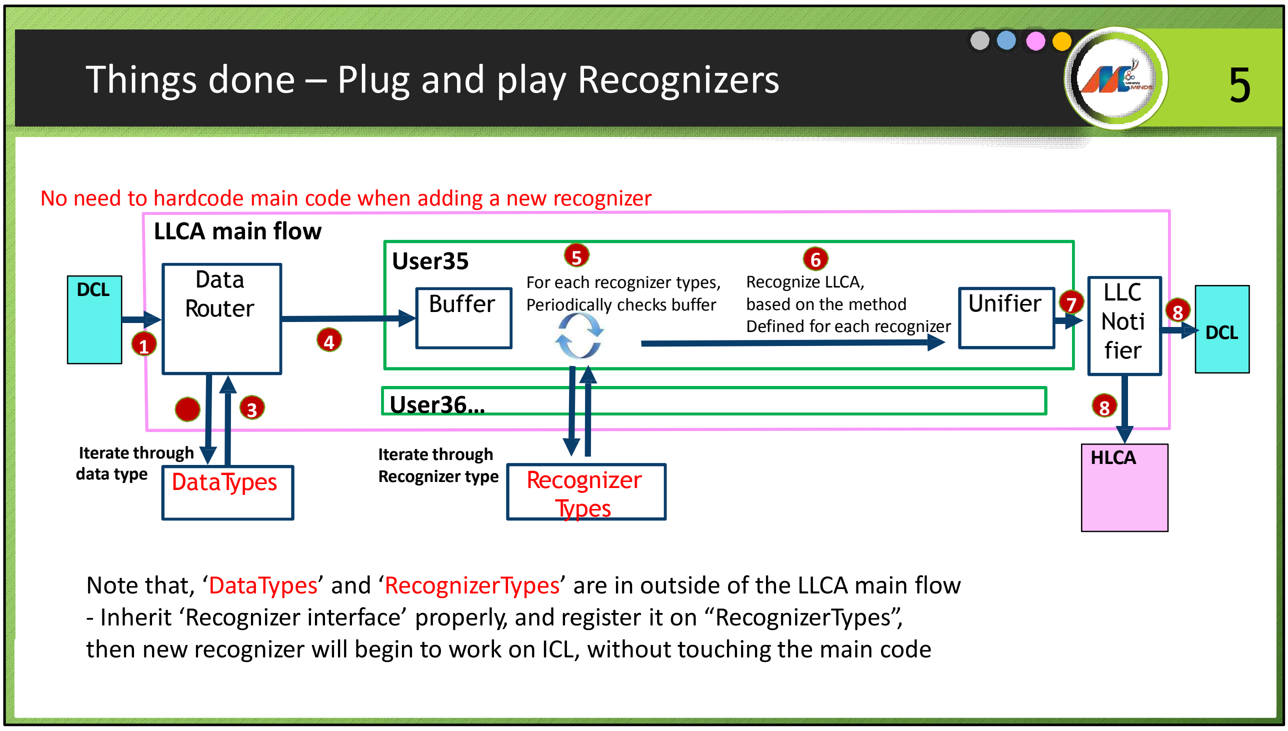
3

http://www.apache.org/legal/resolved.html#category-a

1. Apache 2.0 licensed project cannot use GPL 2.0
   - however, GPL 2.0 licensed library in ICL_LLCA (javax.servlet, jersey) can select also CDDL license (We need to label the inclusion of CDDL licenses though)

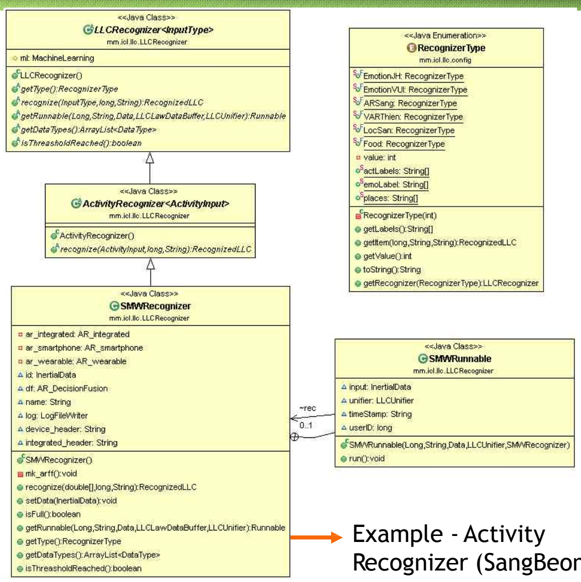
2. Compatible licenses – EPL, MIT, BSD


Mining Minds
ICL Server & Android Data Collection App
Dong Uk Kang
경희대학교 KYUNG HEE UNIVERSITY
KITECH 한국생산기술연구원
POSTECH 포항공과대학교
UNIVERSITY of TASMANIA
녹십자헬스케어
TAPACROSS
U2 system User & Usability system
KOCSA 한국컨설팅서비스협회


Things done – Plug and play Recognizers
5
No need to hardcode main code when adding a new recognizer
LLCA main flow
DCL
Data Router
User35
Buffer
For each recognizer types, Periodically checks buffer
Recognize LLCA, based on the method Defined for each recognizer
Unifier
LLC Notifier
DCL
User36...
HLCA
Iterate through data type
DataTypes
Iterate through Recognizer type
Recognizer Types
1
3
4
5
6
7
8
8
Note that, 'DataTypes' and 'RecognizerTypes' are in outside of the LLCA main flow
- Inherit 'Recognizer interface' properly, and register it on "RecognizerTypes",
then new recognizer will begin to work on ICL, without touching the main code

Plug and play Interface With OOP principle

6



Example - Activity Recognizer (SangBeom's)

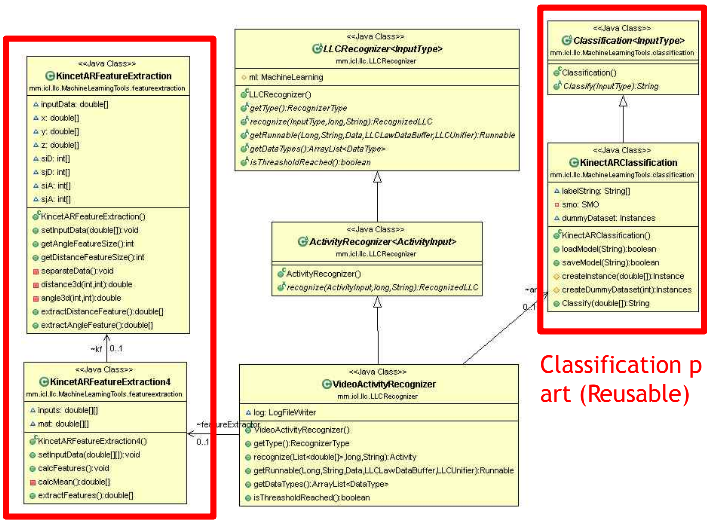
To add a new recognizer,

1. Inherit LLCRecognizer interface
2. Implement abstract methods, which will be functioning on LLCA main flow
   Otherwise new recognizer can't be instantiated
3. Implements Runnable, which will be on a thread of the LLCA main flow to recognize LLC
4. Register your recognizer to RecognizerType(Enum)

If finished, new recognizer will work on ICL, without touching the main code

Recognizers Interface – Cont

7



Classification part (Reusable)

FeatureExtraction part (Reusable)

And while adding a new recognizer,

User can reuse the Segmentation / Feature extraction / Classification classes used by previous classifiers

Utilized Singleton pattern for the configuration class, since configurations do not have to be made many times

Code refactoring 1 – Plug & play recognizer

8

Utilized Polymorphism + Enum for Plug & play recognizers

As-is: Hard-coded recognition process
Every time new recognizer introduced
-> need to modify below code

```
VideoActivityRecognizer vr = new VideoActivityRecognizer(userID);
GPSLocationRecognizer gr = new GPSLocationRecognizer(userID);
AudioEmotionRecognizer emr = new AudioEmotionRecognizer(ext, null,

vr.recognize(input, userID, timeStamp);
gr.recognize(input, userID, timeStamp)
emr.recognize(segmented, userID, timeStamp)
```

Too add recognizer, just
1. inherit the interface,
2. implement abstract functions, and
3. register the recognizer
No need to modify below main process

Current: Registration based, plug & play

```
for(RecognizerType rec: RecognizerType.values()){ //for each registered recognizers
    recognizers.get(rec.getValue()).ConsumeData(it, data); //consume the current sensor data

    if(recognizers.get(rec.getValue()).isThreasholdReached()) //if threashold reached, call the recognizer
    {
        es.execute(recognizers.get(rec.getValue()).getRunnable(userID, curTime, unifier));
    }
}
```

Code refactoring 2 – Exception handling

9

Utilized log4j, and proper exception handling

```
    }
    threads.remove(this);
    return;
} catch (ClassNotFoundException e) {
    logger.error("error at reading data class (smartphone) from socket inputstream");
    logger.error("serializable data class does not matched. Make sure to use right class and "
            + "right sererialization number");
    logger.error(e);
}
```

Using log4j, provides crucial information for debugging

```
if (!contextMng.correctInitialization()) {
    logger.fatal("Failed to initialize ContextOntologyManager.");
    logger.fatal("Check whether valid URI is provided for the ContextOntologyManager.");
    throw new IllegalStateException(); // this exception will be handled at above level
}
```

We need to stop HLC server when this error occurs, but we can't do it on this level
-> we pass it to "upper level"

Comments for every public functions, and for crucial private function

경희대학교
KYUNG HEE UNIVERSITY

Code refactoring 3 – Design patterns

10

Utilized Factory pattern for classes with polymorphism,
Singleton pattern to prevent memory wasting

```
public User getUser(long userID) {
    if (MainServlet.getUserMap().get(userID) == null) {
        Logger.info("User null, creating new user");
        User user = UserFactory.getUser(userID, es);
        MainServlet.getUserMap().put(userID, user);
    }
    return MainServlet.getUserMap().get(userID);
}
```

→ Calls factory function to get instance

```
private static HLCServer singleton = new HLCServer();
public static HLCServer getHLCServer(){
    return singleton;
}
```

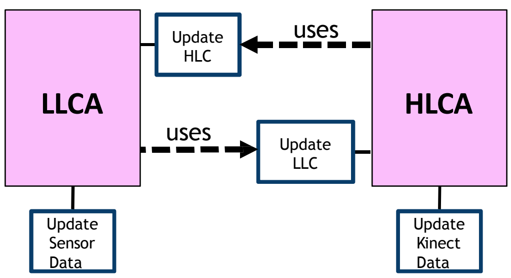
→ Singleton Pattern, to prevent the initialization of two HLC server

경희대학교 KYUNG HEE UNIVERSITY

API Design

11

Rest APIs required for co-operation of LLCA and HLCA



API Name	Input Param	return	Side-effect
UpdateHLC	userID, Timestamp, HLC	Update status	LLCA Receives HLC
UpdateLLC	userID, Timestamp, HLC	Update status	HLCA stores LLC in the context DB
Update SensorData	Sensor Data (JSON form)	Update status	Update Sensor data Of LLCA
Update KinectData	Kinect Data (JSON form)	Update status	Insert Kinect Data into Data buffer

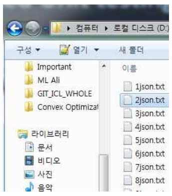
Automated Unit testing

12

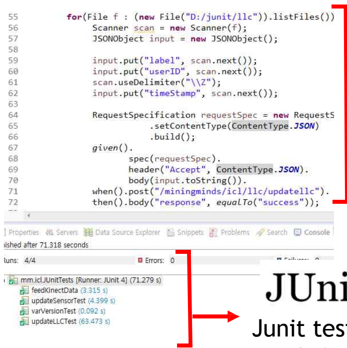
Utilized RestAssured and Junit for the automated testing of 4 endpoints

1. Collected real sensory data

2. More than 100+ test cases in files



3. Automated Test



REST-assured

Used RestAssured to validate rest endpoints

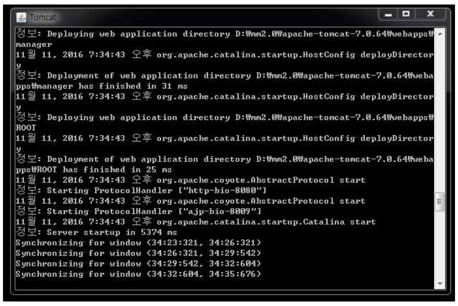
JUnit

Junit tests all successful (Each function includes several test cases)

Build Creation

13

Removed all absolute path dependency, created WAR distribution of ICL



ICL running tested on our laptop

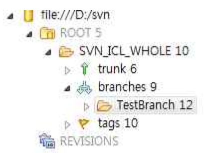
- Only require Java 8 and Tomcat server to run ICL in any machines, without any development environment

Introducing SVN

14

Based on Open source plan, tested SVN locally

Local SVN Repository



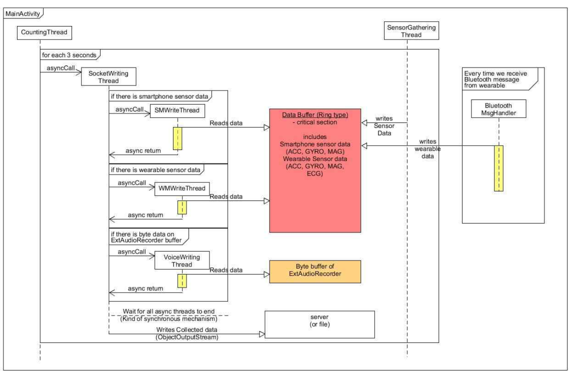
When provided SVN server, will switch to that one immediately

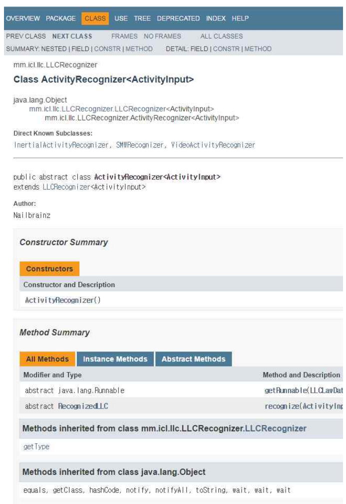
Future plan for Open source - Documentation

15

Documentation (95%)

Documentation for Javadoc, for the open source participants





Mining Minds

Information Curation Layer
Inertial based Recognizer
Open Source Progress

Taeho Hur

경희대학교 KYUNG HEE UNIVERSITY · KITECH 한국생산기술연구원 · POSTECH 포항공과대학교 · UNIVERSITY of TASMANIA · 녹십자헬스케어 · TAPACROSS · U2 system User & Usability system · KOCSA 한국컨설팅서비스협회

Inertial based AR Open Source Code Progress – Task 1

17

- Library Function Identification
  - Packages Categorization
    - AR_DecisionFusion
      - Decision fusion based on each classifier result
    - AR_FeatureExtractor
      - Extract features from inertial sensory data
    - AR_makeARFF
      - Make ARFF file for classification
    - AR_Reasoning
      - Classifies activity based on smartphone, smartwatch and integrated individually
    - InertialActivityRecognizer
      - Activity Recognition execution code

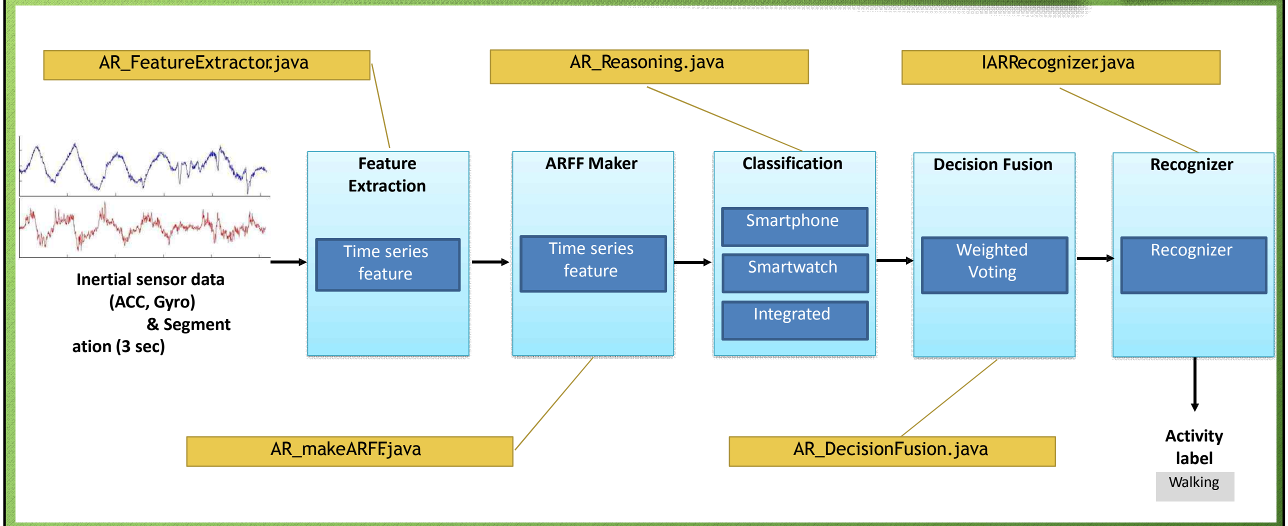
org.uclab.mm.icl.llc.AR.iar
- AR_DecisionFusion.java
- AR_FeatureExtractor.java
- AR_makeARFF.java
- AR_Reasoning.java

org.uclab.mm.icl.llc.LLCRecognizer
- InertialActivityRecognizer.java

경희대학교 KYUNG HEE UNIVERSITY

Architecture

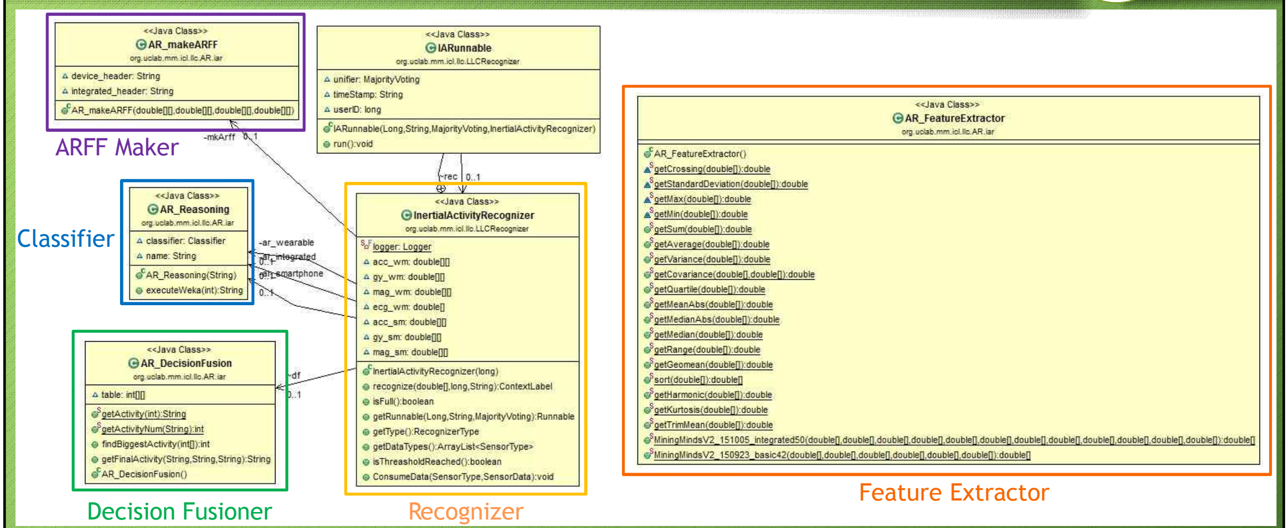
18



경희대학교
KYUNG HEE UNIVERSITY

Inertial based AR Open Source Code Progress – Task 2

19



경희대학교
KYUNG HEE UNIVERSITY

Inertial based AR Open Source Code Progress – Task 3

20

Package org.uclab.mm.icl.llc.AR.iar

Class Summary

Class	Description
AR_DecisionFusion	This class is to fuse the decision among smartphone, smartwatch and integrated AR model
AR_FeatureExtractor	This class is for extracting features.
AR_makeARFF	This class makes ARFF file for classification.
AR_Reasoning	This class loads the smartphone, smartwatch and integrated AR model.

Package org.uclab.mm.icl.llc.LLCRecognizer

Class Summary

Class	Description
InertialActivityRecognizer	This class processes the inertial sensor based activity recognition

경희대학교 KYUNG HEE UNIVERSITY

Inertial based AR Open Source Code Progress – Task 3

21

InertialActivityRecognizer

Constructor Summary

Constructors

Constructor and Description
InertialActivityRecognizer(long userID) This function loads the integrated, smartphone and smartwatch AR model

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	ConsumeData(SensorType it, SensorData obj) This class loads the sensor data
java.util.ArrayList<SensorType>	getDataTypes()
java.lang.Runnable	getRunnable(java.lang.Long userID, java.lang.String timeStamp, MajorityVoting unifier) This will proceed the recognition process
RecognizerType	getType() Returns the RecognizerType of the current instance
boolean	isFull()
boolean	isThreasholdReached()
ContextLabel	recognize(double[] input, long userID, java.lang.String timeStamp) This function proceeds the inertial based activity reasoning

Example Code

```java
/**
 * This class processes the inertial sensor based activity recognition
 * @author Tae ho Hur
 *
 */
public class InertialActivityRecognizer extends LLCRecognizer<double[]>{
    private final static Logger logger = Logger.getLogger(InertialActivityRecognizer.class);
    private AR_Reasoning ar_integrated = null; // Integrated model instance for reasoning
    private AR_Reasoning ar_smartphone = null; // Smartphone model instance for reasoning
    private AR_Reasoning ar_wearable = null; // Wearable model instance for reasoning
    private AR_makeARFF mkArff = null; // Instance for making ARFF file

    AR_DecisionFusion df; // To store decision fusion result

    // Stores the sensor variables
    double acc_wm[][];
    double gy_wm[][];
    double mag_wm[][];
    double ecg_wm[];
    double acc_sm[][];
    double gy_sm[][];
    double mag_sm[][];
```

경희대학교 KYUNG HEE UNIVERSITY

Inertial based AR Open Source Code Progress – Task 3

22

FeatureExtractor

Constructor Summary

Constructors

Constructor and Description
AR_FeatureExtractor()

Method Summary

All Methods | Static Methods | Concrete Methods

Modifier and Type	Method and Description
static double	getAverage(double[] values) get average of array
static double	getCovariance(double[] values_A, double[] values_B) get covariance of two arrays
static double	getGeomean(double[] value) get geomean of array
static double	getHarmonic(double[] values) get harmonic value of array
static double	getKurtosis(double[] values) get kurtosis data of array
static double	getMeanAbs(double[] values) get mean value of abs
static double	getMedian(double[] values) get median value
static double	getMedianAbs(double[] values) get median value of abs
static double	getQuartile(double[] values) get quartile value
static double	getRange(double[] value) get range of array
static double	getSum(double[] values) get sum of array data
static double	getTrimMean(double[] values)
static double	getVariance(double[] values) get variance of array
static double[]	MiningMindsV2_150923_basic42(double[] accX, double[] accY, double[] accZ, double[] gyroX, double[] gyroY, double[] gyroZ) get each device's featured data(basic feature lists)
static double[]	MiningMindsV2_151005_integrated50(double[] SaccX, double[] SaccY, double[] SaccZ, double[] WaccX, double[] WaccY, double[] WaccZ, double[] SgyroX, double[] SgyroY, double[] SgyroZ, double[] WgyroX, double[] WgyroY, double[] WgyroZ) get integrated model's featured data(with 50 features)
static double[]	sort(double[] value) sort array

Example Code

```java
/**
 * This class is for extracting features.
 * It contains feature calculation functions and function to return feature list.
 *
 * @author Tae ho Hur
 *
 */
public class AR_FeatureExtractor {

    /**
     * get Zero Crossing
     *
     * @param double
     *            array(eg:accelerometer data)
     * @return number of zero crossing
     */
    static double getCrossing(double[] signal) {
        double crossing = 0;
        double mean = 0;
        mean = getAverage(signal);
        for (int i = 0; i < signal.length - 1; i++) {
            if ((signal[i] - mean) * (signal[i + 1] - mean) < 0)
                crossing += 1;
        }
        crossing /= signal.length;

        return crossing;
    }
```

경희대학교
KYUNG HEE UNIVERSITY

Inertial based AR Open Source Code Progress – Task 3

23

ARFFMaker

Constructor Summary

Constructors

Constructor and Description
AR_makeARFF(double[][] acc_wm, double[][] gy_wm, double[][] acc_sm, double[][] gy_sm)

Method Summary

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Detail

AR_makeARFF

```java
public AR_makeARFF(double[][] acc_wm,
                   double[][] gy_wm,
                   double[][] acc_sm,
                   double[][] gy_sm)
            throws java.io.IOException
```

Throws:

java.io.IOException

Example Code

```java
/**
 * This class makes ARFF file for classification.
 *
 * @author Tae ho Hur
 *
 */

public class AR_makeARFF {

    String device_header = "@relation sm_acc_gyro\n" + "@attribute accX_average REAL\n"
            + "@attribute accY_average REAL\n" + "@attribute accZ_average REAL\n" + "@attribute accX_max REAL\n"
            + "@attribute accY_max REAL\n" + "@attribute accZ_max REAL\n" + "@attribute accX_min REAL\n"
            + "@attribute accY_min REAL\n" + "@attribute accZ_min REAL\n" + "@attribute accX_deviation REAL\n"
            + "@attribute accY_deviation REAL\n" + "@attribute accZ_deviation REAL\n"
            + "@attribute accX_crossing REAL\n" + "@attribute accY_crossing REAL\n"
            + "@attribute accZ_crossing REAL\n" + "@attribute accX_quartile REAL\n"
            + "@attribute accY_quartile REAL\n" + "@attribute accZ_quartile REAL\n"
            + "@attribute accX_meanabs REAL\n" + "@attribute accY_meanabs REAL\n" + "@attribute accZ_meanabs REAL\n"
            + "@attribute gyroX_average REAL\n" + "@attribute gyroY_average REAL\n"
            + "@attribute gyroZ_average REAL\n" + "@attribute gyroX_max REAL\n" + "@attribute gyroY_max REAL\n"
            + "@attribute gyroZ_max REAL\n" + "@attribute gyroX_min REAL\n" + "@attribute gyroY_min REAL\n"
            + "@attribute gyroZ_min REAL\n" + "@attribute gyroX_deviation REAL\n"
            + "@attribute gyroY_deviation REAL\n" + "@attribute gyroZ_deviation REAL\n"
            + "@attribute gyroX_crossing REAL\n" + "@attribute gyroY_crossing REAL\n"
            + "@attribute gyroZ_crossing REAL\n" + "@attribute gyroX_quartile REAL\n"
            + "@attribute gyroY_quartile REAL\n" + "@attribute gyroZ_quartile REAL\n"
            + "@attribute gyroX_meanabs REAL\n" + "@attribute gyroY_meanabs REAL\n"
            + "@attribute gyroZ_meanabs REAL\n"
            + "@attribute action {Eating,Running,Sitting,Standing,Walking,Stretching,Sweeping,LyingDown}\n"
            + "@data\n";
```

경희대학교
KYUNG HEE UNIVERSITY

Inertial based AR Open Source Code Progress – Task 3

24

Classification

Constructor Summary

Constructors

Constructor and Description
AR_Reasoning(java.lang.String modelpath) This function loads the AR model

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.String	executeWeka(int modelno) This function executes the Weka to infer the activity and returns the activity label

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Example Code

```java
/**
 * This class loads the smartphone, smartwatch and integrated AR model. It also
 * executes the weka based on the AR model and returns the activity label
 *
 * @author Tae ho Hur
 *
 */
public class AR_Reasoning {
    Classifier classifier;

    String name;

    /**
     * This function loads the AR model
     *
     * @param modelpath
     *
     */
    public AR_Reasoning(String modelpath) {
        try {
            classifier = (Classifier) weka.core.SerializationHelper.read(modelpath);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
```

경희대학교
KYUNG HEE UNIVERSITY

Inertial based AR Open Source Code Progress – Task 3

25

DecisionFusion

Constructor Summary

Constructors

Constructor and Description
AR_DecisionFusion() This function allocates weight to each AR model's activity

Method Summary

All Methods | Static Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
int	findBiggestActivity(int[] arg) This function finds the activity with the most weight
static java.lang.String	getActivity(int input) This function returns the activity label
static int	getActivityNum(java.lang.String arg) This function returns the activity number
java.lang.String	getFinalActivity(java.lang.String in, java.lang.String sm, java.lang.String wm) This function returns the final concluded activity based on the weight

Methods inherited from class java.lang.Object
equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Example Code

```java
/**
 * This class is to fuse the decision among smartphone, smartwatch and integrated AR model
 *
 * @author Tae ho Hur
 *
 */

public class AR_DecisionFusion {
    int[][] table;  // Contains the list of activity

    /**
     * This function returns the activity label
     * @param input
     * @return
     */
    public static String getActivity(int input) {
        if (input == 0) {
            return "Eating";
        } else if (input == 1) {
            return "Running";
        } else if (input == 2) {
            return "Sitting";
        } else if (input == 3) {
            return "Standing";
        } else if (input == 4) {
            return "Walking";
        } else if (input == 5) {
            return "Stretching";
        } else if (input == 6) {
            return "Sweeping";
        } else if (input == 7) {
            return "LyingDown";
        } else
            return "UnidentifiedActivity";
    }
```

경희대학교
KYUNG HEE UNIVERSITY

Inertial based AR Open Source Code Progress – Task 3

26

```java
/**
 * get integrated model's featured data(with 20 features)
 * @param Smartphone's accelerometer dataX
 * @param Smartphone's accelerometer dataY
 * @param Smartphone's accelerometer dataZ
 * @param Smart watch's accelerometer dataX
 * @param Smart watch's accelerometer dataY
 * @param Smart watch's accelerometer dataZ
 * @param Smartphone's gyroscope dataX
 * @param Smartphone's gyroscope dataY
 * @param Smartphone's gyroscope dataZ
 * @param Smart watch's gyroscope dataX
 * @param Smart watch's gyroscope dataY
 * @param Smart watch's gyroscope dataZ
 * @return feature list.
 */
public static double[] MiningMindsV2_151005_integrated20(double[] SaccX, double[] SaccY, double[] SaccZ,
        double[] WaccX, double[] WaccY, double[] WaccZ, double[] SgyroX, double[] SgyroY, double[] SgyroZ,
        double[] WgyroX, double[] WgyroY, double[] WgyroZ) {

    double[] feature = new double[20];

    feature[0] = getRange(WgyroZ);
    feature[1] = getMax(SaccY);
    feature[2] = getRange(SaccZ);
    feature[3] = getStandardDeviation(SaccY);
    feature[4] = getStandardDeviation(SaccZ);
    feature[5] = getQuartile(SaccY);
    feature[6] = getQuartile(SaccZ);
    feature[7] = getMeanAbs(SaccY);
    feature[8] = getMeanAbs(SaccZ);
    feature[9] = getMedianAbs(SaccY);
```

Delete unused codes:
Multiple feature extraction functions for testing

```java
String in = null;
String sm = null;
String wm = null;
try{
    in = AR_DecisionFusion.getActivity(Integer.parseInt
    sm = AR_DecisionFusion.getActivity(Integer.parseInt
    wm = AR_DecisionFusion.getActivity(Integer.parseInt
```

```java
String integratedModel = null;
String smartphoneModel = null;
String wearableModel = null;
try{
    integratedModel = AR_DecisionFusion.getActivity(Integer.parseInt
    smartphoneModel = AR_DecisionFusion.getActivity(Integer.parseInt
    wearableModel = AR_DecisionFusion.getActivity(Integer.parseInt(a
```

Rename variables for intuitiveness

경희대학교
KYUNG HEE UNIVERSITY

Inertial based AR Open Source Code Progress – Task 3

27

```java
/***
 * Class for recognizing activity
 * @author euguzns
 *
 */
public class SMWRecognizer extends ActivityRecognizer<double[]>{
    // Varialbes for executing classification
    private AR_integrated ar_integrated = null;
    private AR_smartphone ar_smartphone = null;
    private AR_wearable ar_wearable = null;
    InertialData id = null;

    // Activity labels definition
    public static String[] actLabels = {"Standing", "Eating", "Stretching", "LyingDown", "Sitting", "Sweeping

    // Variables for showing final activity and writing to the file
    AR_DecisionFusion df;
    String name = null;
    LogFileWriter log = new LogFileWriter(FileUtil.getPath()+"\\log\\activity.txt");

    // Variables to load and save sensor data
    double acc_wm[][];
    double gy_wm[][];
    double mag_wm[][];
    double ecg_wm[];
    double acc_sm[][];
    double gy_sm[][];
    double mag_sm[][];

    /***
     * Load smartphone, wearable and integrated AR model
     */
    public SMWRecognizer(){
        ar_integrated = new AR_integrated(FileUtil.getPath()+"/armodel/integrated model_0126.model", name);
        ar_smartphone = new AR_smartphone(FileUtil.getPath()+"/armodel/sm model_0126.model", name);
        ar_wearable = new AR_wearable(FileUtil.getPath()+"/armodel/wm model_0126.model", name);
```

Comments for class, functions, and each paragraph for better understanding

Inertial based AR Open Source Code Progress – Task 4, 7

28

```
<!-- Main Body of the Document -->

# 1. Introduction

- The Inertial based Activity Recognizer (IAR) is designed to infer user physical activity such as "walking" or "running" using
- IAR currently uses two devices which are smartphone and smartwatch.
- The operation flow of IAR is similar to general activity recognition flow as data input, segmentation, feature extraction and
- In our work, we use weighted voting decision fusion to conclude the final activity.

## 1.1 Core concepts

- Example:
- Module: Modules are separated by different function, every module can communicate with other one through their own services.
- Service: Services are the interface of the specifically module.

# 2. Getting Started

This consist of the following sub-headings.

## 2.1 Requirements

-This component requires Java Development Kit and Weka tools to be installed.
- Our setup envirionment was JDK v1.7 and Weka 3.8.
```

Make API document under unified format

Mining Minds

Video-based Activity Recognition (VAR)
Open Source Progress

Thien Huynh-The

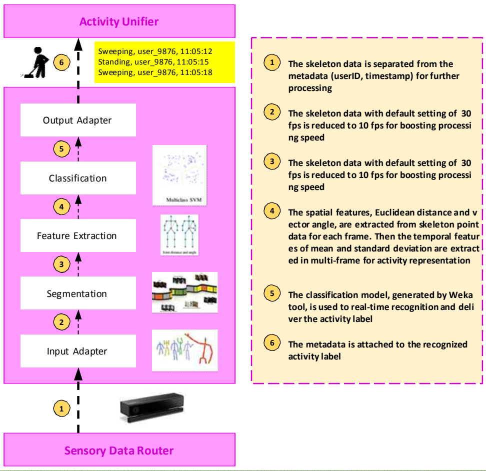
경희대학교 KYUNG HEE UNIVERSITY · KITECH 한국생산기술연구원 · POSTECH 포항공과대학교 · UNIVERSITY of TASMANIA · 녹십자헬스케어 · TAPACROSS · U2 system User & Usability system · KOCSA 한국컨설팅서비스협회

VAR - Open Source Code Progress – Task 1&2

30

VAR – Operation

- Offline training:
  - Feature selection: perform in Matlab to export the index numbers of selective features.
  - Model training: perform using Weka tool
- Online recognition (figure):
  - All components are written in Java
  - The output is the activity label recognized from Kinect.



경희대학교
KYUNG HEE UNIVERSITY

VAR - Open Source Code Progress – Task 1&2

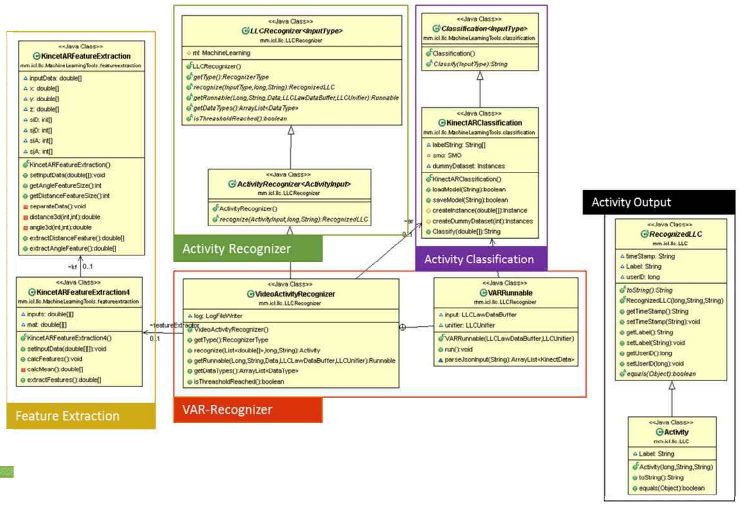
31

- Library Function Identification
  - Packages Categorization
    - mm.icl.llc.MachineLearningTools. featureextraction
      - ➢ Provide general functions for feature extraction (used for both skeleton and inertial data)
    - mm.icl.llc.MachineLearningTools.classification
      - ➢ Provide classification classes (SVM, J48, kNN) used for AR
    - mm.icl.llc.LLRecognizer
      - ➢ Provide execution recognizers
    - mm.icl.llc.LLC
      - ➢ Output activity and LLC label

경희대학교
KYUNG HEE UNIVERSITY

VAR - Open Source Code Progress – Task 1&2

32



경희대학교
KYUNG HEE UNIVERSITY

VAR - Open Source Code Progress – Task 1&2

33

Package mm.icl.llc.LLCRecognizer

Constructor Summary

Constructors

Constructor and Description
ActivityRecognizer()

Method Summary

All Methods | Instance Methods | Abstract Methods

Modifier and Type
abstract java.lang.Runnable
abstract RecognizedLLC

Methods inherited from class mm.icl.llc.LLCRecog

getType

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toStrin

Constructor Summary

Constructors

Constructor and Description
VideoActivityRecognizer()

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.Runnable	getRunnable(LLCLawDataBuffer input, LLCUnifier unifier)
RecognizerType	getType()
Activity	recognize(java.util.List<double[]> input, long userID, java.lang.String timeStamp)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

ActivityRecognizer indicates a appropriate recognizer module for new coming data.

경희대학교
KYUNG HEE UNIVERSITY

VAR - Open Source Code Progress – Task 1&2

34

Package mm.icl.llc.MachineLearningTools. featureextraction

```
public class KincetARFeatureExtraction
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
KincetARFeatureExtraction()

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
double[]	extractAngleFeature()
double[]	extractDistanceFeature()
int	getAngleFeatureSize()
int	getDistanceFeatureSize()
void	setInputData(double[] inputData)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Spatial Feature

Method Detail

setInputData

`public void setInputData(double[] inputData)`

getAngleFeatureSize

`public int getAngleFeatureSize()`

getDistanceFeatureSize

`public int getDistanceFeatureSize()`

extractDistanceFeature

`public double[] extractDistanceFeature()`

extractAngleFeature

`public double[] extractAngleFeature()`

Temporal Feature

Method Detail

setInputData

`public void setInputData(double[][] inputs)`

set input data to extract features for kinect activity recognition

Parameters:

inputs - input data to set

calcFeatures

`public void calcFeatures()`

calculate features. This function must be called after calling setInputData()

extractFeatures

`public double[] extractFeatures()`

extract features. this function must be called after calling setInputData() function.

Returns:

features for kinect activity recognition

경희대학교 KYUNG HEE UNIVERSITY

VAR - Open Source Code Progress – Task 1&2

35

Package mm.icl.llc.MachineLearningTools.classification

Constructor Summary

Constructors

Constructor and Description
KinectARClassification()

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
java.lang.String	Classify(double[] data)
boolean	loadModel(java.lang.String filename)
boolean	saveModel(java.lang.String filename)

Methods inherited from class java.lang.Object

equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

<<Java Class>>

KinectARClassification

mm.icl.llc.MachineLearningTools.classification

- labelString: String[]
- smo: SMO
- dummyDataset: Instances

- KinectARClassification()
- loadModel(String):boolean
- saveModel(String):boolean
- createInstance(double[]):Instance
- createDummyDataset(int):Instances
- Classify(double[]):String

```
public String Classify(double[] data) {
    // TODO Auto-generated method stub
    try {
        // Convert features to an instance
        Instance instance = createInstance(data);
        instance.setDataset(dummyDataset);

        // Classify
        double classIndex = smo.classifyInstance(instance);
        int labelIndex =  Double.valueOf(smo.classifyInstance(instance)).intValue();

        return labelString[labelIndex];
    } catch (Exception ex) {
        ex.printStackTrace();
    }
    return null;
}
```

경희대학교 KYUNG HEE UNIVERSITY

VAR - Open Source Code Progress – Task 1&2

36

Package mm.icl.llc.LLC

Constructor Summary

Constructors

Constructor and Description
Activity(long userID, java.lang.String Label, java.lang.String timeStamp) The only constructor for activity

Method Summary

All Methods | Instance Methods | Concrete Methods

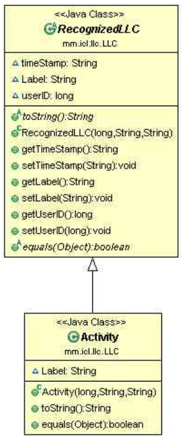
Modifier and Type	Method and Description
boolean	equals(java.lang.Object other) Please define the notion of logical equal of your class
java.lang.String	toString()

Methods inherited from class mm.icl.llc.LLC.RecognizedLLC

getLabel, getTimeStamp, getUserID, setLabel, setTimeStamp, setUserID

Methods inherited from class java.lang.Object

getClass, hashCode, notify, notifyAll, wait, wait, wait



경희대학교
KYUNG HEE UNIVERSITY

Goal & Objective – Task 3

37

- **Code Re-factoring – Task 3**
  - **Task Goal**
    - Shaping VAR source code with defined packages and classes in Task 1 & 2
  - **Objective**
    - Modify source code of the VAR component for readable and understandable
    - Optimizing source code for execution

경희대학교
KYUNG HEE UNIVERSITY

VAR - Open Source Code Progress – Task 3

38

Import code comments

```
/** Re-construct the data from DCL for processing
 *  One vector data with 75 values of x,y, and z to three vectors with 25 values
 *  25 values corresponding to 25 joints of each complete skeleton
 */
private void separateData(){
    x = new double[25];
    y = new double[25];
    z = new double[25];
    for (int i = 0;  i < 25; i++){
        x[i]=inputData[i*3];
        y[i]=inputData[i*3+1];
        z[i]=inputData[i*3+2];
    }
}
```

```
/** Calculating a distance feature between joint i and j
 *  This is the Euclidean distance
 */
private double distance3d(int i, int j){
    double d= 0;
    try{
        d=Math.sqrt(Math.pow(x[i]-x[j],2)+Math.pow(y[i]-y[j],2)+Math.pow(z[i]-z[j],2));
    }catch(Exception e){
    }
    return d;
}
```

경희대학교 KYUNG HEE UNIVERSITY

VAR - Open Source Code Progress – Task 3

39

Optimize code for execution

Before

```
private double angle3d(int i,int j){
    double deltax=x[j]-x[i];
    double deltay=y[j]-y[i];
    double a = Math.atan(deltay/deltax);
    // Normalizing angle from degree to gradian
    if (deltay == 0)
            a=0;
    else if (deltay > 0 && deltax > 0)
            a=a/(2*Math.PI);
    else if (deltay > 0 && deltax < 0)
            a=(a+Math.PI)/(2*Math.PI);
    else if (deltay <0 && deltax < 0)
            a=(a+Math.PI)/(2*Math.PI);
    else if (deltay <0 && deltax > 0)
            a=(a+2*Math.PI)/(2*Math.PI);
    return a;
}
```

After

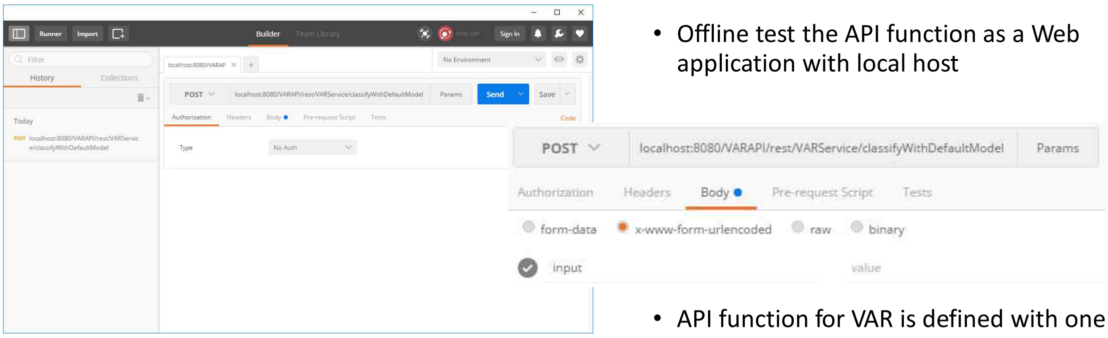
```
private double angle3d(int i,int j){
    double deltax=x[j]-x[i];
    double deltay=y[j]-y[i];
    double a = Math.atan(deltay/deltax);
    // Normalizing angle from degree to gradian
    if (deltay == 0)
            a=0;
    else if (deltay > 0 && deltax > 0)
            a=a;
    else if (deltay > 0 && deltax < 0)
            a=a+Math.PI;
    else if (deltay <0 && deltax < 0)
            a=a+Math.PI;
    else if (deltay <0 && deltax > 0)
            a=a+2*Math.PI;
    a=a/(2*Math.PI);
    return a;
}
```

경희대학교 KYUNG HEE UNIVERSITY

VAR - Open Source Code Progress – Task 4-7

40

Package VAR code to a Web Archive file that is to test, publish, deploy the resources.



- Offline test the API function as a Web application with local host
- API function for VAR is defined with one input as the skeleton data

경희대학교
KYUNG HEE UNIVERSITY

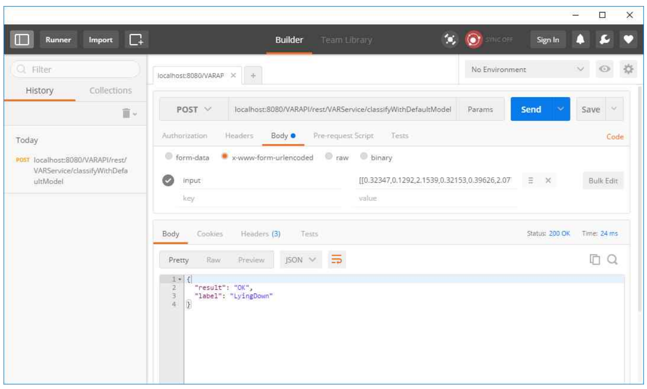
VAR - Open Source Code Progress – Task 4-7

41

Skeleton data of one body frame

Input data for 3-second window recognition

[[x1,y1,z1,…,x25,y25,z25],
[x1,y1,z1,…,x25,y25,z25],
…
[x1,y1,z1,…,x25,y25,z25]]



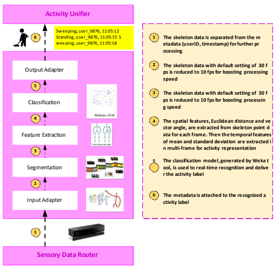
The VAR code is further tested using Junit by Donguk on ICL server

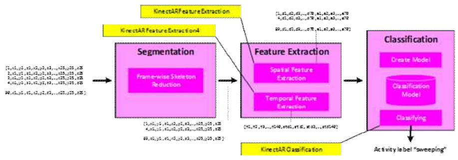
경희대학교
KYUNG HEE UNIVERSITY

VAR - Open Source Code Progress – Task 4-7

API Document

- Video based Activity Recognizer
  - Introduction
  - Overall WorkFlow
  - API Description

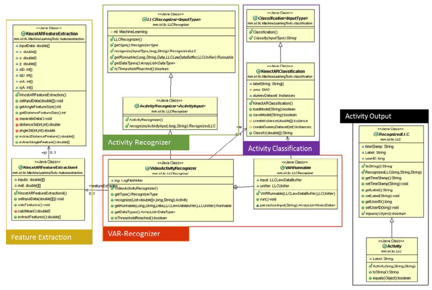




4-2-3. API Description

Class Summary

Class	Description
KinectARFeatureExtraction	This Class calculates the spatial features of joint distance and angle
KinectARFeatureExtraction4	This Class extracts temporal features of mean and standard deviation



Mining Minds

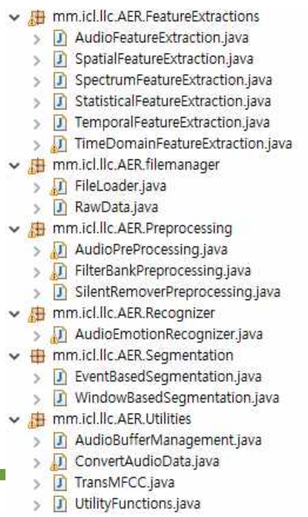
Information Curation Layer
Audio Emotion Recognizer
Open Source Progress

Jaehun Bang

Audio based ER Open Source Code Progress – Task1

44

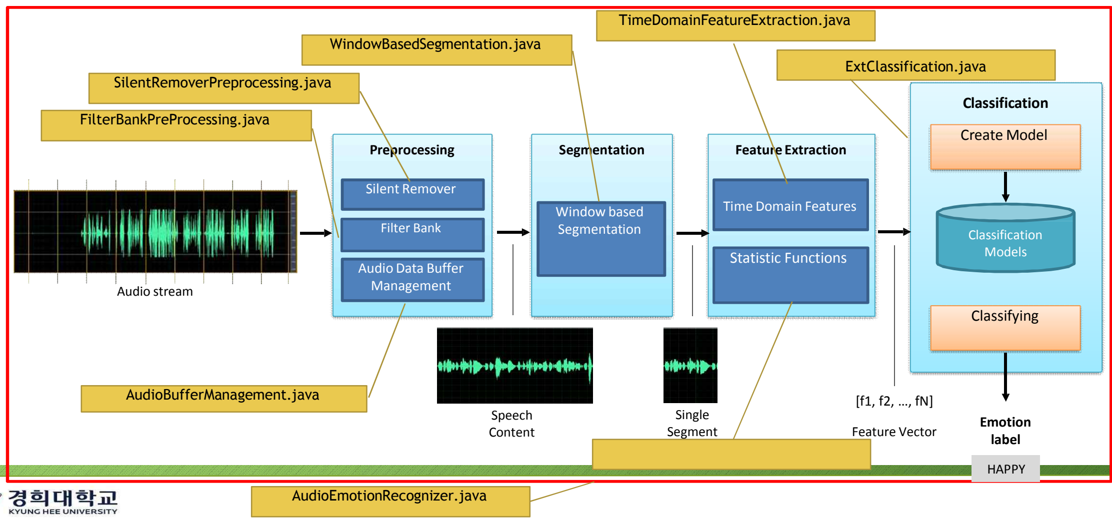
- Library Function Identification
  - Packages Categorization
    - mm.icl.llc.AER.Recognizer
      - Audio Emotion Recognizer execution code
    - mm.icl.llc.AER.FeatureExtractions
      - Feature Extraction Functions for AER
    - mm.icl.llc.AER.filemanager
      - Load audio file and writing arff file (weka)
    - mm.icl.llc.AER.Segmentation
      - Segmentation function for audio file
    - mm.icl.llc.AER.Preprocessing
      - Preprocessing function for audio file
    - mm.icl.llc.AER.Utilities
      - Utility function for processing of audio file



경희대학교
KYUNG HEE UNIVERSITY

Audio based ER Open Source Code Progress – Task1

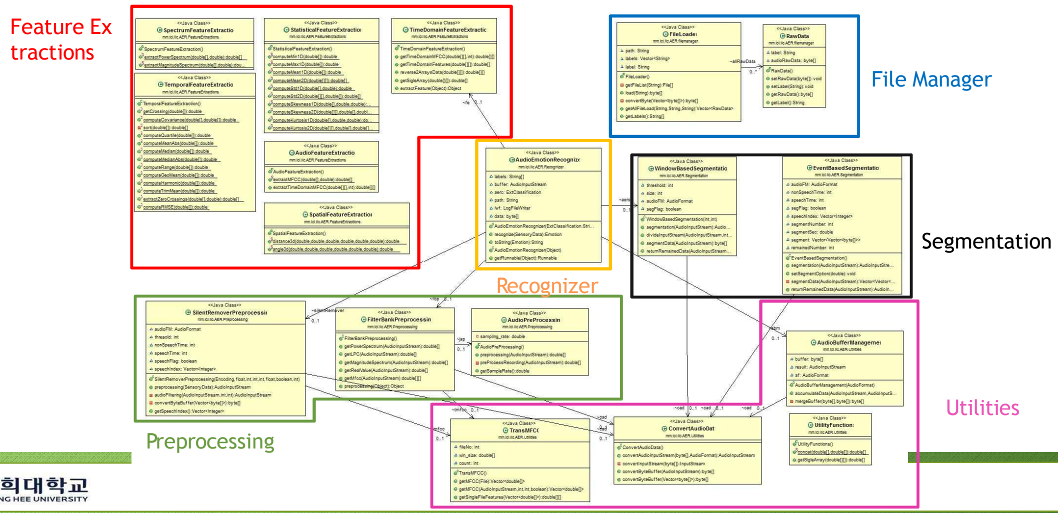
45



경희대학교
KYUNG HEE UNIVERSITY

Audio based ER Open Source Code Progress – Task2

46



경희대학교 KYUNG HEE UNIVERSITY

Audio based ER Open Source Code Progress – Task2

47

- Core component for execution of audio based emotion recognizer

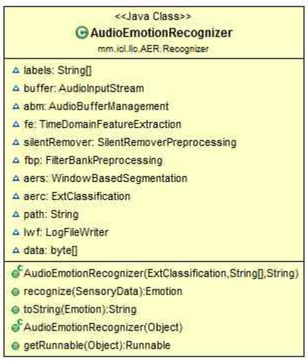
Package mm.icl.llc.AER.Recognizer

Class Summary

Class	Description
AudioEmotionRecognizer	This Class support Audio based Emotion Recognition on MM v2.5

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
java.lang.Runnable	getRunnable(java.lang.Object data)
mm.icl.llc.LLC.Emotion	recognize(mm.icl.llc.sensorydata.SensoryData input)
java.lang.String	toString(mm.icl.llc.LLC.Emotion emotion)



Example Code

```
SMO svm = new SMO(); // Define Classifier with Weka
ExtClassification classifier = new ExtClassification(path, 82, labels,svm);
AudioEmotionRecognizer aer = new AudioEmotionRecognizer(classifier, labels, path);
Emotion emotion = aer.recognize(sd);
```

경희대학교 KYUNG HEE UNIVERSITY

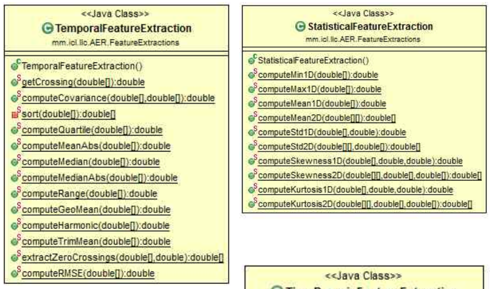
Audio based ER Open Source Code Progress – Task2

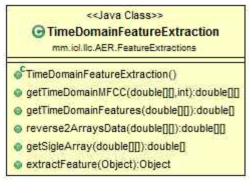
48

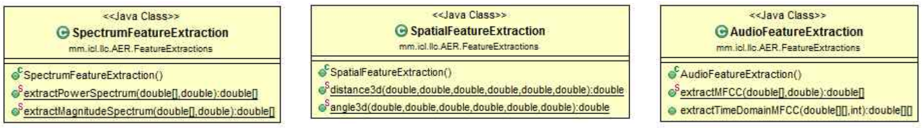
Package mm.icl.llc.AER.FeatureExtractions

Class Summary

Class	Description
AudioFeatureExtraction	This class extracts audio features from the data
SpatialFeatureExtraction	This class extracts spatial features from the data
SpectrumFeatureExtraction	This class extracts spectrum features from the data
StatisticalFeatureExtraction	This class contains common statistical functions
TemporalFeatureExtraction	This class extracts temporal features from the data
TimeDomainFeatureExtraction	This Class extract time-domain features based on seconds







경희대학교
KYUNG HEE UNIVERSITY

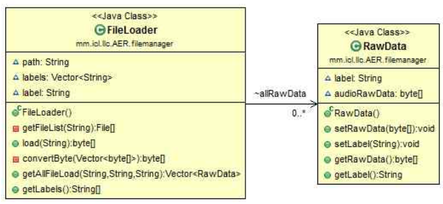
Audio based ER Open Source Code Progress – Task2

49

Package mm.icl.llc.AER.filemanager

Class Summary

Class	Description
FileLoader	This class support file loader for audio file
RawData	Structure class for Audio Data



All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
java.util.Vector<RawData>	getAllFileLoad(java.lang.String path, java.lang.String format, java.lang.String delimeter) load multiple data
java.lang.String[]	getLabels() Load label information
byte[]	load(java.lang.String path) Load single audio file

FileLoader Example Code

```
String path = "D:/데이터셋/abc/";
FileLoader fl = new FileLoader();
Vector<RawData> rawData = new Vector<RawData>();
rawData = fl.getAllFileLoad(path, "wav", "_");
```

경희대학교
KYUNG HEE UNIVERSITY

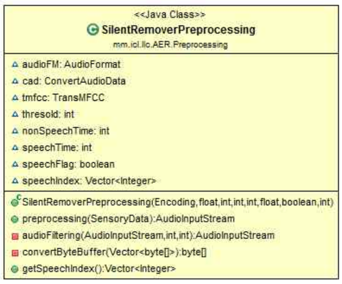
Audio based ER Open Source Code Progress – Task2

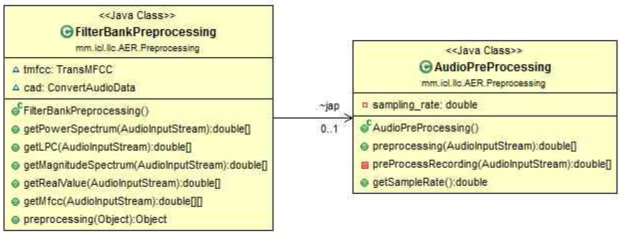
50

Package mm.icl.llc.AER.Preprocessing

Class Summary

Class	Description
AudioPreProcessing	This class support to preprocess with filterbank algorithm (mfcc, lpc, spectrums, realvalue)
FilterBankPreprocessing	This class extract computable data with Filter-Bank Algorithms such as LPC, MFCC, Spectrums
SilentRemoverPreprocessing	This Class remove silent from audio data





경희대학교
KYUNG HEE UNIVERSITY

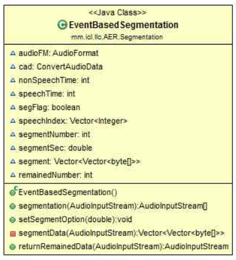
Audio based ER Open Source Code Progress – Task2

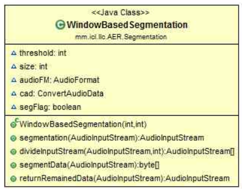
51

Package mm.icl.llc.AER.Segmentation

Class Summary

Class	Description
EventBasedSegmentation	This Class segment based on sentence
WindowBasedSegmentation	This Class segment audio data based on seconds





WindowBasedSegmentation Example Code

```
WindowBasedSegmentation aers = new WindowBasedSegmentation(3, 44100);
AudioInputStream segmented = aers.segmentation(buffer);
```

EventBasedSegmentation Example Code

```
EventBasedSegmentation ebs = new EventBasedSegmentation();
AudioInputStream[] segmented = ebs.segmentation(buffer);
```

경희대학교
KYUNG HEE UNIVERSITY

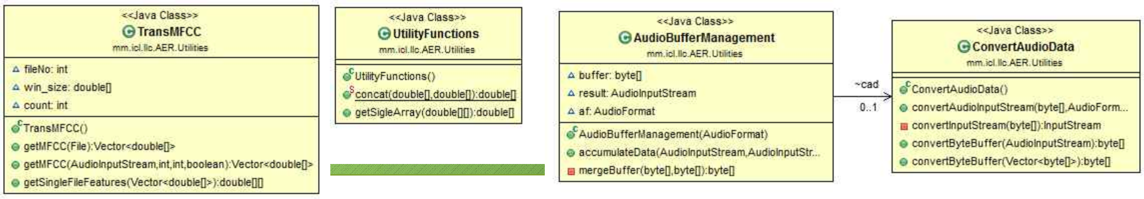
Audio based ER Open Source Code Progress – Task2

52

Package mm.icl.llc.AER.Utilities

Class Summary

Class	Description
AudioBufferManagement	This Class support to accumulate audio buffer for extracting the voice
ConvertAudioData	Convert the audio data between oInputStream to Byte
TransMFCC	This Class support to extract MFCC filterbank feature using comirva library
UtilityFunctions	This class contains utility functions



KYUNG HEE UNIVERSITY

Audio based ER Open Source Code Progress – Task3

53

- **Deletion of Duplicated Code**

FilterBankPreProcessing.java AERPreprocessing.java StatisticalFeatureExtraction.java AERFeatureExtraction.java

Audio based ER Open Source Code Progress – Task3

- **Dividing Long Methods**

```java
public Emotion recognize(SensoryData input) {
    AudioInputStream preprocessed = silentRemover.preprocessing(input); //Audio Input Stream for removing silent
    Emotion emotion = new Emotion();
    if (preprocessed != null) {
        abm = new AudioBufferManagement(preprocessed.getFormat()); // AudioBufferManagement for accumulating Audio Buffer
        buffer = abm.accumulateData(buffer, preprocessed);
        if (buffer != null) {
            AudioInputStream segmented = aers.segmentation(buffer);
            buffer = aers.returnRemainedData(buffer);
            double[] lpcFeature = null;
            double[][] mfccPreprocessing = null;
            double[][] mfccfeature = null;
            if(segmented != null){
                try {
                    mfccPreprocessing = fbp.getMfcc(segmented); // extract MFCC by dobule[][]
                    mfccfeature = fe.getTimeDomainMFCC(mfccPreprocessing, 3); //calculate Time Domain MFCC feature by double[][]
                } catch (IOException | UnsupportedAudioFileException e) {
                    e.printStackTrace();
                }

                double[] mfccfeaturevt = fe.getSigleArray(mfccfeature); // transform double[] from double[][]
                double[] featurevt = new double[mfccfeaturevt.length+lpcFeature.length];
                for(int k=0 ; k< mfccfeaturevt.length; k++){
                    featurevt[k] = mfccfeaturevt[k];
                }

                for(int k=mfccfeaturevt.length; k<featurevt.length;k++){
                    featurevt[k] = lpcFeature[k-mfccfeaturevt.length];
                }

                emotion.setEmotion(aerc.classify(featurevt)); //classify emotion
                lwf.writeLogFeature(featurevt, toString(emotion));
                lwf.close();
                return emotion;
            }
        }
    }
    emotion.setEmotion(labels.length+1);
    return emotion;
}
```

```java
@Override
public void ConsumeData(InputType it, Object obj) {
    // TODO Auto-generated method stub
    if(it.equals(InputType.SmartPhone)){
        SensoryData data = (SensoryData) obj;
        timeStamp = data.getTimeStamp();
        userID = data.getUserID();
        //emoJHinput = silentRemover.preprocessing(data); //Audio Input Stream for removing silent
        if(data.getAudioSensoryData() != null){
            this.audiodata = data.getAudioSensoryData();
            AudioInputStream preprocessed = silentRemover.preprocessing(data); //Audio Input Stream for removing silent

            if (preprocessed != null) {
                abm = new AudioBufferManagement(preprocessed.getFormat()); // AudioBufferManagement for accumulating Audio Buffer

                buffer = abm.accumulateData(buffer, preprocessed);
                if (buffer != null) {
                    inputData = aers.segmentation(buffer);
                    buffer = aers.returnRemainedData(buffer);
                }
            }
        }

    }
}
```

```java
@Override
public Emotion recognize(AudioInputStream segmented, long userID, String timeStamp) {
    System.err.println("Emo Silenceless working");
    Emotion emotion = new Emotion(userID, "NoEmotion", timeStamp);
    try {
        double[][] mfccPreprocessing = null;
        // Divide audio stream for extracting time domain feature
        AudioInputStream[] devideSegmented = aers.divideInputStream(segmented, 3);
        Vector<double[]> featureVector = new Vector<double[]>();

        for(int k =0; k<devideSegmented.length; k++){
            try {
                // Preprocessing for extracting MFCC vector
                mfccPreprocessing = fbp.getMfcc(devideSegmented[k]);
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            double[] featureVT = fe.getTimeDomainFeatures(mfccPreprocessing);
            featureVector.add(featureVT);
        }
        // Extracting time domain feature vector
        double[] timeDomainFeatureVt = fe.convertVector(featureVector);

        // Classifying emotions based on time domain feature vector
        emotion.setLabel(emoLabel[aerc.classify(timeDomainFeatureVt)]);
        log.writeLogFile(userID + "\t" + emotion.getLabel() + "\t" + timeStamp);

        // Check current time
        long localTime = TimeChecker.time;
        System.err.println("Time test of emotion JH = " + (TimeChecker.getCurTime() - localTime));
        FileUtil.WriteTimeLog("EmoJH", (TimeChecker.getCurTime() - localTime) + "\r\n");

        return emotion;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;

}
```

Example: AudioEmotionRecognizer.java - Recognize Method

Audio based ER Open Source Code Progress – Task3

- **Comment in Code**
  - **Describe summarized method and detailed processing**
  - **Show current working situation by console**

Example: AudioEmotionRecognizer.java

```java
/***
 * recognize emotion from audio sensory data
 * @param segmented - Segemented audio stream
 * @param userID - userID for identification user
 * @param timeStamp - Acquisition time
 */
@Override
public Emotion recognize(AudioInputStream segmented, long userID, String timeStamp) {
    System.err.println("Emo Silenceless working");
    Emotion emotion = new Emotion(userID, "NoEmotion", timeStamp);
    try {
        double[][] mfccPreprocessing = null;
        // Divide audio stream for extracting time domain feature
        AudioInputStream[] devideSegmented = aers.divideInputStream(segmented, 3);
        Vector<double[]> featureVector = new Vector<double[]>();

        for(int k =0; k<devideSegmented.length; k++){
            try {
                // Preprocessing for extracting MFCC vector
                mfccPreprocessing = fbp.getMfcc(devideSegmented[k]);
            } catch (IOException e) {
                // TODO Auto-generated catch block
                e.printStackTrace();
            }
            double[] featureVT = fe.getTimeDomainFeatures(mfccPreprocessing);
            featureVector.add(featureVT);
        }
        // Extracting time domain feature vector
        double[] timeDomainFeatureVt = fe.convertVector(featureVector);

        // Classifying emotions based on time domain feature vector
        emotion.setLabel(emoLabel[aerc.classify(timeDomainFeatureVt)]);
        log.writeLogFile(userID + "\t" + emotion.getLabel() + "\t" + timeStamp);

        // Check current time
        long localTime = TimeChecker.time;
        System.err.println("Time test of emotion JH = " + (TimeChecker.getCurTime() - localTime));
        FileUtil.WriteTimeLog("EmoJH", (TimeChecker.getCurTime() - localTime) + "\r\n");

        return emotion;
    } catch (Exception e) {
        e.printStackTrace();
    }
    return null;
```

Summarized methods Description

Printout current situation by console screen

Detailed pr ocessing D escription

Audio based ER Open Source Code Progress – Task4

56

- **Minimization class for running Audio Emotion Recognizer**

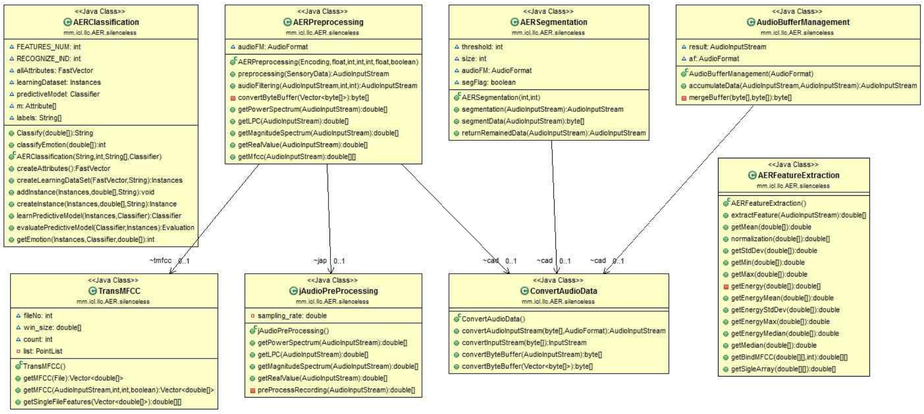


Diagram of API Class Structures

Audio based ER Open Source Code Progress – Task4

57

- **API example code**

```java
public class AERTesting {

    @SuppressWarnings("resource")
    public static void main(String[] args) {

        /////////////////File Load Part/////////////////////
        File file = new File("angry1.wav"); // Definition File object for recognizing
        SensoryData sd = new SensoryData(); // Creation of new SensoryData Instance
        int featureNum = 82; // Overall number of features
        byte[] data  = load(file); //Byte data from wav file
        sd.setAudioSensoryData(data); //Set AudioSensory Data

        /////////////////Create Classification Model Part/////////////////////
        String path = "Model.txt"; // Path of training data set file
        String[] Labels = { "Happy", "Angry", "Sadness" }; // Label list of recognizing emotion
        SMO svm = new SMO();  // Create Classifier from Weka API
        ExtClassification classifier = new ExtClassification(path, featureNum, Labels, svm); //Creation Classification model through training process

        /////////////////Recognize Emotion Part/////////////////////
        AudioEmotionRecognizer aer = new AudioEmotionRecognizer(classifier); // Definition of Audio Emotion Recognizer with predefined Classification model
        Emotion emotion = aer.recognize(sd); // Recognize Emotion from SensoryData
        System.out.println("emotion: " + emotion.toString()); // Print out recognized emotion result
    }


    ///////////////Method of file load///////////////
    static byte[] load(File file)  {
        byte[] data =null;
        byte[] Header = new byte[44]; // Audio file Header size
        try {
            FileInputStream fis = new FileInputStream(file);
            data = new byte[(int) file.length() - Header.length];
            fis.read(Header);
            while (fis.available() > 0) {
                fis.read(data);
            }
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return data;
    }
}
```

Mining Minds

High Level Context Awareness (HLCA)

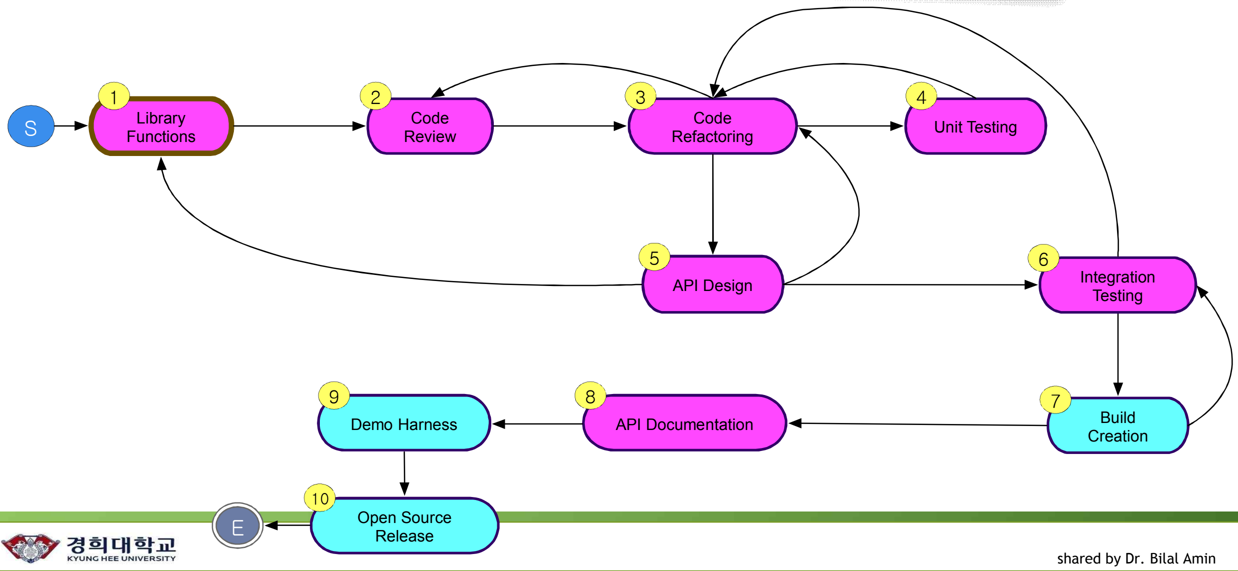
Open Source MMV3.0

Agenda

20
6

- Open Source Roadmap
  - Library Function
  - Class Diagram
  - Refactoring
    - Code Refactoring
    - Log Maintenance
    - Refactoring Impact
  - Unit Testing
  - API Design
  - Build Creation
  - API Documentation
  - Demo Harness
  - Open Source Release

Open Source Release Process - RoadMap



shared by Dr. Bilal Amin

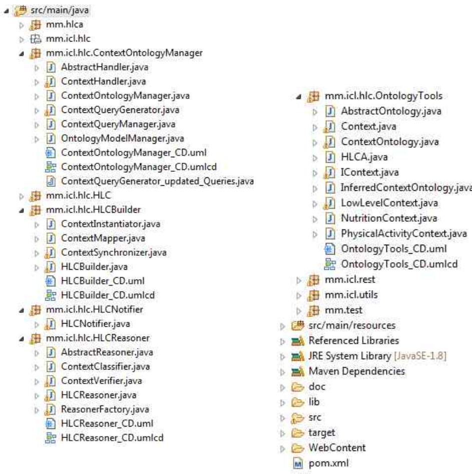
HLCA Open Source Code – Task 1&2

- **High Level Context Library Functions Identification – Task 1**
  - **Task Goal**
    - Identification of High Level Context Functions through reviewing implemented code V2.5
  - **Objective**
    - Relating High Level, Low Level Context based classes with Mining Minds HLCA V2.5 architecture
    - Categorizing packages based on functionalities of classes.
- **Code Review for ICL – Task 2**
  - **Task Goal**
    - Review HLCA V2.5 source code for determining Quality code.
  - **Objective**
    - Analysis all of the HLCA source code
    - Finalize classes structure with HLCA architecture

HLCA Open Source Code – Task 1&2

209

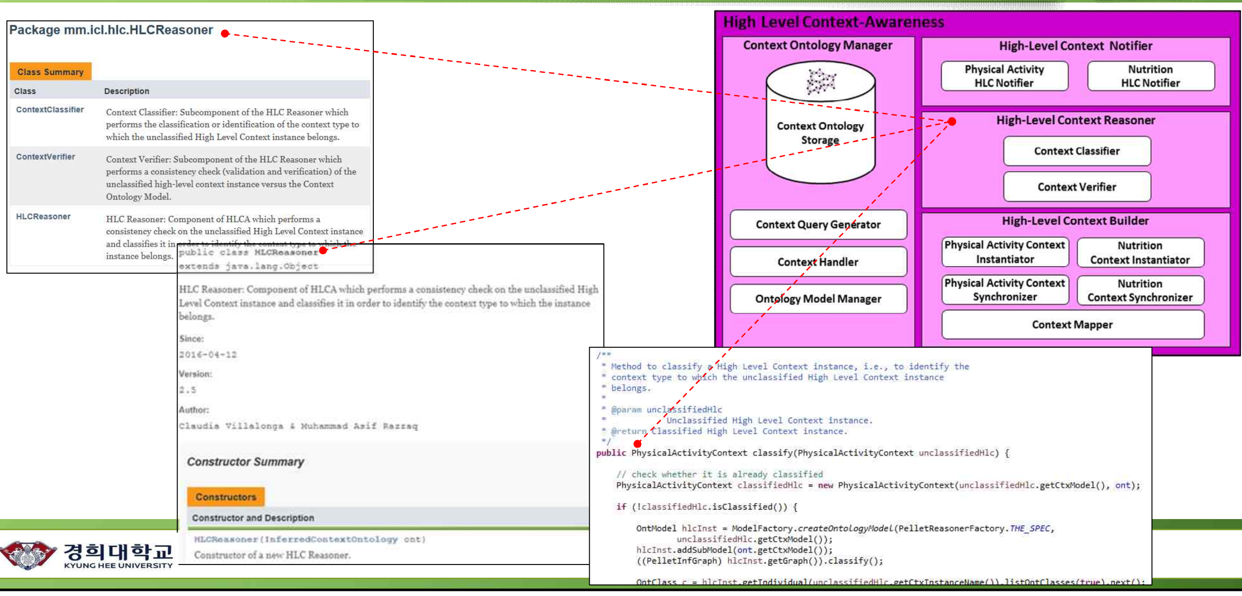
- Library Function Identification
  - Packages Categorization
    - mm.icl.hlc.HLCBuilder
      - Maps, Synchronizes, & Instantiates
    - mm.icl.hlc.HLCReasoner
      - Verifies and Classifies UC-HLC
    - mm.icl.hlc.HLCNotifier
      - Notifies recognized HLC
    - mm.icl.hlc.OntologyTools
      - Parent Classes, methods for PA, Nut, HLC, LLC & Context.
    - mm.icl.hlc.ContextOntologyManager
      - Loads OntModel, Manages SPARQL
    - mm.icl.hlc.HLCA
      - To Execute HLCA Offline
    - mm.icl.hlc.HLC
      - Time Utility for proper time format



경희대학교
KYUNG HEE UNIVERSITY

HLCA Open Source Code – Task 1&2

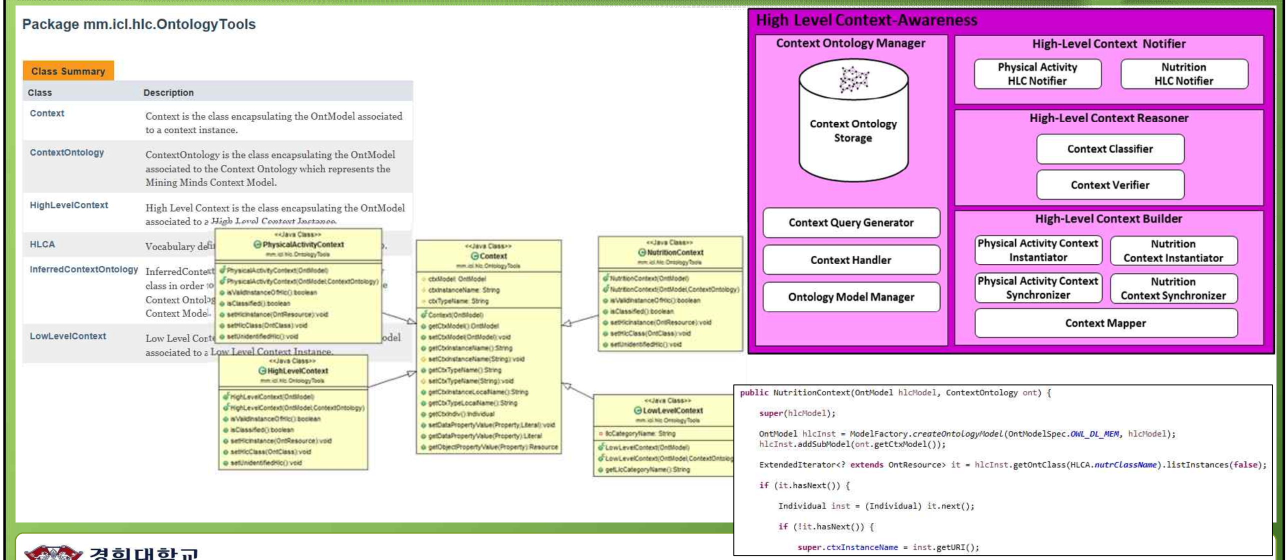
210



경희대학교
KYUNG HEE UNIVERSITY

HLCA Open Source Code – Task 1&2

21
1



경희대학교
KYUNG HEE UNIVERSITY

HLCA Open Source Code – Task 3

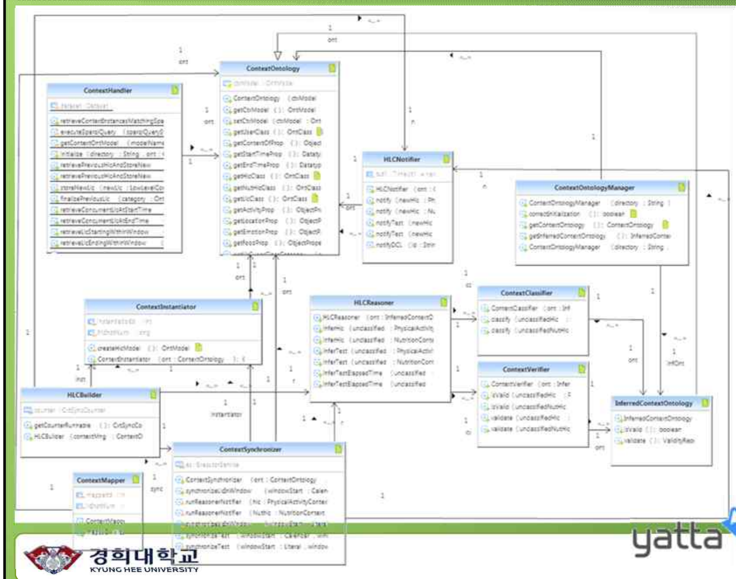
21
2

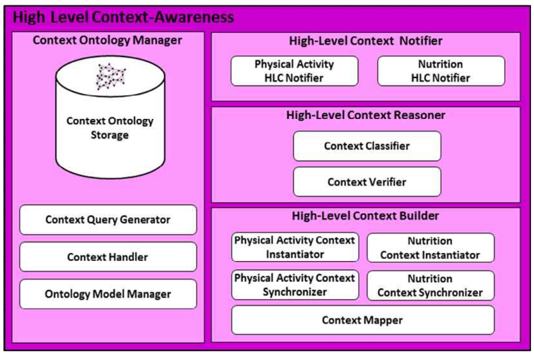
- **Code Re-factoring – Task 3**
  - **Task Goal**
    - Refining HLCA source code with defined classes and structures with designed by Task 1 and Task 2
  - **Objective**
    - Modify source code for consistency maintenance with other codes of ICL components.
    - Optimizing source code for execution for its Horizontal Growth
    - Introduction of Design Patterns
    - Introduction of Maintenance Logs

경희대학교
KYUNG HEE UNIVERSITY

Class Diagram before Refactoring

213

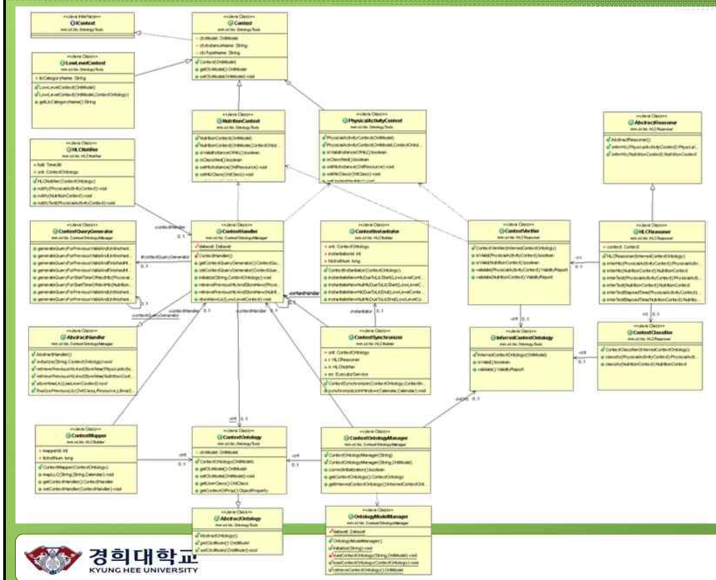


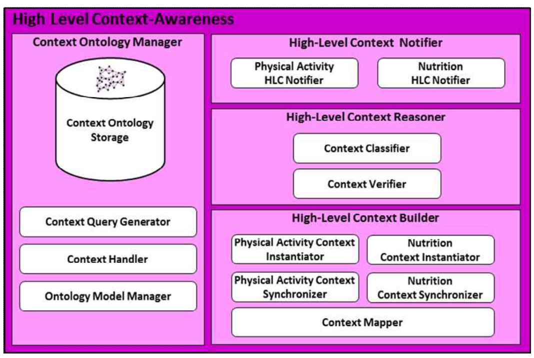


UML LAB

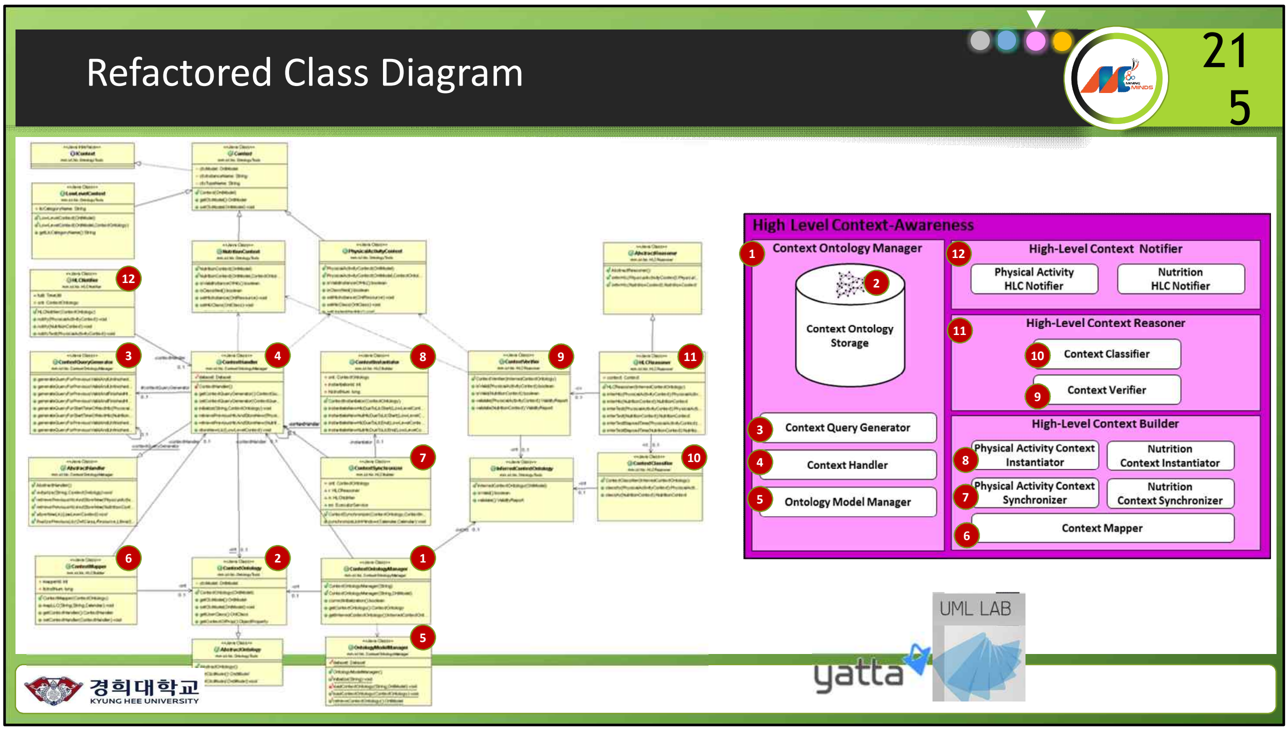
Refactored Class Diagram

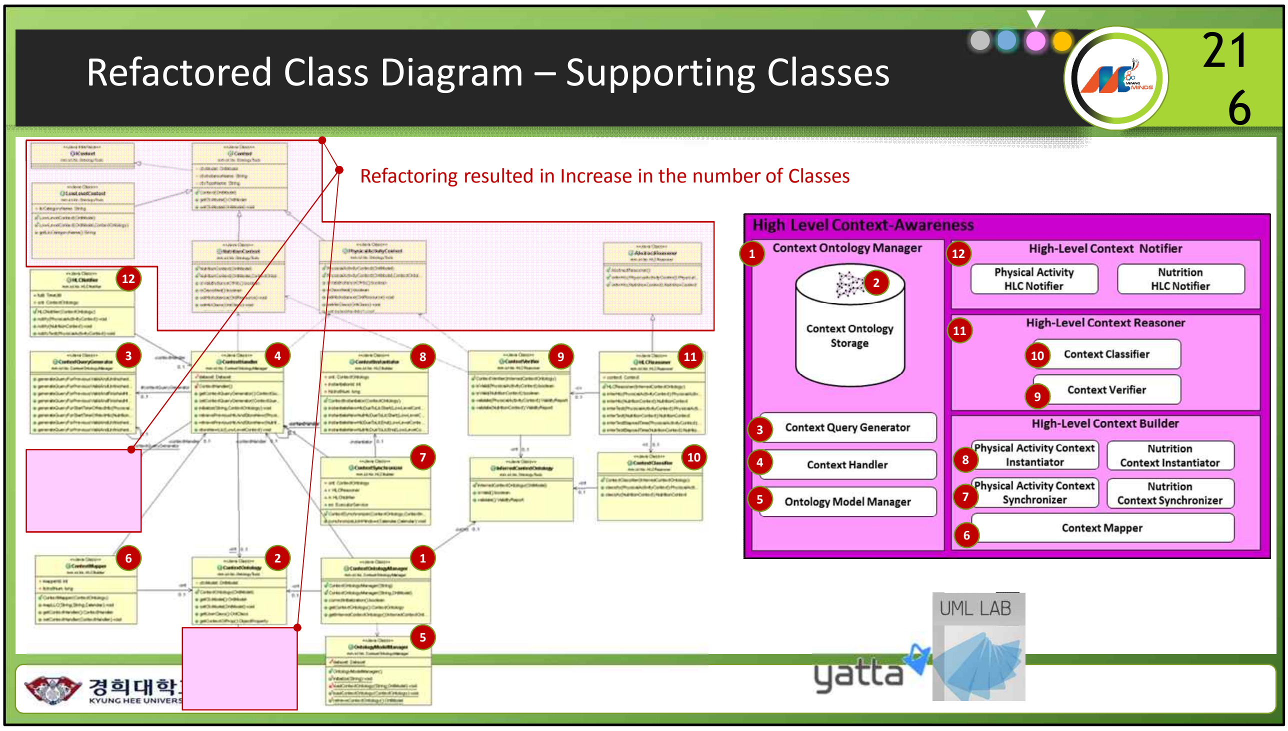
214

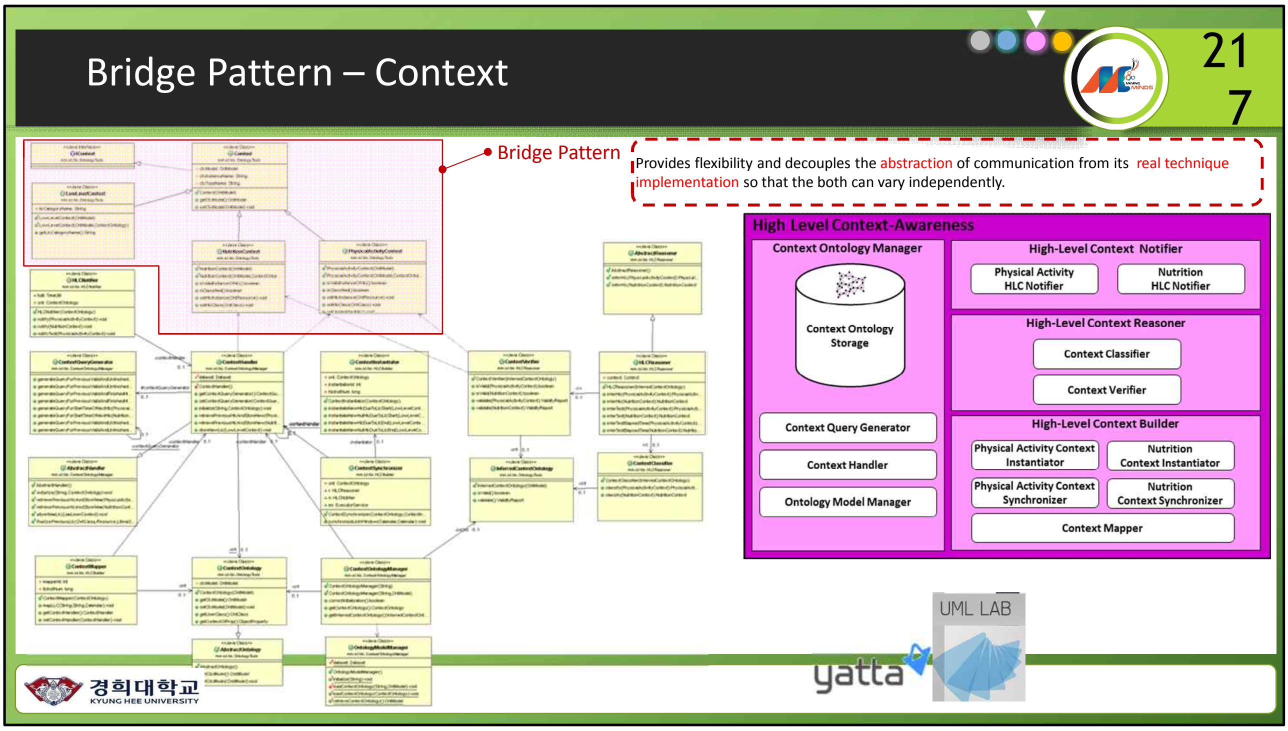


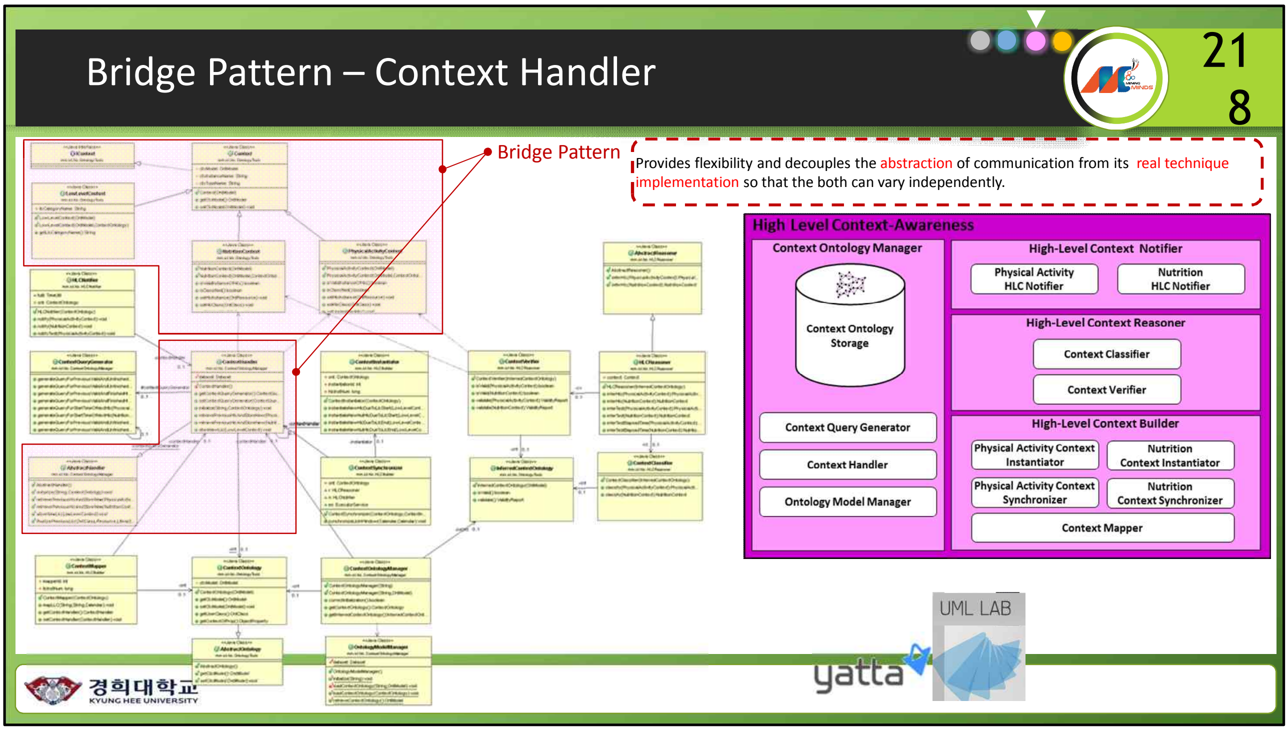


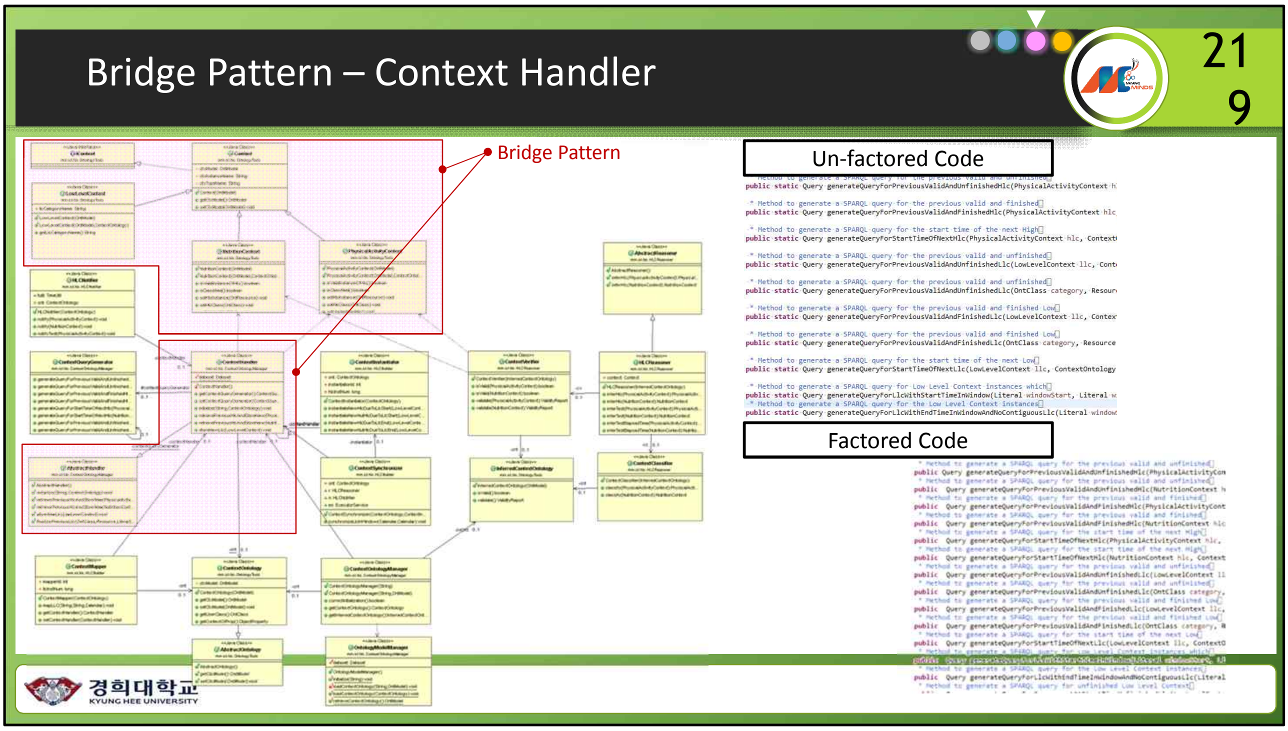
UML LAB


Refactored Class Diagram
215
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
Physical Activity HLC Notifier
Nutrition HLC Notifier
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Physical Activity Context Instantiator
Nutrition Context Instantiator
Physical Activity Context Synchronizer
Nutrition Context Synchronizer
Context Mapper
UML LAB
yatta
경희대학교
KYUNG HEE UNIVERSITY

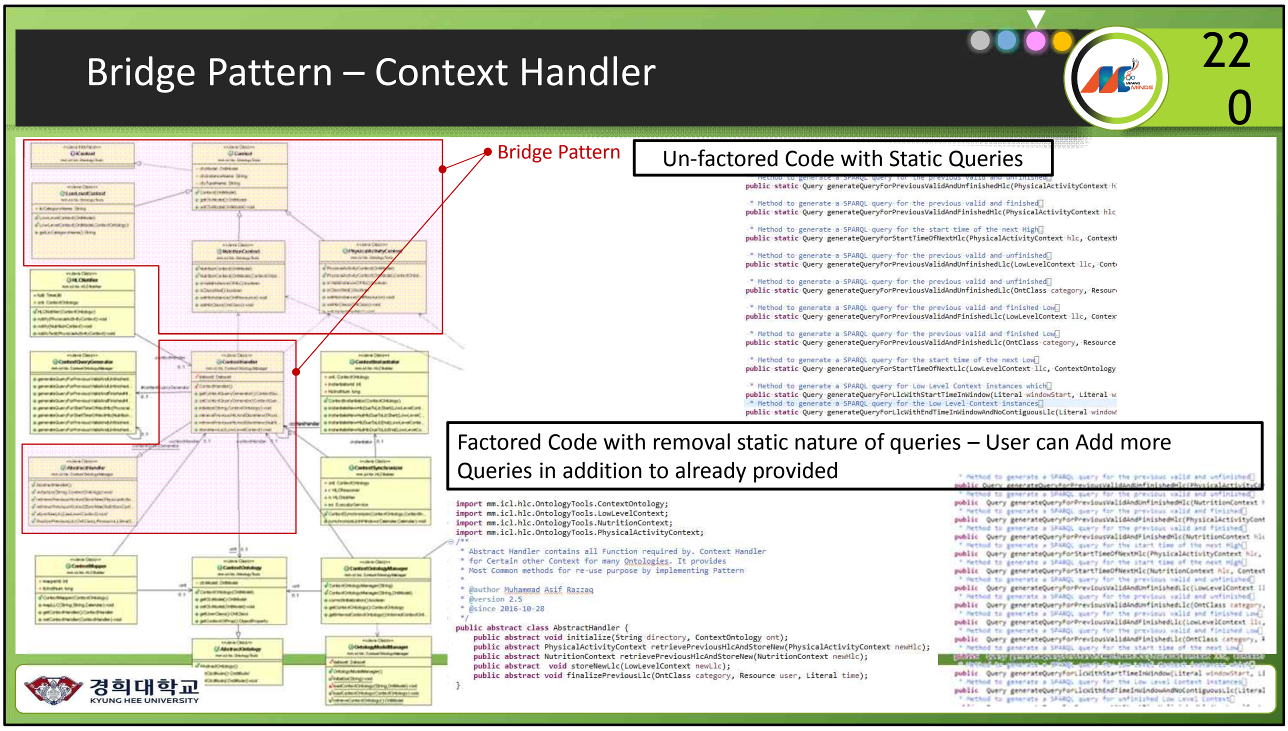

Refactored Class Diagram – Supporting Classes
216
Refactoring resulted in Increase in the number of Classes
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
Physical Activity HLC Notifier
Nutrition HLC Notifier
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Physical Activity Context Instantiator
Nutrition Context Instantiator
Physical Activity Context Synchronizer
Nutrition Context Synchronizer
Context Mapper
UML LAB
yatta


Bridge Pattern – Context
21
7
Bridge Pattern
Provides flexibility and decouples the abstraction of communication from its real technique implementation so that the both can vary independently.
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
Physical Activity HLC Notifier
Nutrition HLC Notifier
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Physical Activity Context Instantiator
Nutrition Context Instantiator
Physical Activity Context Synchronizer
Nutrition Context Synchronizer
Context Mapper
UML LAB
yatta
경희대학교
KYUNG HEE UNIVERSITY


Bridge Pattern – Context Handler
21
8
Bridge Pattern
Provides flexibility and decouples the abstraction of communication from its real technique implementation so that the both can vary independently.
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
Physical Activity HLC Notifier
Nutrition HLC Notifier
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Physical Activity Context Instantiator
Nutrition Context Instantiator
Physical Activity Context Synchronizer
Nutrition Context Synchronizer
Context Mapper
UML LAB
yatta
경희대학교
KYUNG HEE UNIVERSITY

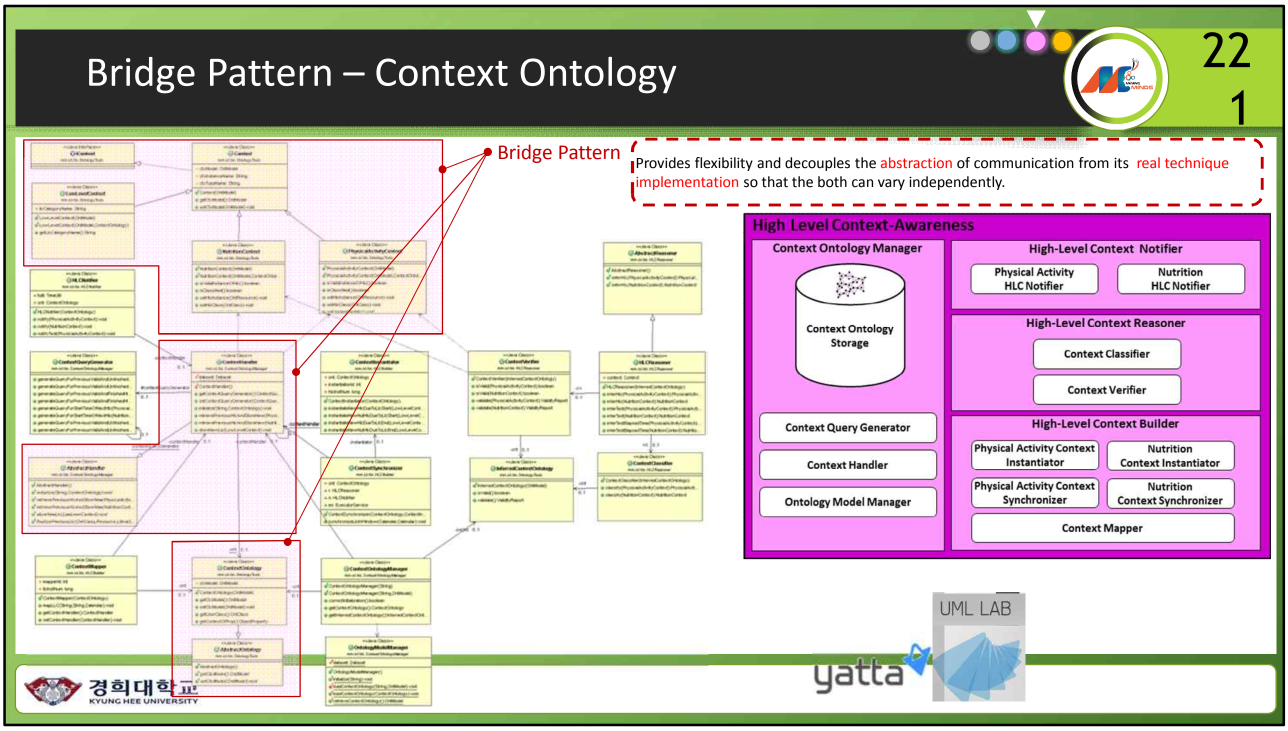


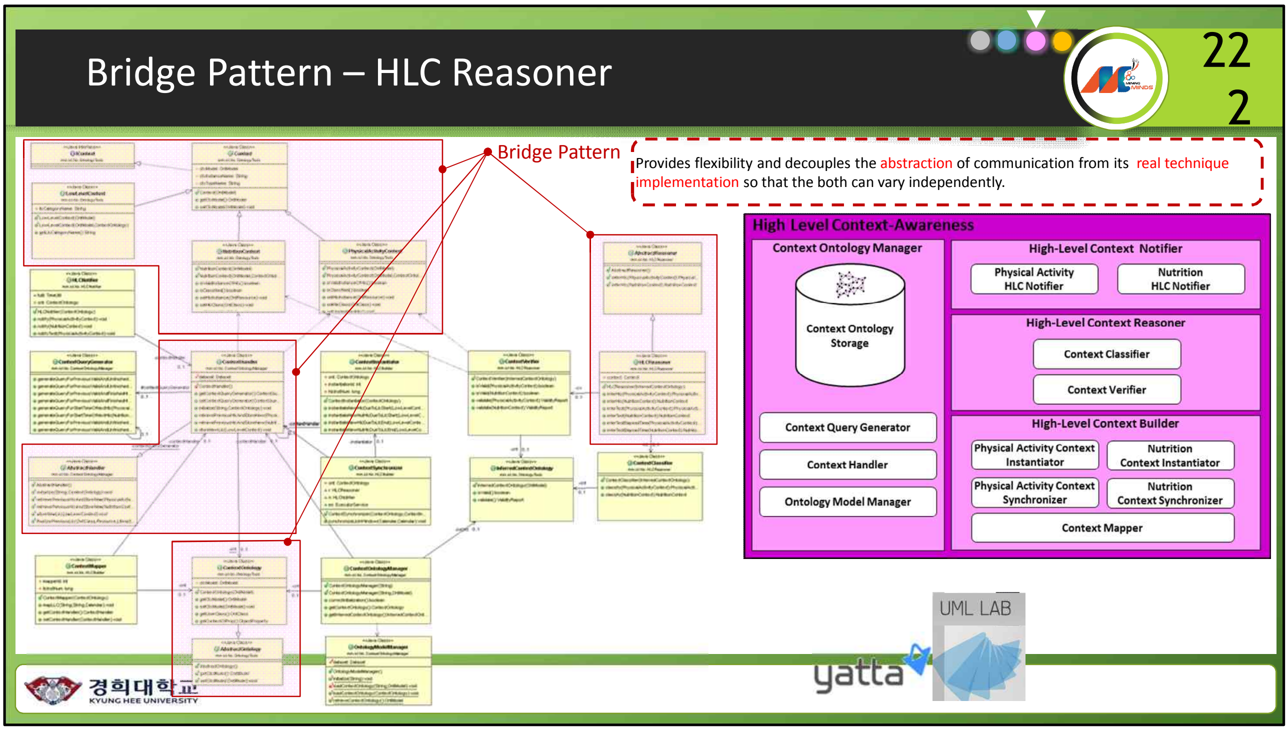
Bridge Pattern – Context Handler
21
9
Bridge Pattern
Un-factored Code
Factored Code
경희대학교
KYUNG HEE UNIVERSITY

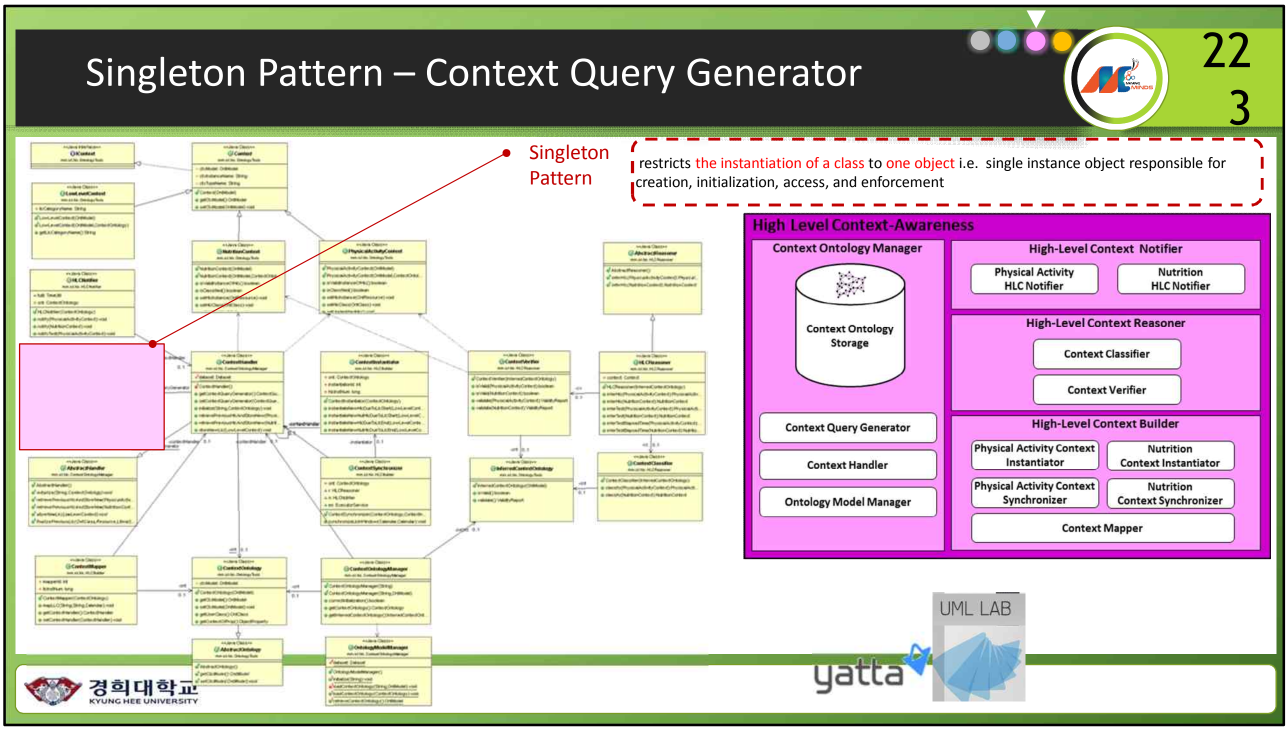


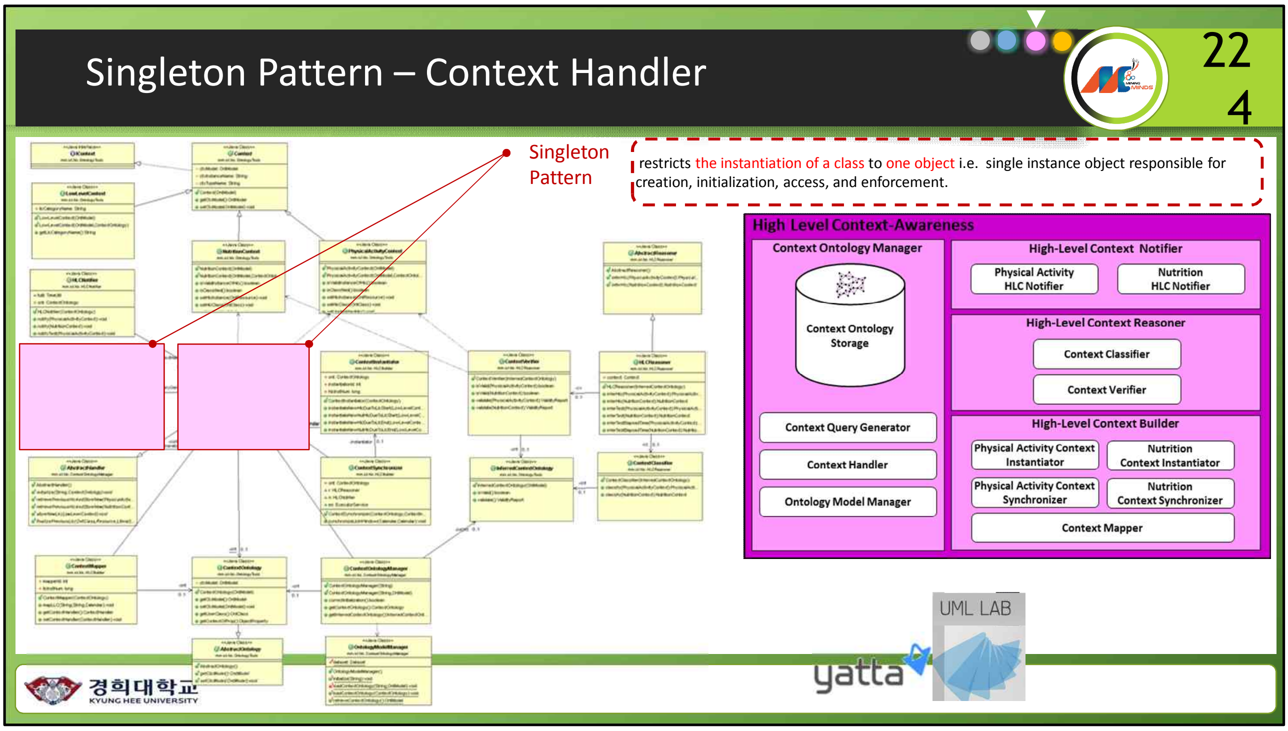
Bridge Pattern – Context Handler
22
0
Bridge Pattern
Un-factored Code with Static Queries
Factored Code with removal static nature of queries – User can Add more Queries in addition to already provided
import mm.icl.hlc.OntologyTools.ContextOntology;
import mm.icl.hlc.OntologyTools.LowLevelContext;
import mm.icl.hlc.OntologyTools.NutritionContext;
import mm.icl.hlc.OntologyTools.PhysicalActivityContext;
/**
* Abstract Handler contains all Function required by. Context Handler
* for Certain other Context for many Ontologies. It provides
* Most Common methods for re-use purpose by implementing Pattern
*
* @author Muhammad Asif Razzaq
* @version 2.5
* @since 2016-10-28
*/
public abstract class AbstractHandler {
public abstract void initialize(String directory, ContextOntology ont);
public abstract PhysicalActivityContext retrievePreviousHlcAndStoreNew(PhysicalActivityContext newHlc);
public abstract NutritionContext retrievePreviousHlcAndStoreNew(NutritionContext newHlc);
public abstract void storeNewLlc(LowLevelContext newLlc);
public abstract void finalizePreviousLlc(OntClass category, Resource user, Literal time);
}
경희대학교
KYUNG HEE UNIVERSITY

326


Bridge Pattern – Context Ontology
221
Bridge Pattern
Provides flexibility and decouples the abstraction of communication from its real technique implementation so that the both can vary independently.
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
Physical Activity HLC Notifier
Nutrition HLC Notifier
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Physical Activity Context Instantiator
Nutrition Context Instantiator
Physical Activity Context Synchronizer
Nutrition Context Synchronizer
Context Mapper
UML LAB
yatta
경희대학교
KYUNG HEE UNIVERSITY


Bridge Pattern – HLC Reasoner
222
Bridge Pattern
Provides flexibility and decouples the abstraction of communication from its real technique implementation so that the both can vary independently.
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
Physical Activity HLC Notifier
Nutrition HLC Notifier
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Physical Activity Context Instantiator
Nutrition Context Instantiator
Physical Activity Context Synchronizer
Nutrition Context Synchronizer
Context Mapper
UML LAB
yatta
경희대학교
KYUNG HEE UNIVERSITY


Singleton Pattern – Context Query Generator
22
3
Singleton Pattern
restricts the instantiation of a class to one object i.e. single instance object responsible for creation, initialization, access, and enforcement
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
Physical Activity HLC Notifier
Nutrition HLC Notifier
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Physical Activity Context Instantiator
Nutrition Context Instantiator
Physical Activity Context Synchronizer
Nutrition Context Synchronizer
Context Mapper
UML LAB
yatta
경희대학교
KYUNG HEE UNIVERSITY


Singleton Pattern – Context Handler
22
4
Singleton Pattern
restricts the instantiation of a class to one object i.e. single instance object responsible for creation, initialization, access, and enforcement.
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
Physical Activity HLC Notifier
Nutrition HLC Notifier
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Physical Activity Context Instantiator
Nutrition Context Instantiator
Physical Activity Context Synchronizer
Nutrition Context Synchronizer
Context Mapper
UML LAB
yatta
경희대학교
KYUNG HEE UNIVERSITY

Code Refactoring- Maintaining Log

225

- **Apache Log4j for Errors and Debugging**
  - Convenient and flexible logging mechanism as well as fast performance.

```
<dependency>
    <groupId>asm</groupId>
    <artifactId>asm</artifactId>
    <version>3.3.1</version>
</dependency>
<dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-api</artifactId>
        <version>${log4j.version}</version>
    </dependency>
    <dependency>
        <groupId>org.apache.logging.log4j</groupId>
        <artifactId>log4j-core</artifactId>
        <version>${log4j.version}</version>
    </dependency>
```

Before Refactoring

```java
System.out.println(
        "[HLC Notifier] DCL Notification Message: " + id + ", " + label + ", " + userID + ", " + timestampS);
try {
    System.out.println("********--------------------*******************************----------------*******************");
    System.out.println(label);
    System.out.println("rest hlc of label "+ label + " " +RestClients.addUserRecognizedHLC(351, label, timestampS));
    System.out.println("********--------------------*******************************----------------*******************");
} catch (Exception e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
```

After Refactoring using Log4j

```java
String timestampS = tutil.parseCal(dateStart);
Log.info("[HLC Notifier] DCL Notification Message: " + id + ", " + label + ", " + userID + ", " + timestampS);
try {
    Log.info("********--------------------*******************************----------------*******************");
    Log.info(label);
    Log.info("rest hlc of label "+ label + " " + RestClients.addUserRecognizedHLC(351, label, timestampS));
    Log.info("********--------------------*******************************----------------*******************");

} catch (Exception e) {
    Log.error("Issue with HLC Notifier");
}
}
```

```
2016-11-19 02:48:53 INFO  HL
2016-11-19 02:48:53 INFO  HL
2016-11-19 02:48:53 INFO  HL
2016-11-19 02:48:53 INFO  HL
2016-11-19 02:48:54 INFO  HLCNotifier:142 - [HLC Notifier: PhysicalActivityContext: ] New instance: hlc_0966046156_0000000000000000004 - Inactiv
Previous Instance: none. Notify DCL.
2016-11-19 02:48:54 INFO  HLCNotifier:235 - [HLC Notifier] DCL Notification Message: hlc_0966046156_0000000000000000004, Inactivity, 9735, 2016
10:11:17
2016-11-19 02:48:54 INFO  HLCNotifier:238 - ********--------------------*******************************----------------*******************
2016-11-19 02:48:54 INFO  HLCNotifier:239 - Inactivity
2016-11-19 02:48:54 INFO  HLCNotifier:240 - rest hlc of label Inactivity ["39755","No Error"]
2016-11-19 02:48:54 INFO  HLCNotifier:241 - ********--------------------*******************************----------------*******************
2016-11-19 03:27:21 ERROR ContextMapper:105 - [Context Mapper] LLC could not be mapped. Cake is not a valid label.
2016-11-19 03:27:45 ERROR ContextMapper:105 - [Context Mapper] LLC could not be mapped. Cake is not a valid label.
```

ALL	All levels including custom levels.
DEBUG	Designates fine-grained informational events that are most useful to debug an application.
ERROR	Designates error events that might still allow the application to continue running.
FATAL	Designates very severe error events that will presumably lead the application to abort.
INFO	Designates informational messages that highlight the progress of the application at coarse-grained level.
OFF	The highest possible rank and is intended to turn off logging.
TRACE	Designates finer-grained informational events than the DEBUG.
WARN	Designates potentially harmful situations.

HLCA Open Source Code Progress – Task 4-8

226

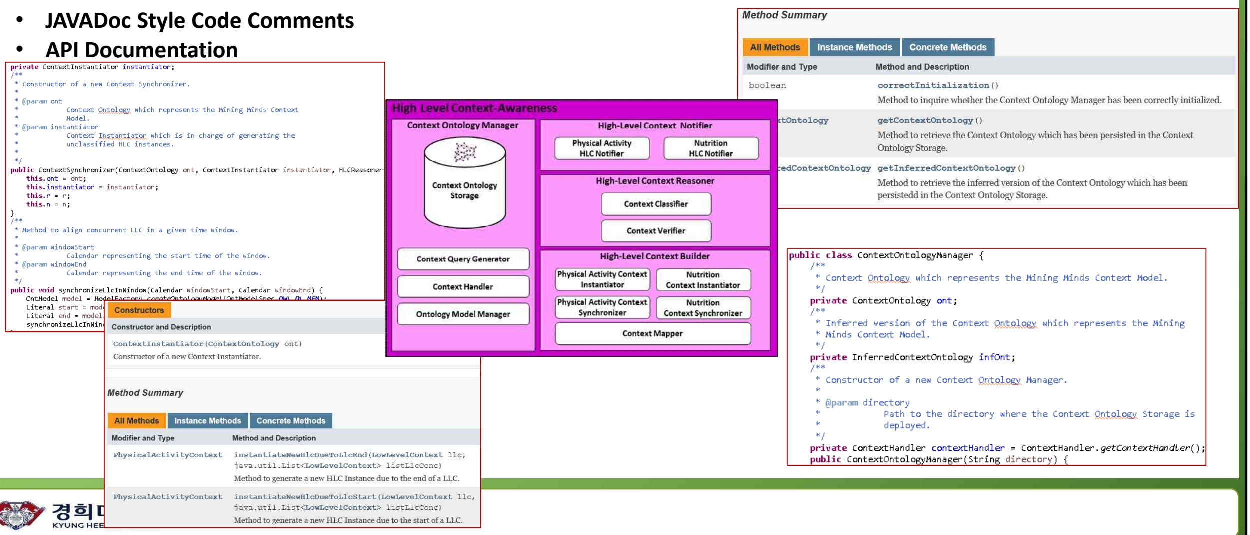
- **Design API – Task 4-5**
  - **Task Goal**
    - Unit Testing using Junit & Build Creation
    - Designing API for the ease of Developers
  - **Objective**
    - Designing API Structures which are extendable and understandable
    - Making draft of API document
- **Build Creation – Task 7**
- **API Document– Task 8**
  - **Task Goal**
    - Compose the document for API explanation
  - **Objective**
    - Making draft of API document based on Javadoc Style code comments

경희대학교
KYUNG HEE UNIVERSITY

HLCA Open Source Code Progress – Task 4-7

22
7

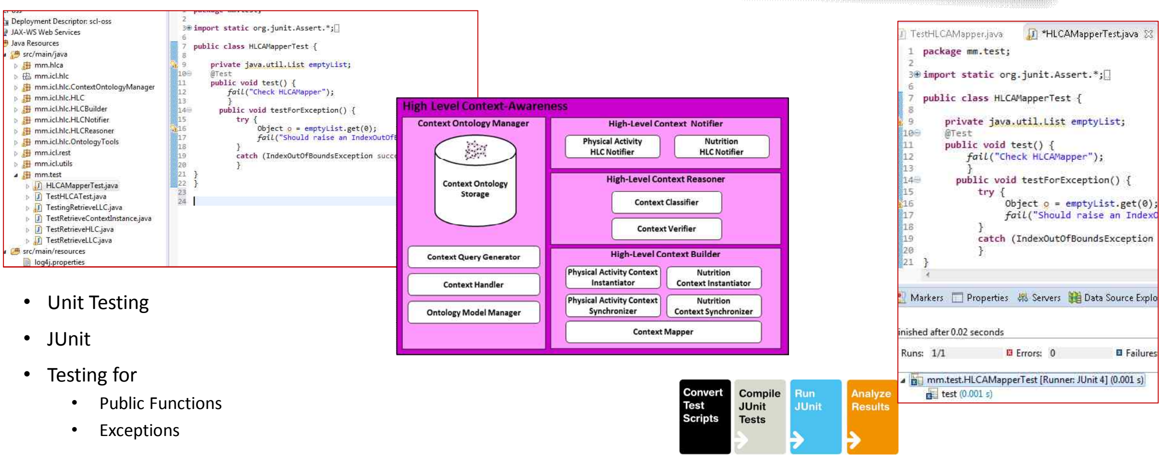
- **JAVADoc Style Code Comments**
- **API Documentation**



경희대학교
KYUNG HEE UNIVERSITY

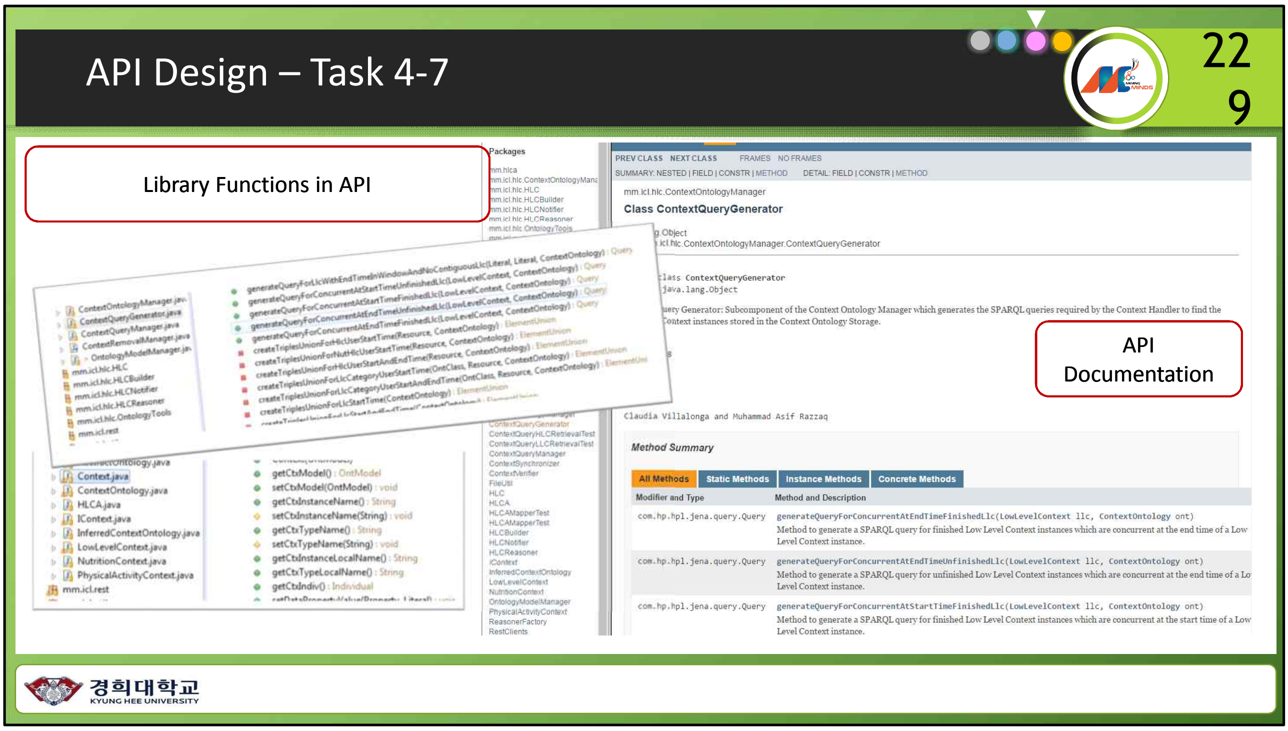
Unit Testing – Task 4-7

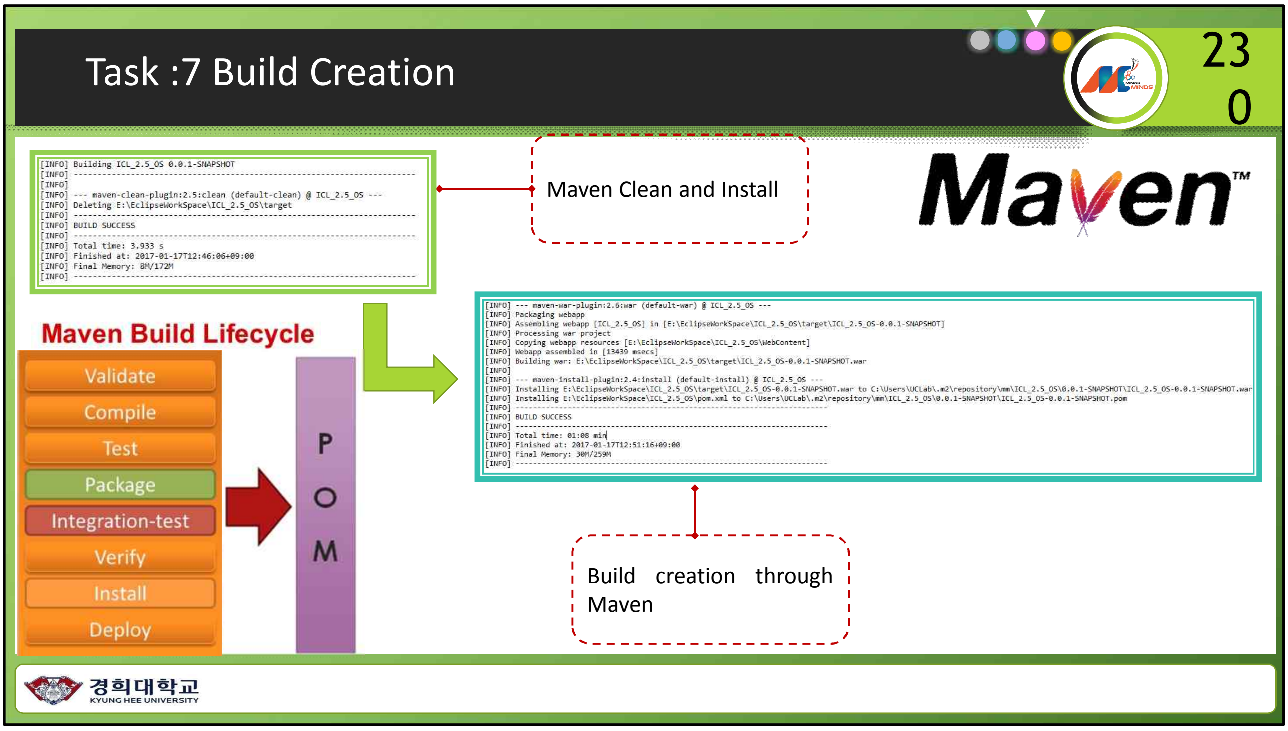
22
8



- Unit Testing
- JUnit
- Testing for
  - Public Functions
  - Exceptions

경희대학교
KYUNG HEE UNIVERSITY

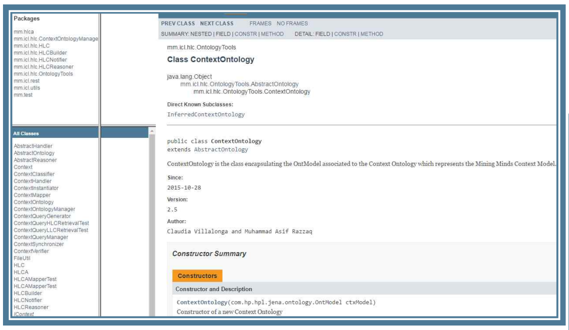

API Design – Task 4-7
22
9
Library Functions in API
API
Documentation
Class ContextQueryGenerator
Method Summary
All Methods
Static Methods
Instance Methods
Concrete Methods
경희대학교
KYUNG HEE UNIVERSITY

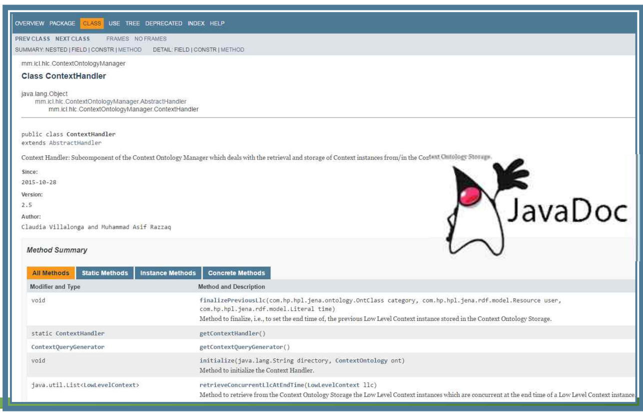

Task :7 Build Creation
23
0
Maven Clean and Install
Maven
Maven Build Lifecycle
Validate
Compile
Test
Package
Integration-test
Verify
Install
Deploy
P
O
M
Build creation through Maven
경희대학교
KYUNG HEE UNIVERSITY

Task :8 API Documentation

23
1

The Java Platform API Specification is defined by the documentation comments in the source code and any documents marked as specifications reachable from those comments.



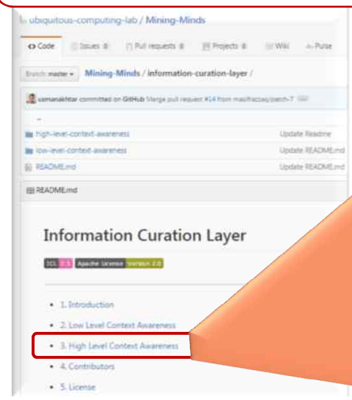


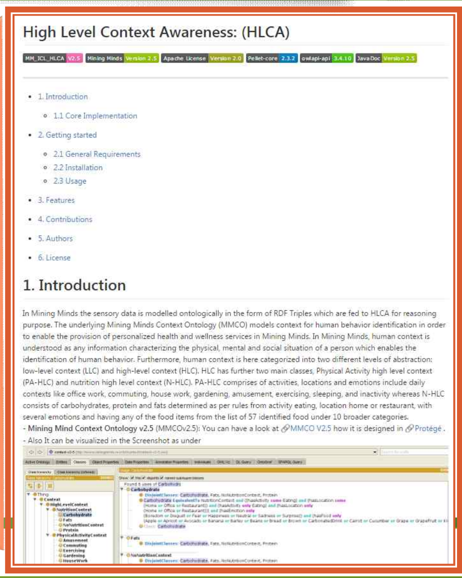
경희대학교
KYUNG HEE UNIVERSITY

Task :9 Demo Harness

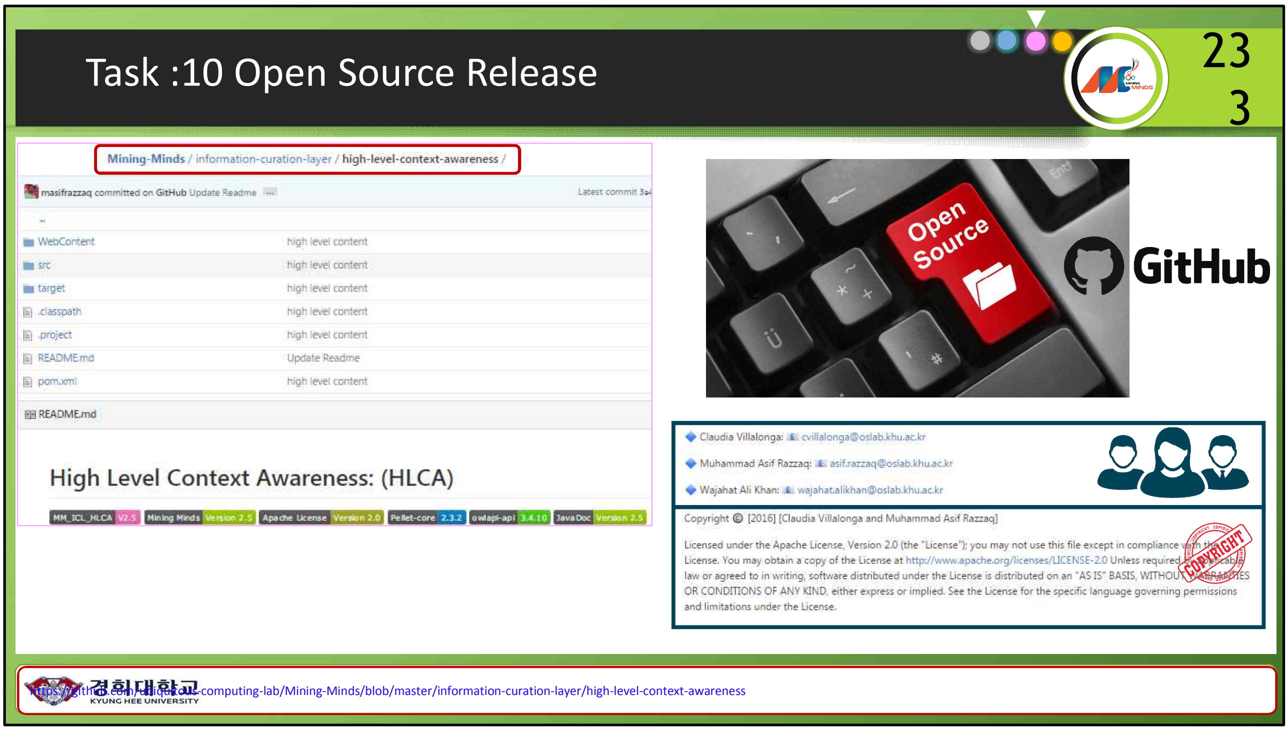
23
2

Installation and usage information available as demo harness:
https://github.com/ubiquitous-computing-lab/Mining-Minds/blob/master/information-curation-layer/high-level-context-awareness/README.md.





경희대학교
KYUNG HEE UNIVERSITY



Task :10 Open Source Release
233
Mining-Minds / information-curation-layer / high-level-context-awareness /
masifrazzaq committed on GitHub Update Readme
Latest commit
WebContent high level content
src high level content
target high level content
.classpath high level content
.project high level content
README.md Update Readme
pom.xml high level content
README.md
High Level Context Awareness: (HLCA)
MM_ICL_HLCA V2.5
Mining Minds Version 2.5
Apache License Version 2.0
Pellet-core 2.3.2
owlapi-api 3.4.10
Java Doc Version 2.5
Open Source
GitHub
Claudia Villalonga: cvillalonga@oslab.khu.ac.kr
Muhammad Asif Razzaq: asif.razzaq@oslab.khu.ac.kr
Wajahat Ali Khan: wajahat.alikhan@oslab.khu.ac.kr
Copyright © [2016] [Claudia Villalonga and Muhammad Asif Razzaq]
Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file except in compliance with the License. You may obtain a copy of the License at http://www.apache.org/licenses/LICENSE-2.0 Unless required by applicable law or agreed to in writing, software distributed under the License is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied. See the License for the specific language governing permissions and limitations under the License.
COPYRIGHT
경희대학교
KYUNG HEE UNIVERSITY
-computing-lab/Mining-Minds/blob/master/information-curation-layer/high-level-context-awareness

MINING MINDS
Mining Minds
Thank you
경희대학교
KYUNG HEE UNIVERSITY
KITECH
한국생산기술연구원
POSTECH
포항공과대학교
UNIVERSITY of TASMANIA
녹십자헬스케어
TAPACROSS
U2 system
User & Usability system
KOCSA
한국컨설팅서비스협회

MINING MINDS

Mining Minds

KCL

Knowledge Curation Layer MMV-3.0

경희대학교 KYUNG HEE UNIVERSITY | KITECH 한국생산기술연구원 | POSTECH 포항공과대학교 | UNIVERSITY of TASMANIA | 녹십자헬스케어 | TAPACROSS | U2 system User & Usability system | KOCSA 한국컨설팅서비스협회

MINING MINDS

Mining Minds

Knowledge Curation Layer Expert-Driven Knowledge Acquisition

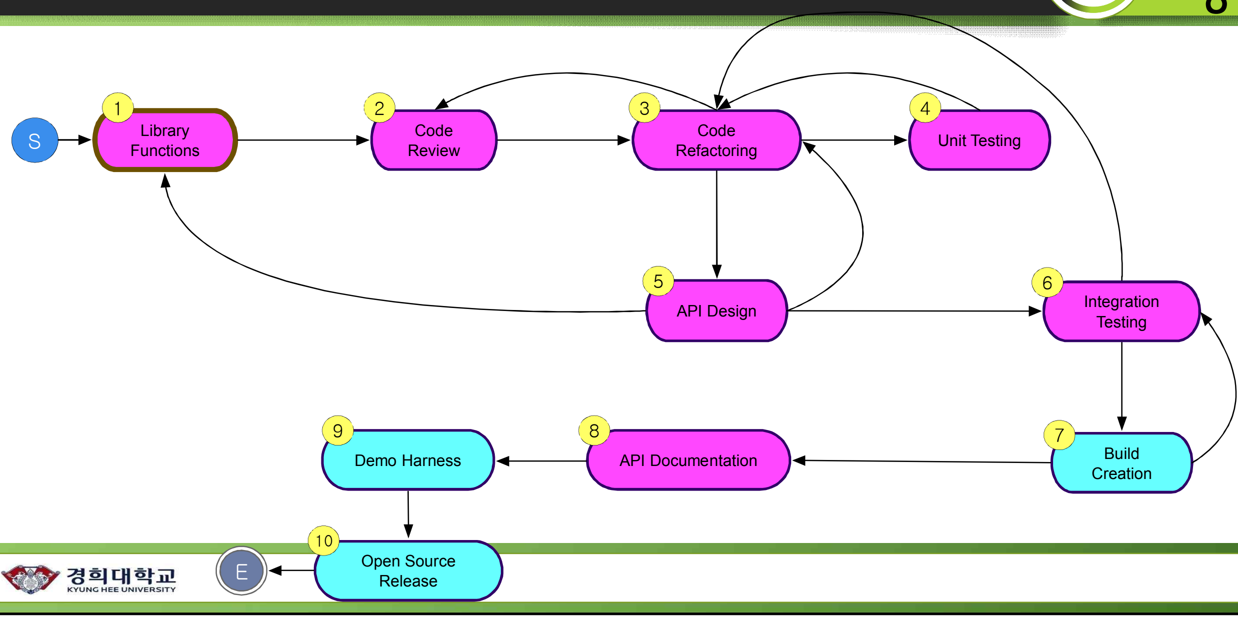
Knowledge Curation Layer MMV-3.0

경희대학교 KYUNG HEE UNIVERSITY | KITECH 한국생산기술연구원 | POSTECH 포항공과대학교 | UNIVERSITY of TASMANIA | 녹십자헬스케어 | TAPACROSS | U2 system User & Usability system | KOCSA 한국컨설팅서비스협회

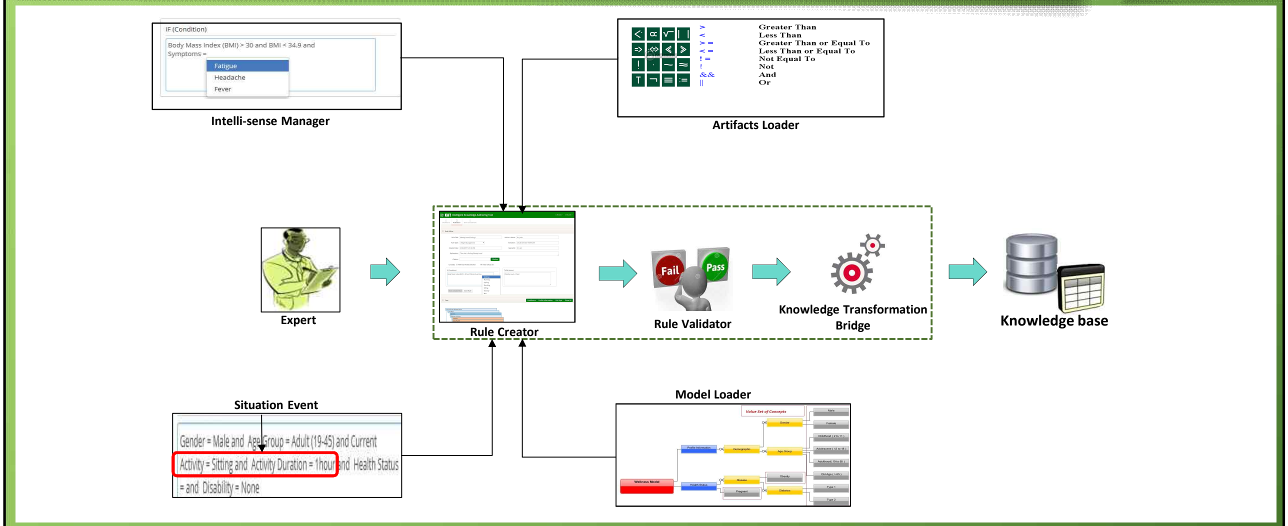
Agenda

- Open Source Roadmap
  - Library Function
  - Code Review
  - Refactoring
  - Unit Testing
  - Integration Testing
  - API Design
  - Build Creation
  - API Documentation
  - Demo Harness
  - Open Source Release

Open Source Release Process - Roadmap



Expert-Driven KA – Conceptual View



Task-1 : Library Functions

KCL: Function Identification

Public Functions

① Rule Base

Rule
- ❑ getAllRules()
- ❑ addRule()
- ❑ updateRule()
- ❑ getRuleByID()
- ❑ removeRule()
- ❑ validateRule()

Conclusions
- ❑ getAllConclusions()
- ❑ addConclusion()
- ❑ updateConclusion()
- ❑ getConclusionByID()
- ❑ removeConclusion()

Conditions
- ❑ getAllConditions()
- ❑ addCondition()
- ❑ updateCondition()
- ❑ getConditionByID()
- ❑ removeCondition()
- ❑ getConditionByFields()

Situations
- ❑ getAllSituations()
- ❑ addSituation()
- ❑ updateSituation()
- ❑ getSituationByID()
- ❑ removeSituation()

② Users
- ❑ validateUser()
- ❑ getUserByID()
- ❑ addUser()
- ❑ updateUser()
- ❑ getAllUsers()
- ❑ removeUser()

③ Wellness Model
- ❑ getAllWellnessConcepts()
- ❑ getWellnessConceptsByKey()
- ❑ addWellnessConcept()
- ❑ upateWellnessConcept()
- ❑ getWellnessConceptByID()
- ❑ removeWellnessConcept()

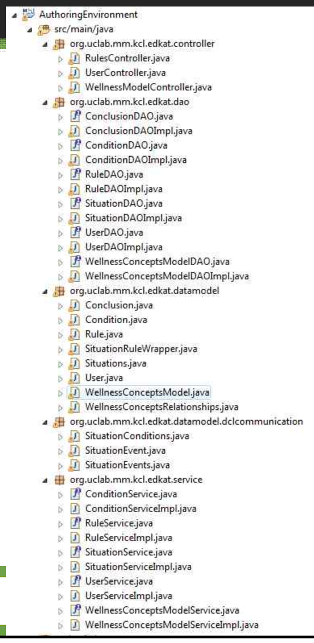
Total Public Functions: 34
Total Private Functions: 68
Total number of classes: 36

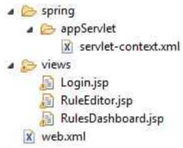
Task-1 : Library Functions

24
1

KCL: Packages Identification

- Library Function Identification
  - Packages Categorization
    - org.uclab.mm.kcl.edkat.controller
      - Controller classes to process user request and building appropriate model to render on views
    - org.uclab.mm.kcl.edkat.dao
      - Categorized classes for data access objects interfaces and their implementation
    - org.uclab.mm.kcl.edkat.datamodel
      - Provides data model classes to use in communication among different components
    - org.uclab.mm.kcl.edkat.service
      - Categorized classes of local services as interfaces and their implementations.
    - org.uclab.mm.kcl.edkat.datamodel.dclcommunication
      - Categorized classes of communication with DCL and SCL





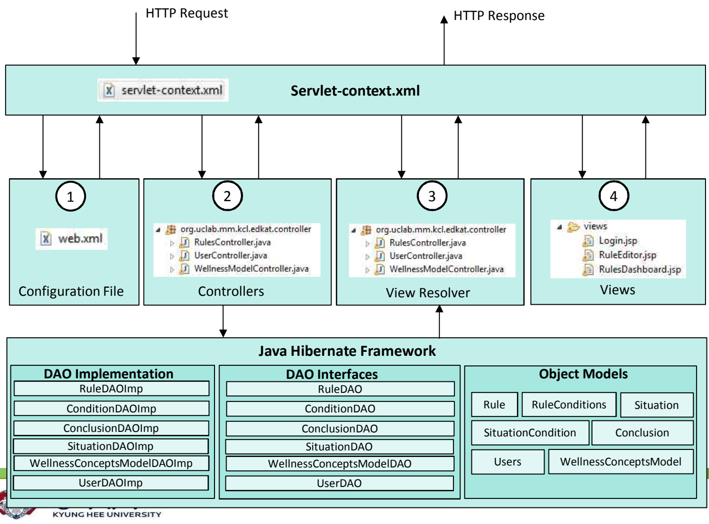
- Spring : provides Dispatcher Servlet to handle requests and responses
- Views: is responsible for rendering the model data and in general it generates HTML

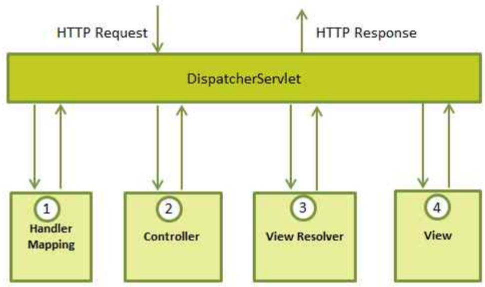
경희대학교 KYUNG HEE UNIVERSITY

Task-2 : Code Review

24
2

KCL: Design Pattern





Advantages

provides model-view-controller architecture

- To develop flexible applications
- To develop loosely coupled applications
- To provide separation in
  - Input logic
  - Business logic
  - UI logic

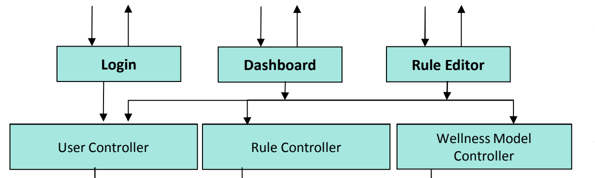
KYUNG HEE UNIVERSITY

http://www.tutorialspoint.com/spring/spring_web_mvc_framework.htm

Task-2 : Code Review

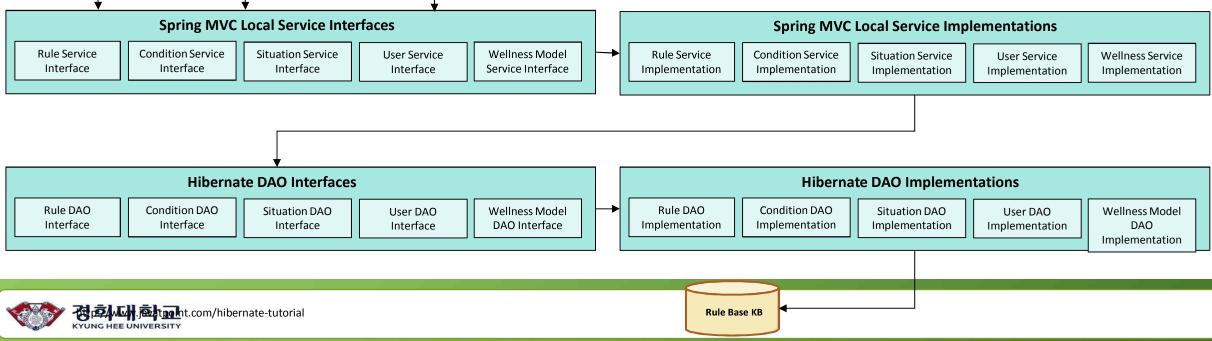
243

KCL: Design Pattern



Advantages

1) Open source and Lightweight: Hibernate framework is open source under the LGPL license and lightweight
2) Fast performance: due to two types of internal cache
3) Database Independent query: HQL (Hibernate Query Language) is the object-oriented version of SQL
4) Automatic table creation
5) Simplifies complex join
6) Provides query statistics and database status

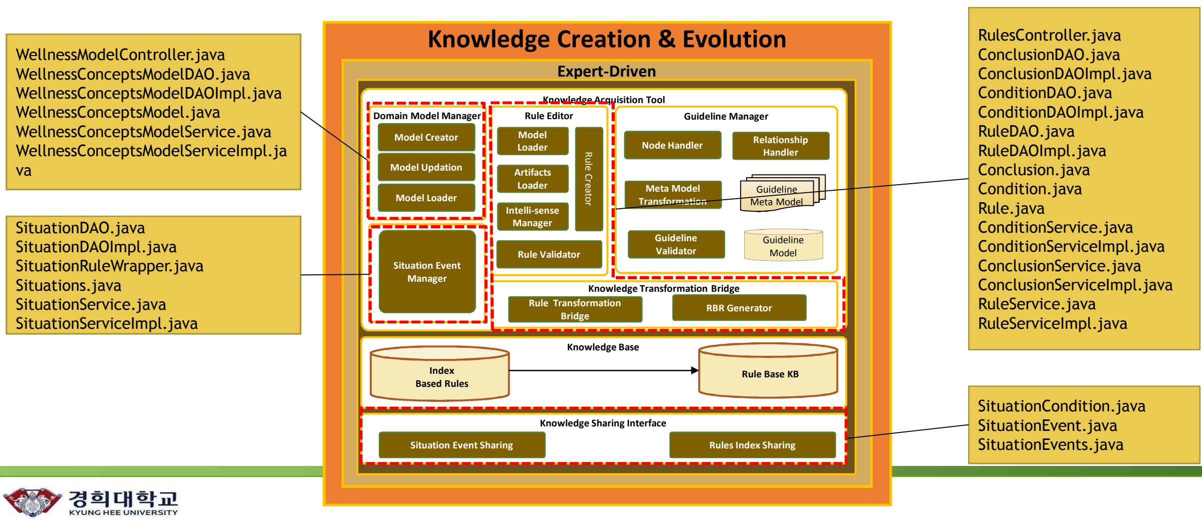


경희대학교 KYUNG HEE UNIVERSITY nt.com/hibernate-tutorial

Task-2 : Code Review

244

KCL: Component packages implementation



경희대학교 KYUNG HEE UNIVERSITY

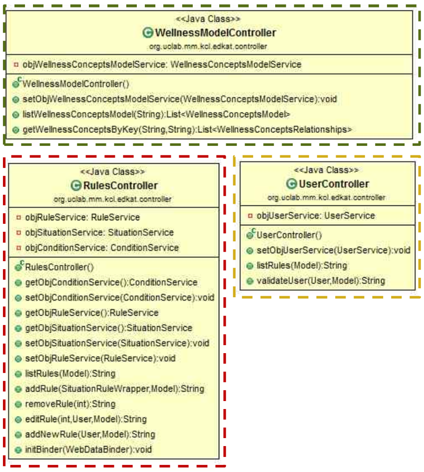
Task-2 : Code Review

245

KCL: Controller Implementation

Controllers:
are responsible for processing user requests and building appropriate model and passes it to the view for rendering.

1) Rules Controller
    1) Controls Rule
    2) Controls Conditions
    3) Controls Conclusion
    4) Control Situations
2) Wellness Model Controller
    1) Controls Wellness Concepts Model
3) User Controller
    1) Controls users



경희대학교
KYUNG HEE UNIVERSITY

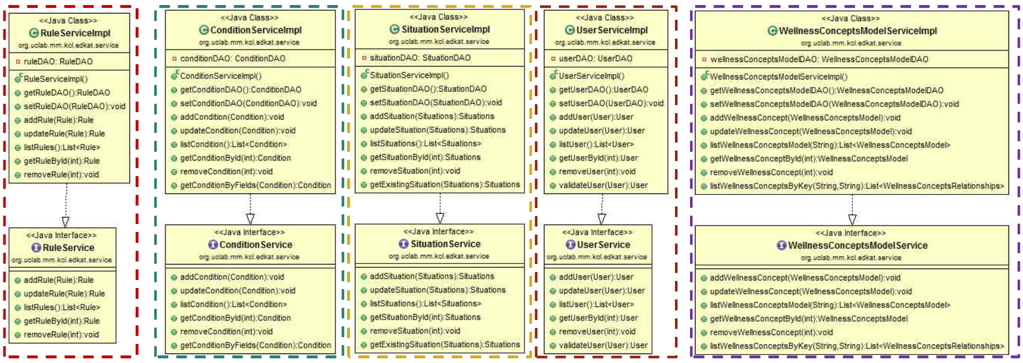
Task-2 : Code Review

246

KCL: Local Services Implementation

Local Services:
are using to communicate with java hibernate frame with DAO classes.

1) Rules Service
2) Condition Service
3) Situation Service
4) User Service
5) Wellness Concepts Service



경희대학교
KYUNG HEE UNIVERSITY

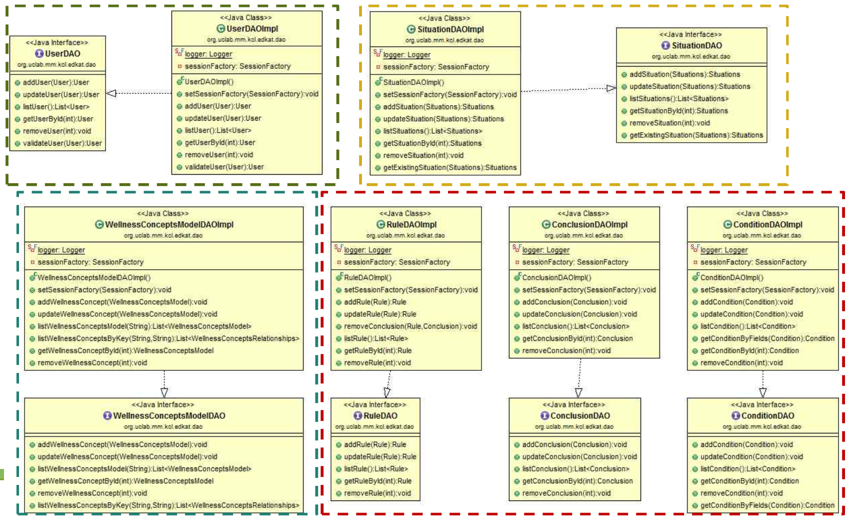
Task-2 : Code Review

247

KCL: DAO Pattern

Data Access Object:

1) Rule Data Access Object
    1) Rule DAO Interface
    2) Rule DAO Implementation
    3) Condition DAO Interface
    4) Condition DAO Implementation
    5) Conclusion DAO Interface
    6) Conclusion DAO Implementation
2) Wellness Concepts Data Access Object
    1) Wellness Concepts Model DAO Interface
    2) Wellness Concepts Model DAO Implementation
3) Situation Data Access Objects
    1) Situation DAO Interface
    2) Situation DAO Implementation
4) User Data Access Objects
    1) User DAO Interface
    2) User DAO Implementation

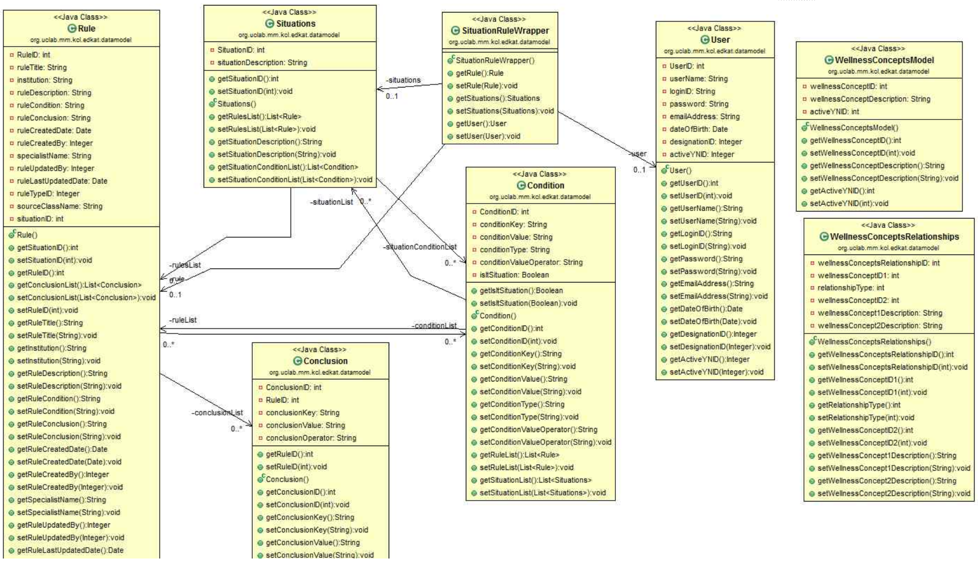


경희대학교
KYUNG HEE UNIVERSITY

Task-2 : Code Review

248

KCL: Data Model

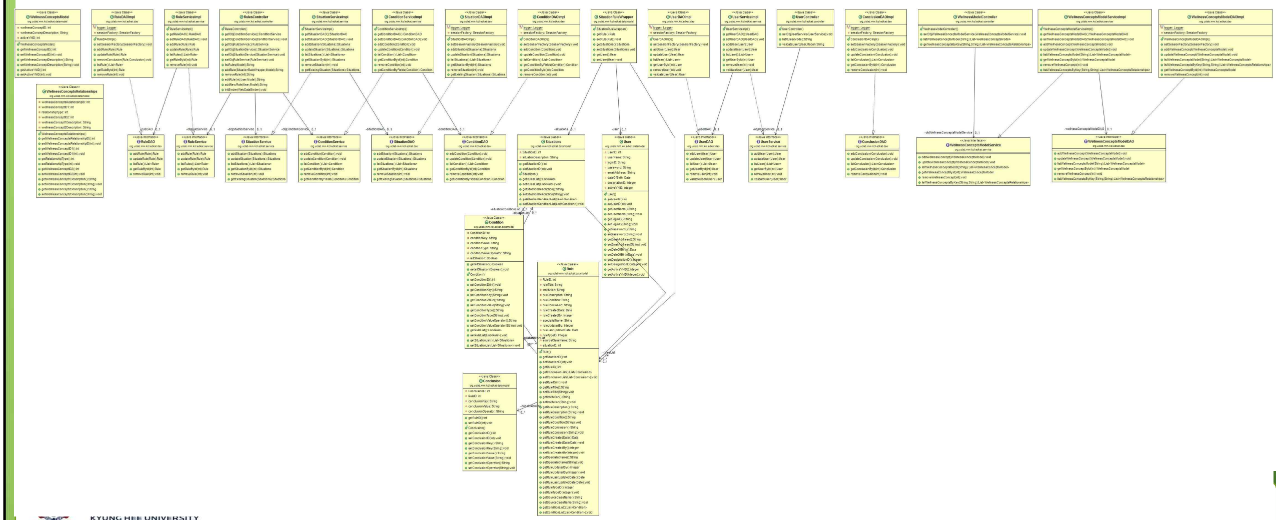


경희대학교
KYUNG HEE UNIVERSITY

Task-2 : Code Review

249

KCL: MVC Pattern



KYUNG HEE UNIVERSITY

Task-2 : Code Review

250

- Core class of Rule that contains situations and conditions

Package org.uclab.mm.kcl.edkat.controller

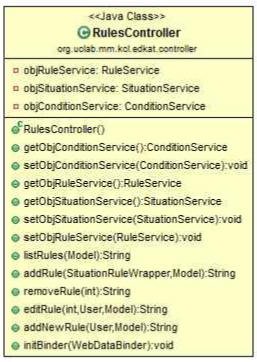
Class Summary

Class	Description
RulesController	
UserController	
WellnessModelController	

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.String	addNewRule(User user, org.springframework.ui.Model model)
java.lang.String	addRule(SituationRuleWrapper situationRuleWrapper, org.springframework.ui.Model model)
java.lang.String	editRule(int id, User user, org.springframework.ui.Model model)
ConditionService	getObjConditionService()
RuleService	getObjRuleService()
SituationService	getObjSituationService()
void	initBinder(org.springframework.web.bind.WebDataBinder binder)
java.lang.String	listRules(org.springframework.ui.Model model)
java.lang.String	removeRule(int id)
void	setObjConditionService(ConditionService objConditionService)
void	setObjRuleService(RuleService objRuleService)
void	setObjSituationService(SituationService objSituationService)



Task-2 : Code Review

org.uclab.mm.kcl.edkat.controller

Class WellnessModelController

java.lang.Object
    org.uclab.mm.kcl.edkat.controller.WellnessModelController

```
@Controller
public class WellnessModelController
extends java.lang.Object
```

<<Java Class>>
WellnessModelController
org.uclab.mm.kcl.edkat.controller

- objWellnessConceptsModelService: WellnessConceptsModelService

- WellnessModelController()
- setObjWellnessConceptsModelService(WellnessConceptsModelService):void
- listWellnessConceptsModel(String):List<WellnessConceptsModel>
- getWellnessConceptsByKey(String,String):List<WellnessConceptsRelationships>

Constructor Summary

Constructors

Constructor and Description
`WellnessModelController()`

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`java.util.List<WellnessConceptsRelationships>`	`getWellnessConceptsByKey(java.lang.String query, java.lang.String strSelectedKey)`
`java.util.List<WellnessConceptsModel>`	`listWellnessConceptsModel(java.lang.String query)`
`void`	`setObjWellnessConceptsModelService(WellnessConceptsModelService objWellnessConceptsModelService)`

Task-2 : Code Review

org.uclab.mm.kcl.edkat.controller

Class UserController

java.lang.Object
    org.uclab.mm.kcl.edkat.controller.UserController

```
@Controller
 @SessionAttributes(value="user")
public class UserController
extends java.lang.Object
```

<<Java Class>>
UserController
org.uclab.mm.kcl.edkat.controller

- objUserService: UserService

- UserController()
- setObjUserService(UserService):void
- listRules(Model):String
- validateUser(User,Model):String

Constructor Summary

Constructors

Constructor and Description
`UserController()`

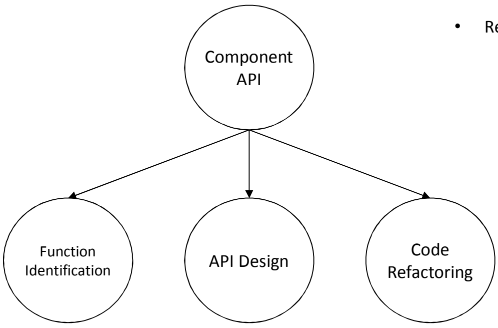
Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
`java.lang.String`	`listRules(org.springframework.ui.Model model)`
`void`	`setObjUserService(UserService objUserService)`
`java.lang.String`	`validateUser(User user, org.springframework.ui.Model model)`

Task-3 : Code Refactoring

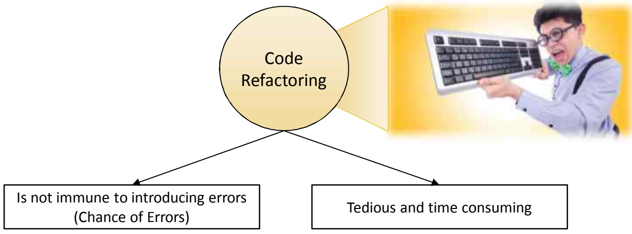
253



- Refactoring, as originally defined by Martin Fowler and Kent Beck

  "A change made to the internal structure of software to make it easier to understand and cheaper to modify without changing its observable behavior… It is a disciplined way to clean up code that minimizes the chances of introducing bugs"

An emerging software development activity



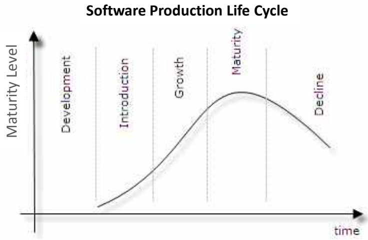
https://www.javacodegeeks.com/2012/04/what-refactoring-is-and-what-it-isnt.html

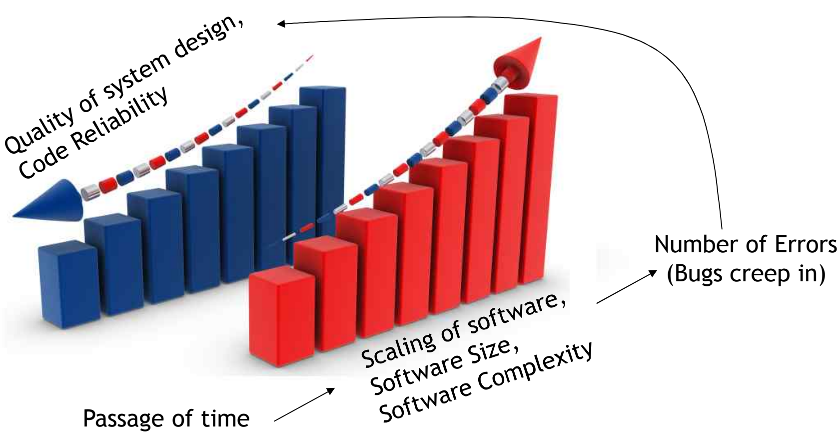
Task-3 : Code Refactoring

254

Who needs refactoring?

To avoid →





경희대학교 KYUNG HEE UNIVERSITY

Task-3 : Code Refactoring

Refactoring Process

Step		
Detect a problem	01	Is there a problem? What is the problem?
Characterize the problem	02	Why is it necessary to change something? What are the benefits? Are there any risk?
Design a solution	03	What should be the “goal state” of the code? Which code transformation(s) will move the code towards the desired state?
Modify the code	04	Steps that will carry out the code transformation(s) that leave the code functioning the same way as it did before?

Task-3 : Code Refactoring

Refactoring Benefits

- Easier to spot the bug location
- Time saving due to easy finding of bug location
- Enhance quality of code
- Well-structured code is less error-prone
- Martin Fowler [Fowler, p.60] states that
  - Hard to read → Hard to modify
  - Duplicate Logic → Hard to modify
  - Changing running code → Hard to modify
  - Complex conditional logic → Hard to modify

Refactoring

Refactoring Examples

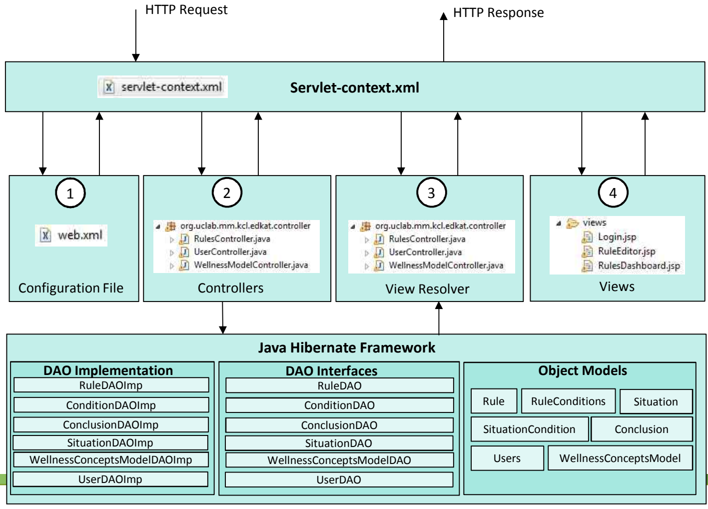
- **Rename:** Changing name of misleading and confusing method, variable, class, or any other item
- **Move Class:** Moving a class from wrong package to better fit package
- **Extract Methods:** A long method needs to be broken up to enhance readability and maintainability
- **Extract Supper Class:** The common behavior is “pulled up” into new parent abstract class.
- **Replace conditional with polymorphism:** Replace multiple conditional operators (If, Switch) with polymorphism to reduce extensive change in code.
- **Exception Handling:** Handle exception at each level of code.

Task-3 : Code Refactoring

25
7

KCL: API Design

- **MVC Pattern**
- **Hibernate Framework**



MVC Advantages

provides model-view-controller architecture

- To develop flexible applications
- To develop loosely coupled applications
- To provide separation in
  - Input logic
  - Business logic
  - UI logic

Hibernate Advantages

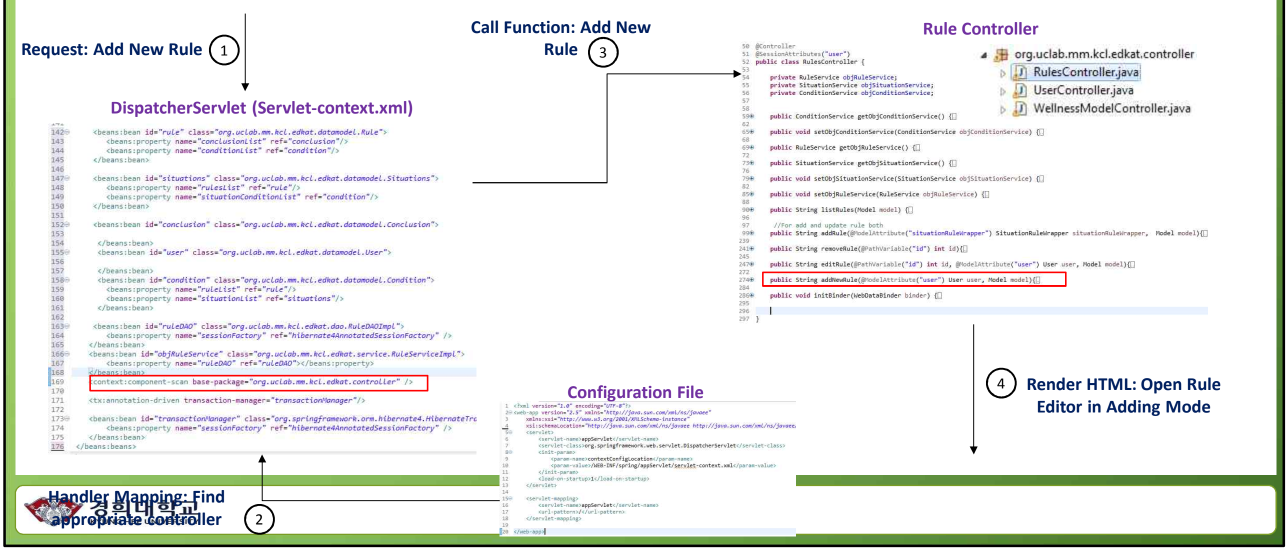
1) Open source and Lightweight: Hibernate framework is open source under the LGPL license and lightweight
2) Fast performance: due to two types of internal cache
3) Database Independent query: HQL (Hibernate Query Language) is the object-oriented version of SQL
4) Automatic table creation
5) Simplifies complex join
6) Provides query statistics and database status

경희대학교 KYUNG HEE UNIVERSITY

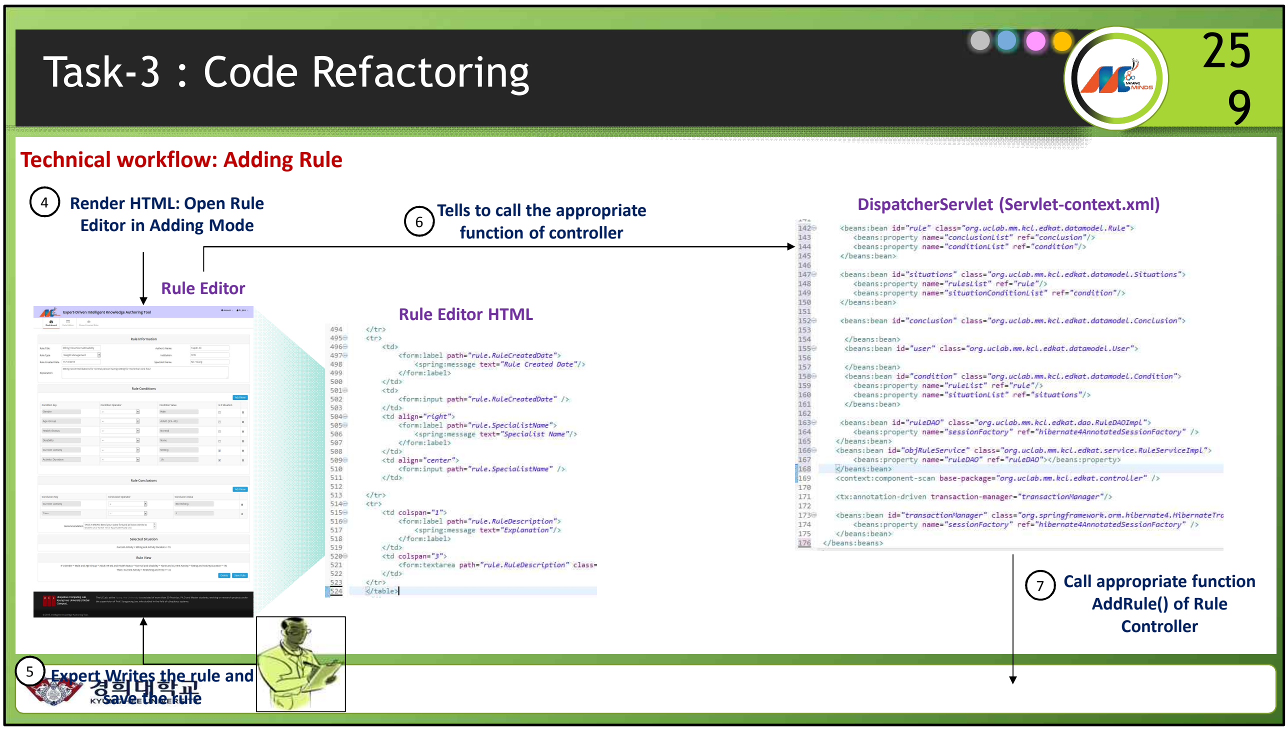
Task-3 : Code Refactoring

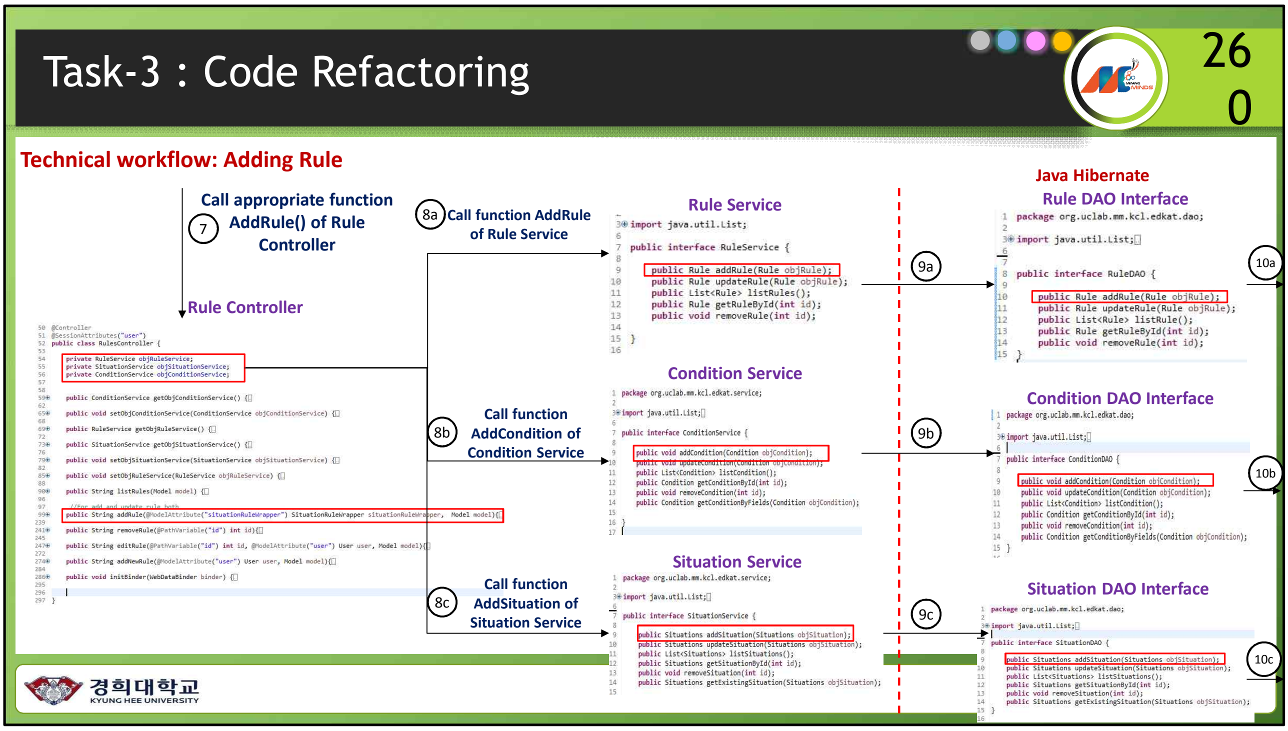
25
8

Technical workflow: Adding Rule



경희대학교 KYUNG HEE UNIVERSITY

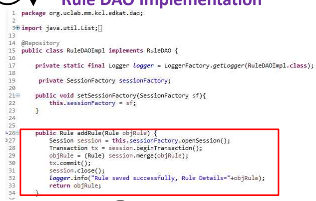

Task-3 : Code Refactoring
259
Technical workflow: Adding Rule
4 Render HTML: Open Rule Editor in Adding Mode
6 Tells to call the appropriate function of controller
DispatcherServlet (Servlet-context.xml)
Rule Editor
Rule Editor HTML
7 Call appropriate function AddRule() of Rule Controller
5 Expert Writes the rule and save the rule
경희대학교
KYUNG HEE UNIVERSITY


Task-3 : Code Refactoring
260
Technical workflow: Adding Rule
7 Call appropriate function AddRule() of Rule Controller
Rule Controller
8a Call function AddRule of Rule Service
8b Call function AddCondition of Condition Service
8c Call function AddSituation of Situation Service
Rule Service
Condition Service
Situation Service
9a
9b
9c
Java Hibernate
Rule DAO Interface
Condition DAO Interface
Situation DAO Interface
10a
10b
10c
경희대학교
KYUNG HEE UNIVERSITY

Task-3 : Code Refactoring

Technical workflow: Adding Rule

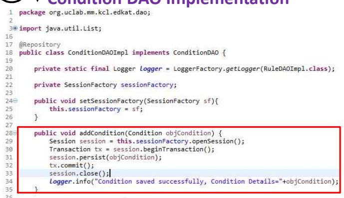
10a — **Rule DAO Implementation**



11a — **Rule Table**

tblRules
- RuleID
- RuleTitle
- Institution
- RuleDescription
- RuleCondition
- RuleConclusion
- RuleCreatedDate
- RuleCreatedBy
- SpecialistName
- RuleUpdatedBy
- RuleLastUpdatedDate
- RuleTypeID

10b — **Condition DAO Implementation**

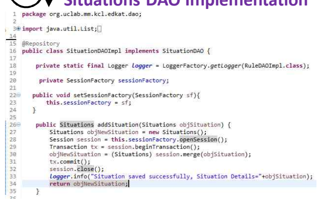


11b — **Rule Condition Tables**

tblRulesConditions
- RulesConditionID
- RuleID
- ConditionID

tblConditions
- ConditionID
- ConditionKey
- ConditionValue
- ConditionType
- ConditionValueOperator

10c — **Situations DAO Implementation**



11c — **Situation Tables**

tblSituations
- SituationID
- SituationDescription

tblSituationConditions
- SituationConditionID
- SituationID
- ConditionID

Task-3 : Code Refactoring

KCL: Identified Problems

Refactoring Examples

- **Rename:** Changing name of misleading and confusing method, variable, class, or any other item
- **Scalable and flexible Design Pattern:** Using a design pattern for scalability and flexibility.
- **Move Class:** Moving a class from wrong package to better fit package
- **Extract Methods:** A long method needs to be broken up to enhance readability and maintainability
- **Extract Supper Class:** The common behavior is "pulled up" into new parent abstract class.
- **Replace conditional with polymorphism:** Replace multiple conditional operators (If, Switch) with polymorphism to reduce extensive change in code.
- **Exception Handling:** Handle exception at each level of code.

Already Implemented maximum good practices of Software Engineering

(100% Implemented)

65% Implemented

Identified Problems

1. Exception Handling
2. Remove unnecessary Comments and spaces
3. Put functional comments

Task-3 : Code Refactoring

KCL: Code Refactoring

① Exception Handling

- 01 Detect a problem
- 02 Characterize the problem
- 03 Design a solution
- 04 Modify the code

Is there a problem? Yes.
What is the problem? 35% of exceptions are not handled.

What are the benefits? It will handle all type of exception when it occurs.
Are there any risk? Yes code crash risk is high.

What should be the “goal state” of the code? Catch all types of exception.
Which code transformation(s) will move the code towards the desired state? All the interface level code will be change.

Steps that will carry out the code transformation(s) that leave the code functioning the same way as it did before? The functions at interface level will be put in try-catch, it will not disturb the external behavior.

Task-3 : Code Refactoring

KCL: Code Refactoring

Before Refactoring

```java
public Rule addRule(Rule objRule) {
    Session session = this.sessionFactory.openSession();
    Transaction tx = session.beginTransaction();
    objRule = (Rule) session.merge(objRule);
    tx.commit();
    session.close();
    logger.info("Rule saved successfully, Rule Details="+objRule);
    return objRule;
}

public Rule updateRule(Rule objRule) {

    Session session = this.sessionFactory.openSession();
    Transaction tx = session.beginTransaction();
    session.update(objRule); // Replaced with updated rule
    tx.commit();
    session.close();
    logger.info("Rule updated successfully, Rule Details="+objRule);
    return objRule;
}
```

After Refactoring

```java
public Rule addRule(Rule objRule) {
    try
    {
        Session session = this.sessionFactory.openSession();
        Transaction tx = session.beginTransaction();
        objRule = (Rule) session.merge(objRule);
        tx.commit();
        session.close();
        logger.info("Rule saved successfully, Rule Details="+objRule);
        return objRule;
    }
    catch(Exception ex)
    {
        logger.info("Error occured in saving new rule, Error Details="+ex.getMessage());
        return objRule;
    }

}

public Rule updateRule(Rule objRule) {
    try
    {
        Session session = this.sessionFactory.openSession();
        Transaction tx = session.beginTransaction();
        session.update(objRule); // Replaced with updated rule
        tx.commit();
        session.close();
        logger.info("Rule updated successfully, Rule Details="+objRule);
        return objRule;
    }
    catch(Exception ex)
    {
        logger.info("Error occured in updating rule, Error Details="+ex.getMessage());
        return objRule;
    }

}
```

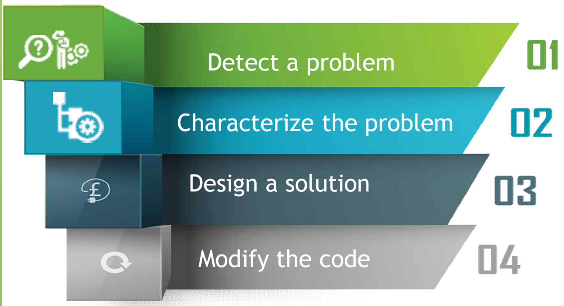
Task-3 : Code Refactoring

26 5

KCL: Code Refactoring

2

Remove unnecessary Comments and spaces



Is there a problem? Yes.
What is the problem? There are some unnecessary committed lines.

What are the benefits? Enhance cleanliness of the code.
Are there any risk? No, there is no risk.

What should be the “goal state” of the code? To get the clean and clear code for open source.
Which code transformation(s) will move the code towards the desired state? All the classes, they have unnecessary comments.

Steps that will carry out the code transformation(s) that leave the code functioning the same way as it did before? We will remove all the committed lines and it will not disturb the behavior of the functions.

경희대학교
KYUNG HEE UNIVERSITY

Task-3 : Code Refactoring

26 6

Before Refactoring

```
 //For add and update rule both
@RequestMapping(value= "/rules/add", method = RequestMethod.POST)
public String addRule(@ModelAttribute("situationRuleWrapper") SituationRuleWrapper situationRuleWrapper,  Model model)
    Rule objRule = new Rule();
    Situations objSituation = new Situations();
    User objUser = new User();

    objSituation = situationRuleWrapper.getSituations();
    objRule = situationRuleWrapper.getRule();
    objUser = situationRuleWrapper.getUser();

    // Start Preparation of Situation and Existing conditions

    String strSituationDesc = "";
    boolean blnIsSituationAttached = false;
    for(int i = 0; i < objRule.getConditionList().size(); i++){
        Condition objCondition = objRule.getConditionList().get(i);
        boolean blnIsItSituation = objCondition.getIsItSituation();
        objCondition = objConditionService.getConditionByFields(objCondition);
        //if(objCondition.getConditionID() != 0)
        //{
            //objRule.getConditionList().set(i, objCondition);
            objRule.getConditionList().get(i).setConditionID(objCondition.getConditionID());
            objRule.getConditionList().get(i).setConditionKey(objCondition.getConditionKey());
            objRule.getConditionList().get(i).setConditionValue(objCondition.getConditionValue());
            objRule.getConditionList().get(i).setConditionValueOperator(objCondition.getConditionValueOperator());
        //}

        if(blnIsItSituation)
        {
            blnIsSituationAttached = true;
            objSituation.getSituationConditionList().add(objCondition);
            if(strSituationDesc.equals(""))
            {
                strSituationDesc = objCondition.getConditionKey() + " " + objCondition.getConditionValueOperator() + "
            }
            else
            {
                strSituationDesc = strSituationDesc + " and " +  objCondition.getConditionKey() + " " + objCondition.g
            }
        }
```

After Refactoring

```
@RequestMapping(value= "/rules/add", method = RequestMethod.POST)
public String addRule(@ModelAttribute("situationRuleWrapper") SituationRuleWrapper situationRuleWrapper,  Model model){
    Rule objRule = new Rule();
    Situations objSituation = new Situations();
    User objUser = new User();

    objSituation = situationRuleWrapper.getSituations();
    objRule = situationRuleWrapper.getRule();
    objUser = situationRuleWrapper.getUser();

    // Start Preparation of Situation and Existing conditions

    String strSituationDesc = "";
    boolean blnIsSituationAttached = false;
    for(int i = 0; i < objRule.getConditionList().size(); i++){
        Condition objCondition = objRule.getConditionList().get(i);
        boolean blnIsItSituation = objCondition.getIsItSituation();
        objCondition = objConditionService.getConditionByFields(objCondition);
        objRule.getConditionList().get(i).setConditionID(objCondition.getConditionID());
        objRule.getConditionList().get(i).setConditionKey(objCondition.getConditionKey());
        objRule.getConditionList().get(i).setConditionValue(objCondition.getConditionValue());
        objRule.getConditionList().get(i).setConditionValueOperator(objCondition.getConditionValueOperator());

        if(blnIsItSituation)
        {
            blnIsSituationAttached = true;
            objSituation.getSituationConditionList().add(objCondition);
            if(strSituationDesc.equals(""))
            {
                strSituationDesc = objCondition.getConditionKey() + " " + objCondition.getConditionValueOperator() + "
            }
            else
            {
                strSituationDesc = strSituationDesc + " and " +  objCondition.getConditionKey() + " " + objCondition.ge
            }
        }

    }

    // End Preparation of Situation and Existing conditions
```

경희대학교
KYUNG HEE UNIVERSITY

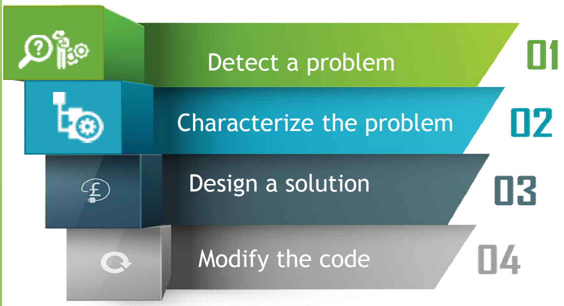
Task-3 : Code Refactoring

26
7

KCL: Code Refactoring

③

Put functional comments



Is there a problem? Yes.
What is the problem? Some functional comments are needed on each function.

What are the benefits? Enhance understandability of the code.
Are there any risk? No, there is no risk.

What should be the “goal state” of the code? To easily understand the code by other developers.
Which code transformation(s) will move the code towards the desired state? All functions of each class need comments.

Steps that will carry out the code transformation(s) that leave the code functioning the same way as it did before? We will put the functional comments on each function and it will not disturb the behavior of the functions.

경희대학교 KYUNG HEE UNIVERSITY

Task-3 : Code Refactoring

26
8

Function Level Comments

```
    /*
     * This function is the implementation for retrieving a single rule by id
     */
    public Rule getRuleById(int id) {
        try
        {
            Session session = this.sessionFactory.openSession();
            Rule objRule = (Rule) session.createQuery("from Rule r left outer join fetch r.c
            logger.info("Rule loaded successfully, Rule details="+objRule);
            session.close();
            return objRule;
        }
        catch(Exception  ex )
        {
            logger.info("Error occurred in rule loading "+ ex.getMessage());
            return null;
        }
    }

    /*
     * This function is the implementation for deleting existing selected rule
     */
    public void removeRule(int id) {
        try
        {
             Session session = this.sessionFactory.openSession();
             Rule objRule = (Rule) session.load(Rule.class, new Integer(id));
                if(null != objRule){
                    session.delete(objRule);
                }
                session.close();
                logger.info("Rule deleted successfully, Rule details="+objRule);
        }
        catch(Exception  ex )
        {
            logger.info("Error occurred in rule deleting "+ ex.getMessage());
        }

    }
```

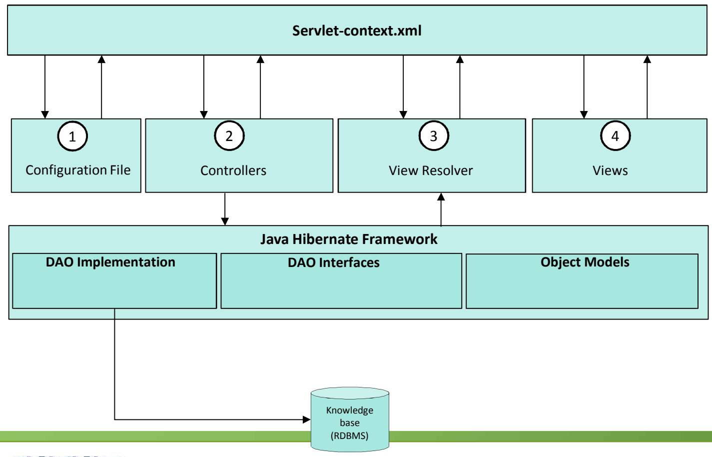
Class Level Comments

```
import org.hibernate.Session;
import org.hibernate.SessionFactory;
import org.hibernate.Transaction;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.stereotype.Repository;
import org.uclab.mm.kcl.edkat.datamodel.Conclusion;
import org.uclab.mm.kcl.edkat.datamodel.Rule;

/*
* This is the Data Access Object implementation class for the rule, this DAO implements rule dao
* interface to create, update, retrieve the rules with conditions, conclusions, and situations
* */

@Repository
public class RuleDAOImpl implements RuleDAO {

    /*
     * This creation of static object of logger for Rule
     */
    private static final Logger logger = LoggerFactory.getLogger(RuleDAOImpl.class);

    /*
     * This creation of Session for factory
     */
     private SessionFactory sessionFactory;

    public void setSessionFactory(SessionFactory sf){
        this.sessionFactory = sf;
    }

    /*
     * This function is the implementation for add new rule
     */
    public Rule addRule(Rule objRule) {
        try
        {
```

Task-4 : Unit Testing

269

Junit Testing



- Used Technologies
  - Spring MVC
  - Hibernate
  - RDBM (SQL Server)

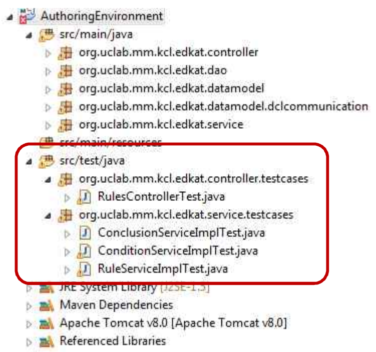
- Required for Unit Testing
  - Controllers are not able to test on Junit testing (Due to URI calls)
  - Only DAO classes (Hibernate) can test
  - Need in-memory database
    - HSQLDB

경희대학교 KYUNG HEE UNIVERSITY

Task-4 : Unit Testing

270

- Unit Testing
- JUnit
- Testing for
  - Public Functions
    - Data Range Values {Test Classes}
  - Exceptions



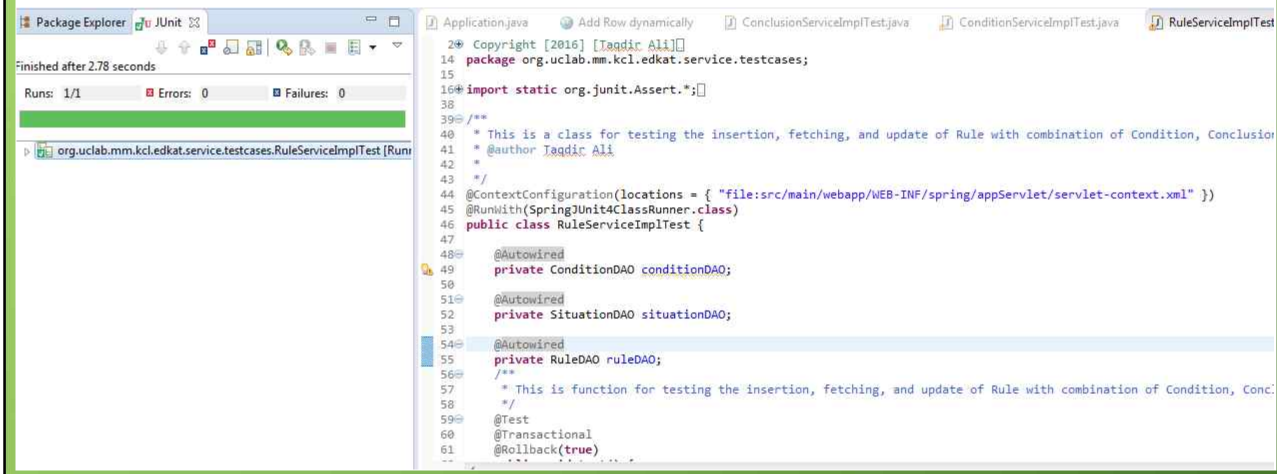
JUnit Testing Framework

경희대학교 KYUNG HEE UNIVERSITY

Task-4 : Unit Testing

271

Junit Testing – Rule Testing

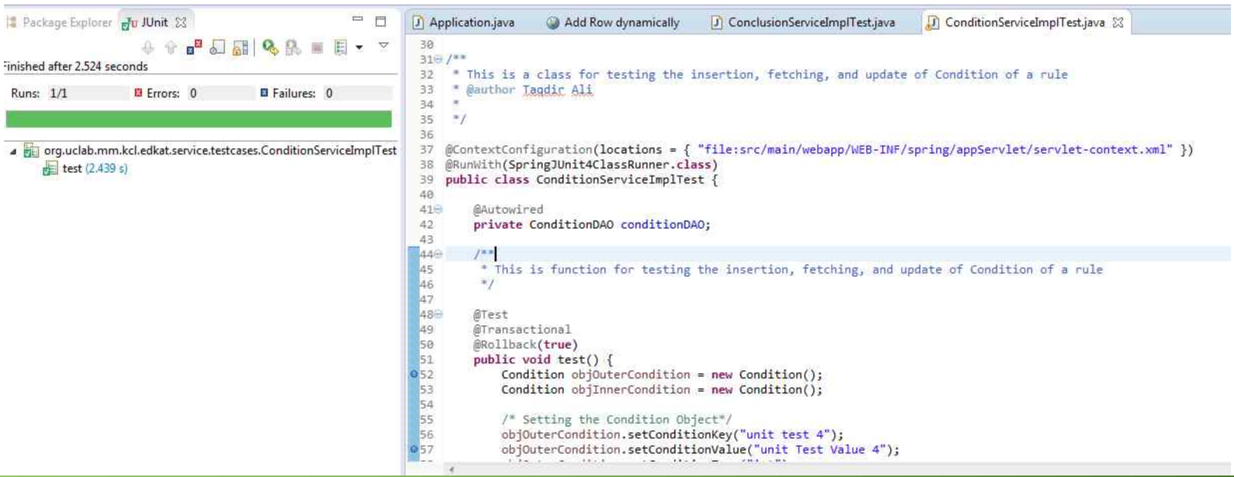


경희대학교
KYUNG HEE UNIVERSITY

Task-4 : Unit Testing

272

Junit Testing – Condition Testing

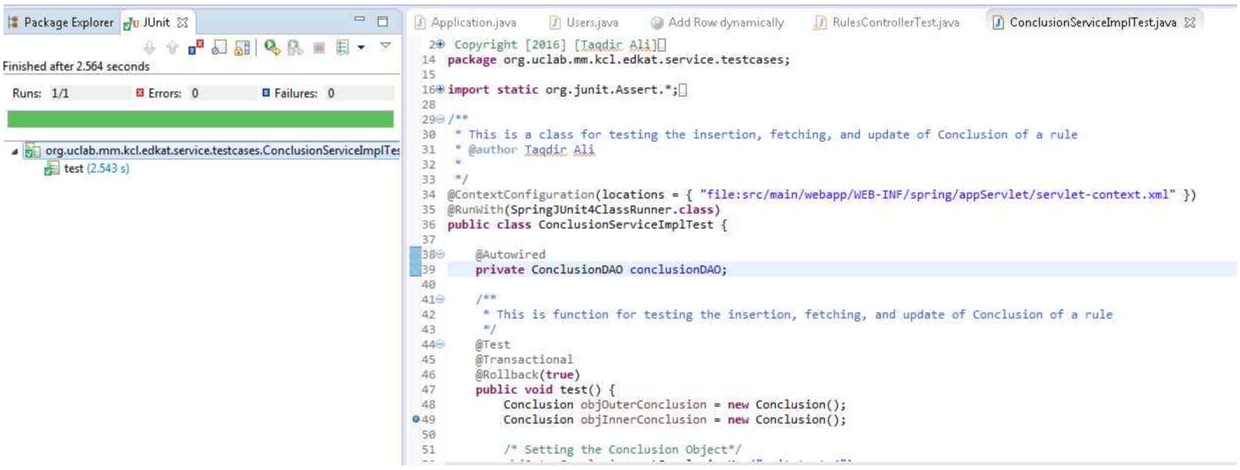


경희대학교
KYUNG HEE UNIVERSITY

Task-4 : Unit Testing

273

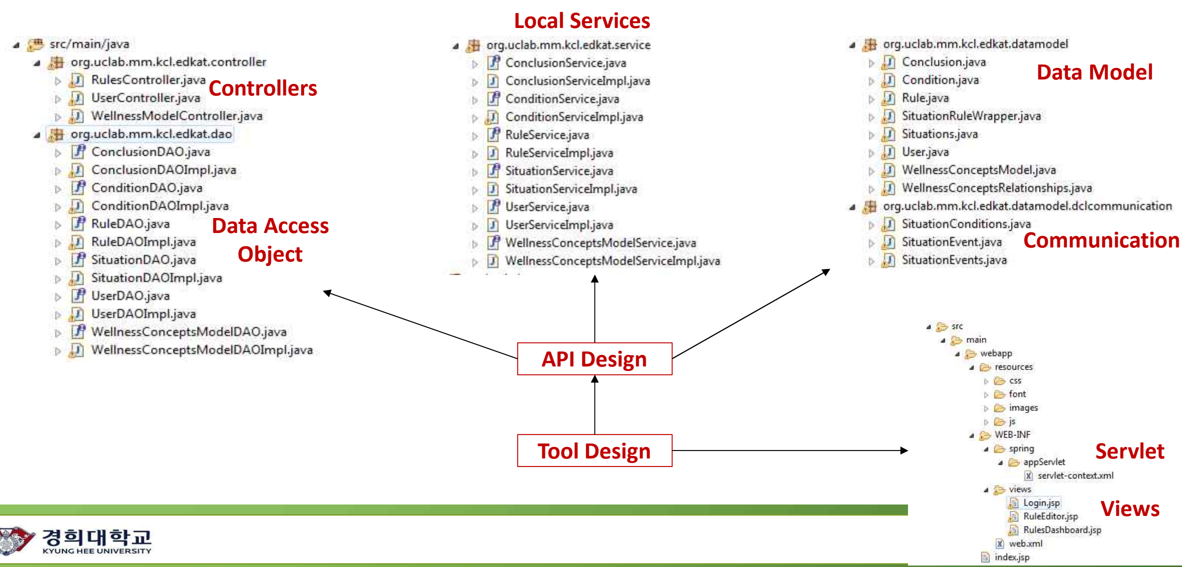
Junit Testing – Conclusion Testing



경희대학교
KYUNG HEE UNIVERSITY

Task-5 : API Design

274

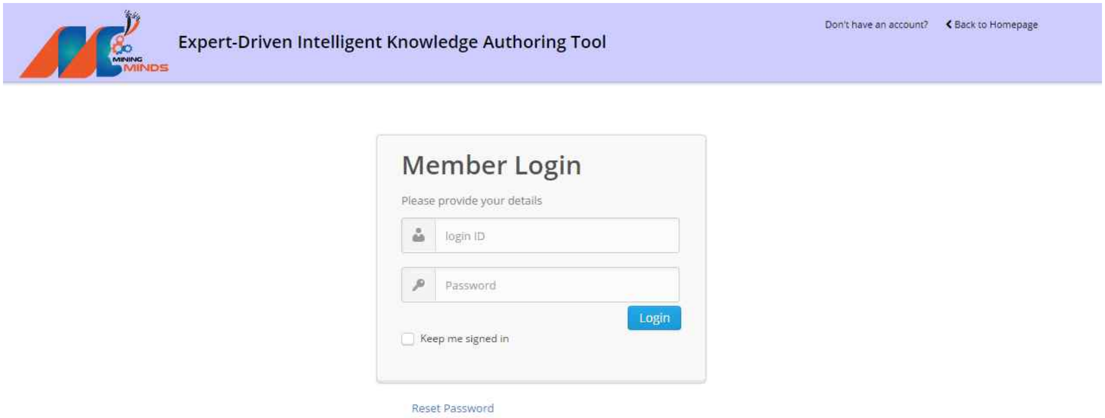


경희대학교
KYUNG HEE UNIVERSITY

Task-6 : Integration Test

275

1. Successfully login to System

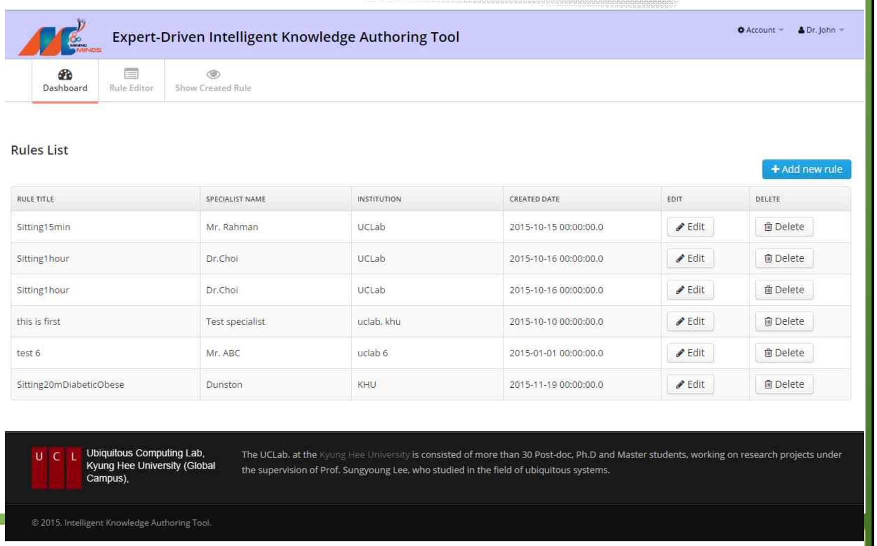


경희대학교
KYUNG HEE UNIVERSITY

Task-6 : Integration Test

276

1. **View and analyze existing rules using dashboard**
2. **Test editing of existing rule**
3. **Test creating new rule by add new rule button**
4. **Test deletion of rule**

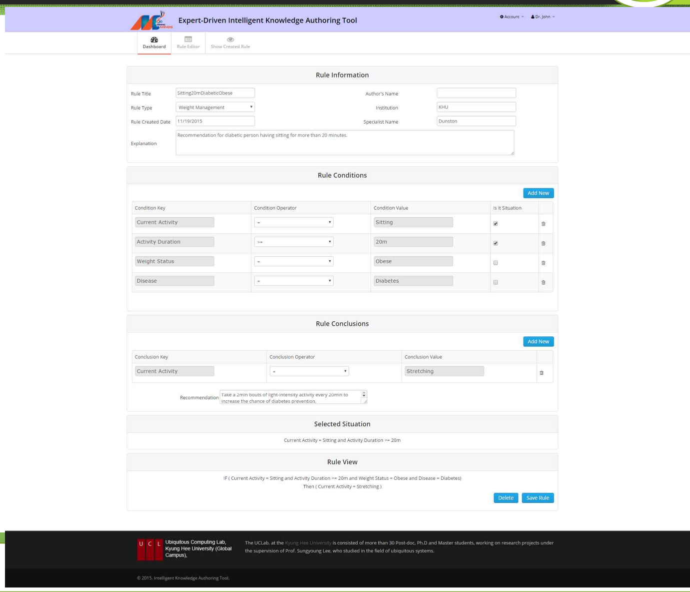


경희대학교
KYUNG HEE UNIVERSITY

Task-6 : Integration Test

27
7

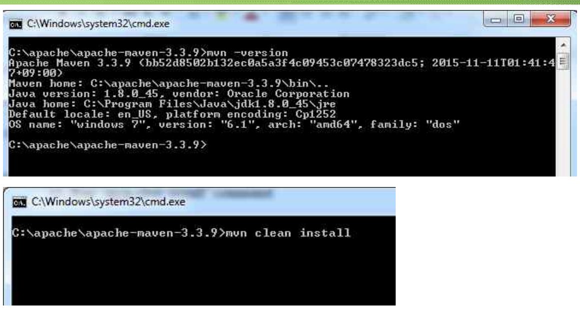
1. Test inserting meta information
2. Adding multiple conditions using Add new button
3. Add multiple conclusion
4. Test selection of situation
5. Test the saving new and existing rule



경희대학교
KYUNG HEE UNIVERSITY

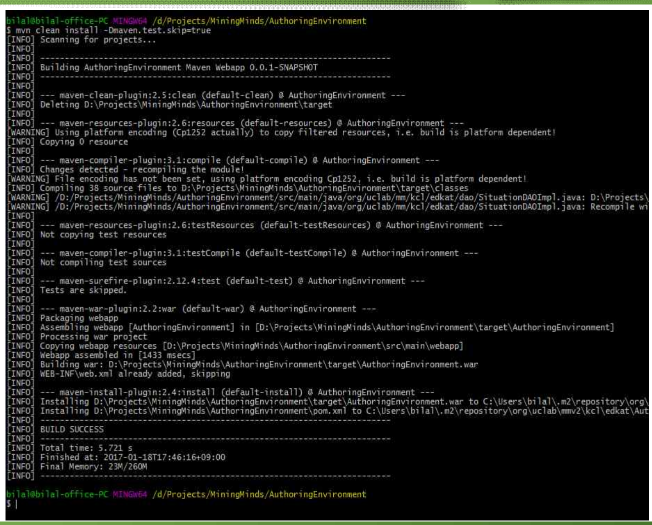
Task-7 : Build Creation

27
8



Maven install and start

- Maven lets us get our package dependencies easily
- Maven forces us to have a standard directory structure



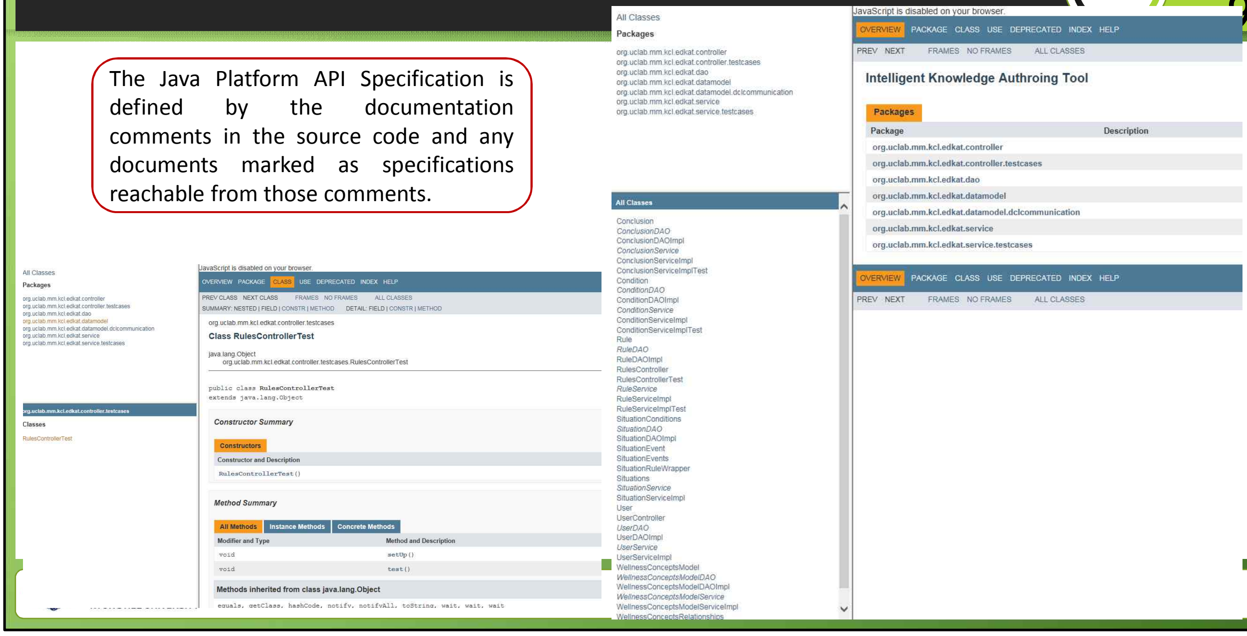
경희대학교
KYUNG HEE UNIVERSITY

Build creation through Maven

Task-8 : API Documentation

27
9

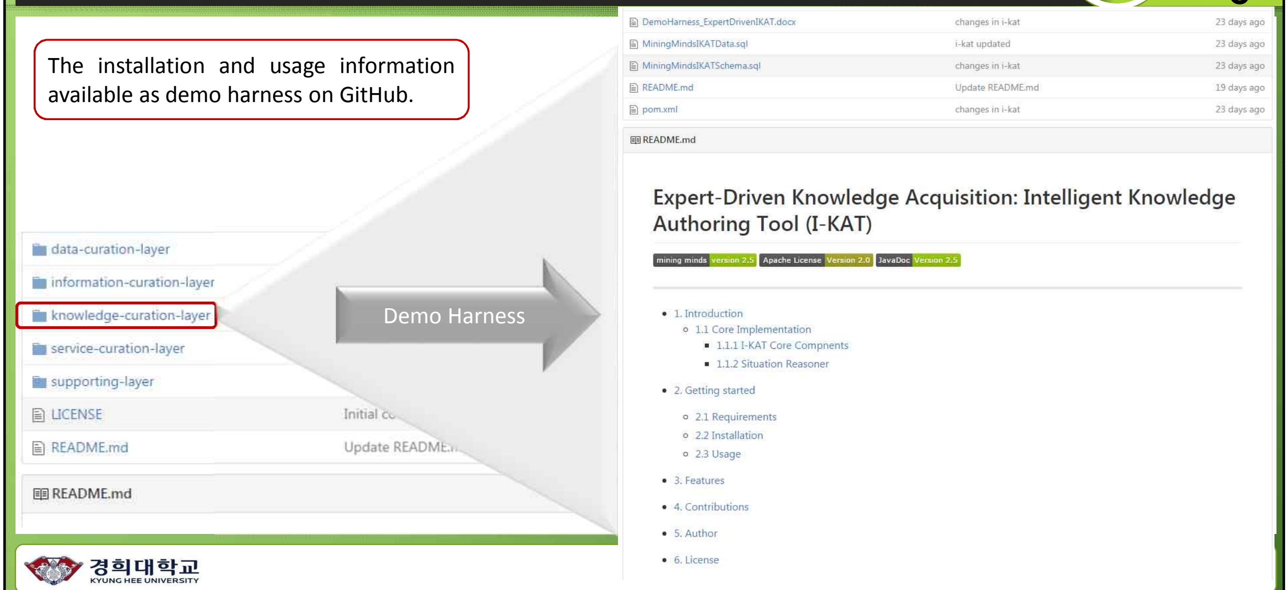
The Java Platform API Specification is defined by the documentation comments in the source code and any documents marked as specifications reachable from those comments.



Task :9 Demo Harness

28
0

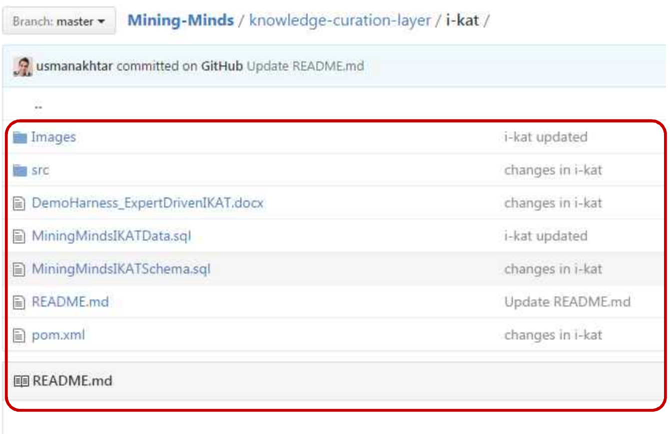
The installation and usage information available as demo harness on GitHub.



경희대학교
KYUNG HEE UNIVERSITY

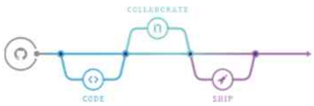
Task :10 Open Source Release

281



The source code is released under the following License:

- The Apache License, Version 2.0 (the "License")
- The copy of License may be obtained at http://www.apache.org/licenses/LICENSE-2.0



GitHub

경희대학교 KYUNG HEE UNIVERSITY

282

Thanks

경희대학교 KYUNG HEE UNIVERSITY

Mining Minds

Service Curation Layer (Open Source Presentation)

December 27, 2016

Mining Minds

Service Curation Layer

Open Source Components

285

1. Recommendation Builder
2. Recommendation Interpreter

경희대학교 KYUNG HEE UNIVERSITY

Mining Minds

Service Curation Layer
Recommendation Builder
Open Source Code Progress

경희대학교 KYUNG HEE UNIVERSITY KITECH 한국생산기술연구원 POSTECH 포항공과대학교 UNIVERSITY of TASMANIA 녹십자헬스케어 TAPACROSS U2 system User & Usability system KOCSA 한국컨설팅서비스협회

Agenda

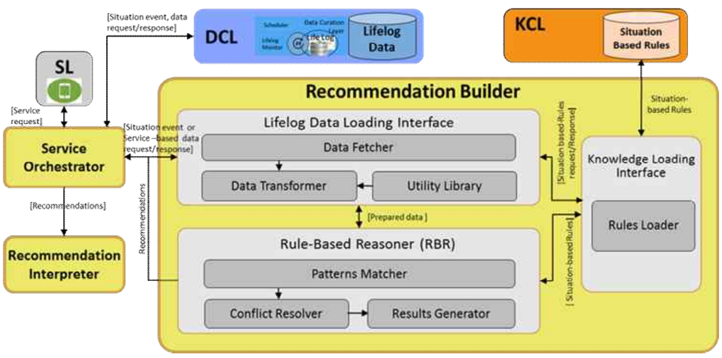
287

- Open Source Status
  - Public Library Functions
  - Code Documentation
    - doc comments for classes
    - doc comments for methods
  - Unit Testing
  - Design Patterns
  - Class Diagrams
  - Java Doc
  - Exception Handling

경희대학교
KYUNG HEE UNIVERSITY

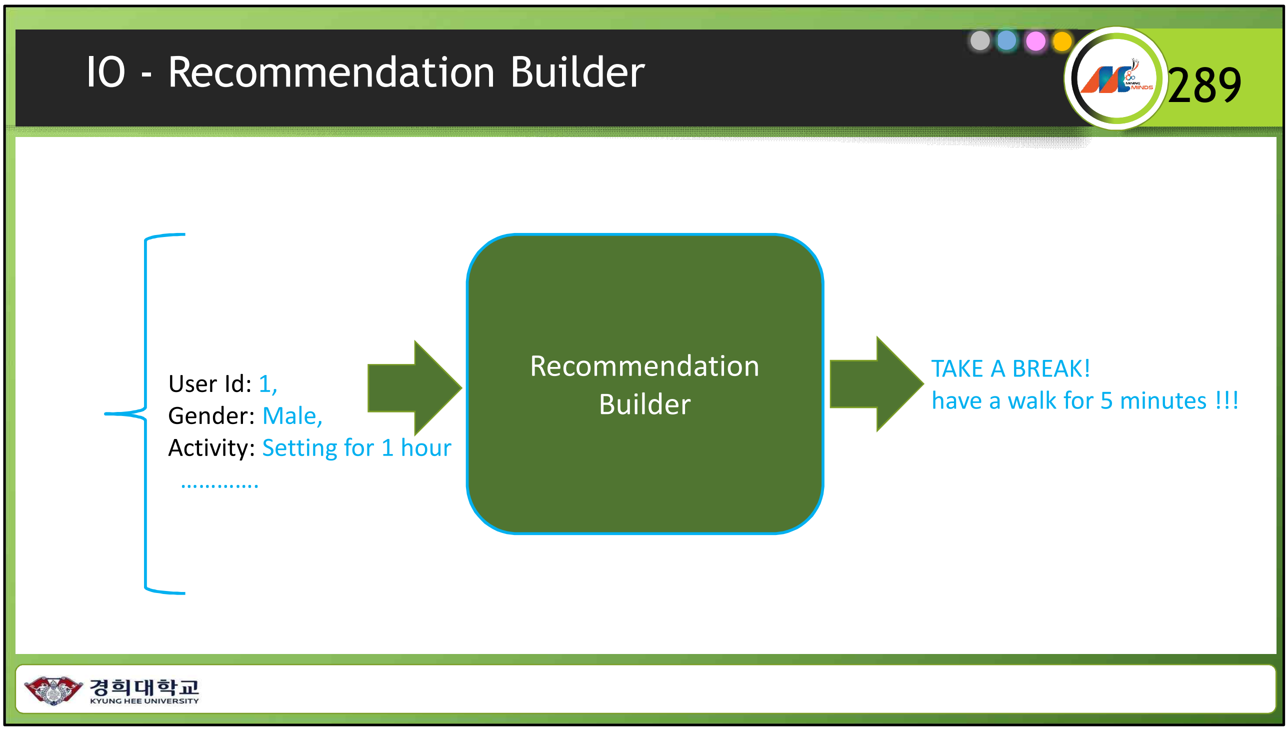
MMv2.5 Recommendation Builder - Architecture

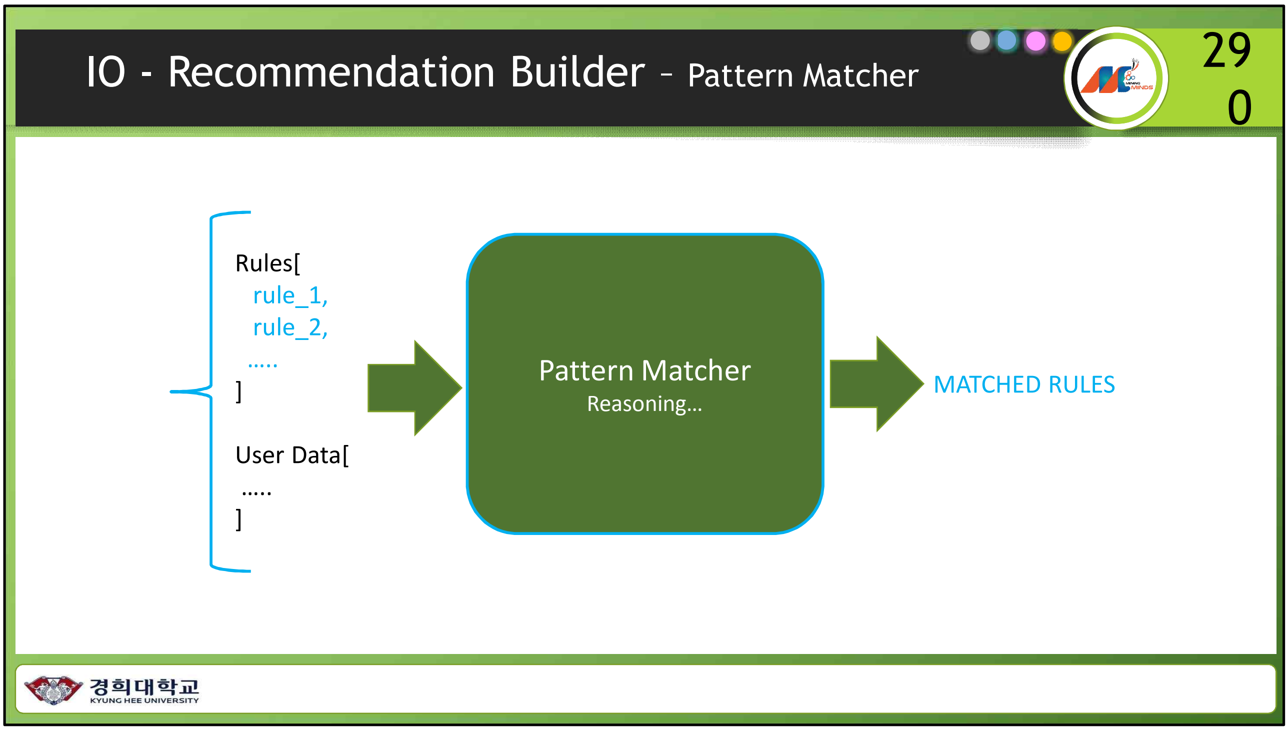
288



- **Input**
  - ✓ Life Log Data, Knowledge Rule
- **Output**
  - ✓ Recommendation contents
- **Implementation Module**
  - ✓ Recommendation Builder Controller
  - ✓ Rule based reasoned
    - ✓ Pattern Matcher
    - ✓ Conflict Resolver
    - ✓ Result Generator

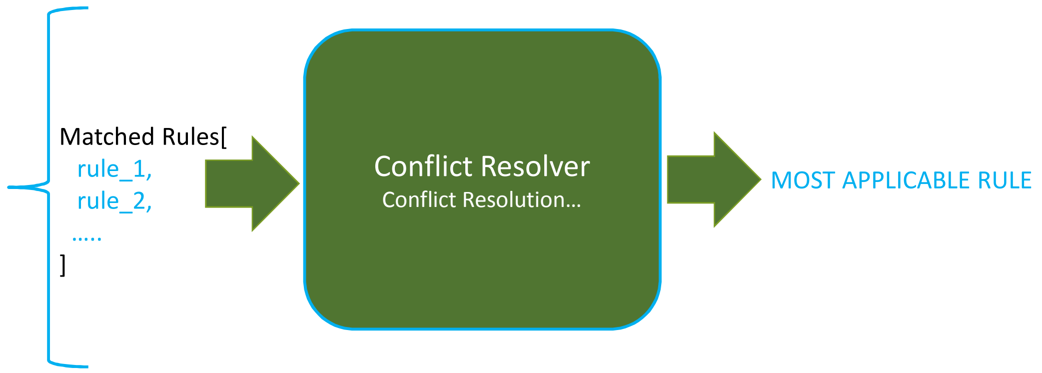
경희대학교
KYUNG HEE UNIVERSITY


IO - Recommendation Builder
289
User Id: 1,
Gender: Male,
Activity: Setting for 1 hour
....
Recommendation Builder
TAKE A BREAK!
have a walk for 5 minutes !!!
경희대학교
KYUNG HEE UNIVERSITY


IO - Recommendation Builder - Pattern Matcher
290
Rules[
rule_1,
rule_2,
.....
]
User Data[
.....
]
Pattern Matcher
Reasoning...
MATCHED RULES
경희대학교
KYUNG HEE UNIVERSITY

IO - Recommendation Builder – Conflict Resolver

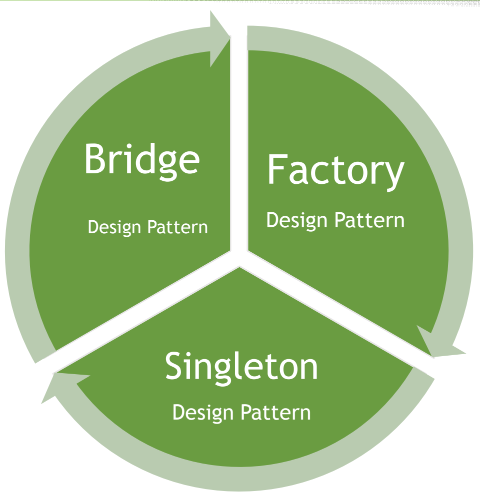
291



경희대학교 KYUNG HEE UNIVERSITY

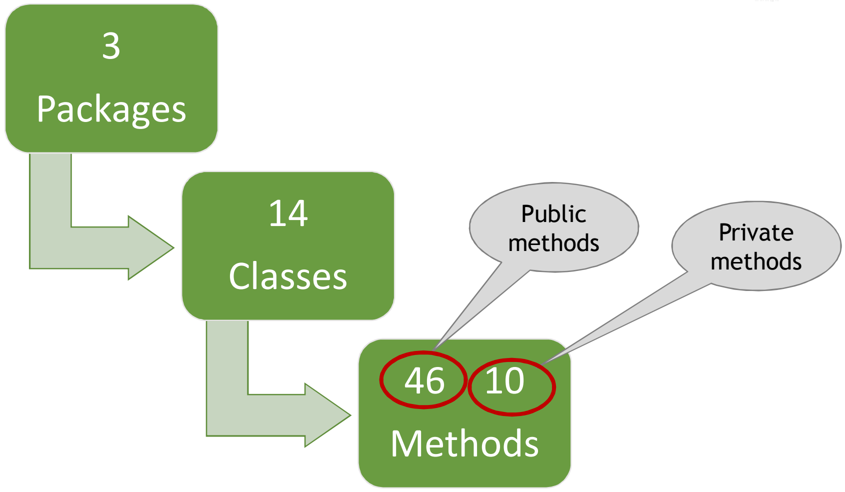
Recommendation Builder – Design Patterns

292



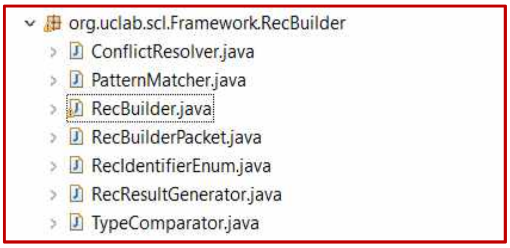
경희대학교 KYUNG HEE UNIVERSITY

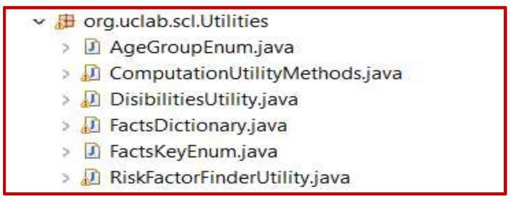
Recommendation Builder - Statistics



Recommendation Builder – Library Functions

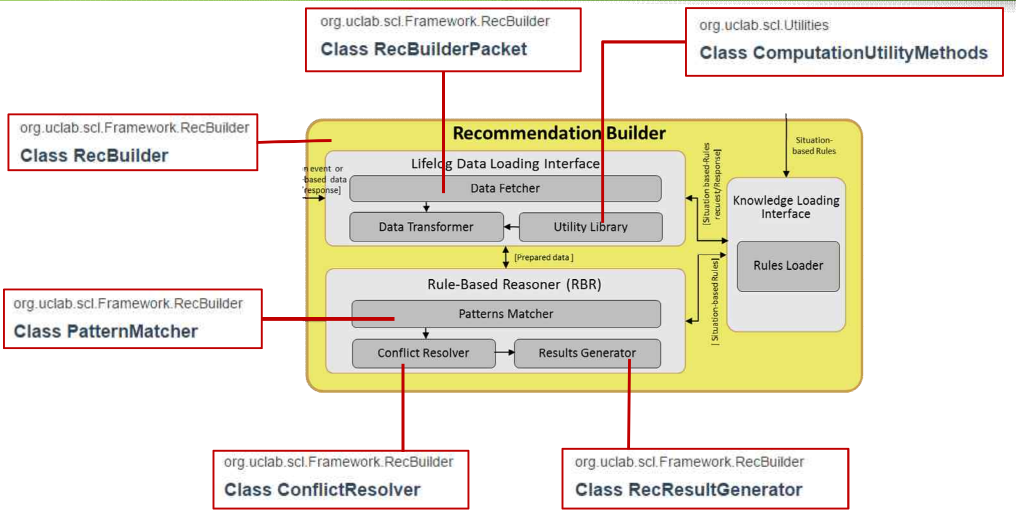
- Library Function Identification
  - Packages Categorization
  - org.uclab.scl.Framework.RecBuilder
    - RecBuilder.Java
      Controller class for loading data and rules and generating recommendations
    - PatternMatcher.java
      - match data against rules
    - ConflictResolver.java
      - Resolves conflict in case more than one rules fired
    - RecResultGenerator.java
      - Generate results
  - org.uclab.scl.Utilities
    - A library of all the utility functions required during data preparation





Recommendation Builder - Architecture

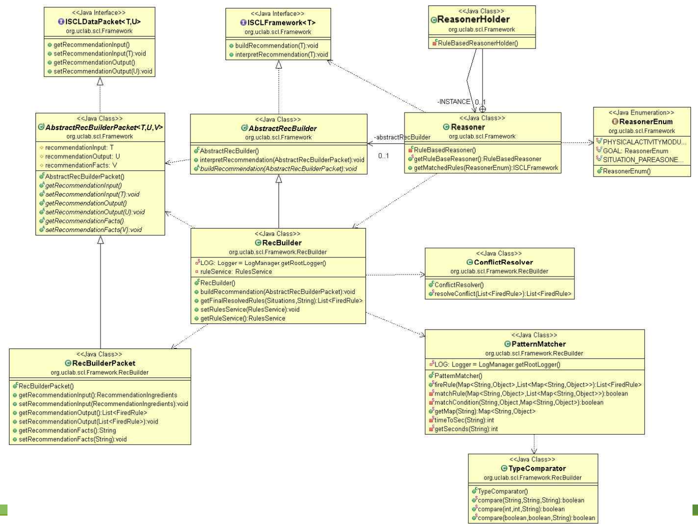
295



경희대학교
KYUNG HEE UNIVERSITY

Recommendation Builder - Class Diagram

296



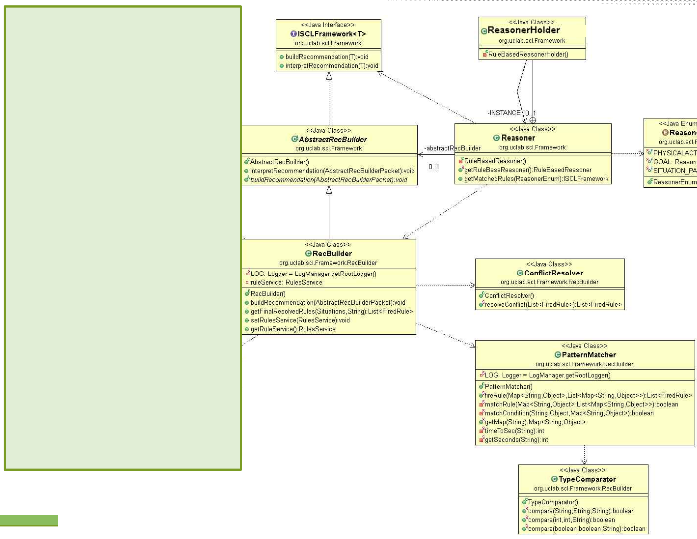
경희대학교
KYUNG HEE UNIVERSITY

Recommendation Builder - Class Diagram

29
7

ISCL Data Packet

Data Input and Output Container Classes

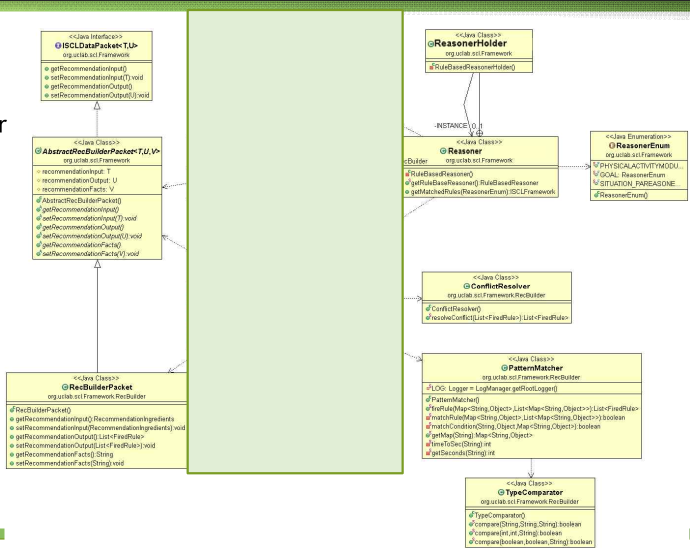


Recommendation Builder - Class Diagram

29
8

Rec Builder

Recommendation Builder Controller Classes

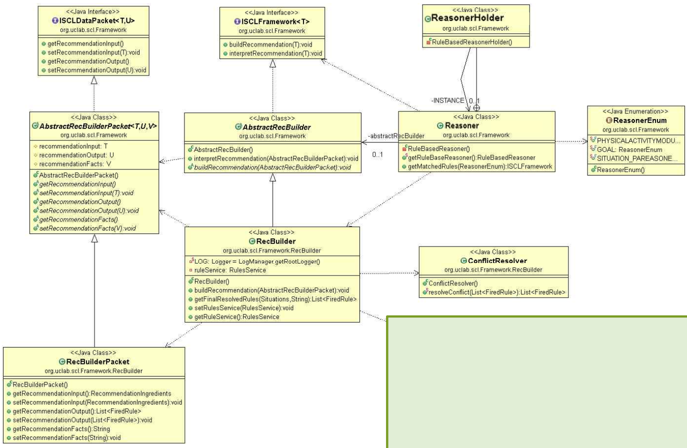


Recommendation Builder - Class Diagram

299

Pattern Matcher

Performs Reasoning to match rules against situation



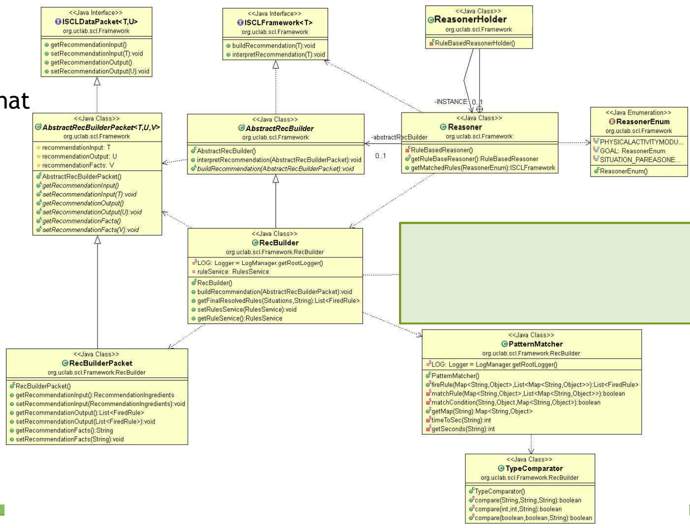
경희대학교
KYUNG HEE UNIVERSITY

Recommendation Builder - Class Diagram

300

Conflict Resolver

Resolves Conflict e.g. what if pattern matcher fires more than one rule ??

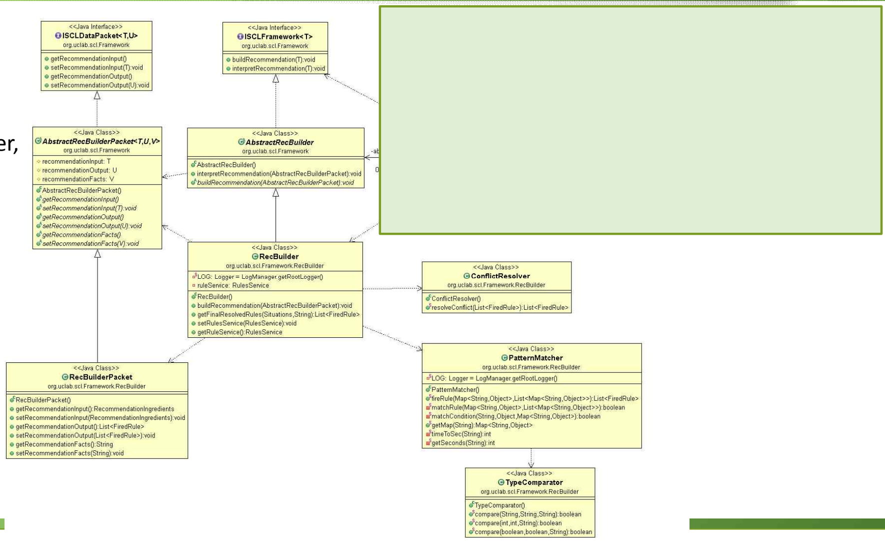


경희대학교
KYUNG HEE UNIVERSITY

Recommendation Builder - Class Diagram

Rule Base Reasoner

Selects a type of reasoner such as Rule-based reasoner, case-based reasoner, etc.



Recommendation Builder – Open Source Code Progress

org.uclab.scl.Framework.RecBuilder

Class RecBuilder

java.lang.Object
    org.uclab.scl.Framework.AbstractRecBuilder
        org.uclab.scl.Framework.RecBuilder.RecBuilder

All Implemented Interfaces:

ISCLFramework<AbstractRecBuilderPacket>

```
public class RecBuilder extends AbstractRecBuilder {
    private static Logger LOG = LogManager.getRootLogger();
    private RulesService ruleService;

    /**
     * generates recommendation based on the ISCLDataPacket
     * <p>
     *
     * @param ISCLDataPacket
     * @return
     */
    @Override
    public void buildRecommendation(AbstractRecBuilderPacket abstractRecBuilderPacket) {
        LOG.debug("building recommendation...");
```

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
void	buildRecommendation(AbstractRecBuilderPacket abstractRecBuilderPacket) generates recommendation based on the recommendationBuilderPacket
java.util.List<org.uclab.scl.datamodel.FiredRule>	getFinalResolvedRules(org.uclab.scl.datamodel.Situations situation, java.lang.String conditionValueJSON) returns the final resolved rules after resolving conflict
RulesService	getRuleService() returns the rule service instance
void	setRulesService(RulesService ruleService) sets the rule service

Pattern Matcher – Open Source Code Progress

30
3

org.uclab.scl.Framework.RecBuilder

Class PatternMatcher

java.lang.Object
org.uclab.scl.Framework.RecBuilder.PatternMatcher

```java
public class PatternMatcher {
  private static Logger LOG = LogManager.getRootLogger();

  /**
   * fires rule for a specific situation after reasoning
   * <p>
   *
   * @param conditionsValue
   * @param rules
   * @return listOfMatchedRules
   */
  @SuppressWarnings("unchecked")
  public static List<FiredRule> fireRule(Map<String, Object> conditionsValue, List<Map<String,
    List<FiredRule> firedRules = new ArrayList<FiredRule>();
```

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`static java.util.List<org.uclab.scl.datamodel.FiredRule>`	`fireRule(java.util.Map<java.lang.String,java.lang.Object> conditionsValue, java.util.List<java.util.Map<java.lang.String,java.lang.Object>> rules)` fires rule for a specific condition after reasoning
`static java.util.Map<java.lang.String,java.lang.Object>`	`getMap(java.lang.String json)` Converts stringJson to map and list rules from it.

Conflict Resolver – Open Source Code Progress

30
4

org.uclab.scl.Framework.RecBuilder

Class ConflictResolver

java.lang.Object
org.uclab.scl.Framework.RecBuilder.ConflictResolver

```java
public class ConflictResolver {

  /**
   * Resolves conflict and returns the final resolved rule/s
   *
   * @param firedRules
   * @return finalResolvedRules
   */
  public static List<FiredRule> resolveConflict(List<FiredRule> firedRules) {
```

Method Summary

All Methods | **Static Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`static java.util.List<org.uclab.scl.datamodel.FiredRule>`	`resolveConflict(java.util.List<org.uclab.scl.datamodel.FiredRule> firedRules)` Resolves conflict and returns the final resolved rule/s

Methods inherited from class java.lang.Object

`equals, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait`

경희대학교
KYUNG HEE UNIVERSITY

SCLDataPacket – Open Source Code Progress

org.uclab.scl.Framework.RecBuilder

Class RecBuilderPacket

java.lang.Object
org.uclab.scl.Framework.AbstractRecBuilderPacket<org.
org.uclab.scl.Framework.RecBuilder.RecBuilderPac

All Implemented Interfaces:
ISCLDataPacket<org.uclab.scl.datamodel.Recommendat

```java
public class RecBuilderPacket extends AbstractRecBuilderPacket<RecommendationIngred

    /**
     * prepares and returns recommendation input
     * <p>
     *
     * @return RecommendationIngredients
     */
    @Override
    public RecommendationIngredients getRecommendationInput(){
```

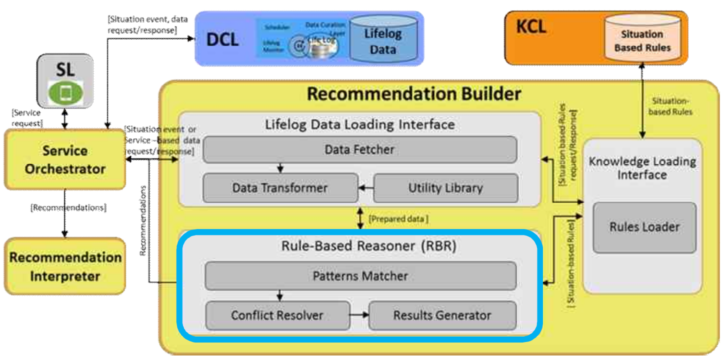
All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
java.lang.String	getRecommendationFacts() returns recommendation facts
org.uclab.scl.datamodel.RecommendationIngredients	getRecommendationInput() returns recommendation input
java.util.List<org.uclab.scl.datamodel.FiredRule>	getRecommendationOutput() returns recommendation output
void	setRecommendationFacts(java.lang.String recommendationFacts) sets recommendation facts
void	setRecommendationInput(org.uclab.scl.datamodel.RecommendationIngredients recommendationInput) sets recommendation input
void	setRecommendationOutput(java.util.List<org.uclab.scl.datamodel.FiredRule> recommendationOutput) sets recommendation output

Future - Recommendation Builder

New Challenges

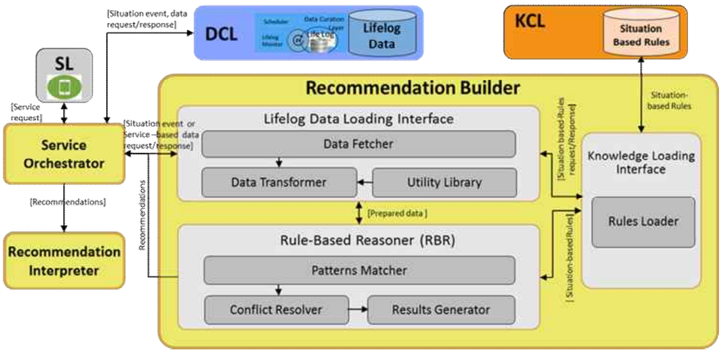
- **Dynamic Data Integration**
- Case Based Reasoning



경희대학교
KYUNG HEE UNIVERSITY

MMv2.5 Recommendation Builder - Architecture

307



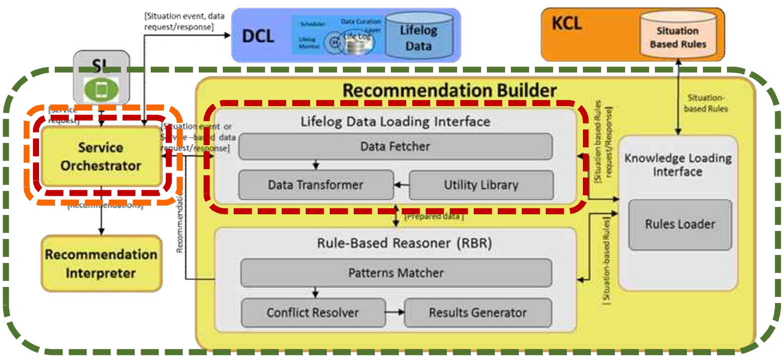
- **Input**
  - ✓ Life Log Data, Knowledge Rule
- **Output**
  - ✓ Recommendation contents
- **Implementation Module**
  - ✓ Recommendation Builder Controller
  - ✓ Rule based reasoned
    - ✓ Pattern Matcher
    - ✓ Conflict Resolver
    - ✓ Result Generator

경희대학교 KYUNG HEE UNIVERSITY

3rd Party Libraries And Components

308

- Jackson-Jaxrs v1.9.13
- Jackson-Mapper-asl v1.9.13
- Log4j-api v2.3
- Log4j-core v2.3
- Jersey-Bundle v1.7
- Junit v4.8.2

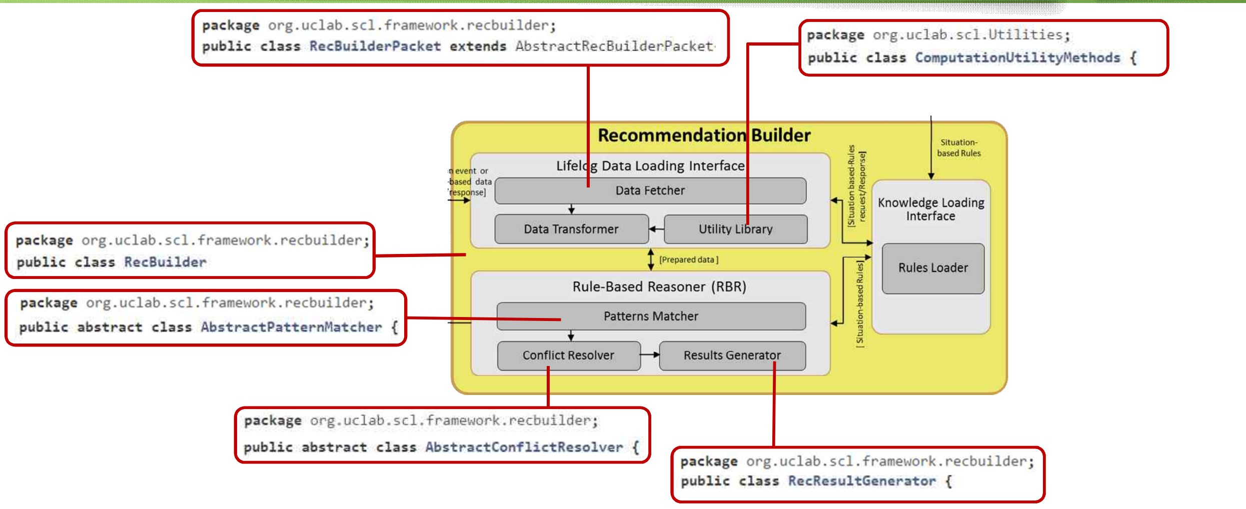


★ Above libraries are used without modification

경희대학교 KYUNG HEE UNIVERSITY

Recommendation Builder - Architecture

309



경희대학교 KYUNG HEE UNIVERSITY

Library Classes

310

master ▾ | scl-miningmind-2.5 / src / main / java / org / uclab / scl /

- ..
- Utilities
- communication
- datamodel
- framework
- outputModel
- rs

master ▾ | scl-miningmind-2.5 / src / main / java / org / uclab / scl / framework / recbuilder /

..		
AbstractConflictResolver.java	415 B	3 days ago
AbstractPatternMatcher.java	494 B	3 days ago
ConflictResolver.java	967 B	3 days ago
PatternMatcher.java	4.8 KB	3 days ago
RecBuilder.java	3.3 KB	3 days ago
RecBuilderPacket.java	1.7 KB	3 days ago
RecIdentifierEnum.java	195 B	3 days ago
RecResultGenerator.java	876 B	3 days ago
TypeComparator.java	1.7 KB	3 days ago

경희대학교 KYUNG HEE UNIVERSITY

Doc Comments - Classes

311

```java
package org.uclab.scl.framework.recbuilder;

import java.util.List;

import org.uclab.scl.datamodel.FiredRule;

/**
 * AbstractConflictResolver
 * <p>
 * enables developers to provide conflict resolution strategy when org.uclab.scl.framework.recbuilder.AbstractPatternMatcher matches or
 * fires more than one rule for a particular situation or condition
 * <p>
 * AbstractConflictResolver provides a way through which the user can provide the implementation for
 * conflict resolution strategy strategy.
 *
 * @author Sadiq
 * @version 2.5
 * @since July 2014
 *
 */

public abstract class AbstractConflictResolver {

```

경희대학교 KYUNG HEE UNIVERSITY

Doc Comments - Classes

312

```java
package org.uclab.scl.framework.recbuilder;

import java.util.List;
import java.util.Map;

import org.uclab.scl.datamodel.FiredRule;

/**
 * AbstractPatternMatcher
 * <p>
 * enables developers to provide rule selection strateg
 *
 * @author Sadiq
 * @version 2.5
 * @since july 2014
 *
 */
public abstract class AbstractPatternMatcher {
```

```java
package org.uclab.scl.framework;

/**
 * AbstractRecommendationBuilder
 * <p>
 * provides recommendation for a particular situation
 *
 * @author Sadiq
 * @version 2.5
 * @since july 2014
 *
 */
public abstract class AbstractRecBuilder implements ISCLFramework
```

경희대학교 KYUNG HEE UNIVERSITY

Doc Comments - Classes


```java
package org.uclab.scl.framework.recbuilder;

import java.util.List;
import java.util.Map;

import org.uclab.scl.datamodel.FiredRule;

/**
 * AbstractPatternMatcher
 * <p>
 * enables developers to provide rule selection strategy for a particular situation or condition
 *
 * @author Sadiq
 * @version 2.5
 * @since july 2014
 *
 */
public abstract class AbstractPatternMatcher {
```


Doc Comments - Methods


```java
/**
 * generates recommendation based on the ISCLDataPacket
 * <p>
 *
 * @param ISCLDataPacket
 * @return
 */
@Override
public void buildRecommendation(AbstractRecBuilderPacket abstractRecBuilderPacket) {
  LOG.debug("building recommendation...");
```

```java
/**
 * Resolves conflict by using maximum specificity technique and returns the final resolved rule/s
 *
 * @param firedRules
 * @return finalResolvedRules
 */
@Override
public List<FiredRule> resolveConflict(List<FiredRule> firedRules) {
  if (firedRules == null || firedRules.size() < 2) {
```

Doc Comments - Methods

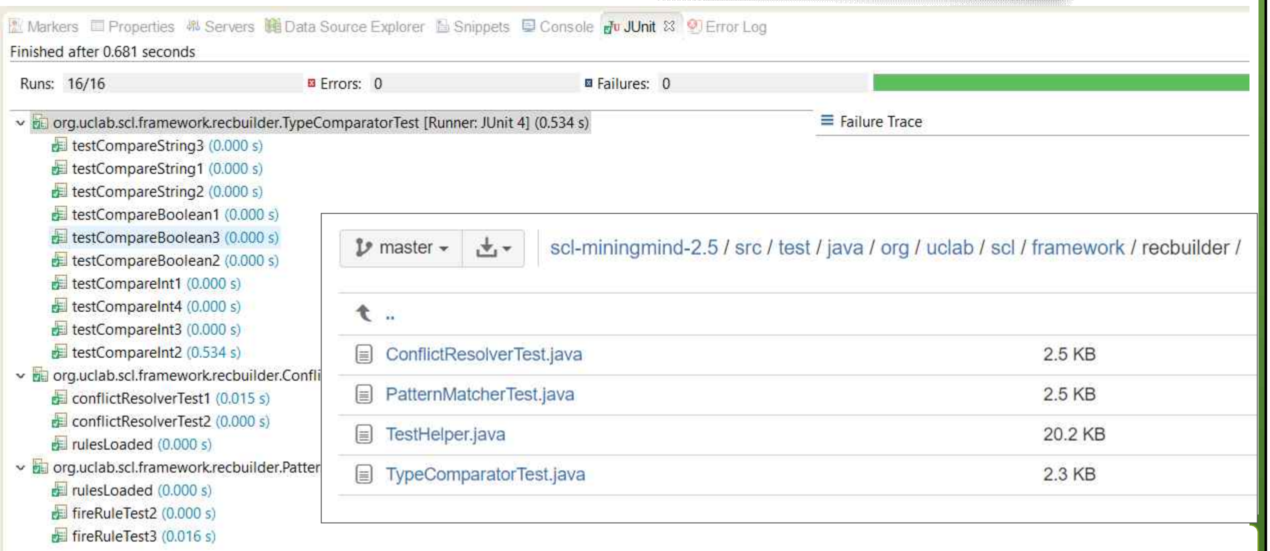
315

```java
/**
 * fires rule for a specific situation after reasoning
 * <p>
 *
 * @param conditionsValue
 * @param rules
 * @return listOfMatchedRules
 * @throws UnsupportedTypeException
 */
@SuppressWarnings("unchecked")
@Override
public List<FiredRule> fireRule(Map<String, Object> conditionsValue, List<Map<String, Object>> rules) throws UnsupportedTypeException {

/**
 * returns true if all conditions matched false other wise
 *
 * @param conditionsValue
 * @param conditionsKeyList
 * @return boolean
 * @throws UnsupportedTypeException
 */
private boolean matchRule(Map<String, Object> conditionsValue, List<Map<String, Object>> conditionsKeyList) throws
  boolean CONDITION_MATCHED = false;
```

경희대학교
KYUNG HEE UNIVERSITY

Unit Testing – jUnit

316



KYUNG HEE UNIVERSITY

Unit Testing – Test Methods


```java
@Test
public void fireRuleTest2() {
  String jsonFact = "{\"Current Activity\":\"LyingDown\",\"Activity Duration\":\"1h\"}";
  Map<String, Object> fact = TestHelper.buildFact(jsonFact);

  String expected = "19";
```

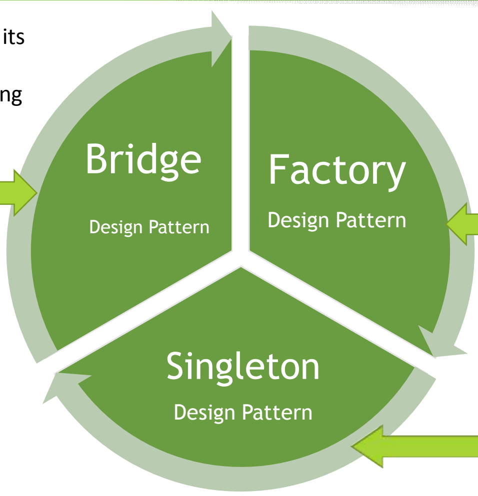
```java
@Test
public void conflictResolverTest1() {
  String jsonFact = "{\"Current Activity\":\"LyingDown\",\"Activity Duration\":\"1h\",\"Acti
  Map<String, Object> fact = TestHelper.buildFact(jsonFact);
  List<FiredRule> firedRules = null;
```

```java
@Test
public void testCompareString2(){
  String v1 = "setting";
  String v2 = "standing";
  String op = "=";

  assertEquals(false, TypeComparator.compare(v1, v2, op));
}
```

```java
@Test
public void testCompareBoolean3(){
  boolean v1 = true;
  boolean v2 = false;
  String op = "!=";

  assertEquals(true, TypeComparator.compare(v1, v2, op));
}
```


Recommendation Builder – Design Patterns

- Decouple the abstraction from its implementations
- Divide the responsibilities among different classes
- Hide details from client

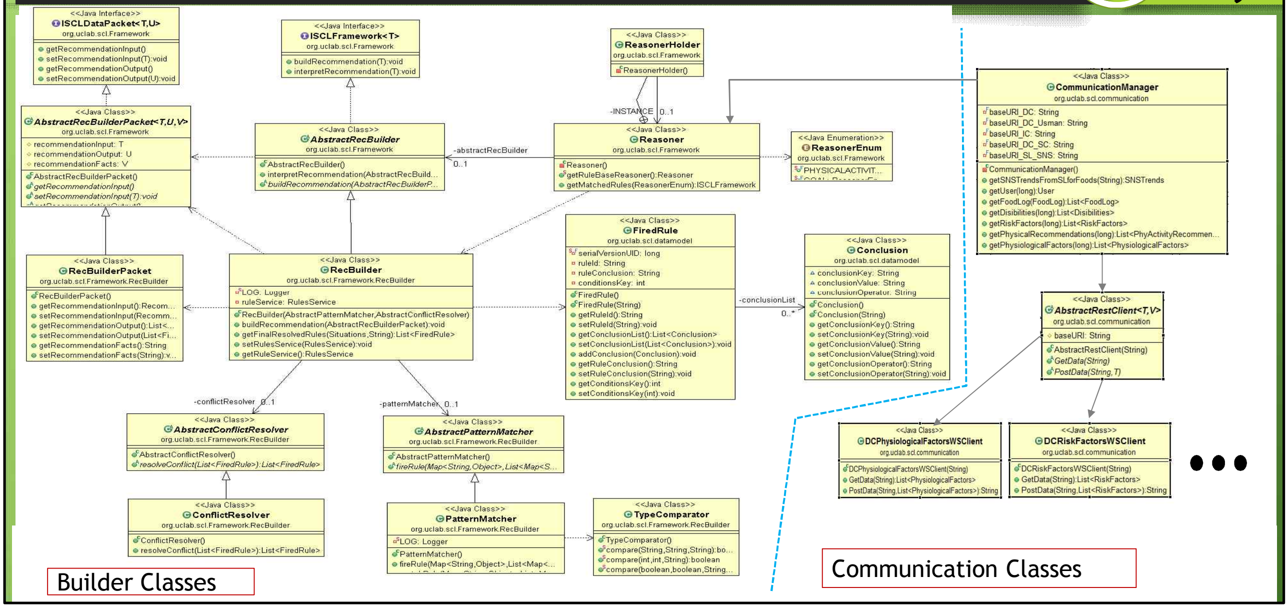


- Create objects without exposing the creation logic to outside world
- Refer to newly created object using a common interface
- Hide object creation complexity

- Restricts the instantiation of a class
- Ensures that only one instance of the class exists
- Provide a global access point to get the instance of the class

Class Diagram – Recommendation Builder

319



Recommendation Builder - Java Doc

320

org.uclab.scl.framework.recbuilder

Class RecBuilder

java.lang.Object
org.uclab.scl.framework.AbstractRecBuilder
org.uclab.scl.framework.recbuilder.RecBuilder

All Implemented Interfaces:

`ISCLFramework<AbstractRecBuilderPacket>`

Constructor Summary

Constructors

Constructor and Description
`RecBuilder(AbstractPatternMatcher patternMatcher, AbstractConflictResolver`   Instantiates RecommendationBuilder by using patternMatcher & conflictResolver

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
`void`	`buildRecommendation(AbstractRecBuilderPacket abstractRecBuilderPacket)` generates recommendation based on the ISCLDataPacket
`java.util.List<FiredRule>`	`getFinalResolvedRules(Situations situation, java.lang.String conditionValueJSON)` returns the final resolved rules after resolving conflict

Exception Handling

40dfe1d 2 minutes ago Full commit Blame Ra

```java
package org.uclab.scl.framework.recbuilder;

/**
 * UnsupportedTypeException
 * <p>
 * This class handles all sort of exceptions that occurs due to unsupported type (data types not supported by this
 * application) of parameters or operators
 *
 * @author Sadiq
 * @version 2.5
 * @since july 2014
 *
 */
public class UnsupportedTypeException extends Exception {
```

```java
        try{
          List<Map<String, Object>> rules = getRuleService().setSituati
          Map<String, Object> conditionsValue = ConditionsValueService.
          List<FiredRule> firedRules = patternMatcher.fireRule(conditio
          LOG.debug("Resolving conflict...");
          finalResolvedRules = conflictResolver.resolveConflict(firedRu
        }catch(Exception e){
          StringWriter sw = new StringWriter();
          PrintWriter pw = new PrintWriter(sw);
          e.printStackTrace(pw);
          LOG.error(sw.toString());
        }
```

```java
@Override
public List<FiredRule> fireRule(Map<String, Object> conditionsValue, List<Map<String, Object>> rules) throws UnsupportedTypeException {
  List<FiredRule> firedRules = new ArrayList<FiredRule>();
```

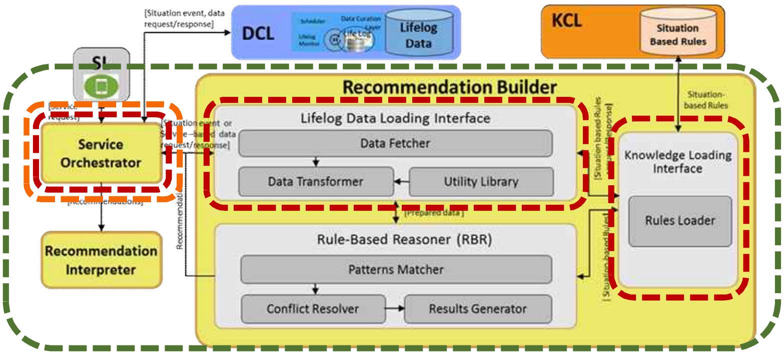

SCL RB - Code Refactoring Activities

- Rename (Meaningful naming)
  - Methods
  - Variables
  - Classes
- Exception Handling
  - Separating Error-Handling Code from "Regular" Code
  - Propagating Errors Up the Call Stack
- Design Patterns
  - Recurring solutions to software design problems
  - An array of tried and tested solutions to common problems
- Move Classes
  - Moving classes to better fit packages
- Code Cleaning
  - Remove unnecessary code

3rd Party Libraries And Components

32
3

- Jackson-Jaxrs v1.9.13
- Jackson-Mapper-asl v1.9.13
- Log4j-api v2.3
- Log4j-core v2.3
- Jersey-Bundle v1.7
- Junit v4.8.2



★ Above libraries are used without modification

경희대학교 KYUNG HEE UNIVERSITY

Utilization And License Of The Library

32
4

Library	Utilization	License
Jackson-Jaxrs	• Used to parse a JSON from a string and create an object graph (Java Object) representing the parsed JSON • Convert Java Object to JSON representation	Apache 2.0, LGPL 2.1
Jackson-Mapper		Apache 2.0
Log4j-core	• Used to control which log statements should be output	Apache 2.0
Log4j-api		Apache 2.0
Jersey-Bundle	• Used for restful web services	CDDL 1.1, GPL 2.0
Junit	• Used for unit testing	EPL 1.0

경희대학교 KYUNG HEE UNIVERSITY

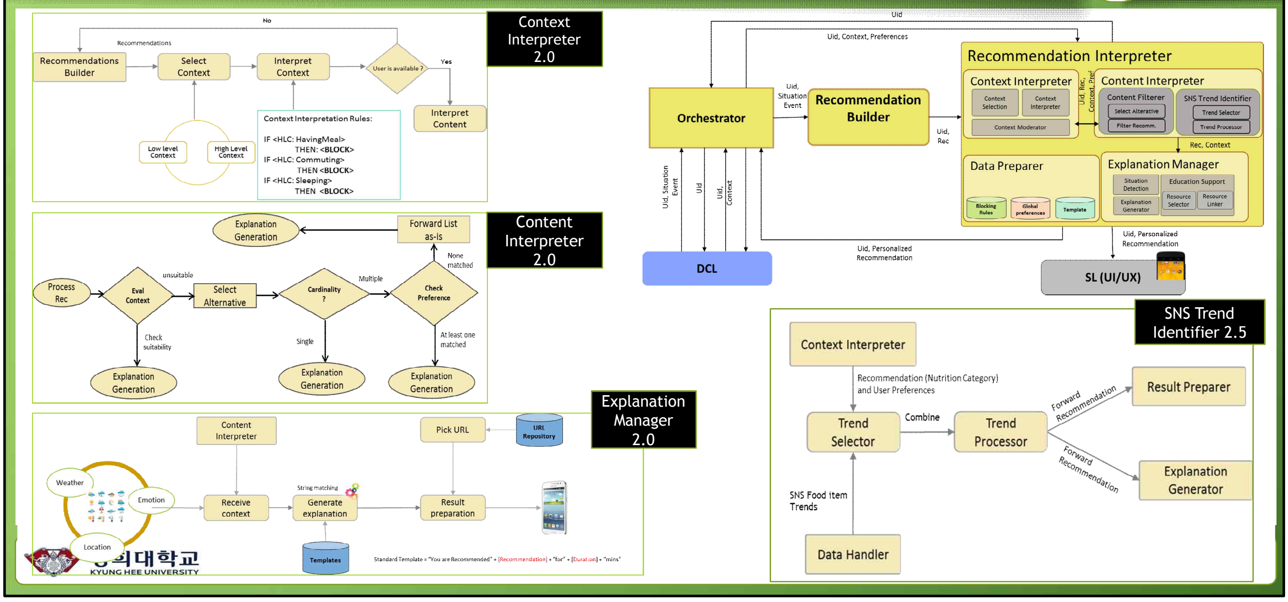
License and Its Features

License	Commercial Use	Modify	Distribute	Sublicense	Private Use	Use Patent Claims
Apache 2.0	✓	✓	✓	✓	✓	✓
LGPL 2.1	✓	✓	✓			
EPL 1.0	✓	✓	✓	✓	✓	✓
CDDL 1.1	✓	✓	✓	✓		✓
GPL 2.0	✓	✓	✓			

Mining Minds Platform

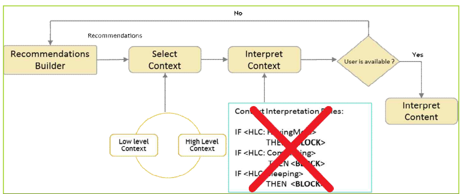
Service Curation Layer
Recommendation
Interpreter
Source Code Progress

Recommendation Interpreter– Main Components



Current Objectives for Open Source Process

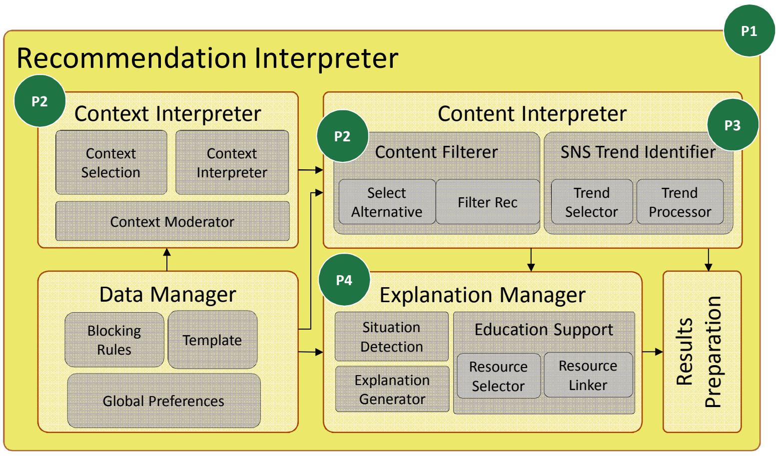
- **Improve the source code –**
  - **Objectives**
    - Improved Context Selector by eliminating "Heuristic rules"
    - Integrate classes to the SCL architecture
    - Categorize classes into packages
    - Remove unused code segments
    - Reduce number of classes
- **Identify Library Functions –**
  - **Objectives**
    - Understand open source requirements for the SCL Framework
    - Provide suitable code documentation for reuse



Contextual Matrix						
Context/Rec	Hiking	Jogging	Sitting	Stretching	Cyclng	Walking
Location= Office	1	1	0	1	1	0
HLC= Inactivity	1	1	0	1	1	1
Weather= Rainy	0	1	1	1	0	1

Overview of Main Packages

329



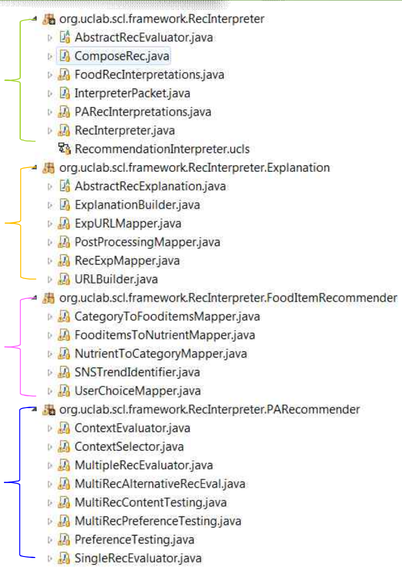
- P1 org.uclab.scl.framework.RecInterpreter
- P2 org.uclab.scl.framework.RecInterpreter.PARecommender
- P3 org.uclab.scl.framework.RecInterpreter.FoodItemRecommender
- P4 org.uclab.scl.framework.RecInterpreter.Explanation

경희대학교 KYUNG HEE UNIVERSITY

Recommendation Interpreter Open Source Code Progress

330

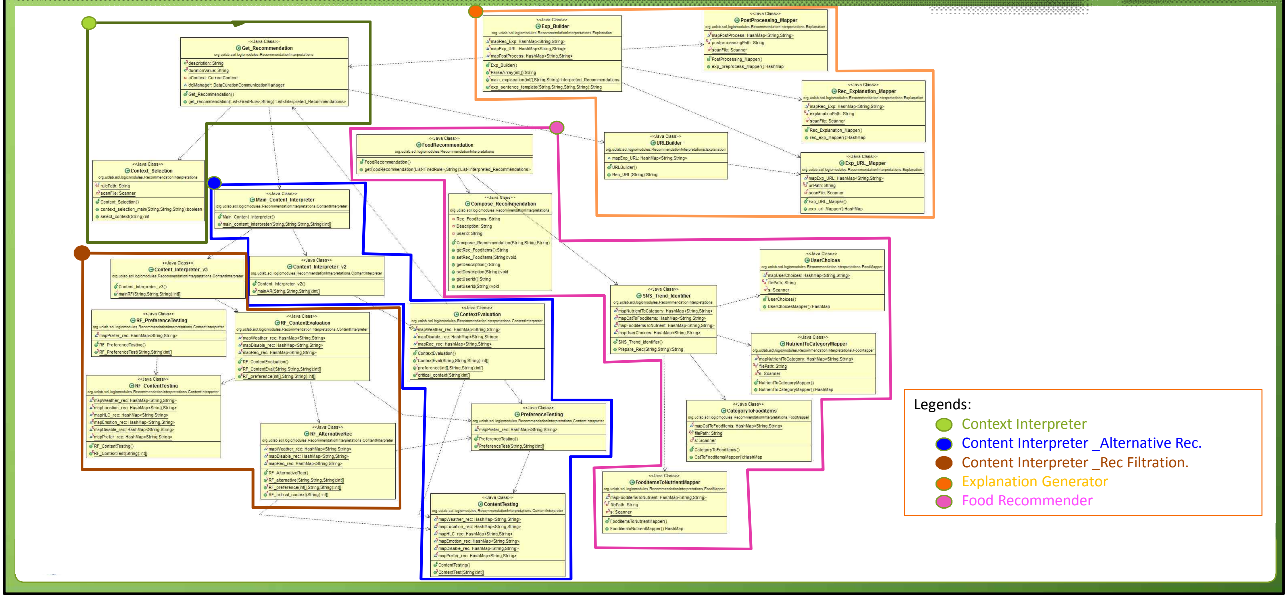
- Package Categorization
    - org.uclab.scl.framework.RecInterpreter
        - Contains Controller classes
        - Packages finally generated recommendation
    - org.uclab.scl.framework.RecInterpreter.Explanation
        - Methods for Sentence generation
        - Methods for Audio/Visual Aid (multimedia) embedding with recommendation
        - Methods for URL embedding with recommendation
    - org.uclab.scl.framework.RecInterpreter.FoodItemRecommender
        - Methods for mapping between Nutrient to Food Category
        - Methods for mapping between Food Category and Food items
        - Methods for mapping between Food items to Nutrient
    - org.uclab.scl.framework.RecInterpreter.PARecommender
        - Methods deciding user's interruptibility
        - Methods for Alternative Recommendation
        - Methods for Recommendation Filtration
        - Methods for Preference Evaluation



KYUNG HEE UNIVERSITY

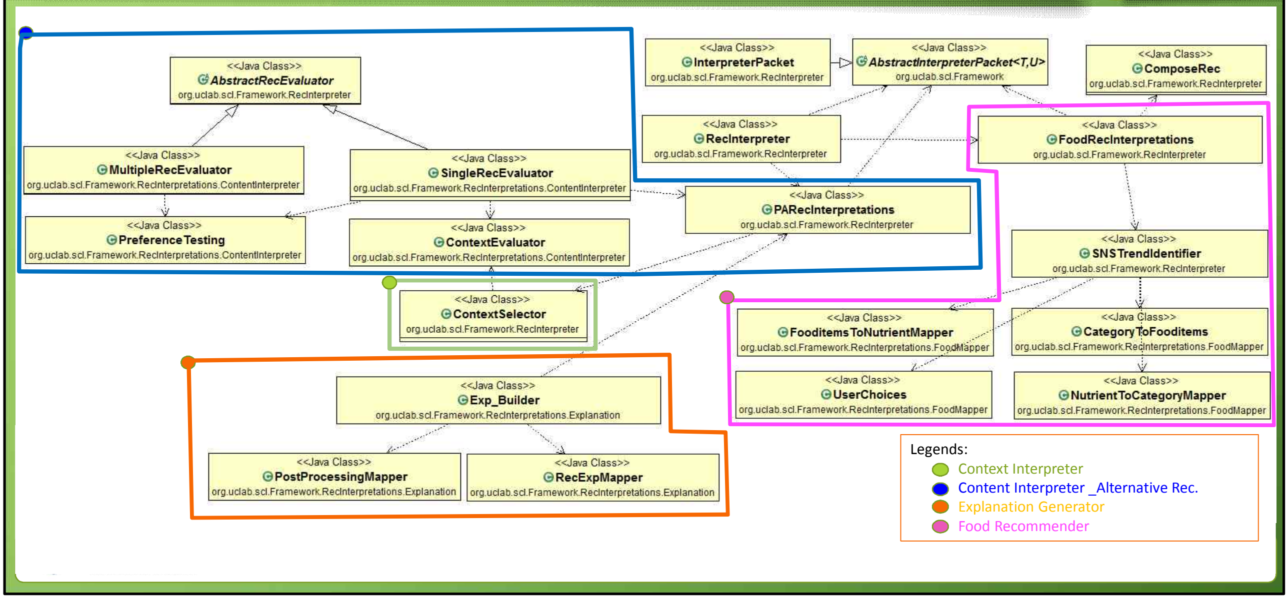
Recommendation Interpreter Class Diagram Evolution (Ver. 1)

331

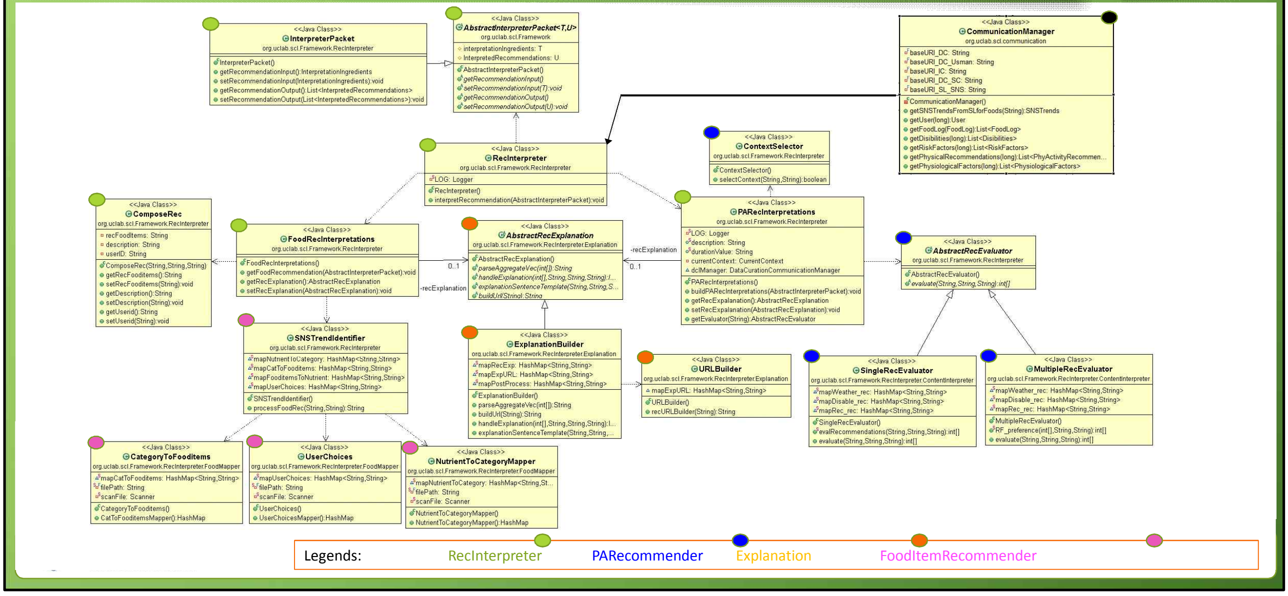


Recommendation Interpreter Class Diagram Evolution (Ver. 2)

332



Recommendation Interpreter Class Diagram Evolution (Ver. 3)



Code Refactoring

- **Types of Refactoring**
  - Standardizing variable names
  - Moving Classes to appropriate packages
  - Structural Changes
  - Handling Exceptions
  - Code Commenting and Logging

```
public static int[] RF_PreferenceTest(String userid, String context_string)
    throws Exception {
    // Check Context with Preference
    PreferenceMapper preferenceMapper = new PreferenceMapper();

    mapPrefer_rec = preferenceMapper.preferMapper();

    String[] prefer1 = mapPrefer_rec.get(userid).split(",");
    int[] prefer = new int[prefer1.length];

    for (int i = 0; i < prefer1.length; i++) {
      String numberAsString = prefer1[i];
      prefer[i] = Integer.parseInt(numberAsString);
    }
```

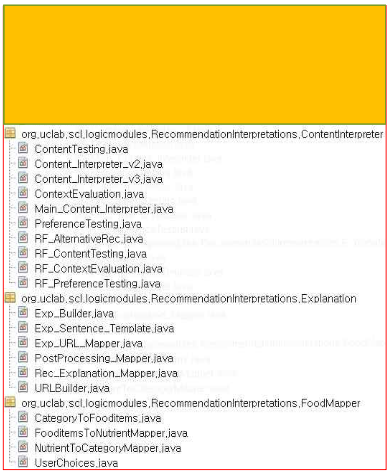
```
public static int[] mrecPreferenceTest(String userid, String currentContext) throws Exception {
    // Check Context with Preference
    PreferenceMapper preferenceMapper = new PreferenceMapper();
    mapPreferRec = preferenceMapper.preferMapper();
    String[] preferStr = mapPreferRec.get(userid).split(",");
    int[] prefer = new int[preferStr.length];

    for (int i = 0; i < preferStr.length; i++) {
      String numberAsString = preferStr[i];
      prefer[i] = Integer.parseInt(numberAsString);
    }
```

경희대학교
KYUNG HEE UNIVERSITY

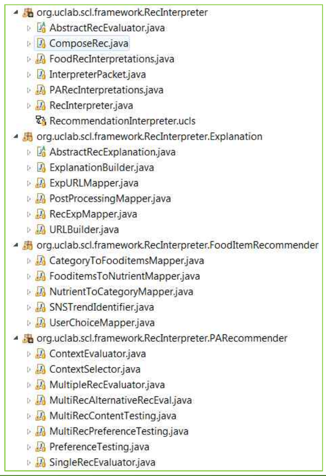
Code Refactoring

- **Types of Refactoring**
  - Standardizing variable names
  - Moving Classes to appropriate packages
  - Structural Changes
  - Handling Exceptions
  - Code Commenting and Logging

Not Refactored

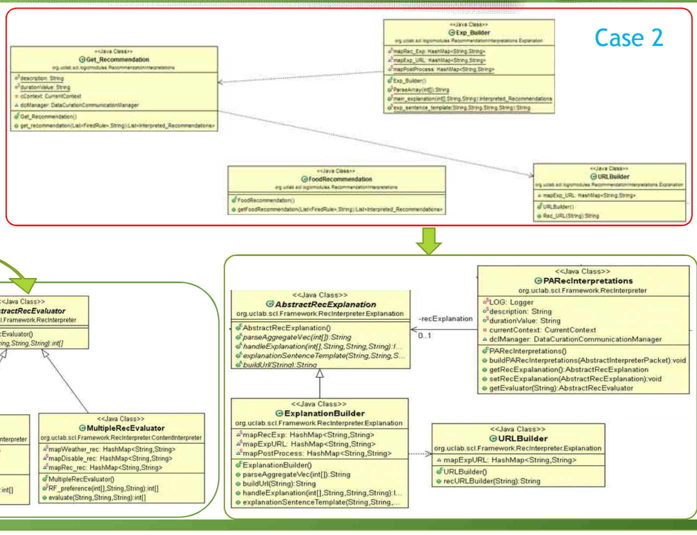


Refactored

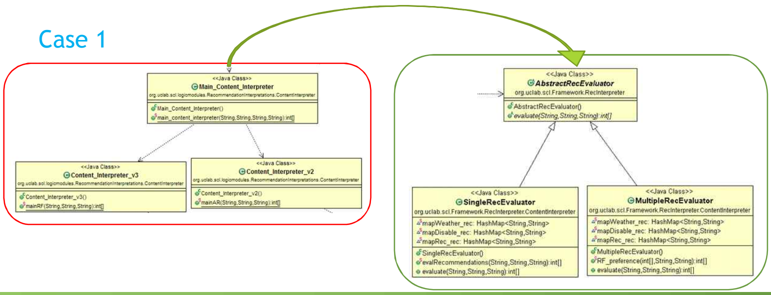


Code Refactoring

- **Types of Refactoring**
  - Standardizing variable names
  - Moving Classes to appropriate packages
  - Structural Changes
  - Handling Exceptions
  - Code Commenting and Logging



Case 1



Code Refactoring

33
7

- **Types of Refactoring**
  - Standardizing variable names
  - Moving Classes to appropriate packages
  - Structural Changes
  - Handling Exceptions
  - Code Commenting and Logging

```
public LocationMapper() {
    scanFile = new Scanner(this.getClass().getClassLoader().getResourceAsStream(LocatoinPath));
}
```

```
public LocationMapper() {
    try {
        scanFile = new Scanner(this.getClass().getClassLoader().getResourceAsStream(LocatoinPath));
      } catch (Exception exp) {
        System.err.println(exp.getMessage());
      }
    }
```

```
public int[] evaluate(String rec, String userid, String context) {
    RecommendationMapper recommendationMapper = new RecommendationMapper();
    // context example = "Office,OfficeWork,Rainy,Neutral,No";
    LOG.debug("Context String: " + context);
    try {
        mapRec = recommendationMapper.recMapper();
    } catch (FileNotFoundException e) {
        LOG.error("RecMapper data file doesn't exist");
```


Code Refactoring

33
8

- **Types of Refactoring**
  - Standardizing variable names
  - Moving Classes to appropriate packages
  - Structural Changes
  - Handling Exceptions
  - Code Commenting and Logging

```
/** ContextSelector
 * <p>
 * This function evaluates User's Interruptibility Status.
 * This function uses information provided by generateContextMatrix function for deciding
 * whether to provide user with the recommended physical activity as the present moment or not
 * <p>
 * @param userid
 * @param receivedContext
 * @return boolean
 * @author Imran
 * @version 2.5
 * @Since September 2015
 */
public boolean selectContext(String userid, String receivedContext){
```

```
// Reading data from HashMap
String temp = mapRec.containsKey(rec) ? mapRec.get(rec) : mapRec.get(null);
int index = 0;
if (temp != null)
    index = Integer.parseInt(temp);
else {
  LOG.debug("Recommendation NOT FOUND in the repository");
  LOG.debug("Default case applied");
    index = Integer.parseInt(mapRec.get("D"));
```

ContextSelector Snapshot Example

33
9

org.uclab.scl.Framework.RecInterpreter

Class ContextSelector

java.lang.Object
    org.uclab.scl.Framework.RecInterpreter.ContextSelector

```
public class ContextSelector
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
`ContextSelector()`

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
boolean	`selectContext(java.lang.String userid, java.lang.String receivedContext)`

<<Java Class>>
ContextSelector
org.uclab.scl.Framework.RecInterpreter

- ContextSelector()
- selectContext(String,String):boolean

```java
public boolean selectContext(String userid, String receivedContext){
  int[] matResultVec = new int[5];
  int sum=0;
  boolean flag;
  try {
    matResultVec= ContextEvaluator.generateContextMatrix(receivedContext);
  } catch (Exception e) {
    e.printStackTrace();
  }
  for (int i = 0; i < matResultVec.length; i++) {
    sum += matResultVec[i];
  }
  switch(sum){
  case 0:
   flag = false;
  break;
  default:
    flag = true;
  }
  return flag;
}
private static int[] generateContextMatrix(String receivedContext) throws Exception {

  DisableMapper mapDisable = new DisableMapper();
  EmotionMapper mapEmotion = new EmotionMapper();
  HLCMapper mapHLC = new HLCMapper();
  LocationMapper mapLoc = new LocationMapper();
  PreferenceMapper mapPref = new PreferenceMapper();
  WeatherMapper mapWeather = new WeatherMapper();

  mapWeatherRec = mapWeather.weatherMapper();
  mapLocationRec = mapLoc.locationMapper();
  mapHLCRec = mapHLC.hlcMapper();
  mapEmotionRec = mapEmotion.emotionMapper();
  mapDisableRec = mapDisable.disableMapper();
  mapPreferRec = mapPref.preferMapper();

  String context[] = receivedContext.split(",");
  LOG.debug("Rec: Walking,Running,Stretching,Cyclng,Sitting");

  char[] loc = mapLocationRec.get(context[0]).replaceAll(",", "").toCharArray();
  LOG.debug("Location"  + mapLocationRec.get(context[0]));

  char[] hlc = mapHLCRec.get(context[1]).replaceAll(",", "").toCharArray();
  LOG.debug("HLC"  + mapLocationRec.get(context[1]));
```

경희대학교
KYUNG HEE UNIVERSITY

PARecInterpretations Snapshot Example

34
0

org.uclab.scl.framework.RecInterpreter

Class PARecInterpretations

java.lang.Object
    org.uclab.scl.framework.RecInterpreter.PARecInterpretations

```
public class PARecInterpretations
extends java.lang.Object
```

Field Summary

Fields

Modifier and Type	Field and Description
static java.lang.String	`description`
static java.lang.String	`durationValue`

Constructor Summary

Constructors

Constructor and Description
`PARecInterpretations()`

Method Summary

All Methods | **Instance Methods** | **Concrete Methods**

Modifier and Type	Method and Description
void	`buildPARecInterpretations(AbstractInterpreterPacket interpreterPacket)` Receives control flow for Physical Recommendation and evaluates its contextually.
AbstractRecEvaluator	`getEvaluator(java.lang.String rec)`
AbstractRecExplanation	`getRecExpalanation()`
void	`setRecExpalanation(AbstractRecExplanation recExpalanation)`

<<Java Class>>
PARecInterpretations
org.uclab.scl.Framework.RecInterpreter

- LOG: Logger
- description: String
- durationValue: String
- currentContext: CurrentContext
- dclManager: DataCurationCommunicationManager

- PARecInterpretations()
- buildPARecInterpretations(AbstractInterpreterPacket):void
- getRecExpalanation():AbstractRecExplanation
- setRecExpalanation(AbstractRecExplanation):void
- getEvaluator(String):AbstractRecEvaluator

```java
public class PARecInterpretations {
  private static Logger LOG = LogManager.getRootLogger();
  public static String description = "";
  public static String durationValue = "";
  private CurrentContext currentContext = new CurrentContext();
  CommunicationManager commManager = CommunicationManager.getDataCurationCommunicationManager();
  private AbstractRecExplanation recExplanation;

  /**
   * Receives control flow for Physical Recommendation and evaluates its contextually.
   * If required generates an alternative recommendation.
   * This function serves as a control unit for Recommendation Interpreter
   * @param interpreterPacket
   */
  public void buildPARecInterpretations(AbstractInterpreterPacket interpreterPacket) {


   public AbstractRecExplanation getRecExpalanation() {
       return recExplanation;
   }
   public void setRecExpalanation(AbstractRecExplanation recExpalanation) {
       this.recExplanation = recExpalanation;
   }
   public AbstractRecEvaluator getEvaluator(String rec){
       int contentLength = rec.split(",").length;
       AbstractRecEvaluator evaluator = null;

       switch (contentLength) {
       case 1:
           evaluator = new SingleRecEvaluator();
       break;
       default:
           evaluator = new MultipleRecEvaluator();
       }

       return evaluator;
   }
```

Explanation Builder Snapshot Example

34
1

<<Java Class>>
ExplanationBuilder
org.uclab.scl.Framework.RecInterpreter.Explanation

- mapRecExp: HashMap<String,String>
- mapExpURL: HashMap<String,String>
- mapPostProcess: HashMap<String,String>

- ExplanationBuilder()
- parseAggregateVec(int[]):String
- buildUrl(String):String
- handleExplanation(int[],String,String,String):I...
- explanationSentenceTemplate(String,String,...

org.uclab.scl.framework.RecInterpreter.Explanation

Class ExplanationBuilder

java.lang.Object
org.uclab.scl.framework.RecInterpreter.Explanation.AbstractRecExplanation
org.uclab.scl.framework.RecInterpreter.Explanation.ExplanationBuilder

```
public class ExplanationBuilder
extends AbstractRecExplanation
```

Constructor Summary

Constructors

Constructor and Description
ExplanationBuilder()

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.String	buildUrl(java.lang.String recValue)
java.lang.String	explanationSentenceTemplate(java.lang.String rec, java.lang.St java.lang.String context)
InterpretedRecommendations	handleExplanation(int[] interpRec, java.lang.String duration, Generates explanatory statement for the received recommendation in case the explanatory statement moreover, post processing adds contextual information
java.lang.String	parseAggregateVec(int[] aggregateVec) Receives an input vector and provides corresponding recommendation

```
public String buildUrl(String recValue ) throws Exception{
    return (new URLBuilder().recURLBuilder(recValue));
}
/**
 * Generates explanatory statement for the received recommendation in case the received recommendation
 * an explanatory statement moreover, post processing adds contextual information to the statement for
 *
 * @author Imran
 * @version 2.5
 * @since September 2015
 *
 */
@SuppressWarnings("unchecked")
public InterpretedRecommendations handleExplanation(int[] interpRec, String duration, String context,
  RecExpMapper recExpMap = new RecExpMapper();
  InterpretedRecommendations interpretedRec = new InterpretedRecommendations();
  List<String> explanationArr = new ArrayList<String>();
  String rec = this.parseAggregateVec(interpRec);

  LOG.debug("Recommended Activity: " + rec);
  explanationArr.add(rec);
  interpretedRec.setRecActivity(rec);
  interpretedRec.setCurrentContext(context);

  LOG.debug("Duration (mins): " + duration);
  explanationArr.add(duration);
  interpretedRec.setActDuration(duration);
```

KYUNG HEE UNIVERSITY

FoodRecInterpretations Snapshot Example

34
2

<<Java Class>>
FoodRecInterpretations
org.uclab.scl.Framework.RecInterpreter

- FoodRecInterpretations()
- getFoodRecommendation(AbstractInterpreterPacket):void
- getRecExplanation():AbstractRecExplanation
- setRecExplanation(AbstractRecExplanation):void

org.uclab.scl.framework.RecInterpreter.FoodItemRecommender

Class SNSTrendIdentifier

java.lang.Object
org.uclab.scl.framework.RecInterpreter.FoodItemRecommender.SNSTrendIdentifier

```
public class SNSTrendIdentifier
extends java.lang.Object
```

Constructor Summary

Constructors

Constructor and Description
SNSTrendIdentifier()

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method and Description
java.lang.String	processFoodRec(java.lang.String nutrientLabel, java.lang.String userid) The main purpose of this class is to process food related recommendations.

```
public class SNSTrendIdentifier{
  private static Logger LOG = LogManager.getRootLogger();

  static HashMap<String, String> mapNutrientToCategory = new HashMap<>();
  static HashMap<String, String> mapCatToFooditems = new HashMap<>();
  static HashMap<String, String> mapFooditemsToNutrient = new HashMap<>();
  static HashMap<String, String> mapUserChoices = new HashMap<>();

  /**
   * The main purpose of this class is to process food related recommendations. This function rece
   * Current trends against the food category are received from [SO] which in turn calls SNS servi
   * Received food items are filtered based on the major nutrients and user's dislike list
   * Final list of food items is provided to getFoodRecommendation() function
   *
   * @param nutrientLabel
   * @param userid
   * @return
   * @throws Exception
   */
  @SuppressWarnings("unchecked")
  public String processFoodRec(String nutrientLabel, String userid) throws Exception {
    NutrientToCategoryMapper nutritionToCat = new NutrientToCategoryMapper();
    CategoryToFooditemsMapper catToFood = new CategoryToFooditemsMapper();
    mapNutrientToCategory = nutritionToCat.NutrientToCategoryMapper();
    mapCatToFooditems = catToFood.CatToFooditemsMapper();
    String foodCategory = mapNutrientToCategory.containsKey(nutrientLabel) ? mapNutrientToCategory
    if (foodCategory == null)
      LOG.debug("FoodCategory NOT FOUND in the repository");
    LOG.debug("Nutrient to Category: " + nutrientLabel + ": " + foodCategory);
    CommunicationManager commManager = CommunicationManager.getDataCurationCommunicationManager();
```

Mining Minds

Supporting Layer
Project Progress

Shujaat Hussain
Jamil Hussain
Anees ul Hassan

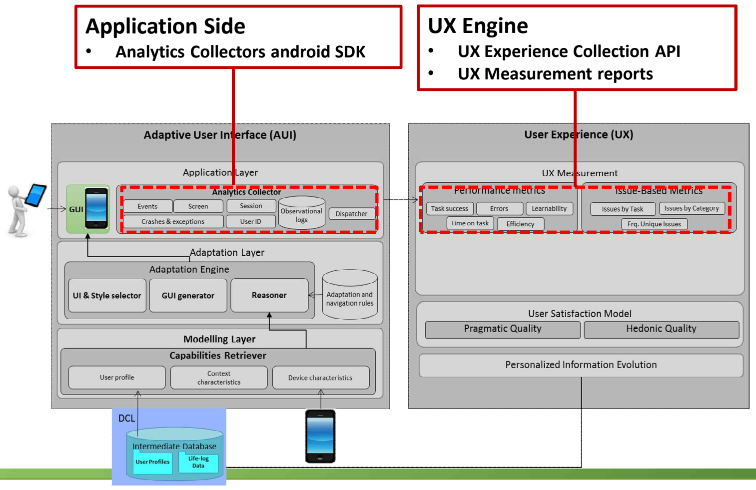
Mining Minds

UI/UX Authoring Tool
Project Progress

Jamil Hussain
Anees Ul Hassan

MMv3.0 UI/UX Authoring Tool

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain



UI/UX Authoring Tool: Open Source Code Progress

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

- UI/UX Authoring Tool development
  - Adaptive UI application development
  - UX Measurement tool kit as web application for UX quantification

UI/UX Authoring Tool: MVC Design Pattern

34
7

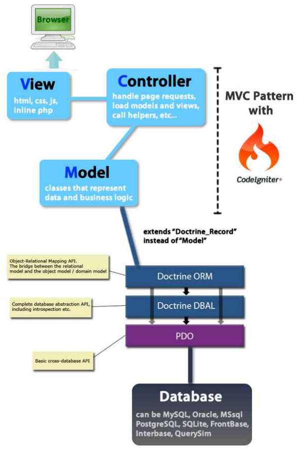
MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

UX Measurement tool kit as web application for UX quantification

Advantages

provides model-view-controller architecture

- To develop flexible applications
- To develop loosely coupled applications
- To provide separation in
  - Input logic
  - Business logic
  - UI logic



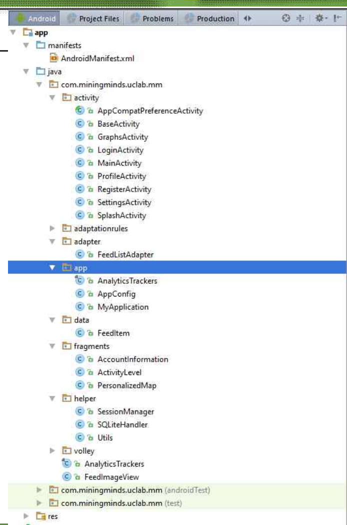
경희대학교
KYUNG HEE UNIVERSITY

UI/UX Authoring Tool: Open Source Code Progress

34
8

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

- Library Function Identification
  - Packages Categorization
    - com.miningminds.uclab.mm.app
    - com.miningminds.uclab.mm.activity
    - com.miningminds.uclab.mm.adaptationrules
    - com.miningminds.uclab.mm.adapter
    - com.miningminds.uclab.mm.data
    - com.miningminds.uclab.mm.fragments
    - com.miningminds.uclab.mm.helper
    - com.miningminds.uclab.mm.volleyapi



경희대학교
KYUNG HEE UNIVERSITY

UI/UX Authoring Tool: Open Source Code Progress

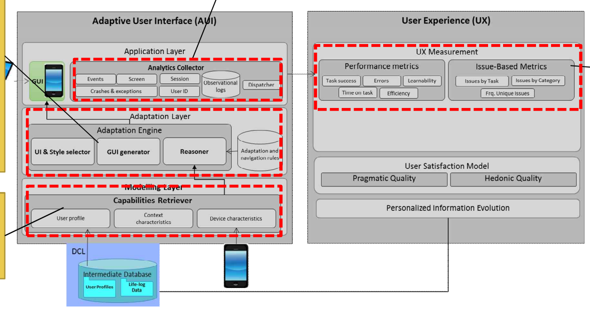
34
9

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

BaseActivity.java
MainActivity.java
RegisterActivity.java
easyRuleEngine.java
WellnessConceptsModelDAO.java
WellnessConceptsModelDAOImpl.java
ColourBlind.java
ColourBlindLowVision.java
LowVision.java
LowLevel.java
NosieLevel.java

AppConfig.java
Retrivedata.java
RetriveDeviceinformation.java
RetriveContext.java

AnalyticsTrackers.java
MyApplication.java



GooglеAnalyticsServiceModel.php
PerformanceMeteric
Controller.php
IssueBasedMetricsController.php

경희대학교
KYUNG HEE UNIVERSITY

UI/UX Authoring Tool: Open Source Code Progress

35
0

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

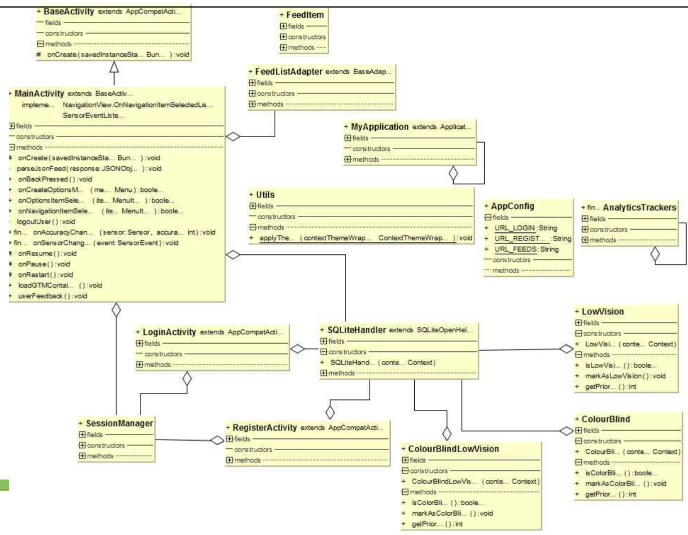
#	Title		Oct 2 Oct 8	Oct 9 Oct 15	Oct 16 Oct 22	Oct 23 Oct 29	Oct 30 Nov 5	Nov 6 Nov 12	Nov 13 Nov 19	Nov 20 Nov 26	Nov 27 Dec 3	Dec 4 Dec 10
1	Descriptive Analytics Library Functions Identification (Shujaat Hussain)	2w										
2	UI/UX Functions Identification (Jamil Hussain)	2w										
4	Descriptive Analytics Code Review (Shujaat Hussain)	1w										
5	UI/UX Code Review (Jamil Hussain)	1w										
6	Code Refactoring & Unit Testing (Shujaat and Jamil)	6w										
7	API Design & Documentation (Shujaat and Jamil)	4w										
8	Demo Harness (Shujaat and Jamil)	3w										
9	Build Creation (Shujaat and Jamil)	3w										
10	Open Source Release (Shujaat and Jamil)	2w										

KYUNG HEE UNIVERSITY

UI/UX Authoring Tool: Open Source Code Progress

351

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain



경희대학교 KYUNG HEE UNIVERSITY

App Package Open Source Code Progress

352

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

Package com.miningminds.uclab.mm.app

Class Summary

Class	Description
AnalyticsTrackers	A collection of Google Analytics trackers.
AppConfig	The type App config.
MyApplication	Created by Jamil on 13/08/15.

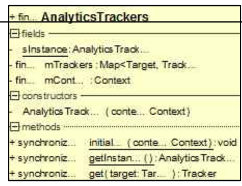
Class AppConfig

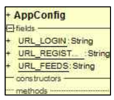
java.lang.Object
com.miningminds.uclab.mm.app.AppConfig

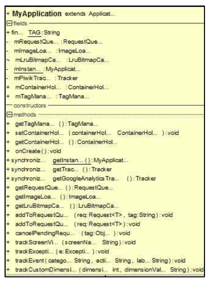
Field Summary

Fields

Modifier and Type	Field and Description
static java.lang.String	URL_FEEDS The constant URL_FEEDS.
static java.lang.String	URL_LOGIN The constant URL_LOGIN.
static java.lang.String	URL_REGISTER The constant URL_REGISTER.







경희대학교 KYUNG HEE UNIVERSITY

Main Application Class Open Source Code Progress

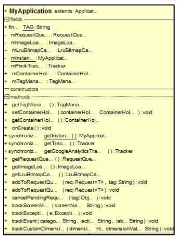
353

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

- Main Class that handle Adaptive UI/UX

Class MyApplication

java.lang.Object
android.content.Context
android.content.ContextWrapper
android.app.Application
com.miningminds.uclab.mm.app.MyApplication



Myapplication Ex
Code

```java
public void trackException(Exception e) {
    if (e != null) {
        Tracker t = getGoogleAnalyticsTracker();

        t.send(new HitBuilders.ExceptionBuilder()
                .setDescription(
                        new StandardExceptionParser(this, null)
                                .getDescription(Thread.currentThread().getName(), e))
                .setFatal(false)
                .build()
        );
    }
}
```

Methods

Modifier and Type	Method and Description
<T> void	addToRequestQueue(com.android.volley.Request<T> req) Add to request queue.
<T> void	addToRequestQueue(com.android.volley.Request<T> req, java.lang.String tag) Add to request queue.
void	cancelPendingRequests(java.lang.Object tag) Cancel pending requests.
com.google.android.gms.tagmanager.ContainerHolder	getContainerHolder() Gets container holder.
com.google.android.gms.analytics.Tracker	getGoogleAnalyticsTracker() Gets google analytics tracker.
com.android.volley.toolbox.ImageLoader	getImageLoader() Gets image loader.
static MyApplication	getInstance() Gets instance.
LruBitmapCache	getLruBitmapCache() Gets lru bitmap cache.
com.android.volley.RequestQueue	getRequestQueue() Gets request queue.
com.google.android.gms.tagmanager.TagManager	getTagManager() Gets tag manager.
org.piwik.sdk.Tracker	getTracker() Gets tracker.
void	onCreate() Called when the application is starting, before any activity, service, or receiver objects (excluding content providers) have been created.
void	setContainerHolder(com.google.android.gms.tagmanager.ContainerHolder containerHolder) Sets container holder.
void	trackCustomDimension(int dimension, java.lang.String dimensionValue) Track Custom Dimension
void	trackEvent(java.lang.String category, java.lang.String action, java.lang.String label) Tracking event
void	trackException(java.lang.Exception e) Tracking exception
void	trackScreenView(java.lang.String screenName) Tracking screen view

경희대학교
KYUNG HEE UNIVERSITY

Analytics Tracker Open Source Code Progress

354

- Core component for collection of user observational data

Class AnalyticsTrackers

java.lang.Object
com.miningminds.uclab.mm.AnalyticsTrackers

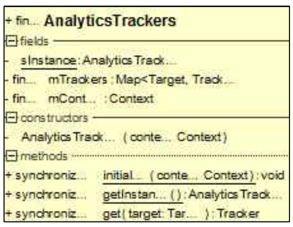
```java
public final class AnalyticsTrackers
extends java.lang.Object
```

A collection of Google Analytics trackers. Fetch the tracker you need using AnalyticsTrackers.getInstance().get(...)

Method Summary

Methods

Modifier and Type	Method and Description
com.google.android.gms.analytics.Tracker	get(AnalyticsTrackers.Target target) Get tracker.
static AnalyticsTrackers	getInstance() Gets instance.
static void	initialize(android.content.Context context)



Example Code

```java
public synchronized Tracker get(Target target) {
  if (!mTrackers.containsKey(target)) {
    Tracker tracker;
    switch (target) {
      case APP:
        tracker = GoogleAnalytics.getInstance(mContext).newTracker(R.xml.app_tracker);
        break;
      default:
        throw new IllegalArgumentException("Unhandled analytics target " + target);
```

경희대학교
KYUNG HEE UNIVERSITY

Adaptation Rule Open Source Code Progress

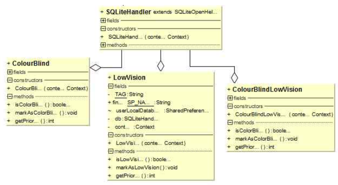
355

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

Package com.miningminds.uclab.mm.adaptationrules

Class Summary

Class	Description
ColourBlind	The type Colour blind.
ColourBlindLowVision	The type Colour blind low vision.
LowVision	Created by jamil on 11/3/2015.



Low Vision Rule

Method Summary

Adaptation Rules

Methods

Modifier and Type	Method and Description
int	getPriority() Gets priority.
boolean	isLowVision() Is low vision boolean.
void	markAsLowVision() Mark as low vision.

Example Code

```
@Rule(name = "ColourBlind",
      description = "Check if the person's is colour-blind and marks the person as adult")
public class LowVision {

    private static String TAG = LowVision.class.getSimpleName();
    /**
     * The constant SP_NAME.
     */
    public static final String SP_NAME = "theme";
```

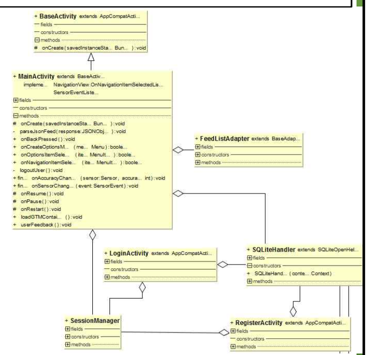
경희대학교 KYUNG HEE UNIVERSITY

Activities Open Source Code Progress

356

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

Package com.miningminds.uclab.mm.activity



Class Summary

Class	Description
AppCompatPreferenceActivity	A PreferenceActivity which implements and proxies the necessary calls to be used with AppCompat.
BaseActivity	The type Base activity.
GraphsActivity	The type Graphs activity.
LoginActivity	The type Login activity.
MainActivity	The type Main activity.
ProfileActivity	The type Profile activity.
RegisterActivity	The type Register activity.
SettingsActivity	A PreferenceActivity that presents a set of application settings.
SettingsActivity.AdaptiveUIPreferenceFragment	This fragment shows data and sync preferences only.
SettingsActivity.DataSyncPreferenceFragment	This fragment shows data and sync preferences only.
SettingsActivity.GeneralPreferenceFragment	This fragment shows general preferences only.
SettingsActivity.NotificationPreferenceFragment	This fragment shows notification preferences only.
SplashActivity	The type Splash activity.

경희대학교 KYUNG HEE UNIVERSITY

Main Activity class Open Source Code Progress

35
7

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

Main Activity Class

Method Summary

Methods

Modifier and Type	Method and Description
void	loadGTMContainer() Load gtm container.
private void	logoutUser()
void	onAccuracyChanged(android.hardware.Sensor sensor, int accuracy) Called when the accuracy of the registered sensor has changed.
void	onBackPressed() Take care of popping the fragment back stack or finishing the activity as appropriate.
protected void	onCreate(android.os.Bundle savedInstanceState) Perform initialization of all fragments and loaders.
boolean	onCreateOptionsMenu(android.view.Menu menu) Initialize the contents of the Activity's standard options menu.
android.view.View	onCreateView(java.lang.String name, android.content.Context context, android.util.AttributeSet attrs) Standard implementation of LayoutInflater.Factory.onCreateView(java.lang.String, android.content.Context, android.util.AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.String).
android.view.View	onCreateView(android.view.View parent, java.lang.String name, android.content.Context context, android.util.AttributeSet attrs) Standard implementation of LayoutInflater.Factory2.onCreateView(View, String, Context, AttributeSet) used when inflating with the LayoutInflater returned by Activity.getSystemService(java.lang.String).
boolean	onNavigationItemSelected(android.view.MenuItem item) Called when an item in the navigation menu is selected.
boolean	onOptionsItemSelected(android.view.MenuItem item) This hook is called whenever an item in your options menu is selected.
protected void	onPause() Dispatch onPause() to fragments.
protected void	onRestart() Called after Activity.onStop() when the current activity is being re-displayed to the user (the user has navigated back to it).
protected void	onResume() Dispatch onResume() to fragments.
void	onSensorChanged(android.hardware.SensorEvent event) Called when sensor values have changed.
private void	parseJsonFeed(org.json.JSONObject response)
void	userFeedback() User feedback.

```
+ MainActivity extends BaseActiv...
    impleme... NavigationView.OnNavigationItemSelectedLis...
               SensorEventListe...
+ fields
— constructors
- methods
#  onCreate(savedInstanceSta... Bun... ):void
-  parseJsonFeed(response:JSONObj... ):void
+  onBackPressed():void
+  onCreateOptionsM... (me... Menu):boole...
+  onOptionsItemSele... (ite... MenuIt... ):boole...
+  onNavigationItemSele... (ite... MenuIt... ):boole...
-  logoutUser():void
+ fin... onAccuracyChan... (sensor:Sensor, accura... int):void
+ fin... onSensorChang... (event:SensorEvent):void
#  onResume():void
#  onPause():void
#  onRestart():void
+  loadGTMContai... ():void
+  userFeedback():void
```

Example Code

```
@Override
public final void onSensorChanged(SensorEvent event) {
    // The light sensor returns a single value.
    // Many sensors return 3 values, one for each axis.
    float lux = event.values[0];
    // Do something with this sensor value.

    float average=0;
    int STOP_RAPID_CHANGES=30000;
    float  MIN_LIGHT=50;
     average = (4 * average + lux) / 5;
  // Toast.makeText(getApplicationContext(),"Light Intensity"+ lux , Toast.LENGTH_LONG).show();
  // Toast.makeText(getApplicationContext(),"Change Theme timer"+ Utils.themeChanged + "System (

    SharedPreferences SP = PreferenceManager.getDefaultSharedPreferences(getBaseContext());
```

경희대학교
KYUNG HEE UNIVERSITY

Helper Open Source Code Progress

35
8

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

Package com.miningminds.uclab.mm.helper

Class Summary

Class	Description
SessionManager	The type Session manager.
SQLiteHandler	The type Sq lite handler.
Utils	The type Utils.

Method Summary

Methods

Modifier and Type	Method and Description
boolean	isLoggedIn() Is logged in boolean.
void	setLogin(boolean isLoggedIn) Sets login.

```
+ SessionManager
- fields
-  TAG:String
-  pref:SharedPreferen...
-  editor:Editor
-  _cont... :Context
-  PRIVATE_MO... :int
- fin... PREF_NAME:String
- fin... KEY_IS_LOGGED... :String
- constructors
+  SessionMana... (conte... Context)
- methods
+  setLo... (isLogged... boole... ):void
+  isLogge... ():boole...
```

Example Code

```
public SessionManager(Context context) {
    this._context = context;
    pref = _context.getSharedPreferences(PREF_NAME, PRIVATE_MODE);
    editor = pref.edit();
}
/**
 * Sets login.
 *
 * @param isLoggedIn the is logged in
```

경희대학교
KYUNG HEE UNIVERSITY

Open Source Code Plan

359

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

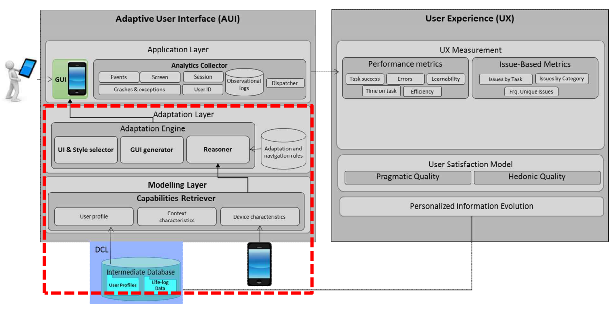
#	Title		Oct 2 Oct 8	Oct 9 Oct 15	Oct 16 Oct 22	Oct 23 Oct 29	Oct 30 Nov 5	Nov 6 Nov 12	Nov 13 Nov 19	Nov 20 Nov 26	Nov 27 Dec 3	Dec 4 Dec 10
1	Descriptive Analytics Library Functions Identification (Shujaat Hussain)	2w										
2	UI/UX Functions Identification (Jamil Hussain)	2w										
4	Descriptive Analytics Code Review (Shujaat Hussain)	1w										
5	UI/UX Code Review (Jamil Hussain)	1w										
6	Code Refactoring & Unit Testing (Shujaat and Jamil)	6w										
7	API Design & Documentation (Shujaat and Jamil)	4w										
8	Demo Harness (Shujaat and Jamil)	3w										
9	Build Creation (Shujaat and Jamil)	3w										
10	Open Source Release (Shujaat and Jamil)	2w										

경희대학교 KYUNG HEE UNIVERSITY

MMv3.0 UI/UX Authoring Tool: Adaptive UI

360

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain



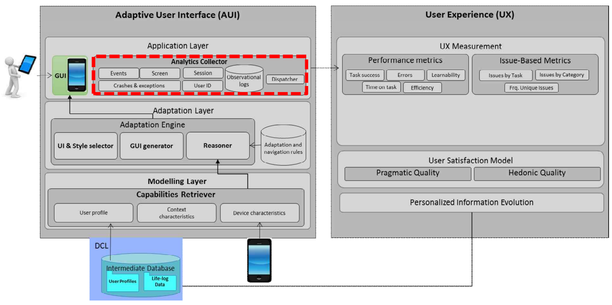
- **Input**
  - ✓ User Information
  - ✓ Device Information
  - ✓ Environmental information
- **Implementation Module**
  - ✓ Data modeling
  - ✓ Reasoner
  - ✓ Adaptation Rules
  - ✓ GUI Generator
- **Output**
  - ✓ Adaptive UI Generation based on user context

경희대학교 KYUNG HEE UNIVERSITY

MMv3.0 UI/UX Authoring Tool :Analytics Tracker

36
1

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain



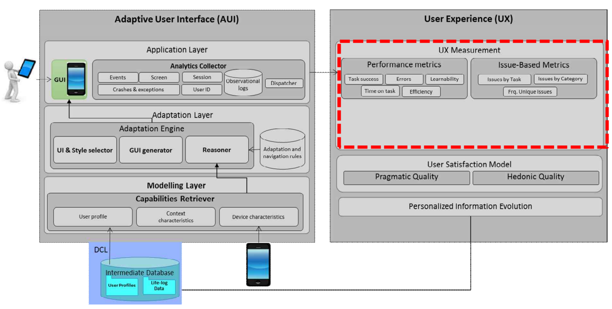
- **Input**
  - ✓ User actions while interacting with the system
- **Implementation Module**
  - ✓ Analytics Tracker
- **Output**
  - ✓ User Observational log in the form of UX metrics

경희대학교
KYUNG HEE UNIVERSITY

MMv3.0 UI/UX Authoring Tool : UX Measurements

36
2

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain



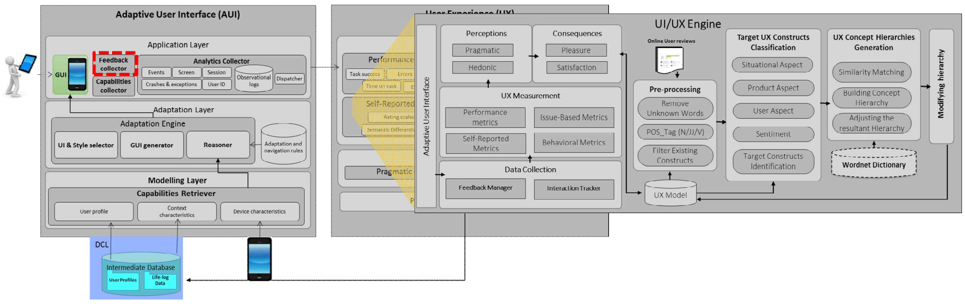
- **Input**
  - ✓ User Observational log
- **Implementation Module**
  - ✓ User experience Measurement
    - Performance Metrics
    - Issue based Metrics
- **Output**
  - ✓ UX quantification in the form of usability

경희대학교
KYUNG HEE UNIVERSITY

MMv3.0 UI/UX Authoring Tool: UX Engine

363

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain



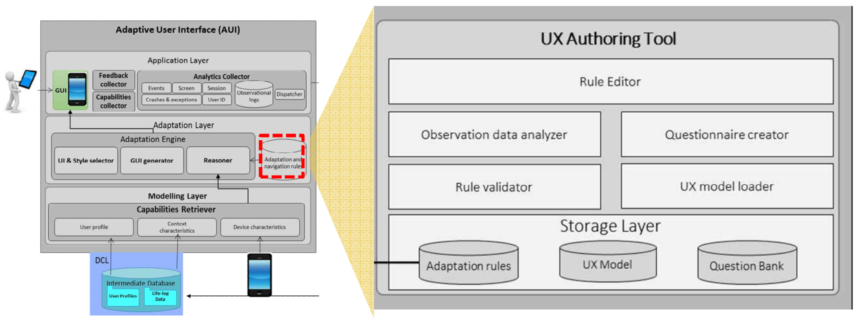
- **Input**
  - ✓ UX metrics
    - ▪ Self-reported metrics
    - ▪ Performance metrics
- **Implementation Module**
  - ✓ Feed Collector
  - ✓ UX Measurement
  - ✓ UX Model creation
- **Output**
  - ✓ Overall User Experience of MM3.0

경희대학교 KYUNG HEE UNIVERSITY

MMv3.0 UI/UX Authoring Tool:
UI Authoring Tool For Adaptive UI

364

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain



- **Input**
  - ✓ UX metrics
    - ▪ Self-reported metrics
    - ▪ Performance metrics
- **Implementation Module**
  - ✓ UX Authoring Tool
  - ✓ Rule Editor
  - ✓ UX Rule Validator
  - ✓ UX Model Loader
- **Output**
  - ✓ Overall User Experience of MM3.0

경희대학교 KYUNG HEE UNIVERSITY

UI/UX Authoring Tool: Open Source Code Progress

365

SL-UI/UX: Function Identification MM v3.0_SL_OpenSource_V0.2_20161021_Jamil Hussain

Public Functions

1 Adaptation Rule Engine

Rule Definition
- evaluateCondition()
- executeActions()
- setPriority()

Rule Engine
- RulesEngine().build()
- registerRule()
- CompositeRule()
- fireRules()
- scanner()
- scanner.nextLine()

2 Users
- SessionManager()
- setLogin()
- logoutUser()
- checkLogin()
- addUser()
- deleteUsers()
- updateUser()
- registerUser()
- userFeedback

3 Analytics Tracker
- Initialize()
- getInstanceTrackers()
- getTracker()
- trackEvent()
- trackScreenView()
- trackException()
- trackCustomDimension()

6 UX Engine
- set_analytics_start_date()
- set_analytics_end_date()
- set_metrics()
- set_dimensions()
- set_sort()
- set_filters()
- set_maxResult()
- get_analytics()
- get_dimensional_analytics()
- get_hourly_analytics ()
- get_daily_analytics()
- get_date_analytics()
- get_weekly_analytics()
- get_monthly_analytics()
- get_yearly_analytics()
- get_country_analytics()
- PerformaceMetrics ()
- Self-repotedMetics ()

4 Context monitor
- onSensorChanged()
- CheckCurrentContext()
- checkLightValue()
- checkNosieLevel()
- checkLocation()
- evaluateContext()

5 Adaptation Manger
- CheckCurrentTheme()
- ChangeTheme()

Total Public Functions: 58
Total Private Functions: 30
Total number of classes: 20

UI/UX Authoring Tool: Open Source Code Progress

366

UI/UX: Code Refactoring MM v3.0_SL_OpenSource_V0.2_20161021_Jamil Hussain

Refactoring Examples

- **Rename:** Changing name of misleading and confusing method, variable, class, or any other item
- **Scalable and flexible Design Pattern:** Using a design pattern for scalability and flexibility.
- **Move Class:** Moving a class from wrong package to better fit package
- **Extract Methods:** A long method needs to be broken up to enhance readability and maintainability
- **Extract Supper Class:** The common behavior is "pulled up" into new parent abstract class.
- **Replace conditional with polymorphism:** Replace multiple conditional operators (If, Switch) with polymorphism to reduce extensive change in code.
- **Exception Handling:** Handle exception at each level of code.

Identified Problems

1) Design Pattern implementation
2) Exception Handling
3) Remove unnecessary comments ,spaces and addition of useful comments for code understandability

경희대학교 KYUNG HEE UNIVERSITY

UI/UX: Code Refactoring MM v3.0_SL_OpenSource_V0.2_20161021_Jamil Hussain

1. Design Pattern Implementation

- **For complex rules handling**
  - **composite design pattern:** A composite rule is triggered if all conditions of its composing rules are satisfied. When a composite rule is applied, actions of all composing rules are performed in the natural order of rules which is rules priorities by default.
- **Scheduling a rules engine**
  - **Java scheduler Quartz :** This will schedule the rules engine to start based on specified time-based
- **Handling runtime event condition handling**
  - **Observer Design pattern**: It detect the runtime change in any event condition and change the adaptation rule engine behavior after that change

Current code problems

- ❑ **Current Rule Engine not handling the complex adaptation rules**
- ❑ **Not able to set scheduling for rechecking the current situation for adaptation on UI.**
- ❑ **No handling the runtime change in event condition.**

UI/UX: Code Refactoring MM v3.0_SL_OpenSource_V0.2_20161021_Jamil Hussain

1.Exception Handling

- **All kind of exception must be handle by using try-catch**
  - Catch all types of exception so that MM application will be never crash
  - All interface level code will be change

Current code problems

- ❑ **Most of code is not written in try-catch block**
- ❑ **Some time the MM application crash due to exceptions.**

UI/UX Authoring Tool: Open Source Code Progress

36
9

UI/UX: Code Refactoring MM v3.0_SL_OpenSource_V0.2_20161021_Jamil Hussain

1. Remove unnecessary comments , spaces and addition of useful comments

- **Unnecessary comments and spaces removal from code**
  - Enhance cleanliness of the code
- **Add useful comments to necessary code**
  - Enhance the code understandability

Current code problems

- **In current there some unnecessary comment and spaces**
- **Missing important comments**

경희대학교
KYUNG HEE UNIVERSITY

Open Source Code Plan

37
0

MM v3.0_SL_OpenSource_V0.2_20161021_Jamil Hussain

#	Title		Oct 2 Oct 8	Oct 9 Oct 15	Oct 16 Oct 22	Oct 23 Oct 29	Oct 30 Nov 5	Nov 6 Nov 12	Nov 13 Nov 19	Nov 20 Nov 26	Nov 27 Dec 3	Dec 4 Dec 10
1	Descriptive Analytics Library Functions Identification (Shujaat Hussain)	2w										
2	UI/UX Functions Identification (Jamil Hussain)	2w										
4	Descriptive Analytics Code Review (Shujaat Hussain)	1w										
5	UI/UX Code Review (Jamil Hussain)	1w										
6	Code Refactoring & Unit Testing (Shujaat and Jamil)	6w										
7	API Design & Documentation (Shujaat and Jamil)	4w										
8	Demo Harness (Shujaat and Jamil)	3w										
9	Build Creation (Shujaat and Jamil)	3w										
10	Open Source Release (Shujaat and Jamil)	2w										

KYUNG HEE UNIVERSITY

Open Source Code Plan

371

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

#	Title		Oct 2 – Oct 8	Oct 9 – Oct 15	Oct 16 – Oct 22	Oct 23 – Oct 29	Oct 30 – Nov 5	Nov 6 – Nov 12	Nov 13 – Nov 19	Nov 20 – Nov 26	Nov 27 – Dec 3	Dec 4 – Dec 10
1	Descriptive Analytics Library Functions Identification (Shujaat Hussain)	2w										
2	UI/UX Functions Identification (Jamil Hussain)	2w										
4	Descriptive Analytics Code Review (Shujaat Hussain)	1w										
5	UI/UX Code Review (Jamil Hussain)	1w										
6	Code Refactoring & Unit Testing (Shujaat and Jamil)	6w										
7	API Design & Documentation (Shujaat and Jamil)	4w										
8	Demo Harness (Shujaat and Jamil)	3w										
9	Build Creation (Shujaat and Jamil)	3w										
10	Open Source Release (Shujaat and Jamil)	2w										

경희대학교
KYUNG HEE UNIVERSITY

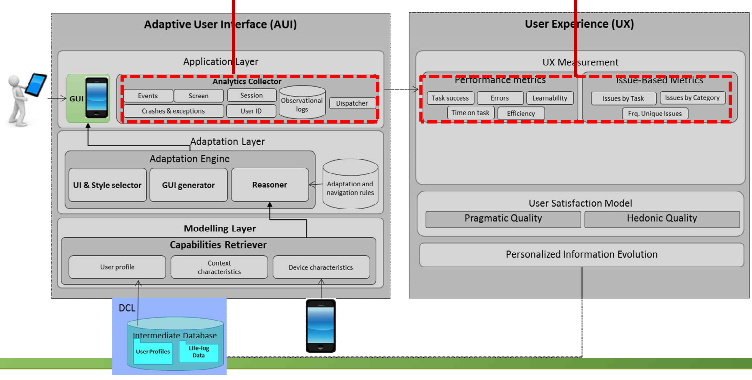
MMv3.0 UI/UX Authoring Tool

372

MM v3.0_SL_OpenSource_V0.1_20160903_Jamil Hussain

Application Side
- **Analytics Collectors android SDK**

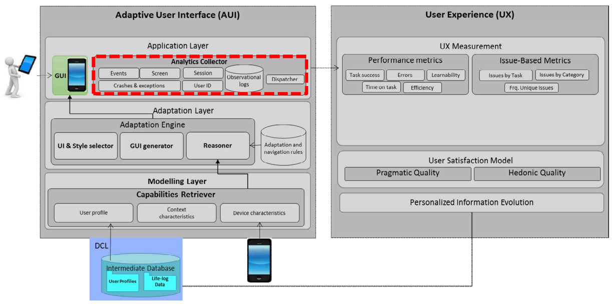
UX Engine
- **UX Experience Collection API**
- **UX Measurement reports**



경희대학교
KYUNG HEE UNIVERSITY

MMv3.0 UI/UX Authoring Tool :Analytics Tracker

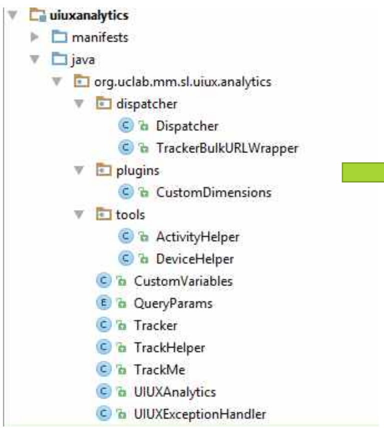
MM v3.0_SL_OpenSource_V0.2_20161203_Jamil Hussain



- **Input**
  - ✓ User actions while interacting with the system
- **Implementation Module**
  - ✓ Analytics Tracker
- **Output**
  - ✓ User Observational log

Analytics Tracker : Library Classes

MM v3.0_SL_OpenSource_V0.2_20161203_Jamil Hussain

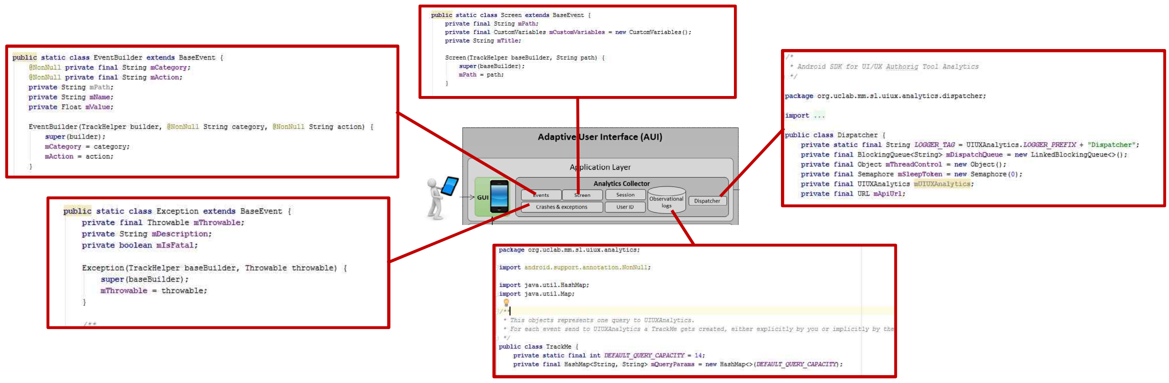


- **Custom Analytics tracker**
  - **Tracker the user interaction with app**
    - **Events**
    - **Screen**
    - **Exceptions**
    - **Session**
    - **User**

MMv3.0 UI/UX Authoring Tool : Analytics Tracker SDK

37
5

MM v3.0_SL_OpenSource_V0.2_20160903_Jamil Hussain



경희대학교
KYUNG HEE UNIVERSITY

Analytics Tracker SDK: Code Refactoring- Maintaining Log

37
6

MM v3.0_SL_OpenSource_V0.2_20161203_Jamil Hussain

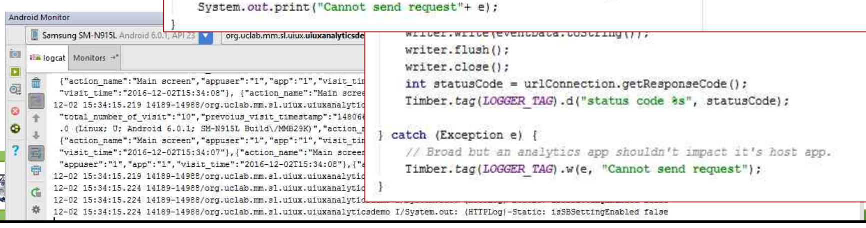
- Write and View Logs with Logcat

Before Refactoring

```
try {
    HttpURLConnection urlConnection = (HttpURLConnection) wrapper.getApiUrl().openConnection();
    urlConnection.setConnectTimeout(mTimeOut);
    urlConnection.setReadTimeout(mTimeOut);
    urlConnection.setDoOutput(true); // Forces post
    urlConnection.setRequestProperty("Content-Type", "application/json");
    urlConnection.setRequestProperty("charset", "utf-8");
    BufferedWriter writer = new BufferedWriter(new OutputStreamWriter(urlConnection.getOutputStream(), "UTF-8"))
    writer.write(eventData.toString());
    writer.flush();
    writer.close();
    int statusCode = urlConnection.getResponseCode();
    System.out.print("status code"+ statusCode);
} catch (Exception e) {
    // Broad but an analytics app shouldn't impact it's host app.
    System.out.print("Cannot send request"+ e);
}
```

After Refactoring using Logcat

```
Connection();
.getOutputSt
    writer.flush();
    writer.close();
    int statusCode = urlConnection.getResponseCode();
    Timber.tag(LOGGER_TAG).d("status code %s", statusCode);

} catch (Exception e) {
    // Broad but an analytics app shouldn't impact it's host app.
    Timber.tag(LOGGER_TAG).w(e, "Cannot send request");
}
```



Analytics Tracker SDK: Code Refactoring- Exception Handling

37
7

MM v3.0_SL_OpenSource_V0.2_20161203_Jamil Hussain

```java
@Override
public void uncaughtException(Thread thread, Throwable ex) {
    try {
        String excInfo = ex.getMessage();
        TrackHelper.track().exception(ex).description(excInfo).fatal(true).with(getTracker());
        // Immediately dispatch as the app might be dying after rethrowing the exception
        getTracker().dispatch();
    } catch (Exception e) {
        Timber.tag(Tracker.LOGGER_
    } finally {
        // re-throw critical excep
        if (getDefaultExceptionHan
            getDefaultExceptionHan
        }
    }
}
```

```java
public TrackMe build() {
    String className;
    try {
        StackTraceElement trace = mThrowable.getStackTrace()[0];
        className = trace.getClassName() + "/" + trace.getMethodName() + ":" + trace.getLineNumber();
    } catch (java.lang.Exception e) {
        Timber.tag(Tracker.LOGGER_TAG).w(e, "Couldn't get stack info");
        className = mThrowable.getClass().getName();
    }
    String actionName = "exception/" + (mIsFatal ? "fatal/" : "") + (className + "/") + mDescription;
    return new TrackMe(getBaseTrackMe())
            .set(QueryParams.ACTION_NAME, actionName)
            .set(QueryParams.EVENT_CATEGORY, "Exception")
            .set(QueryParams.EVENT_ACTION, className)
            .set(QueryParams.EVENT_NAME, mDescription)
            .set(QueryParams.EVENT_VALUE, mIsFatal ? 1 : 0);
}
```

경희대학교
KYUNG HEE UNIVERSITY

Analytics Tracker SDK: Doc Comments - Classes

37
8

MM v3.0_SL_OpenSource_V0.2_20161203_Jamil Hussain

```java
package org.uclab.mm.sl.uiux.analytics.tools;

import ...

/**
 * Helper class to gain information about the device we are running on
 */
public class DeviceHelper {
    private static final String LOGGER_TAG = UIUXAnalytics.LOGGER_PREFIX + "DeviceHelper";

    /**
     * Returns user language
     *
     * @return language
     */
    public static String getUserLanguage() { return Locale.getDefault().getLanguage(); }

    /**
     * Returns user country
     *
     * @return country
     */
    public static String getUserCountry() { return Locale.getDefault().getCountry(); }

    /**
     * Returns android system user agent
     *
     * @return well formatted user agent
     */
    public static String getUserAgent() { return System.getProperty("http.agent"); }
```

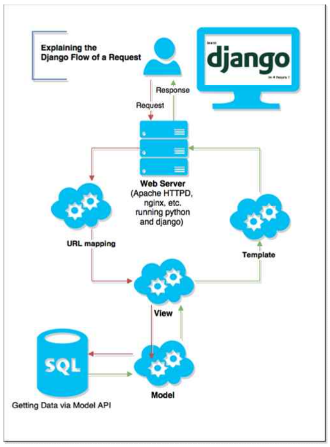
- Remove Useless comments
- Add functional Comments for the API Document

경희대학교
KYUNG HEE UNIVERSITY

UX Engine : Django Framework

MM v3.0_SL_OpenSource_V0.2_20161203_Jamil Hussain

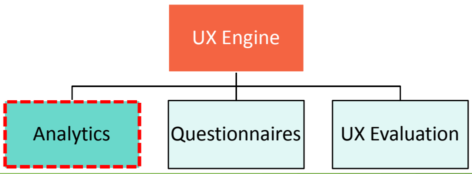
UX Engine tool kit as web application for UX quantification



Advantages

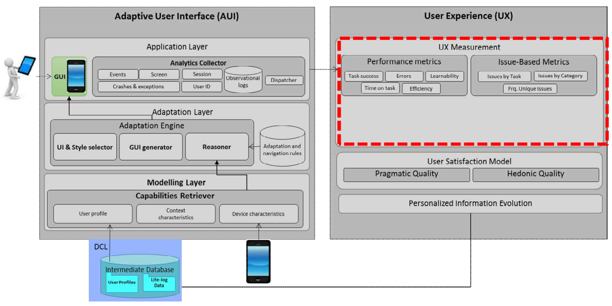
provides model-view-controller architecture

- To develop flexible applications
- To develop loosely coupled applications
- To provide separation in
  - Input logic
  - Business logic
  - UI logic



MMv3.0 UI/UX Authoring Tool : Analytics

MM v3.0_SL_OpenSource_V0.2_20161203_Jamil Hussain



- **Input**
  - ✓ User Observational log
- **Implementation Module**
  - ✓ User experience Measurement
    - Performance Metrics
    - Issue based Metrics
- **Output**
  - ✓ UX quantification in the form of usability

Analytics: Models

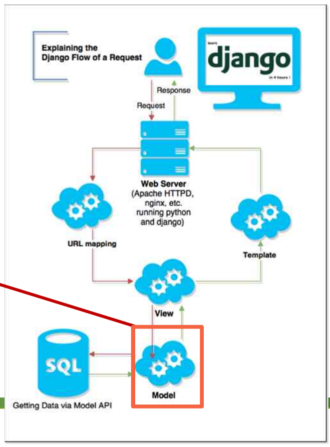
MM v3.0_SL_OpenSource_V0.2_20161203_Jamil Hussain

django REST framework

```
#APP must be create for the collection of Obeservational Log
class App(models.Model):
    app_name = models.CharField(max_length=50)

    def __str__(self):
        return se
```

```
#The data should be collected on the bases of user ID. This Model stor
class AppUser(models.Model):
    user_name = models.CharField(max_length=50)
    user_email = models.EmailField(max_length=50)
    user_password = models.CharField(max_length=50)
    user_sight = models.CharField(max_length=50, null=True)
    user_hearing = models.CharField(max_length=50, null=True)
```

```
class ActionLog(models.Model):
    action_name = models.CharField(max_length=100)
#Main Oberventional log Model that will store the Disptached user action log by
class Log(models.Model):
    actionlog = models.ForeignKey(ActionLog, related_name='logs', on_delete=mod
    app = models.ForeignKey(App, on_delete=models.CASCADE)
    appuser = models.ForeignKey(AppUser, on_delete=models.CASCADE)
    new_visit=models.IntegerField(blank=True,null=True)
    action_name = models.CharField(max_length=50, blank=True)
    entry_screen = models.CharField(max_length=50, blank=True)
    exit_screen = models.CharField(max_length=50, blank=True)
    visit_time = models.DateTimeField(blank=True,null=True)
    first_visit_timestamp = models.BigIntegerField(blank=True,null=True)
    prevoius_visit_timestamp = models.BigIntegerField(blank=True,null=True)
    screen_resolution = models.CharField(max_length=70, blank=True)
    user_agent = models.CharField(max_length=70, blank=True)
```

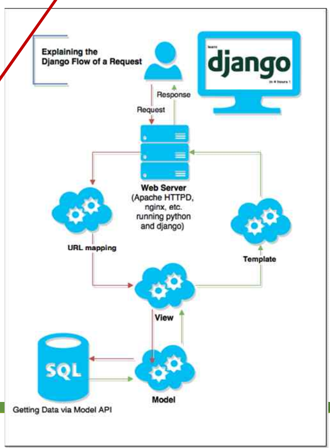


Analytics: API Design - user action log collections from app

MM v3.0_SL_OpenSource_V0.2_20161203_Jamil Hussain

django REST framework

```
from analytics.models import (Log, ActionLog)
from rest_framework import serializers


class LogSerializer(serializers.ModelSerializer):
    class Meta:
        model = Log
        fields = ('app','appuser','screen_resolution','action_name', 'entry_screen', 'exit_screen'

class ActionLogSerializer(serializers.ModelSerializer):
    logs = LogSerializer(many=True)

    action_name = serializers.HiddenField(default="Request")

    class Meta:
        model = ActionLog
        fields = ('action_name', 'logs')

    def create(self, validated_data):
        logs_data = validated_data.pop('logs')
        actionlog = ActionLog.objects.create(**validated_data)
        for logs_data in logs_data:
            Log.objects.create(actionlog=actionlog, **logs_data)
        return actionlog
```



Analytics: API Design - View

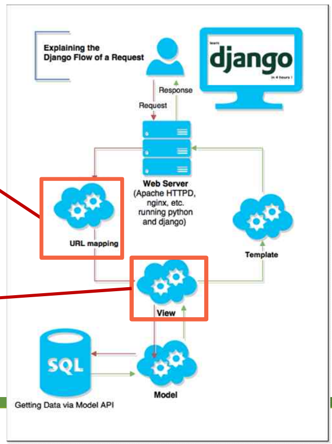
38
3

MM v3.0_SL_OpenSource_V0.2_20161203_Jamil Hussain

```
from django.conf.urls import url, include
from django.contrib import admin
from analytics import views
from rest_framework.routers import DefaultRouter


router = DefaultRouter()
router.register(r'actionlog', views.ActionLogViewSet)


urlpatterns = [

    url(r'^api/', include(router.urls)),
    url(r'^admin/', admin.site.urls),
]
```

```
from rest_framework import viewsets
from .api.serializers import ActionLogSerializer
from .models import ActionLog


#Api View for User Analytics Action Log
class ActionLogViewSet(viewsets.ModelViewSet):
    queryset = ActionLog.objects.all()
    serializer_class = ActionLogSerializer
```

django REST framework



경희대학교
KYUNG HEE UNIVERSITY

Open Source Code Plan

38
4

MM v3.0_SL_OpenSource_V0.2_20161021_Jamil Hussain

#	Title		Oct 2 – Oct 8	Oct 9 – Oct 15	Oct 16 – Oct 22	Oct 23 – Oct 29	Oct 30 – Nov 5	Nov 6 – Nov 12	Nov 13 – Nov 19	Nov 20 – Nov 26	Nov 27 – Dec 3	Dec 4 – Dec 10
1	Descriptive Analytics Library Functions Identification (Shujaat Hussain)	2w										
2	UI/UX Functions Identification (Jamil Hussain)	2w										
4	Descriptive Analytics Code Review (Shujaat Hussain)	1w										
5	UI/UX Code Review (Jamil Hussain)	1w										
6	Code Refactoring & Unit Testing (Shujaat and Jamil)	6w										
7	API Design & Documentation (Shujaat and Jamil)	4w										
8	Demo Harness (Shujaat and Jamil)	3w										
9	Build Creation (Shujaat and Jamil)	3w										
10	Open Source Release (Shujaat and Jamil)	2w										

KYUNG HEE UNIVERSITY

Mining Minds

UI/UX Authoring Tool

Third Party Open Source Libraries
Jamil Hussain
KHU

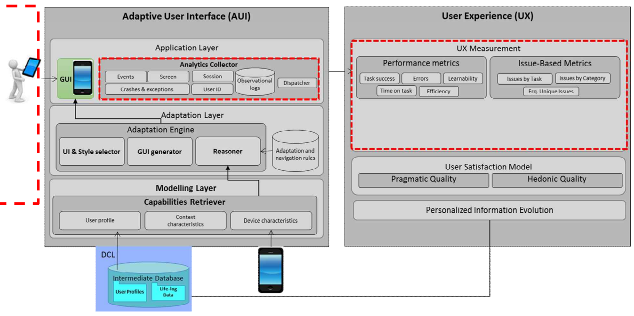
Agenda

- Open Source Third Party Libraries
  - Component wise Libraries
  - Licenses of Libraries

UI/UX Authoring Tool

38
7

- Django Framework
- Django-rest Framework
- SQLite
- Piwik
- Junit API



- We used these open source libraries without/without modification

경희대학교
KYUNG HEE UNIVERSITY

API Detail

38
8

Library	Purpose	License
Django	Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design	BSD license. https://opensource.org/licenses/BSD-3-Clause
Django-rest	Django REST framework is a powerful and flexible toolkit for building Web APIs.	BSD license. https://opensource.org/licenses/BSD-3-Clause
Piwik		BSD license https://opensource.org/licenses/BSD-3-Clause
Junit API	JUnit has been important in the development of test-driven development, and is one of a family of unit testing frameworks	Eclipse Public License 1.0 http://www.eclipse.org/legal/epl-v10.html

경희대학교
KYUNG HEE UNIVERSITY

Licenses comparison

Library	Commercial User	Modify	Distribute	Sublicense	Private use	Use Patent claims
The BSD 3-Clause License	✓	✓	✓	✓	✓	✓
Eclipse Public License 1.0 (EPL-1.0)	✓	✓	✓	✓	✓	✓

Mining Minds

Descriptive Analytics
Open Source Progress

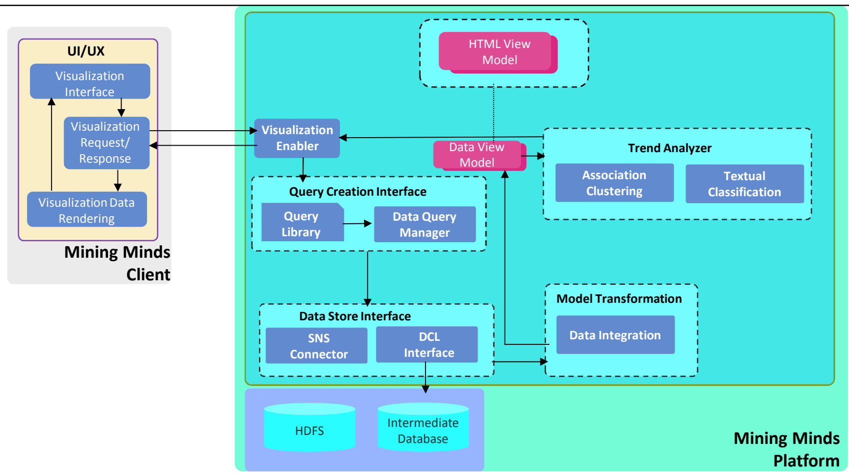
Shujaat Hussain

Agenda

- Open Source Status
  - Public Library Functions
  - Class Diagram
  - Code Refactoring

Descriptive Analytics

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

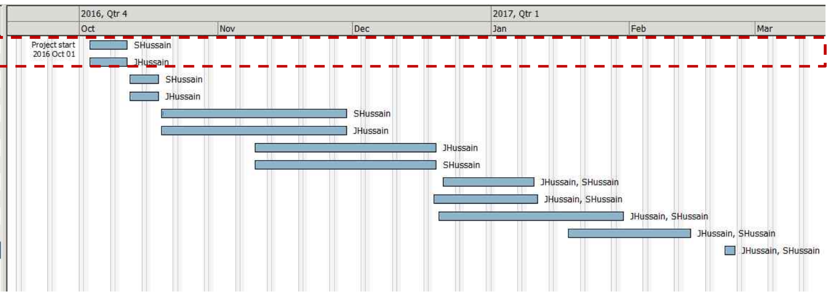


SL 2.5 Open Source Release Gantt Chart

393

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

WBS	Name	Work
1	Descriptive Analytics (DA) Library Function Identification	7d
2	Adaptive UI/UX Library Function Identification	7d
3	Code Review for DA	5d
4	Code Review for UI/UX	5d
5	DA Code Re-factoring	30d
6	UI/UX Code Re-factoring	30d
7	UI/UX API Design	29d
8	Web/Scripting Design for DA	29d
9	Unit Testing	30d
10	Build Creation	35d
11	Documentation	60d
12	Demo	40d
13	Open Source Release	5d



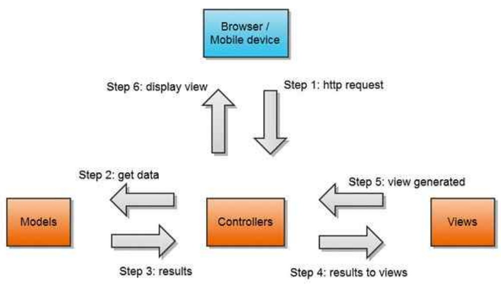
경희대학교 KYUNG HEE UNIVERSITY

Descriptive Analytics: MVC Design Pattern

394

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

- In an MVC framework, controllers manage and funnel data between the browser, models, and views.
- It allows for a faster development cycle.
- Plays well with Agile software development.
- The underlying code will change less often, resulting in a more stable site.



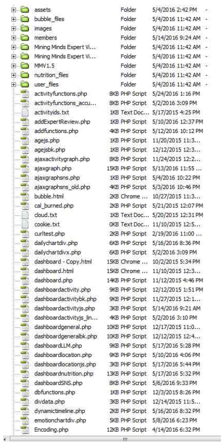
경희대학교 KYUNG HEE UNIVERSITY

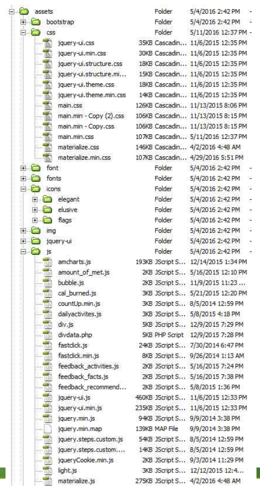
Descriptive Analytics : Open Source Code Progress

395

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

- Library Function Identification
  - Packages Categorization
    - Activity, Nutrition, LLM, Recommendation, SNS
      - Controller classes to process user request and building appropriate model to render on views
    - JSON POST and GET through DCL
      - Provides data model to use in communication among different components and DCL
    - Line, Bar, Pie, Notifications
      - Categorized classes for rendering graphs, user interfaces and expert notifications
    - Javascript functions and CSS files
      - Classes and libraries for rendering views for user interface





경희대학교
KYUNG HEE UNIVERSITY

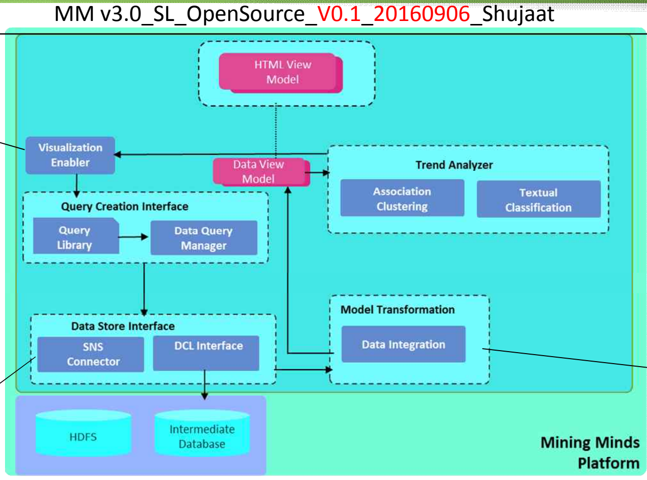
Descriptive Analytics : Open Source Code Progress

396

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

dynamictimeline.php
emotionchartdiv.php
dashboardnutrition.php
hlcchartdiv.php
dailychartdiv.php
monthlyactivites.php
weeklyactivites.php
monthlyemotionchartdiv.php
nutritiongraph.php
querylibrary.php
Jquery.js

snsservice.php
snsfunctions.php
dashboardnutrition.php
factfeedback.php
feedbackfunctions.php
servicecuration.php
nutritionsns.php



locationfacts.php
dashboardactivity.php
dashboardnutrition.php
factfeedback.php
feedbackfunctions.php
servicecuration.php
Calculateoverallactivity.php
Calculateoverallfat.php
Calculateactvityratio.php
Calculatelocationratio.php

Jsonparser.php
JSONConformation.php
Jquery.min.css
Mainstyle.css

경희대학교
KYUNG HEE UNIVERSITY

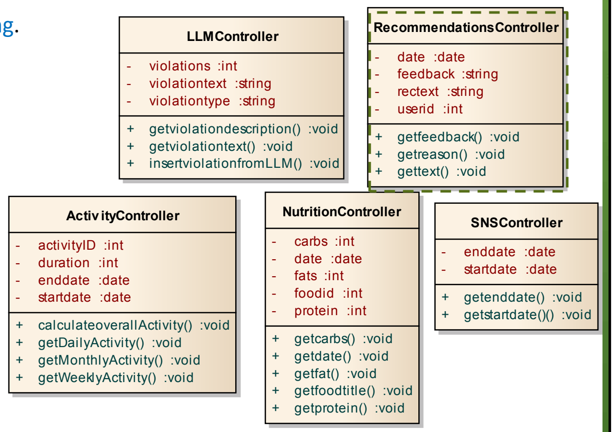
Descriptive Analytics-Open Source Code Progress

39
7

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

Controllers: are responsible for processing user requests and building appropriate model and passes it to the view for rendering.

1) Activity Controller
   - Daily Activities
   - Weekly and Monthly Activities
   - Dashboard Activities
   - Queried Activities
2) Nutrition Controller
   - Weekly and Monthly Nutrition
   - Dashboard Nutrition
   - SNS Nutrition
3) LLM Controller
   - Total User violations
   - Notifications of new situations
   - Priority of notifications
4) Recommendation Controller
   - Recommendation and feedbacks
5) SNS Controller
   - SNS data from Tapacross through dates



경희대학교 KYUNG HEE UNIVERSITY

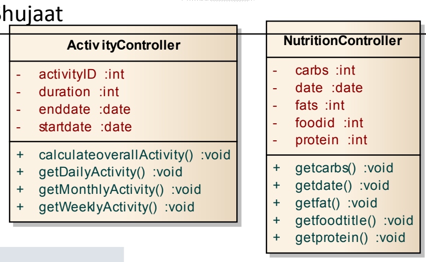
Controller Package: Open Source Code Progress

39
8

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

Core Component : Activity and Nutrition Packages

Class	Description
Activity Controller	Collection of Activity Functions
Nutrition Controller	Collection of Nutrition Functions



Type	Method	Description
JSON	calculateoverallActivity	Calculate the overall activity (monthly, weekly, daily)
JSON	getDailyActivity	Calculate the daily activity hourly
JSON	getWeeklyActivity	Calculate Weekly and monthly activity
Integer	getprotein	Calculate weekly and monthly protein consumption
String	getfoodtitle	Calculate weekly and monthly most consumed food

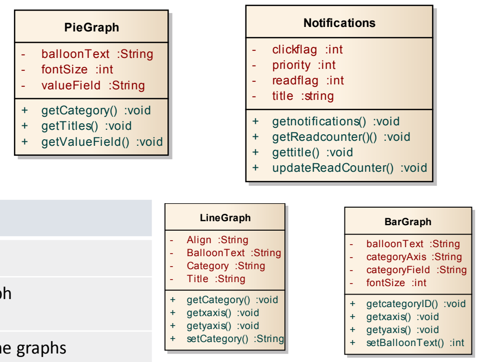
경희대학교 KYUNG HEE UNIVERSITY

Views Package: Open Source Code Progress

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

Core Component : Views (Graph) Packages

Class	Description
PieGraph	Piegraphs for activities
Notifications	Notifications for LLM
LineGraph	Line graph for SNS, Activity and nutrition
BarGraph	Bar graph for SNS, Activity, feedback and nutrition



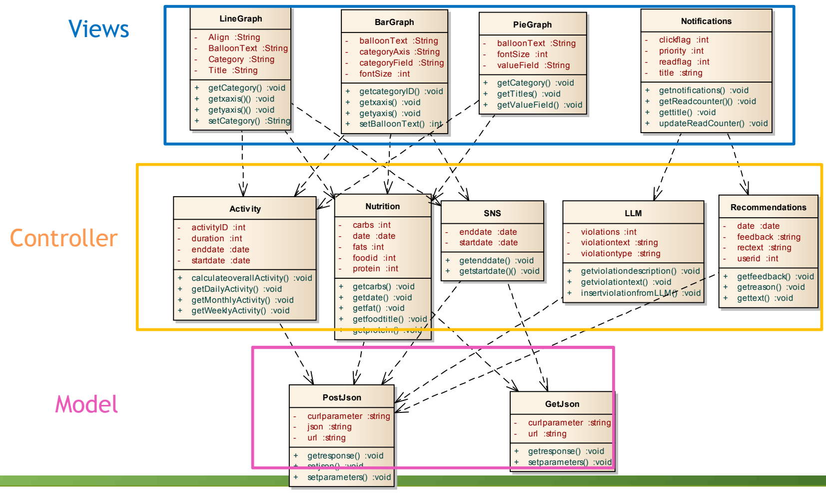
Type	Method	Description
JSON	getNotifications	Get Push notifications from LLM
JSON	getCategory	Get category names to be described in the graph
String	getBalloonText	Mouseover text to be shown and zoomed on the graphs
JSON	getyaxis	Yaxis and Xaxis coordinates in graph compliant format
INT	getValueField	Values for graph scales

Descriptive Analytics-Class Diagram

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

Advantages

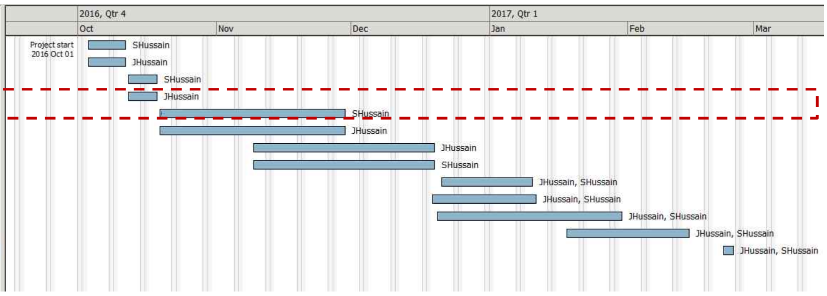
- To develop flexible applications
- To develop loosely coupled applications
- To provide separation in
  - Input logic
  - Business logic
  - UI logic



Open Source Code Plan

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

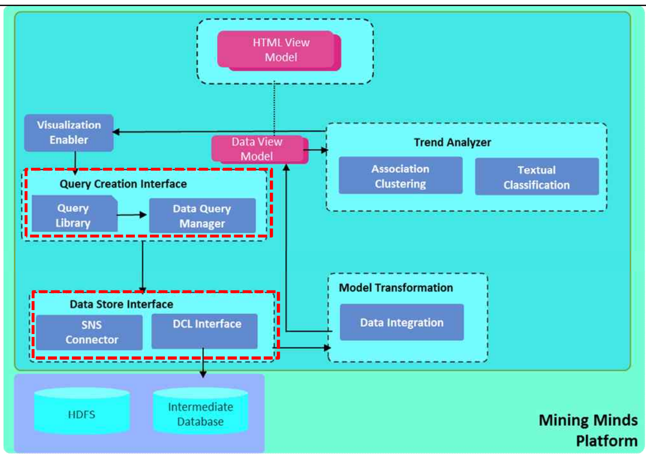
WBS	Name	Work
1	Descriptive Analytics (DA) Library Function Identification	7d
2	Adaptive UI/UX Library Function Identification	7d
3	Code Review for DA	5d
4	Code Review for UI/UX	5d
5	DA Code Re-factoring	30d
6	UI/UX Code Re-factoring	30d
7	UI/UX API Design	29d
8	Web/Scripting Design for DA	29d
9	Unit Testing	30d
10	Build Creation	35d
11	Documentation	60d
12	Demo	40d
13	Open Source Release	5d



Next Phase: Function Identification, Code Refactoring and Designing

Descriptive Analytics in MMv3.0-Query Creation Interface

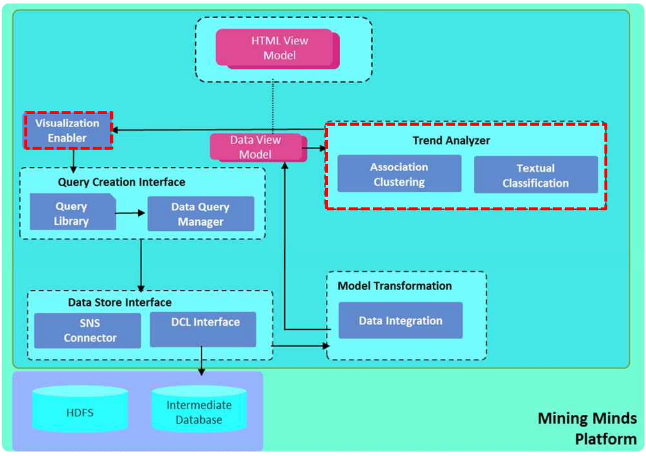
MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat



- Input
  - ✓ Life Log Data
  - ✓ Parameters (date, activity)
- Implementation Module
  - ✓ Data Query Manager
  - ✓ Query Library
  - ✓ SNS Interface
- Output
  - ✓ Queried Activity Data
  - ✓ Queried Nutrition Data
  - ✓ SNS data

Descriptive Analytics in MMv3.0-Trend Analyzer

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat



- Input
  - ✓ Transformed Life Log Data
- Implementation Module
  - ✓ Trend Analyzer
  - ✓ Visualization Enabler
- Output
  - ✓ Clustered Data points
  - ✓ Data Analytic Facts
  - ✓ Graphs and Trends

MMv3.0 Descriptive Analytics

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat

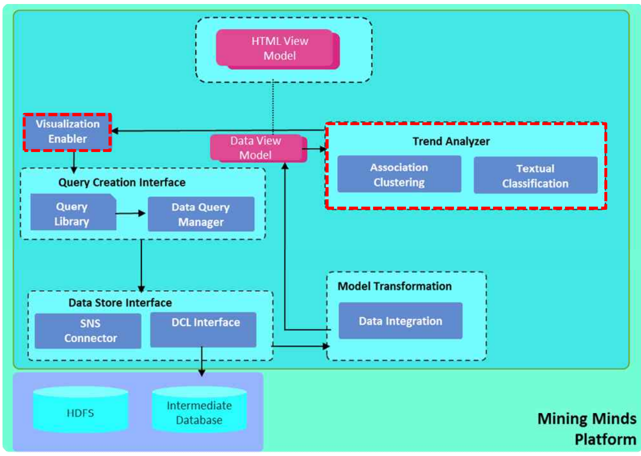
GC Healthcare Requirements

- **Nutrition**
- Management functionality of expert photo evaluation
- Must be able to view the food photo user uploaded
- *Must be able to view user kcal based data*

- **diabetes**

- Blood sugar level monitoring functionality (daily/weekly/monthly)
- Management functionality of user blood sugar level evaluation

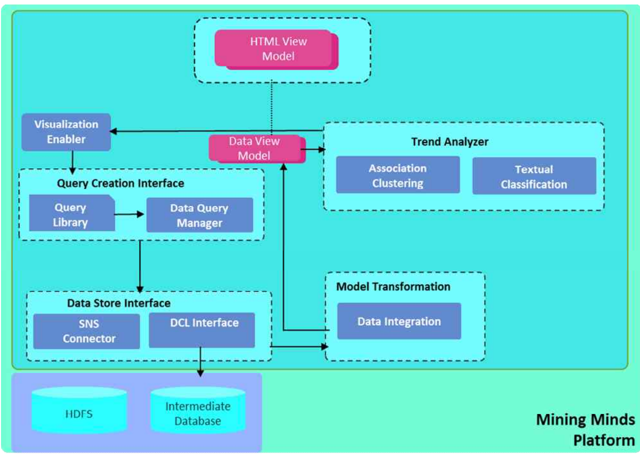
- **Water intake**
- Must be able to monitor the water intake (daily/weekly/monthly)



MMv3.0 Descriptive Analytics

405

MM v3.0_SL_OpenSource_V0.1_20160906_Shujaat



- Input
  - ✓ Life Log Data (Big Data) and clinical data
- Implementation Module
  - ✓ Trend Analyzer
  - ✓ Visualization Enabler
  - ✓ SNS Interface
- Output
  - ✓ Clinical Trends
  - ✓ Water Intake
  - ✓ Wellness and Clinical Analytics

SL Timeline

406

MM v3.0_SL_OpenSource_V0.2_20161022_Shujaat

#	Title		Oct 2 Oct 8	Oct 9 Oct 15	Oct 16 Oct 22	Oct 23 Oct 29	Oct 30 Nov 5	Nov 6 Nov 12	Nov 13 Nov 19	Nov 20 Nov 26	Nov 27 Dec 3	Dec 4 Dec 10
1	Descriptive Analytics Library Functions Identification (Shujaat Hussain)	2w										
2	UI/UX Functions Identification (Jamil Hussain)	2w										
4	Descriptive Analytics Code Review (Shujaat Hussain)	1w										
5	UI/UX Code Review (Jamil Hussain)	1w										
6	Code Refactoring & Unit Testing (Shujaat and Jamil)	6w										
7	API Design & Documentation (Shujaat and Jamil)	4w										
8	Demo Harness (Shujaat and Jamil)	3w										
9	Build Creation (Shujaat and Jamil)	3w										
10	Open Source Release (Shujaat and Jamil)	2w										

경희대학교 KYUNG HEE UNIVERSITY

Library Function Identification

407

MM v3.0_SL_OpenSource_V0.2_20161022_Shujaat

Activity
- getDailyActivity()
- getWeeklyActivity()
- getMonthlyActivity()
- getOverallActivity()
- getQueriedActivity()

High Level Context and Emotion, Location
- getWeeklyHLC()
- getMonthlyHLC()
- getWeeklyEmotion()
- getMonthlyEmotion()
- getMonthlyLocation

Recommendation/ Feedback
- getRecommendation()
- getFeedbackReason()
- getFeedback()
- getRecFeedbackbyUserID()

Query Panel(s)
- getAjaxBasedCategories()
- generateDynamicGraph()
- generateDynamicFacts()
- userSearch()

LLM/SNS
- getSNSbyFoodType()
- getTotalViolations()
- getViolationbyUserID()
- getPushNotifications()
- SNSGateway()

Nutrition
- calFoodFat()
- getFoodProtein()
- getFoodCarbs()
- getPopularFood()

Graphs/Analytics
- calAnalyticsActivity()
- calAnalyticsLocation()
- getWeeklyEmotion()
- getMonthlyEmotion()

Total Public Functions: <40
Total number of classes: <20

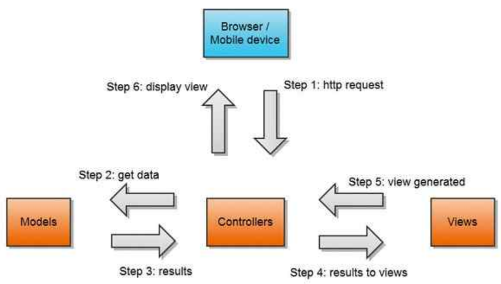
경희대학교 KYUNG HEE UNIVERSITY

Descriptive Analytics: MVC Design Pattern

408

MM v3.0_SL_OpenSource_V0.2_20161022_Shujaat

- In an MVC framework, controllers manage and funnel data between the browser, models, and views.
- It allows for a faster development cycle.
- Plays well with Agile software development.
- The underlying code will change less often, resulting in a more stable site.



경희대학교 KYUNG HEE UNIVERSITY

Controller Package: Open Source Code Progress

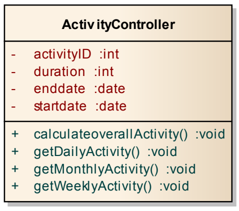
409

MM v3.0_SL_OpenSource_V0.2_20161022_Shujaat

Package Name: Activity and Nutrition

Class	Description
Activity Controller	Collection of Activity Functions

Type	Method	Description
Long	calculateOverallActivity	Calculate the overall activity (monthly, weekly, daily)
JSON	getDailyActivity	Calculate the daily activity hourly
JSON	getWeeklyActivity	Aggregate weekly/monthly activity
JSON	getQueriedActivity	Calculate specific activity based on defined range of dates



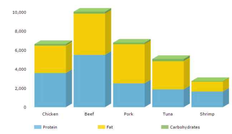
경희대학교 KYUNG HEE UNIVERSITY

Controller Package: Open Source Code Progress

410

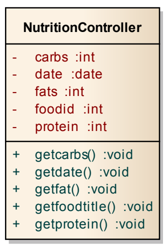
MM v3.0_SL_OpenSource_V0.2_20161022_Shujaat

Package Name: Activity and Nutrition



Class	Description
Nutrition Controller	Collection of Nutrition Functions

Type	Method	Description
Integer	getTotalProtein()	Calculate weekly /monthly protein consumption
String[]	getMostConsumedFood()	Calculate weekly /monthly most consumed food
Integer	getTotalCarbs()	Calculate weekly /monthly carbohydrates consumption
Integer	getTotalFats()	Calculate weekly /monthly fat consumption
JSON	getConsumedFoodbyDate()	Calculate weekly /monthly consumption of all users



경희대학교 KYUNG HEE UNIVERSITY

Code Refactoring: Exception Handling

411

MM v3.0_SL_OpenSource_V0.2_20161022_Shujaat

- Goal is to produce somewhat consistent error messages in case of a abnormal behavior

1. WrongTypeException
   - This occurs when JSON is passed or used as an array. Data structure is an incorrect type
2. MYSQL database connect
   - This occurred in case of push notifications when connecting with MYSQL database
3. Timeout Exception
   - In case of big data query this occurs. It is triggered after 60 seconds of the request with no reply.
4. Server down/ Empty Data
   - This was written when no data is returned.

경희대학교 KYUNG HEE UNIVERSITY

Code Refactoring: Comments

412

MM v3.0_SL_OpenSource_V0.2_20161007_Shujaat

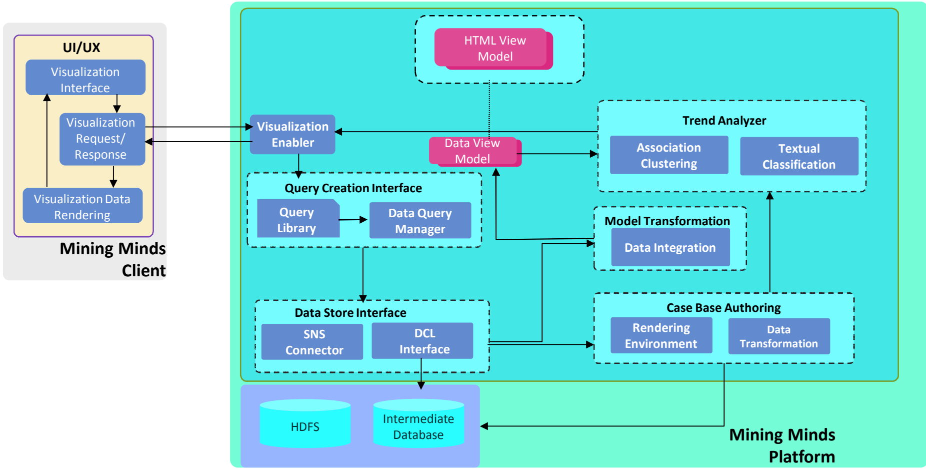
- The goal is to remove unnecessary comments and whitespaces
- It is necessary to improve the cleanliness of the overall code for better understanding
- There are many comments for understanding while coding at that moment.
  - Some are obsolete and unnecessary as the code has changed.
  - 25% of all the code comments are completed.

Good developers write good code; great ones also write good comments.

경희대학교 KYUNG HEE UNIVERSITY

Case Base Authoring

413



경희대학교
KYUNG HEE UNIVERSITY

Open Source Code Plan

414

MM v3.0_SL_OpenSource_V0.2_20161022_Shujaat

#	Title		Oct 2 Oct 8	Oct 9 Oct 15	Oct 16 Oct 22	Oct 23 Oct 29	Oct 30 Nov 5	Nov 6 Nov 12	Nov 13 Nov 19	Nov 20 Nov 26	Nov 27 Dec 3	Dec 4 Dec 10
1	Descriptive Analytics Library Functions Identification (Shujaat Hussain)	2w										
2	UI/UX Functions Identification (Jamil Hussain)	2w										
4	Descriptive Analytics Code Review (Shujaat Hussain)	1w										
5	UI/UX Code Review (Jamil Hussain)	1w										
6	Code Refactoring & Unit Testing (Shujaat and Jamil)	6w										
7	API Design & Documentation (Shujaat and Jamil)	4w										
8	Demo Harness (Shujaat and Jamil)	3w										
9	Build Creation (Shujaat and Jamil)	3w										
10	Open Source Release (Shujaat and Jamil)	2w										

경희대학교
KYUNG HEE UNIVERSITY

Open Source Code Plan

41 5

MM v3.0_SL_OpenSource_V0.3_20161202_Shujaat

#	Title		Oct 2 – Oct 8	Oct 9 – Oct 15	Oct 16 – Oct 22	Oct 23 – Oct 29	Oct 30 – Nov 5	Nov 6 – Nov 12	Nov 13 – Nov 19	Nov 20 – Nov 26	Nov 27 – Dec 3	Dec 4 – Dec 10
1	Descriptive Analytics Library Functions Identification (Shujaat Hussain)	2w										
2	UI/UX Functions Identification (Jamil Hussain)	2w										
4	Descriptive Analytics Code Review (Shujaat Hussain)	1w										
5	UI/UX Code Review (Jamil Hussain)	1w										
6	Code Refactoring & Unit Testing (Shujaat and Jamil)	6w										
7	API Design & Documentation (Shujaat and Jamil)	4w										
8	Demo Harness (Shujaat and Jamil)	3w										
9	Build Creation (Shujaat and Jamil)	3w										
10	Open Source Release (Shujaat and Jamil)	2w										

경희대학교 KYUNG HEE UNIVERSITY

Library Function Identification

41 6

MM v3.0_SL_OpenSource_V0.2_20161022_Shujaat

Activity
getDailyActivity()
getWeeklyActivity()
getMonthlyActivity()
getOverallActivity()
getQueriedActivity()

High Level Context and Emotion, Location
getWeeklyHLC()
getMonthlyHLC()
getWeeklyEmotion()
getMonthlyEmotion()
getMonthlyLocation

Recommendation/ Feedback
getRecommendation()
getFeedbackReason()
getFeedback()
getRecFeedbackbyUserID()

LLM/SNS
getSNSbyFoodType()
getTotalViolations()
getViolationbyUserID()
getPushNotifications()
SNSGateway()

Nutrition
calFoodFat()
getFoodProtein()
getFoodCarbs()
getPopularFood()

Graphs/Analytics
calAnalyticsActivity()
calAnalyticsLocation()
getWeeklyEmotion()
getMonthlyEmotion()

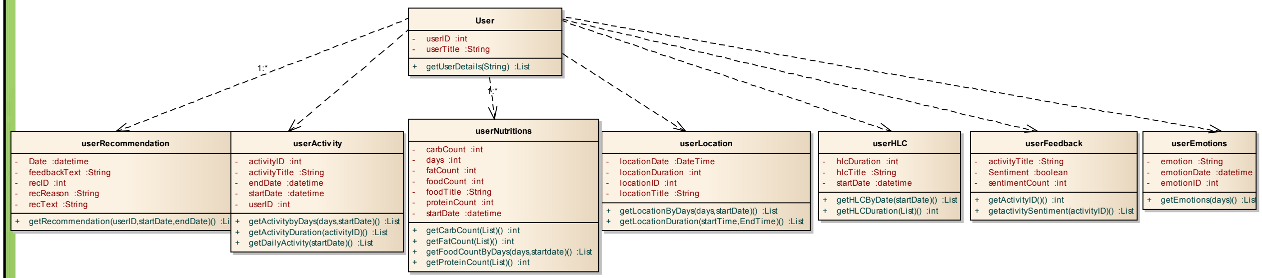
Total Public Functions: <40
Total number of classes: <20

경희대학교 KYUNG HEE UNIVERSITY

Individual User Class Diagram

41
7

MM v3.0_SL_OpenSource_V0.3_20161202_Shujaat



A user has **one to many** relationship with activities, context, HLC, emotions and location

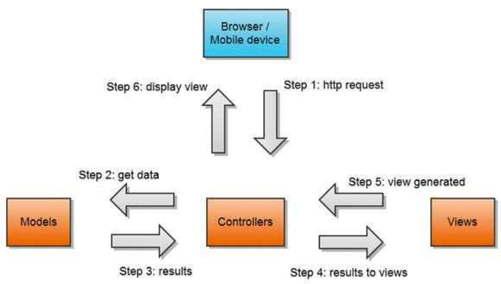
경희대학교
KYUNG HEE UNIVERSITY

Descriptive Analytics: MVC Design Pattern

41
8

MM v3.0_SL_OpenSource_V0.3_20161202_Shujaat

- In an MVC framework, controllers manage and funnel data between the browser, models, and views.
- It allows for a faster development cycle.
- Plays well with Agile software development.
- The underlying code will change less often, resulting in a more stable site.



경희대학교
KYUNG HEE UNIVERSITY

Design Patterns: MVC

MM v3.0_SL_OpenSource_V0.3_20161202_Shujaat

Dashboard Code

- Activity Dashboard
- Users' Activities
- Users' Locations
- Users' Nutrition
- Users' LLM
- Users' General Statistics

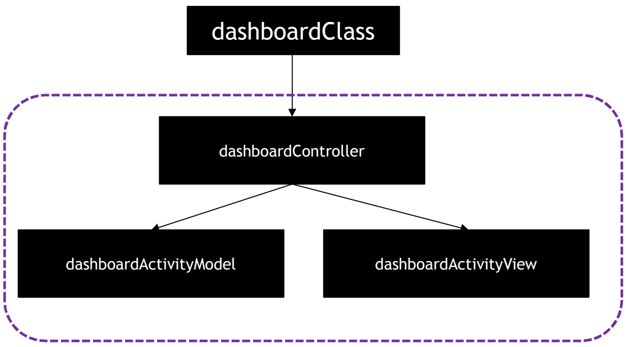
```
//Dashboard Activity Class represents the ration of all the activities.
// The analytics represents all the users in Mining Mind
//setter and getter functions
class dashboardActivity {

        private $data = array();

        public function __set($variable, $value){
        $this->data[$variable] = $value;
    }

        public function __get($variable){
            if(isset($this->data[$variable])){
                return $this->data[$variable];
            }else{
                die('Unknown variable.');
            }
    }

        }

$dashboardData = new dashboardActivity;
$dashboardData->userId = 0;
$dashboardData->startTime = $start;
$dashboardData->endTime = $end;

$data = array(

                "userId"=>$dashboardData->userId,
                "startTime"=>$dashboardData->startTime,
                "endTime"=>$dashboardData->endTime

        );
```


Design Patterns: MVC

MM v3.0_SL_OpenSource_V0.3_20161202_Shujaat

Dashboard Coverage

- Activity Dashboard
- Users' Activities
- Users' Locations
- Users' Nutrition
- Users' LLM
- Users' General Statistics



Code Refactoring

421

MM v3.0_SL_OpenSource_V0.3_20161202_Shujaat

```
global $dclserver;

$url="$dclserver/MMDataCurationRestfulService/webresources/In

//set start time 365 days ago
$starttimeobject = new DateTime('-364 days');
$starttimeobject->setTime(00,00,00);

//set end time today
$endtimeobject = new DateTime('NOW');
$endtimeobject->setTime(23,59,59);

//convert date objects into strings
$start = $starttimeobject->format('Y m d H:i:s');
$end = $endtimeobject->format('Y m d H:i:s');

$data = array(

            "userId"=>0,
        "startTime"=>$start,
        "endTime"=>$end

    );
$json = json_encode($data,true);
$result =postJsonRest($url,$json);
//echo "<pre> hello";
$result=json_decode($result,true);
//print_r($result);
$sittingtime=0;$standingtime=0;$walkingtime=0;$runningtime=0;
for($i=0;$i<count($result);$i++){

    //sitting
    if($result[$i]['activityId']==3){//sitting

        $sittingtime=$sittingtime+$result[$i]['duration'];

    }else if($result[$i]['activityId']==8){//standing
```

- Global Variables
- Structural Programming
- No comments what the code is doing

```
//Dashboard Activity Class represents the ration of all the activities.
// The analytics represents all the users in Mining Mind
//setter and getter functions
class dashboardActivity {


    private $data = array();


    public function __set($variable, $value){
    $this->data[$variable] = $value;
  }

    public function __get($variable){
        if(isset($this->data[$variable])){
            return $this->data[$variable];
        }else{
            die('Unknown variable.');
        }
  }


    }

$dashboardData = new dashboardActivity;
$dashboardData->userId = 0;
$dashboardData->startTime = $start;
$dashboardData->endTime = $end;


$data = array(

        "userId"=>$dashboardData->userId,
        "startTime"=>$dashboardData->startTime,
        "endTime"=>$dashboardData->endTime

    );
```

- Proper Comments and function
- Classes and object oriented approach

경희대학교 KYUNG HEE UNIVERSITY

Advantages and Lessons Learned

422

MM v3.0_SL_OpenSource_V0.3_20161202_Shujaat

Advantages

- ❑ Significantly faster execution time
- ❑ Precise and reliable
- ❑ Better code understandability
- ❑ Code more reusable for different objectives

Lessons Learned

- ❑ Reusability enhances efficiency
- ❑ Crash probability decreased
- ❑ Easy Debugging and testing

경희대학교 KYUNG HEE UNIVERSITY

Mining Minds

Descriptive Analytics

Third Party Open Source Libraries

Shujaat Hussain

KHU

경희대학교 KYUNG HEE UNIVERSITY · KITECH 한국생산기술연구원 · POSTECH 포항공과대학교 · UNIVERSITY of TASMANIA · 녹십자헬스케어 · TAPACROSS · U2 system User & Usability system · KOCSA 한국컨설팅서비스협회

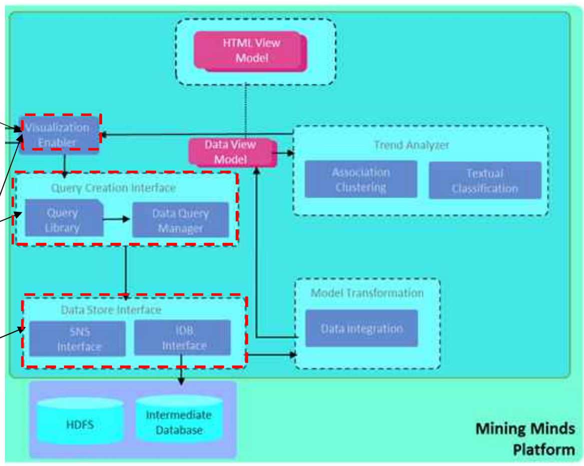
Descriptive Analytics

42
4

Amcharts: JavaScript / HTML5 charts

jQuery: JavaScript Library

Bootstrap: Front End Framework



- QUnit: A JavaScript Unit Testing framework.
- PHPUnit - The PHP Testing Framework

경희대학교 KYUNG HEE UNIVERSITY

API Detail

42
5

Library	Purpose	License
Jquery	• Fast, small, and feature-rich JavaScript library. • HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers.	MIT License http://www.opensource.org/licenses/mit-license.php
Amcharts	• JavaScript / HTML5 charts and maps data-viz libraries for web sites and applications. • Fast and responsive.	Single website license http://www.amcharts.com/online-store/licenses-explained
Bootstrap	• Bootstrap, a sleek, intuitive, and powerful mobile first front-end framework for faster and easier web development.	MIT License http://www.opensource.org/licenses/mit-license.php
Qunit	• QUnit is a JavaScript unit testing framework for testing jQuery, jQuery UI and jQuery Mobile, • It is a generic framework to test any JavaScript code.	MIT License http://www.opensource.org/licenses/mit-license.php
PHPUnit	• PHPUnit is a programmer-oriented testing framework for PHP.	Creative Commons license https://creativecommons.org/licenses/by/3.0/

경희대학교
KYUNG HEE UNIVERSITY

Licenses comparison

42
6

Library	Commercial User	Modify	Distribute	Sublicense	Private use	Use Patent claims
MIT License (Expat)	✓	✓	✓	✓	✓	✗
Creative Commons license	✓	Public Domain	Public Domain	Public Domain	Public Domain	✗
Single website license	✓	✓	✓	✗	✓	✗

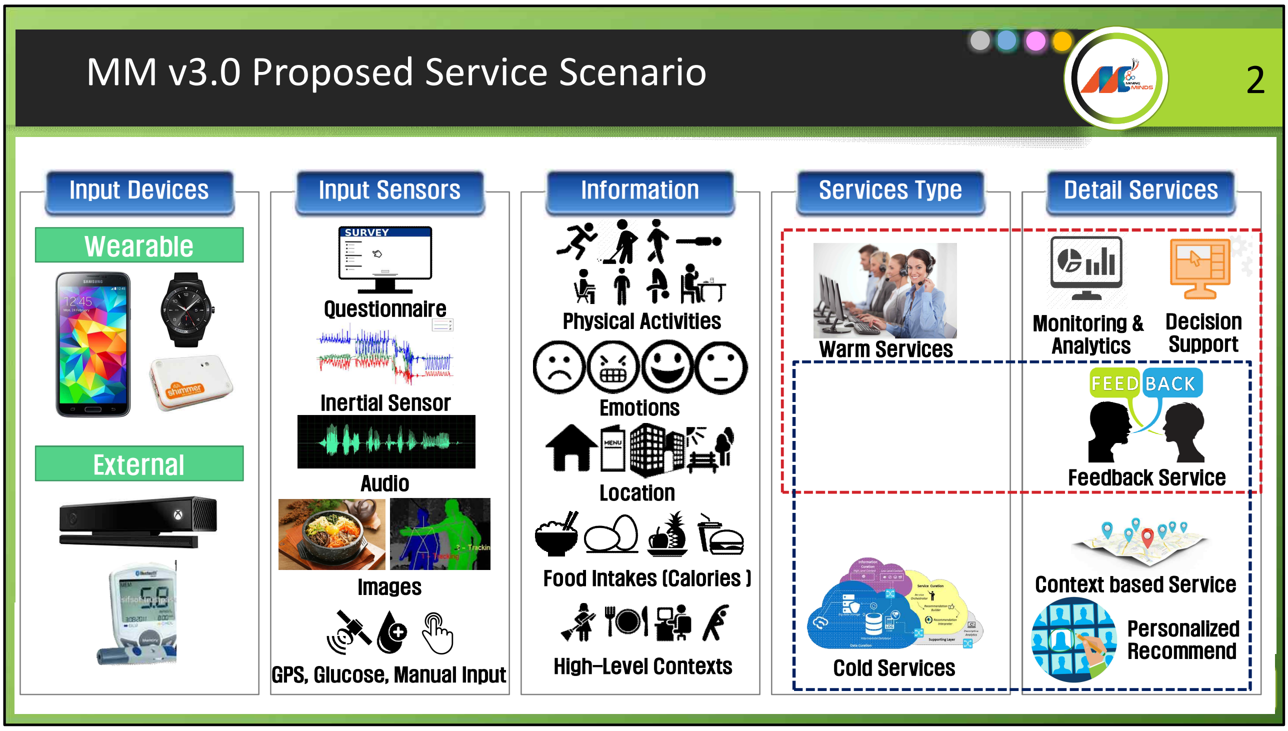
https://en.wikipedia.org/wiki/Comparison_of_free_and_open-source_software_licenses

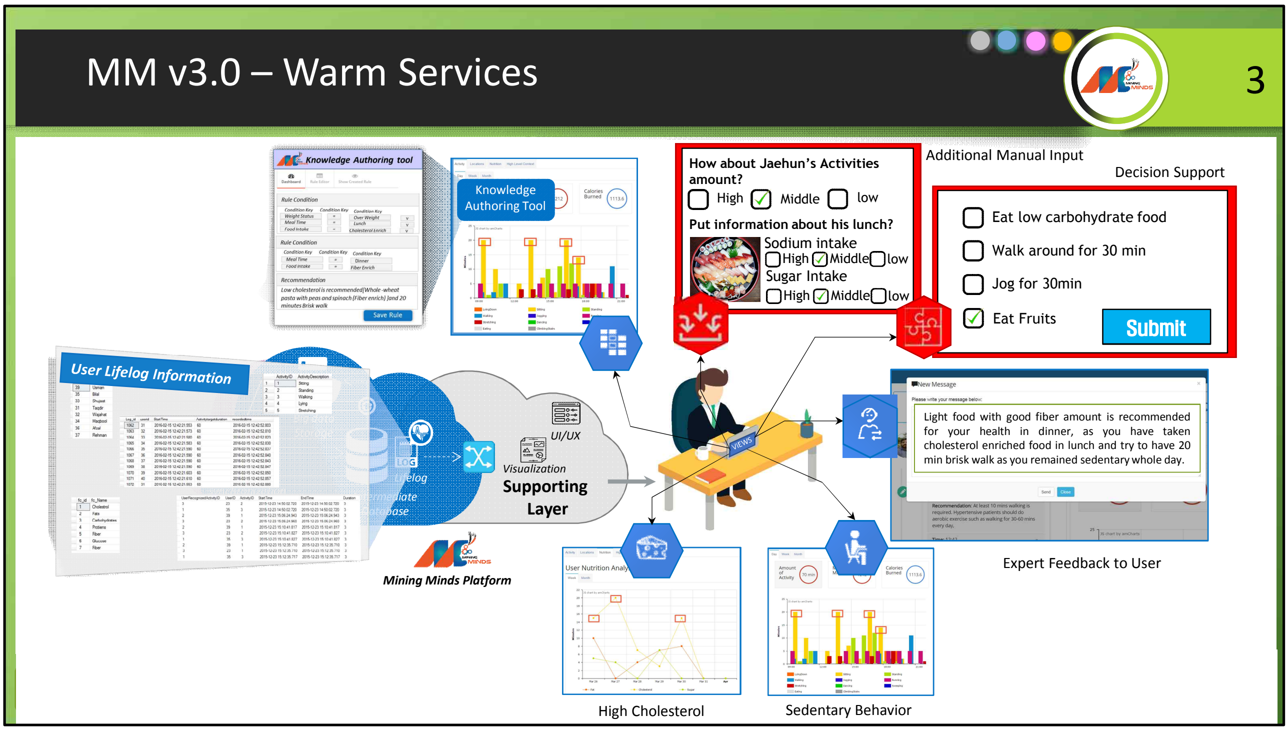
경희대학교
KYUNG HEE UNIVERSITY

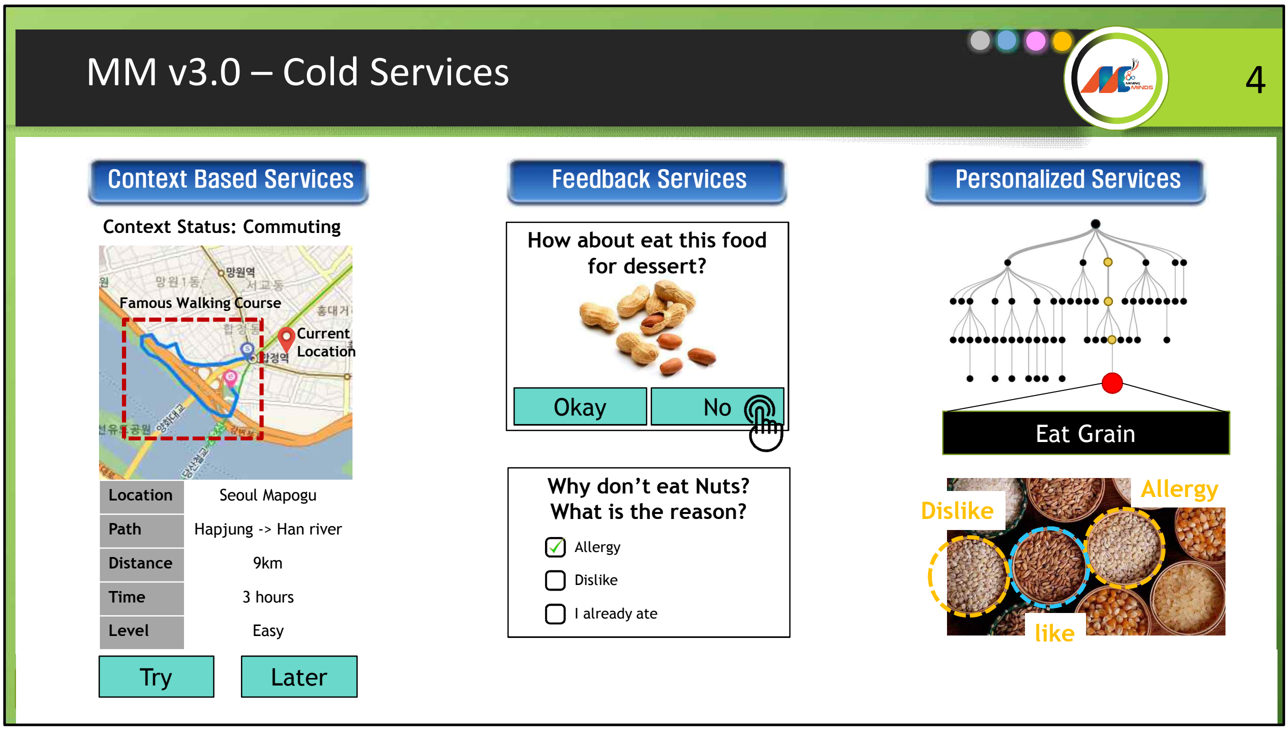
<u>Section 4</u>

Mining Minds Version 3.0 Service Scenario Requirement Analysis


MINING MINDS
MM v3.0 Service Scenario
MM V3.0 Service Scenario
Requirement Analysis
경희대학교
KYUNG HEE UNIVERSITY
KITECH
POSTECH
포항공과대학교
UNIVERSITY of TASMANIA
녹십자헬스케어
TAPACROSS
U2 system
User & Usability system
KOCSA
한국컨설팅서비스협회

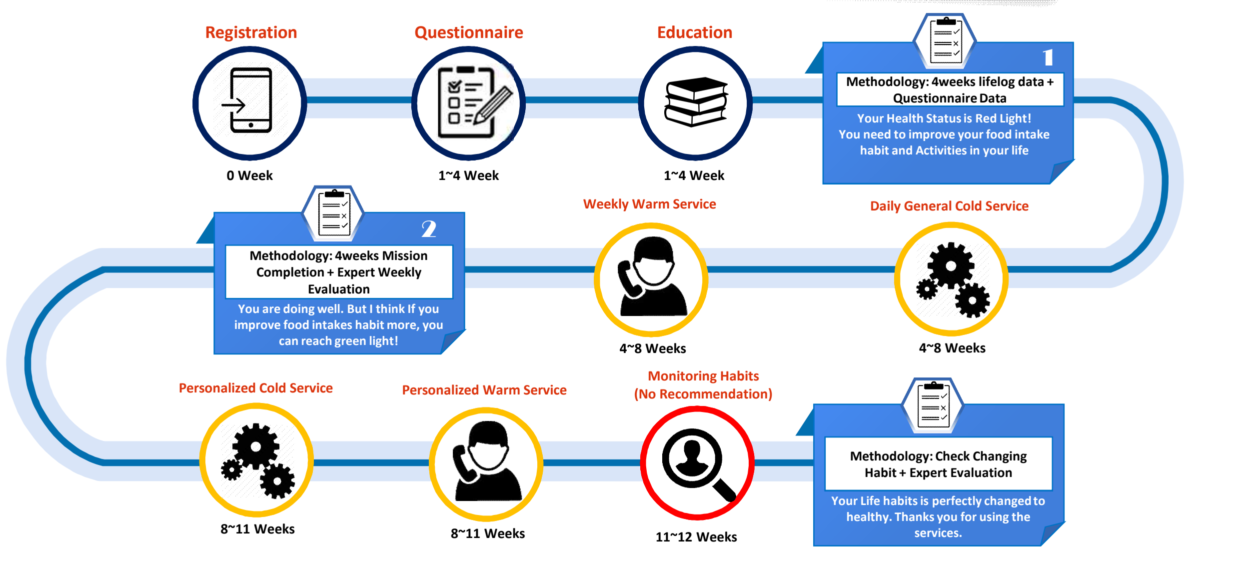

MM v3.0 Proposed Service Scenario
2
Input Devices
Wearable
External
Input Sensors
SURVEY
Questionnaire
Inertial Sensor
Audio
Images
GPS, Glucose, Manual Input
Information
Physical Activities
Emotions
Location
Food Intakes (Calories)
High-Level Contexts
Services Type
Warm Services
Cold Services
Detail Services
Monitoring & Analytics
Decision Support
FEED BACK
Feedback Service
Context based Service
Personalized Recommend


MM v3.0 – Warm Services
3
Knowledge Authoring tool
Knowledge Authoring Tool
How about Jaehun's Activities amount?
High
Middle
low
Put information about his lunch?
Sodium intake
High Middle low
Sugar Intake
High Middle low
Additional Manual Input
Decision Support
Eat low carbohydrate food
Walk around for 30 min
Jog for 30min
Eat Fruits
Submit
User Lifelog Information
UI/UX
Visualization
Supporting Layer
Mining Minds Platform
Light food with good fiber amount is recommended for your health in dinner, as you have taken cholesterol enriched food in lunch and try to have 20 min brisk walk as you remained sedentary whole day.
Expert Feedback to User
High Cholesterol
Sedentary Behavior


MM v3.0 – Cold Services
4
Context Based Services
Context Status: Commuting
Famous Walking Course
Current Location
Location
Seoul Mapogu
Path
Hapjung -> Han river
Distance
9km
Time
3 hours
Level
Easy
Try
Later
Feedback Services
How about eat this food for dessert?
Okay
No
Why don't eat Nuts? What is the reason?
Allergy
Dislike
I already ate
Personalized Services
Eat Grain
Dislike
Allergy
like

MM v3.0 – Service Process

5

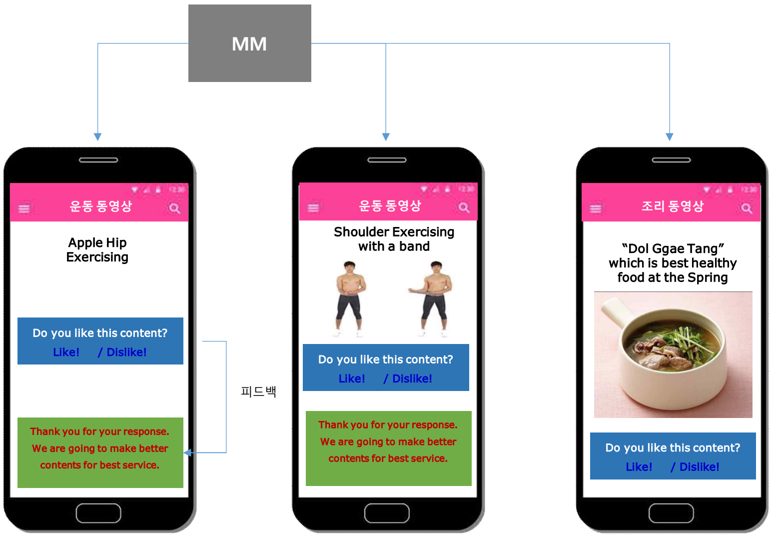


Service Scenario AS-IS TO-BE

6

Service Type	MM v2.5	MM v3.0
Duration	Daily	12 weeks
Physical Activity	Walking for 20 min	Walking 20min in the Young Tong Park
Nutrition	Eat low Sodium Food	Eat Bibimbab (500kcal)
Feedback	Feedback for service Satisfaction	Feedback user prefer and suspected Disease
Trendy Service	SNS based services	Same
Warm Services	Check user life-log with statistical factor	Check Auto generated recommendation list (Decision Support)

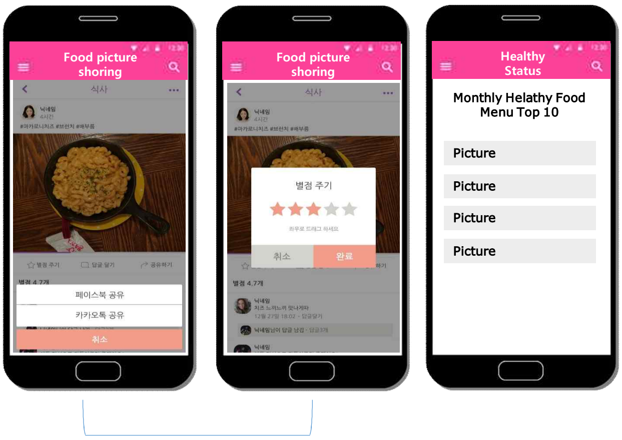
Exercising Movies List/ Food Recipe Movies List (Number of contents is 1 to 15)
Provision Service contents based on Expert Recommendation with a priority



Contents satisfaction distribution with gender, ages, living location, disease level from feedback result

Priority based provision of contents and utilizing for creation of new service contents

7



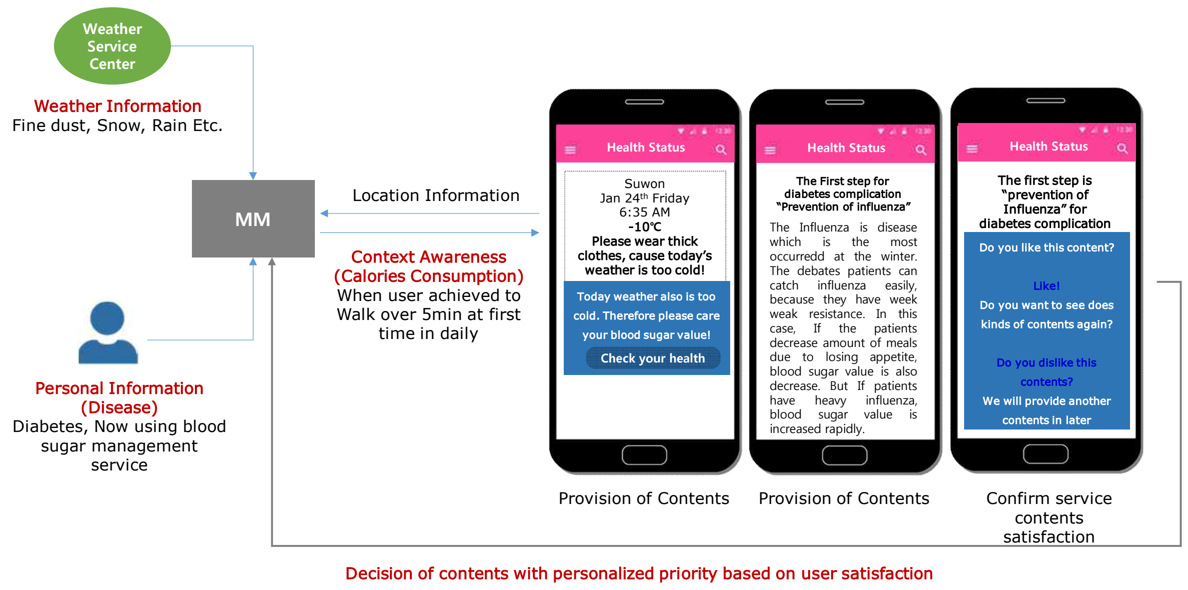
Food intakes photo sharing with user to user

Each user put the score about the food photo

→ Confirm the best food photo from expert and then make new contents

Ex: Healthy food Top 10 with user evaluation in last month

8



9

GCHealth Care Service Scenario

Physical Activity & Nutrition

MMV3.0

경희대학교 KYUNG HEE UNIVERSITY · KITECH 한국생산기술연구원 · POSTECH 포항공과대학교 · UNIVERSITY of TASMANIA · 녹십자헬스케어 · TAPACROSS · U2 system User & Usability system · KOCSA 한국컨설팅서비스협회 Korea Consulting Service Association

Nutritional Management Scenario

11

Registration Stage – Register and Data Gathering

<Registration>

- User ID (account code)
- Nickname
- Password
- Linked SNS (e.g. the FACEBOOK ID, S-HEALTH ID)

<Contact>

- Contact Number(Mobile)
- Address
- e-mail

<Client Information>

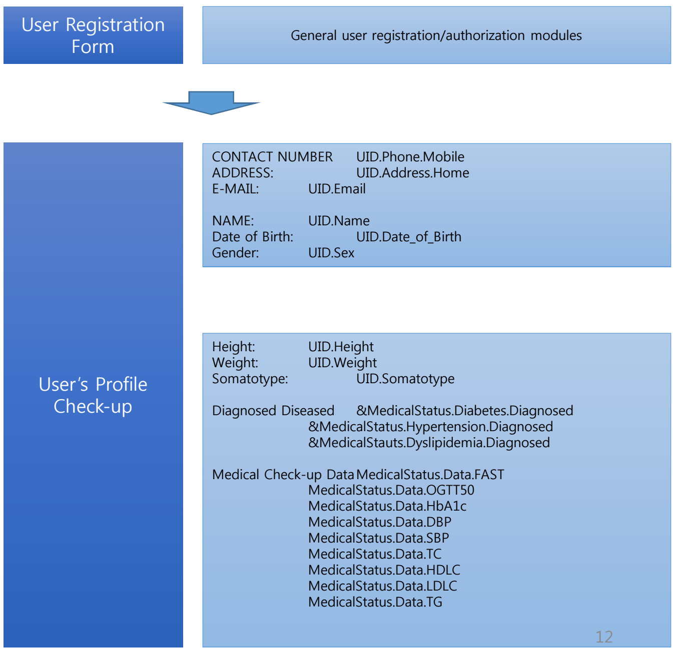
- Client Name
- Date of Birth
- Gender

<Somatology>

- Height
- Weight
- Somatotype of Body

<Medical History on Lifestyle Diseases>

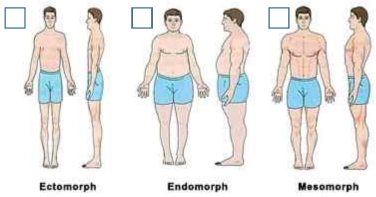
- Diabetes (if checked, additional input form presented)
- Hypertension (additional form)
- Dyslipidemia (additional form)



12

<Somatotype Assessment>

Question: What is your body type? Choose most similar type in the pictures.
Choices:



Data:

UID	Field Name	Date & Time of Assessment	Toolset	Data	Unit	Certified by
####	UID.Height	yyyyMMddHHmmss	{UserInput}	Integer	cm	USER
####	UID.Weight	yyyyMMddHHmmss	{UserInput, DeviceID}	###.#	kg	USER
####	UID.Somatotype	yyyyMMddHHmmss	{UserChoice, ST_msr01, ST_msr02, ...}	{Ectomorph, Endomorph, Mesomorph; NULL}	N/A	USER
####	UID.BMI	yyyyMMddHHmmss	{Rule.Index.BMI}	##.#	kg/m^2	SYSTEM

Rule.Index.BMI
input: UID.Height.Current_date, UID.Weight.Current_date
output: UID.BMI = UID.Weight.Current_date/(UID.Height.Current_date*UID.Height.Current_date)

13

<Medical History Assessment>

Question: Have you ever been diagnosed with any of following diseases or conditions?

- ☐ Diabetes or prediabetes
- ☐ Hypertension or Prehypertension
- ☐ Dyslipidemia

If the boxes are checked, present additional questions.

- ✓ Diabetes or prediabetes
  - Input your recent FAST test result : ( ) mg/dL ☐ Don't Know
  - OGTT 50g test result : ( ) mg/dL ☐ Don't Know
  - HbA1c : ( ) % or ( ) mmol/mol ☐ Don't Know
- ✓ Hypertension or Prehypertension
  - Recent measure of Blood Pressure : Systolic ( )mmHg, Diastolic ( )mmHg
- ✓ Dyslipidemia
  - Recent measure of Total Cholesterol : ( ) mg/dL
  - Recent measure of HDL Cholesterol : ( ) mg/dL
  - Recent measure of LDL Cholesterol : ( ) mg/dL
  - Recent measure of Triglyceride : ( ) mg/dL

Data:

UID	Field Name	Date & Time of Assessment	Toolset	Data	Unit	Certified by
####	MedicalStatus.Diabetes.Diagnosed	yyyyMMddHHmmss	{UserChoice, ExpertInput}	{Yes; NULL}	N/A	USER
####	MedicalStatus.Hypertension.Diagnosed	yyyyMMddHHmmss	{UserChoice, ExpertInput}	{Yes; NULL}	N/A	USER
####	MedicalStauts.Dyslipidemia.Diagnosed	yyyyMMddHHmmss	{UserChoice, ExpertInput}	{Yes; NULL}	N/A	USER
####	MedicalStatus.Data.FAST	yyyyMMddHHmmss	{UserInput, DeviceID}	Integer	mg/dL	USER
####	MedicalStatus.Data.OGTT50	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER
####	MedicalStatus.Data.HbA1c	yyyyMMddHHmmss	{UserInput}	##.# ##.# Integer	% mmol/mol mg/dL	USER
####	MedicalStatus.Data.DBP	yyyyMMddHHmmss	{UserInput, DeviceID}	Integer	mmHg	USER
####	MedicalStatus.Data.SBP	yyyyMMddHHmmss	{UserInput, DeviceID}	Integer	mmHg	USER
####	MedicalStatus.Data.TC	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER
####	MedicalStatus.Data.HDLC	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER
####	MedicalStatus.Data.LDLC	yyyyMMddHHmmss	{UserInput}	###.#	mg/dL	USER
####	MedicalStatus.Data.TG	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER

14

Registration Stage – Register and Data Gathering

<Questionnaire: Dietary Habits>

- Eating Regularity
- Food Group Balance
- Whole Grain Uptakes
- Protein Uptakes
- Vegetable Uptakes
- Fruits Uptakes
- Dairy Uptakes
- Red Meat Uptakes
- Sodium Uptakes
- Added Sugar Uptakes

Dietary Habits Questionnaire

Selected dietary habit assessment tools
(ex, GCH's Questionnaire)
1. During the last month, how regularly do you eat?
2. Do you enjoy a wide variety of nutritious foods from the five groups every day?
3. When you eat cooked-rice, do you usually eat cooked-rice with multi-grains?
4. Do you usually take protein foods* in every servings?
5. Do you usually eat vegetables in every servings?
6. Do you usually eat fruits every day?
7. Do you usually dairy foods every day?
8. Do you usually eat red meat or high fat meat more than twice a week?
9. Do you usually take salty side dishes everyday?
10. Do you usually eat snacks or beverages with added sugars every day?

Generate Initial Values for User Profile

<Dietary Habits Data (in User's Profile DB)>

Eating Regularity:	DietaryHabits.EP.Regularity (1pt ~ 5pt)
Food Group Balance:	DietaryHabits.EP.FGBalance (1pt ~ 5pt)
Whole Grain Uptakes:	DietaryHabits.NT.WGUtk (1pt ~ 5pt)
Protein Uptakes:	DietaryHabits.NT.PtUtk (1pt ~ 5pt)
Vegetable Uptakes:	DietaryHabits.NT.VgUtk (1pt ~ 5pt)
Fruits Uptakes:	DietaryHabits.NT.FtUtk (1pt ~ 5pt)
Dairy Uptakes:	DietaryHabits.NT.DyUtk (1pt ~ 5pt)
Red Meat Uptakes:	DietaryHabits.NT.RMUtk (1pt ~ 5pt)
Sodium Uptakes:	DietaryHabits.NT.NaUtk (1pt ~ 5pt)
Added Sugar Uptakes:	DietaryHabits.NT.SgUtk (1pt ~ 5pt)

15

<Lifestyle Questionnaire – Dietary Habits>

CALL: GCH-DietPic-QN-DH-v001 questionnaire modules

Q1. Eat Regularity (DietaryHabits.EP.Regularity)

During the last month, how regularly (eat three times a day) do you eat?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q2. Food Group Balance (DietaryHabits.EP.FGBalance)

Do you enjoy a wide variety of nutritious foods from the five groups every day? (the five groups: fruits, vegetables, grains, protein foods, and dairy)

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q3. Whole Grain Uptakes (DietaryHabits.NT.WGUtk)

When you eat cooked-rice, do you usually eat cooked-rice with multi-grains?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q4. Protein Uptakes (DietaryHabits.NT.PtUtk)

Do you usually take protein foods* in every servings? * Lean Meat (except red meat and high fat meat), Poultry, Fish and seafood, Eggs, Nuts and Seeds, Legumes/Beans

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q5. Vegetables Uptakes (DietaryHabits.NT.VgUtk)

Do you usually eat vegetables in every servings?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q6. Fruit Uptakes (DietaryHabits.NT.FtUtk)

Do you usually eat fruits everyday?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q7. Dairy Uptakes (DietaryHabits.NT.DyUtk)

Do you usually dairy foods everyday?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q8. Red Meat Uptakes (DietaryHabits.NT.RMUtk)

Do you usually eat red meat or high fat meat more than twice a week?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q9. Sodium Uptakes (DietaryHabits.NT.NaUtk)

Do you usually take salty side dishes everyday?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q10. Added Sugar Uptakes (DietaryHabits.NT.SgUtk)

Do you usually eat snacks or beverages with added sugars every day?

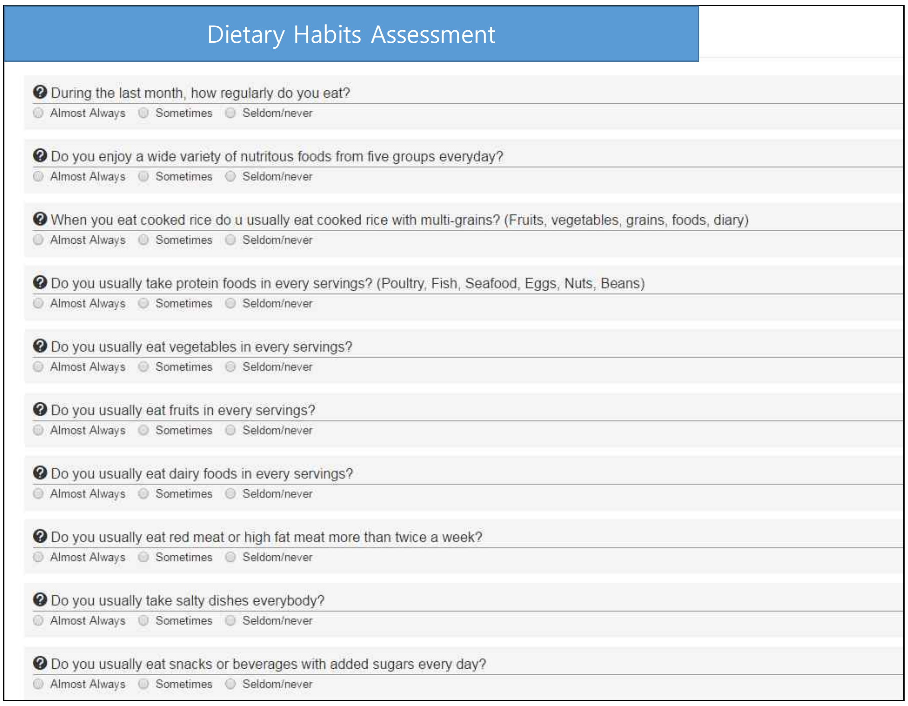
☐ Almost Always ☐ Sometimes ☐ Seldom/Never

16

GCH-DietPic-QN-DH-v001 questionnaire modules

No	Classification			Variable Name	Measure	Score Direction	Question Reference	Answer Choice and Measure Example (Ans to Ref. Question)
1	Dietary Habits Screening	Eating Pattern	Eating Regularity	DQ_EP.Regularity	5-Scale Points	Positive LOW	During the last month, how regularly do you eat? (Eat three times a day)	Almost Always (5) Sometimes (3) Seldom/Never (1)
2	Dietary Habits Screening	Eating Pattern	Food Group Balance	DQ_EP.FGBalance	5-Scale Points	Positive LOW	Do you enjoy a wide variety of nutritious foods from the five groups every day? (the five groups: fruits, vegetables, grains, protein foods, and dairy)	Almost Always (5) Sometimes (3) Seldom/Never (1)
3	Dietary Habits Screening	Nutrient	Whole Grain Uptakes	DQ_NT.WGUtk	5-Scale Points	Positive LOW	When you eat cooked-rice, do you usually eat cooked-rice with multi-grains?	Almost Always (5) Sometimes (3) Seldom/Never (1)
4	Dietary Habits Screening	Nutrient	Protein Uptakes	DQ_NT.PtUtk	5-Scale Points	Positive LOW	Do you usually take protein foods* in every servings? * Lean Meat (except red meat and high fat meat), Poultry, Fish and seafood, Eggs, Nuts and Seeds, Legumens/Beans	Almost Always (5) Sometimes (3) Seldom/Never (1)
5	Dietary Habits Screening	Nutrient	Vegetables Uptakes	DQ_NT.VgUtk	5-Scale Points	Positive LOW	Do you usually eat vegetables in every servings?	Almost Always (5) Sometimes (3) Seldom/Never (1)
6	Dietary Habits Screening	Nutrient	Fruits Uptakes	DQ_NT.FtUtk	5-Scale Points	Positive LOW	Do you usually eat fruits every day?	Almost Always (5) Sometimes (3) Seldom/Never (1)
7	Dietary Habits Screening	Nutrient	Dairy Uptakes	DQ_NT.DyUtk	5-Scale Points	Positive LOW	Do you usually dairy foods every day?	Almost Always (5) Sometimes (3) Seldom/Never (1)
8	Dietary Habits Screening	Nutrient	Red Meat Uptakes	DQ_NT.RMUtk	5-Scale Points	Positive HIGH	Do you usually eat red meat or high fat meat more than twice a week?	Almost Always (1) Sometimes (3) Seldom/Never (5)
9	Dietary Habits Screening	Nutrient	Sodium Uptakes	DQ_NT.NaUtk	5-Scale Points	Positive HIGH	Do you usually take salty side dishes everyday?	Almost Always (1) Sometimes (3) Seldom/Never (5)
10	Dietary Habits Screening	Nutrient	Added Sugar Uptakes	DQ_NT.SgUtk	5-Scale Points	Positive HIGH	Do you usually eat snacks or beverages with added sugars every day?	Almost Always (1) Sometimes (3) Seldom/Never (5)

17


Dietary Habits Assessment

During the last month, how regularly do you eat?
Almost Always Sometimes Seldom/never

Do you enjoy a wide variety of nutritous foods from five groups everyday?
Almost Always Sometimes Seldom/never

When you eat cooked rice do u usually eat cooked rice with multi-grains? (Fruits, vegetables, grains, foods, diary)
Almost Always Sometimes Seldom/never

Do you usually take protein foods in every servings? (Poultry, Fish, Seafood, Eggs, Nuts, Beans)
Almost Always Sometimes Seldom/never

Do you usually eat vegetables in every servings?
Almost Always Sometimes Seldom/never

Do you usually eat fruits in every servings?
Almost Always Sometimes Seldom/never

Do you usually eat dairy foods in every servings?
Almost Always Sometimes Seldom/never

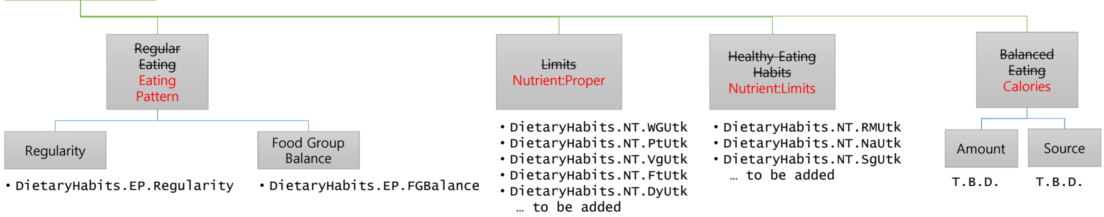
Do you usually eat red meat or high fat meat more than twice a week?
Almost Always Sometimes Seldom/never

Do you usually take salty dishes everybody?
Almost Always Sometimes Seldom/never

Do you usually eat snacks or beverages with added sugars every day?
Almost Always Sometimes Seldom/never

18

Dietary Habits Assessment | Dietary Questionnaire | the initial values



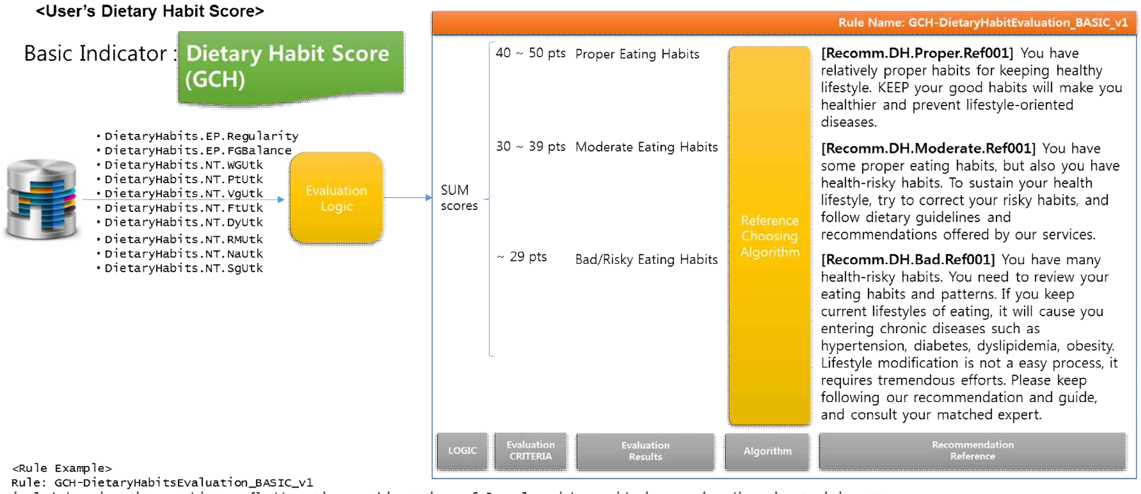
No	Classification			Variable Name	Measure	Score Direction	Question Refereneence	Answer Choice and Measure Example (Ans to Ref. Question)
1	Dietary Habits Screening	Eating Pattern	Eating Regularity	DQ_EP.Regularity	5-Scale Points	Positive LOW	During the last month, how regularly do you eat? (Eat three times a day)	Almost Always (5) Sometimes (3) Seldom/Never (1)
2	Dietary Habits Screening	Eating Pattern	Food Group Balance	DQ_EP.FGBalance	5-Scale Points	Positive LOW	Do you enjoy a wide variety of nutritious foods from the five groups every day? (the five groups: fruits, vegetables, grains, protein foods, and dairy)	Almost Always (5) Sometimes (3) Seldom/Never (1)
3	Dietary Habits Screening	Nutrient	Whole Grain Uptakes	DQ_NT.WGUtk	5-Scale Points	Positive LOW	When you eat cooked-rice, do you usually eat cooked-rice with multi-grains?	Almost Always (5) Sometimes (3) Seldom/Never (1)
4	Dietary Habits Screening	Nutrient	Protein Uptakes	DQ_NT.PtUtk	5-Scale Points	Positive LOW	Do you usually take protein foods* in every servings? * Lean Meat (except red meat and high fat meat), Poultry, Fish and seafood, Eggs, Nuts and Seeds, Legumens/Beans	Almost Always (5) Sometimes (3) Seldom/Never (1)
5	Dietary Habits Screening	Nutrient	Vegetables Uptakes	DQ_NT.VgUtk	5-Scale Points	Positive LOW	Do you usually eat vegetables in every servings?	Almost Always (5) Sometimes (3) Seldom/Never (1)
6	Dietary Habits Screening	Nutrient	Fruits Uptakes	DQ_NT.FtUtk	5-Scale Points	Positive LOW	Do you usually eat fruits every day?	Almost Always (5) Sometimes (3) Seldom/Never (1)
7	Dietary Habits Screening	Nutrient	Dairy Uptakes	DQ_NT.DyUtk	5-Scale Points	Positive LOW	Do you usually dairy foods every day?	Almost Always (5) Sometimes (3) Seldom/Never (1)
8	Dietary Habits Screening	Nutrient	Red Meat Uptakes	DQ_NT.RMUtk	5-Scale Points	Positive HIGH	Do you usually eat red meat or high fat meat more than twice a week?	Almost Always (1) Sometimes (3) Seldom/Never (5)
9	Dietary Habits Screening	Nutrient	Sodium Uptakes	DQ_NT.NaUtk	5-Scale Points	Positive HIGH	Do you usually take salty side dishes everyday?	Almost Always (1) Sometimes (3) Seldom/Never (5)
10	Dietary Habits Screening	Nutrient	Added Sugar Uptakes	DQ_NT.SgUtk	5-Scale Points	Positive HIGH	Do you usually eat snacks or beverages with added sugars every day?	Almost Always (1) Sometimes (3) Seldom/Never (5)

19

Registration Stage – Decision of User Type

<User's Dietary Habit Score>

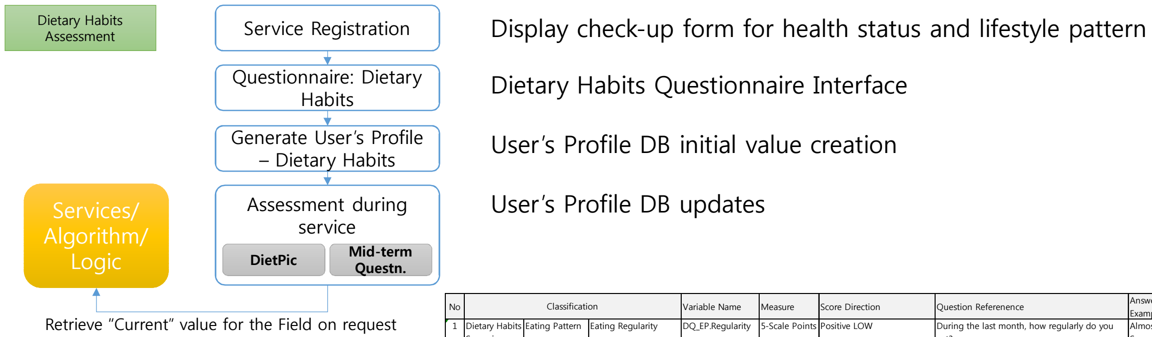
Basic Indicator : **Dietary Habit Score (GCH)**



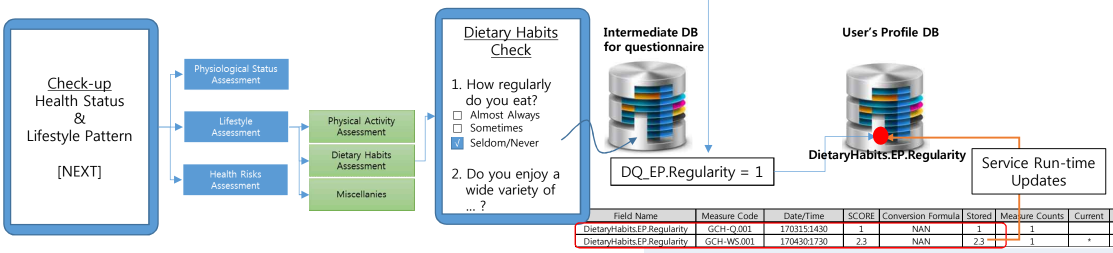

```
<Rule Example>
Rule: GCH-DietaryHabitsEvaluation_BASIC_v1
included Logic: DietaryHabitScore{} //Sum DietaryHabits Points of 5-scale, with considering scoring direction Positive LOW
Result evaluation: if DietaryHabitScore = < 29, DietaryHabitStatus = BAD; 30 ~ 39, DietaryHabitStatus = Moderate, 40+, DietaryHabitStatus = Proper
Choosing Recommendation Reference:
  if DietaryHabitStatus = Moderate, call "Recom.DH.Moderate.Ref001"; if DietaryHabitStatus = Proper, call "Recomm.DH.Proper.Ref001"; if DietaryHabitStatus = Bad, call "Recomm.DH.Bad.Ref001"
 * Reference Recommendation can varies during service time with analyzing service result database or expert's choice/preferences or company/service aims or specific user's preferences/
   context or user group's general responses. So Reference Choosing Algorithm is needed in future work. This could be done by tagging attributes to the "Reference Recommendations",
   which include (in example) user's "I Like it" counts, expert's "I Agree to this reference(Not change the reference)" or choice/use of other experts' generated recommendations rate or
   analysis result of user group's lifestyle modification rate rusults.
```

20

Service Stage – Update User's Data



No	Classification			Variable Name	Measure	Score Direction	Question Reference	Answer Choice and Measure Example (Ans to Ref. Question)
1	Dietary Habits Screening	Eating Pattern	Eating Regularity	DQ_EP.Regularity	5-Scale Points	Positive LOW	During the last month, how regularly do you eat? (Eat three times a day)	Almost Always (5) Sometimes (3) Seldom/Never (1)



Field Name	Measure Code	Date/Time	SCORE	Conversion Formula	Stored	Measure Counts	Current	Certified by
DietaryHabits.EP.Regularity	GCH-Q.001	170315:1430	1	NAN	1	1		System
DietaryHabits.EP.Regularity	GCH-WS.001	170430:1730	2.3	NAN	2.3	1	*	E13123

GCH-Q.001: GCH's User Registration Questionnaire - Nutrition)

GCH-WS.001: GCH's Warm Service – DietPic Expert evaluation on Eating Pattern

21

Physical Activity Management Scenario

22

<Lifestyle Questionnaire – PA & Exercise Habits> CALL: GCH-DietPic-QN-PA-v001 questionnaire modules

No	Classification			Variable Name	Measure	Score Direction	Question Reference	Answer Choice and Measure Example (Ans to Ref. Question)	Questionnaire routine/Comments
1	PA-Exercise Screening	Regular Exercise: General	Performing Exercise Regularly	DQ_RE.REPf	YES/NO	Positive HIGH	Do you currently perform regular, intended exercise?	1=Yes 0=No	If DQ_+K2:K11RE.REPf=0, skip #2~#5 questions (go to #6 question)
2	PA-Exercise Screening	Regular Exercise: Status	Exercise Frequency	DQ_RE.EF	YES/NO	Positive HIGH	Do you currently perform exercise more than 3 to 5 times per week regularly?	1=Yes 0=No	
3	PA-Exercise Screening	Regular Exercise: Status	Exercise Duration per unit session	DQ_RE.Dur_ssn	5-scale pts (conversion)	Positive HIGH	How long do you exericse at each session?	2=More than 1 hour 1=More than 30 minutes 0=Less than 20 minutes	
4	PA-Exercise Screening	Regular Exercise: Status	Exercise Intensity per each session in average	DQ_RE.Int_ssn	5-scale pts (conversion)	Positvie HIGH	How's the intensity level of exercise normaly at each session?	2=Vigorous 1=Moderate 0=Light	
5	PA-Exercise Screening	Regular Exercise: Status	Performing pre, post exercise or daily stretching	DQ_RE.StrPf	YES/NO	Positive HIGH	Do you perform stretching normaly before/after exercise or in daily life regularly?	1=Yes 0=No	End of Questionnaire for the user with DQ_RE.REPf=1
6	PA-Exercise Screening	Regular Exercise: General	Experience performing regular exercise	DQ_RE.REExp	YES/NO	Positive HIGH	Do you have an experience performing regular exercise?	1=Yes 0=No	Begin Questionnaire subroutine for the user with DQ_RE.REPf=0
7	PA-Exercise Screening	Regular Exercise: General	Existence of restrictions on performing exercise	DQ_RE.Rstr	YES/NO	Positive LOW	Are there any restrictions on starting exercise, even though you are willing to do?	1=Yes 0=No	If DQ_RE.Rstr=0, skip #7 question (go to #8 question)
8	PA-Exercise Screening	Regular Exercise: General	Reason of restrictions on performing exericse	DQ_RE.RstrRsn	N/A	NEUTRAL	If you have any restrictions on starting exercise, why?	3=Lack of time or space 2=Feel pains or sick 1=Do not want to exercise alone 0=Don't know how to	
9	PA-Exercise Screening	Regular Exercise: General	Objective of Exercise	DQ_RE.ObjEx	N/A	NEUTRAL	What is your objective of exercise?	2=for Diet 1=for Leisure 0=for Health	
10	PA-Exercise Screening	Regular Exercise: General	Willingness on performing exercise	DQ_RE.WnEx	YES/NO	Positive HIGH	Are you willing to start regular exercise?	1=Yes 0=No	End of Questionnaire for the user with DQ_RE.REPf=0
11	PA-Exercise Screening	PA Level: PA Related Lifestyle	Daily Walking Time	DQ_PA.WkTime	YES/NO	Positive HIGH	Do you usually walk more than 10 minutes on a daily basis?	1=Yes 0=No	
12	PA-Exercise Screening	PA Level: Amount	Lifetime Movement Pattern	DQ_PA.LMP	YES/NO	Positive HIGH	Do you prefer exercise or working by moving your body?	1=Yes 0=No	
.13	PA-Exercise Screening	PA Level: PA Related Lifestyle	Sedentary Lifestyle	DQ_PA.Sdty	YES/NO	Positive LOW	Do you spend more time standing or moving your body rather than sitting?	1=No 0=Yes	
14	PA-Exercise Screening	PA Level: PA Related Lifestyle	Prefer Walking	DQ_PA.PW	YES/NO	Positive HIGH	Do you prefer walking short distance rather than ride?	1=Yes 0=No	

23

Registration Stage – Register and Data Gathering

<Questionnaire: Physical Activity Habits>

- Performing Exercise Regularly
- Exercise Frequency
- Exercise Duration per unit session
- Exercise Intensity per each session in average
- Performing pre, post exercise or daily stretching
- Experience performing regular exercise
- Existence of restrictions on performing exercise
- Reason of restrictions on performing exercise
- Objective of Exercise
- Willingness on performing exercise
- Daily Walking Time
- Lifetime Movement Pattern - Prefer Active Lifestyle
- Sedentary Lifestyle
- Prefer Walking

Physical Activity and Exercise Habits Questionnaire

Selected physical activity and exercise habit assessment tools
(ex, GCH's Questionnaire)

1. Do you currently perform regular, intended exercise?
2. Do you currently perform exercise more than 3 to 5 times per week regularly?
3. How long do you exercise at each session?
4. How's the intensity level of exercise normally at each session?
5. Do you perform stretching normally before/after exercise or in daily life regularly?
6. Do you have an experience performing regular exercise?
7. Are there any restrictions on starting exercise, even though you are willing to do?
8. If you have any restrictions on starting exercise, why?
9. What is your objective of exercise?
10. Are you willing to start regular exercise?
11. Do you usually walk more than 10 minutes on a daily basis?
12. Do you prefer exercise or working by moving your body?
13. Do you spend more time standing or moving your body rather than sitting?
14. Do you prefer walking short distance rather than ride?

Generate Initial Values for User Profile

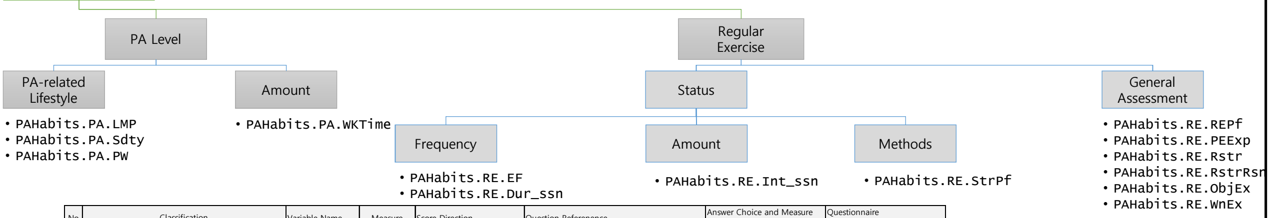
<PA & Exercise Habits Data (in User's Profile DB)>
Exercise Regularity: PAHabits.RE.REPf (0 or 1)
Exercise Frequency: PAHabits.RE.EF (0 or 1)
Exercise Duration: PAHabits.RE.Dur_ssn (5 scale pts)
Exercise Intensity: PAHabits.RE.Int_ssn (5 scale pts)
Stretching: PAHabits.RE.StrPf (0 or 1)
Experience on regular exercise: PAHabits.RE.REExp (0 or 1)
Restrictions on Exercise: PAHabits.RE.Rstr (0 or 1)
Source of Restrictions: PAHabits.RE.RstrRsn
Objective of Exercise: PAHabits.RE.ObjEx
Willingness exercise: PAHabits.RE.WnEx (0 or 1)
Daily Walking Time: PAHabits.PA.WkTime (0 or 1)
Prefer Active Lifestyle: PAHabits.PA.LMP (0 or 1)
Sedentary Lifestyle: PAHabits.PA.Sdty (0 or 1)
Prefer Walking: PAHabits.PA.PW (0 or 1)

24

Physical Activity Assessment

Exercise Questionnaire

the initial values



No	Classification			Variable Name	Measure	Score Direction	Question Reference	Answer Choice and Measure Example (Ans to Ref. Question)	Questionnaire routine/Comments
1	PA-Exercise Screening	Regular Exercise: General	Performing Exercise Regularly	DQ_RE.REPf	YES/NO	Positive HIGH	Do you currently perform regular, intended exercise?	1=Yes 0=No	If DQ_+K2:K11RE.REPf=0, skip #2~#5 questions (go to #6 question)
2	PA-Exercise Screening	Regular Exercise: Status	Exercise Frequency	DQ_RE.EF	YES/NO	Positive HIGH	Do you currently perform exercise more than 3 to 5 times per week regularly?	1=Yes 0=No	
3	PA-Exercise Screening	Regular Exercise: Status	Exercise Duration per unit session	DQ_RE.Dur_ssn	5-scale pts (conversion)	Positive HIGH	How long do you exericse at each session?	2=More than 1 hour 1=More than 30 minutes 0=Less than 20 minutes	
4	PA-Exercise Screening	Regular Exercise: Status	Exercise Intensity per each session in average	DQ_RE.Int_ssn	5-scale pts (conversion)	Positvie HIGH	How's the intensity level of exercise normaly at each session?	2=Vigorous 1=Moderate 0=Light	
5	PA-Exercise Screening	Regular Exercise: Status	Performing pre, post exercise or daily stretching	DQ_RE.StrPf	YES/NO	Positive HIGH	Do you perform stretching normaly before/after exercise or in daily life regularly?	1=Yes 0=No	End of Questionnaire for the user with DQ_RE.REPf=1
6	PA-Exercise Screening	Regular Exercise:	Experience performing regular exercise	DQ_RE.REExp	YES/NO	Positive HIGH	Do you have an experience performing regular exercise?	1=Yes 0=No	Begin Questionnaire subroutine for the user with
7	PA-Exercise Screening	Regular Exercise:	Existence of restrictions on	DQ_RE.Rstr	YES/NO	Positive LOW	Are there any restrictions on starting exercise, even though you are willing to do?	1=Yes 0=No	If DQ_RE.Rstr=0, skip #7 question (go to #8 question)
8	PA-Exercise Screening	Regular Exercise: General	Reason of restrictions on performing exericse	DQ_RE.RstrRsn	N/A	NEUTRAL	If you have any restrictions on starting exercise, why?	3=Lack of time or space 2=Feel pains or sick 1=Do not want to exercise alone	
9	PA-Exercise Screening	Regular Exercise: General	Objective of Exercise	DQ_RE.ObjEx	N/A	NEUTRAL	What is your objective of exercise?	2=for Diet 1=for Leisure 0=for Health	
10	PA-Exercise Screening	Regular Exercise:	Willingness on performing exercise	DQ_RE.WnEx	YES/NO	Positive HIGH	Are you willing to start regular exercise?	1=Yes 0=No	End of Questionnaire for the user with DQ_RE.REPf=0
11	PA-Exercise Screening	PA Level: PA Related	Daily Walking Time	DQ_PA.WkTime	YES/NO	Positive HIGH	Do you usually walk more than 10 minutes on a daily basis?	1=Yes 0=No	
12	PA-Exercise Screening	PA Level: Amount	Lifetime Movement Pattern	DQ_PA.LMP	YES/NO	Positive HIGH	Do you prefer exercise or working by moving your body?	1=Yes 0=No	
.13	PA-Exercise Screening	PA Level: PA Related	Sedentary Lifestyle	DQ_PA.Sdty	YES/NO	Positive LOW	Do you spend more time standing or moving your body rather than sitting?	1=No 0=Yes	
14	PA-Exercise Screening	PA Level: PA Related	Prefer Walking	DQ_PA.PW	YES/NO	Positive HIGH	Do you prefer walking short distance rather than ride?	1=Yes 0=No	

25

Registration Stage – Register and Data Gathering

<Diabetes Risk Assessment>

if MedicalStatus.Diabetes.Diagnosed == 0,
&& MedicalStatus.Data.BGTestResult = NULL
call DiabetesRiskAssessmentTool()

(results)
sum of risk points
5 or less: UID.Diabetes.Risk = Low
6 – 11: UID.Diabetes.Risk = Moderate
12 or more: UID.Diabetes.Risk = High

Diabetes Risk Assessment Tool

Diabetes Risk Assessment Tool
(ex, GCH's Questionnaire)
<Data from User Profile Database>
1. Age
2. Gender
3. Ethnicity (Default)
4. High Blood Pressure
5. Smoking
6. Eat Vegetables or Fruits
7. PA at least 2.5 hours per week

<Additional Questionnaire>
1. Family History
2. High Blood Glucose Level
3. Waist Measurement

Data for Diabetes Assessment

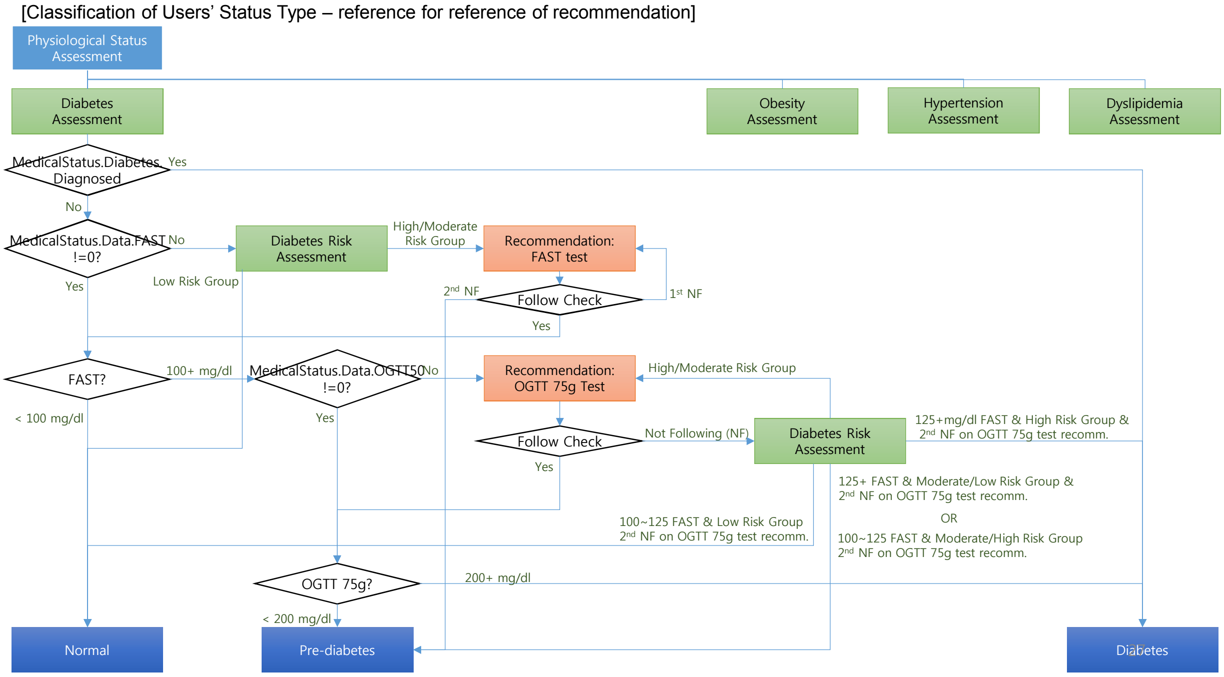
<Data in User's Profile DB)>

Age:	UID.Data_of_Birth
Gender:	UID.Sex
Ethnicity:	Asian(Default)
High BP:	MedicalStatus.Hypertension.Diagnosed or MedicalStatus.Data.DBP/SBP
Smoking:	T.B.D. (*BQ_LH.Smoking*)
Eat Veg or Fruits	DietaryHabits.NT.FtUtk/VgUtk
PA 2.5+ hr/week	*PAHabits.PA.PATimeperWk*
Family History	T.B.D. (*BQ_MI.FH*)
High BG Level	*MedicalStatus.Data.BGTestResult or BQ_MI.HBG*
Waist Measurement	*UID.WaistCircumference*

26

Registration Stage – User Assessment

[Classification of Users' Status Type – reference for reference of recommendation]



Diabetes Assessment

Reference: The Australian Type 2 Diabetes

1. Your Age Group
   - ☐ Under 35 years ;0 points
   - ☐ 35 – 44 years ;2 points
   - ☐ 45 – 54 years ;4 points
   - ☐ 55 – 64 years ;6 points
   - ☐ 65 years or over ;8 points
2. Your Gender
   - ☐ Female ;0 points
   - ☐ Male ;3 points
3. Your ethnicity
   - ☐ Asia, Middle East, North Africa, Southern Europe ;2 points
   - ☐ Other ;0 points
4. Have either of your parents, or any of your brothers or sisters been diagnosed with diabetes (type 1 or type 2)
   - ☐ No ;0 points
   - ☐ Yes ;3 points
5. Have you ever been found to have high blood glucose (sugar) (for example, in a health examination, during an illness, during pregnancy)?
   - ☐ No ;0 points
   - ☐ Yes ;6 points
6. Are you currently taking medication for high blood pressure?
   - ☐ No ;0 points
   - ☐ Yes ;2 points
7. Do you currently smoke cigarettes or any other tobacco products on a daily basis?
   - ☐ No ;0 points
   - ☐ Yes ;2 points
8. How often do you eat vegetables or fruit?
   - ☐ No ;0 points
   - ☐ Yes ;1 points
9. On average, would you say you do at least 2.5 hours of physical activity per week (for example, 30 minutes a day on 5 or more days a week)?
   - ☐ No ;0 points
   - ☐ Yes ;1 points
10. Your waist measurement taken below the ribs (usually at the level of the navel, and while standing)
    - ☐ Waist Measurement (cm) __________

Add up points: ______ points
- ☐ 5 or less: Low Risk
- ☐ 6 – 11: Moderate Risk
- ☐ 12 or more: High Risk

MEN
- ☐ Less than 90cm;0 points
- ☐ 90 – 100cm ;4 points
- ☐ More than 100cm ;7 points

WOMEN
- ☐ Less than 80cm;0 points
- ☐ 80 – 90cm ;4 points
- ☐ More than 90cm ;7 points

28

Registration Stage – Register and Data Gathering

<Hypertension Risk Assessment>

```
if      MedicalStatus.Hypertension.Diagnosed == 0,
         && MedicalStatus.Data.DBP/SBP = NULL
call    HypertensionRiskAssessmentTool()

(results)
sum of risk points
## ~ ##:     UID.Hypertension.Risk = Low
## ~ ##:     UID.Hypertension.Risk = Moderate
## ~ ##:     UID.Hypertension.Risk = High
```

Diabetes Risk Assessment Tool

Hypertension Risk Assessment Tool
(ex, GCH's Questionnaire)
<Data from User Profile Database>

<Additional Questionnaire>

Data for Diabetes Assessment

<Data in User's Profile DB)>

29

Registration Stage – Register and Data Gathering

<Dyslipidemia Risk Assessment>

```
if      MedicalStatus.Dyslipidemia.Diagnosed == 0,
         && MedicalStatus.Data.TC/HDLC/LDLC/TC = NULL
call    DyslipidemiaRiskAssessmentTool()

(results)
sum of risk points
## ~ ##:     UID.Dyslipidemia.Risk = Low
## ~ ##:     UID.Dyslipidemia.Risk = Moderate
## ~ ##:     UID.Dyslipidemia.Risk = High
```

Diabetes Risk Assessment Tool

Dyslipidemia Risk Assessment Tool
(ex, GCH's Questionnaire)
<Data from User Profile Database>

<Additional Questionnaire>

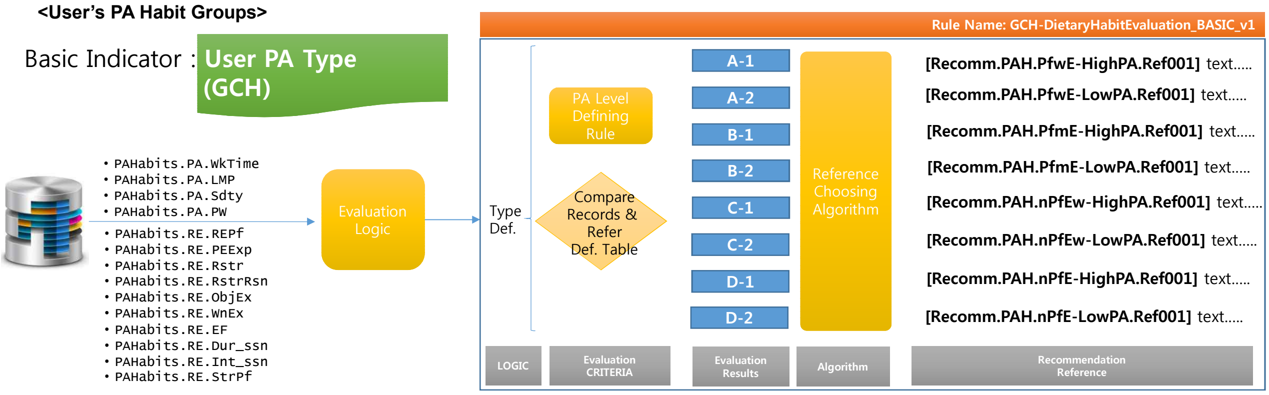
Data for Diabetes Assessment

<Data in User's Profile DB)>

30

Registration Stage – Decision of User Type

<User's PA Habit Groups>




```
<Rule Example>
Rule: GCH-PAHabitsEvaluation_BASIC_v1
included Logic: UserPAType{} // X(A:D)-Y(1:2)
                Determine Y value (1 or 2):  if PALevel=HIGH, Y=1; PALevel=LOW, Y=0, where PALevel is defined by a PALevel defining rule.
                            in case of initial value (based on GCH questionnaire), if PAHabits.PA.WkTime + PAHabits.PA.LMP + ~PAHabits.PA.Sdty +
PAHabits.PA.PW >= 3, PALevel=HIGH; else LOW
                            during the service time, analysis of step counts will replace PALevel value.
                Determine X value (A, B, C, D):  if ((DQ_RE.REPf==1) AND ((PAHabits.PE.EF && PAHabits.RE.StrPf) == 1) AND PAHabits.RE.Int_ssn !=0), X=A
                                                 if ((DQ_RE.REPf==1) AND ((PAHabits.RE.EF && PAHabits.RE.StrPf) != 1) OR
(PAHabits.RE.Dur_ssn*PAHabits.RE.Int_ssn==0)), X=B
                                                 if ((DQ_RE.REPf!=1) AND (DQ_RE.WnEx = 1), X=C
                                                 if ((DQ_RE.REPf!=1) AND (DQ_RE.WnEx = 0), X=D
```

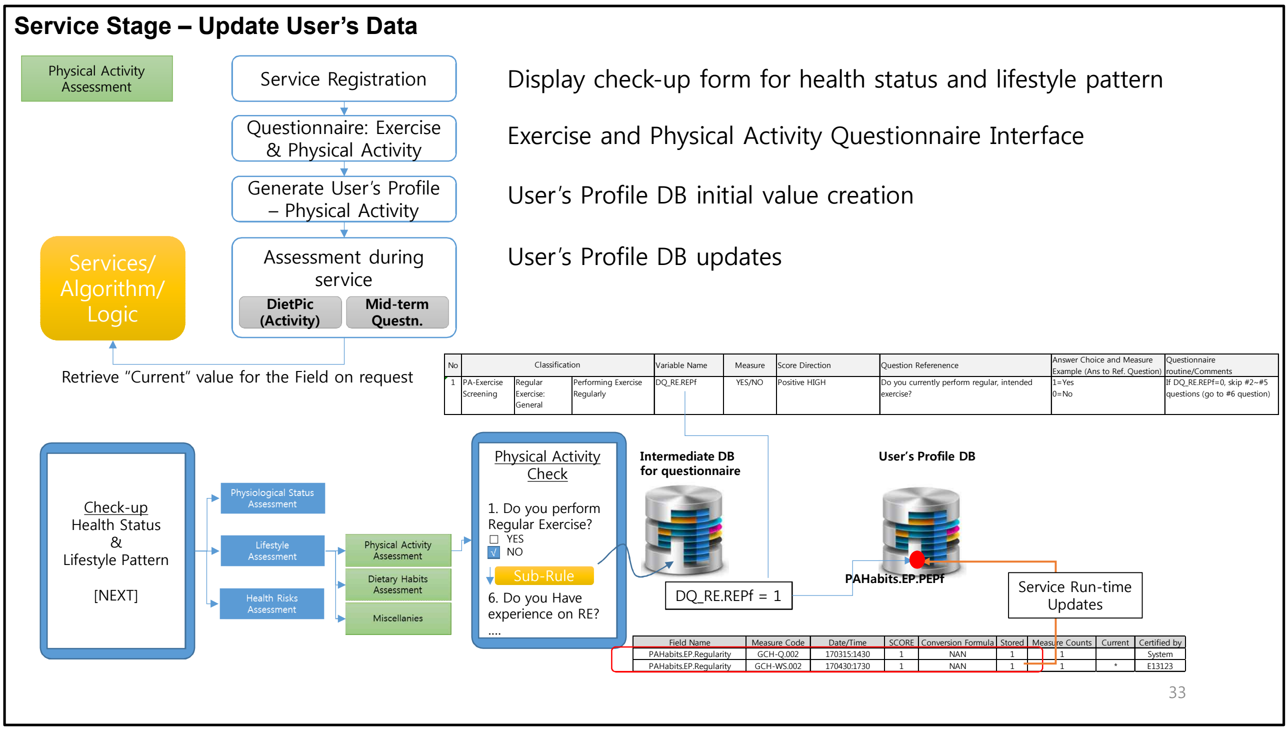
31

Registration Stage – Decision of User Type

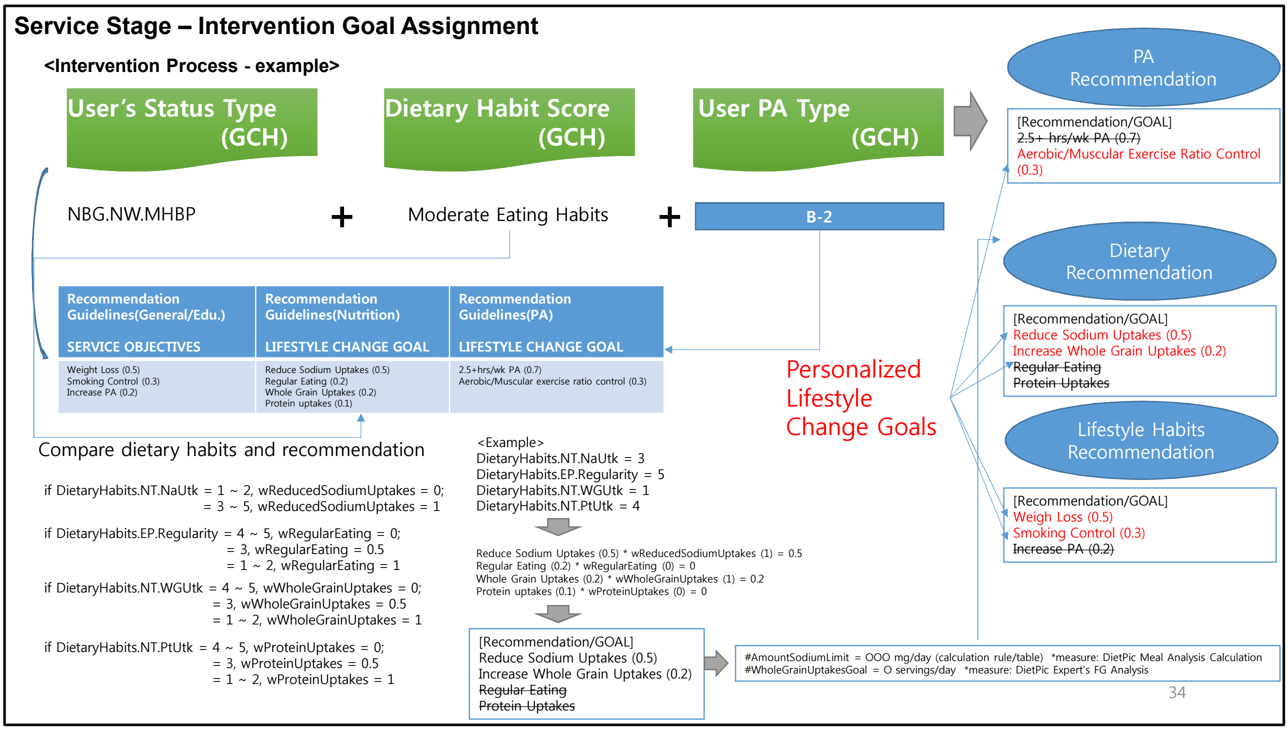
<Classification of User's Status Type>

Diabetes	Obesity	Hypertension	Dyslipidemia	Group Code	Recommendation Guidelines(General/Edu.)	Recommendation Guidelines(Nutrition)	Recommendation Guidelines(PA)
Normal	Low Weight	Normal	Group 1 ~ 5	NBG.LW.NBP.DL0 ~ NBG.LW.NBP.DL4	Carbohydrate uptake control Muscular exercise recomm.	Proper Calories, Regular Eating, 3-times Eating, Protein uptakes, Snack Control, Dairy Intakes	Aerobic/Muscular exercise ratio control
		Pre-Hypertension	Group 1 ~ 5	NBG.LW.MHBP.DL0 ~ NBG.LW.MHBP.DL4			
		Hypertension	Group 1 ~ 5	NBG.LW.HBP.DL0 ~ NBG.LW.HBP.DL4			
	Normal Weight	Normal	Group 1 ~ 5	NBG.NW.NBP			
		Pre-Hypertension	Group 1 ~ 5	NBG.NW.MHBP			
		Hypertension	Group 1 ~ 5	NBG.NW.HBP			
	Over weight/ Obesity	Normal	Group 1 ~ 5	NBG.OW.NBP			
		Pre-Hypertension	Group 1 ~ 5	NBG.OW.MHBP			
		Hypertension	Group 1 ~ 5	NBG.OW.HBP			
Pre-Diabetes	Low Weight	Normal	Group 1 ~ 5	MHBG.LW.NBP			
		Pre-Hypertension	Group 1 ~ 5	MHBG.LW.MHBP			
		Hypertension	Group 1 ~ 5	MHBG.LW.HBP			
	Normal Weight	Normal	Group 1 ~ 5	MHBG.NW.NBP			
		Pre-Hypertension	Group 1 ~ 5	MHBG.NW.MHBP			
		Hypertension	Group 1 ~ 5	MHBG.NW.HBP			
	Over weight/ Obesity	Normal	Group 1 ~ 5	MHBG.OW.NBP			
		Pre-Hypertension	Group 1 ~ 5	MHBG.OW.MHBP			
		Hypertension	Group 1 ~ 5	MHBG.OW.HBP			
Diabetes	Low Weight	Normal	Group 1 ~ 5	HBG.LW.NBP			
		Pre-Hypertension	Group 1 ~ 5	HBG.LW.MHBP			
		Hypertension	Group 1 ~ 5	HBG.LW.HBP			
	Normal Weight	Normal	Group 1 ~ 5	HBG.NW.NBP			
		Pre-Hypertension	Group 1 ~ 5	HBG.NW.MHBP			
		Hypertension	Group 1 ~ 5	HBG.NW.HBP			
	Over weight/ Obesity	Normal	Group 1 ~ 5	HBG.OW.NBP			
		Pre-Hypertension	Group 1 ~ 5	HBG.OW.MHBP			
		Hypertension	Group 1 ~ 5	HBG.OW.HBP			

32

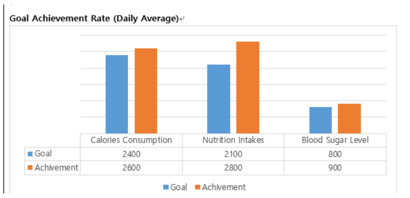


Service Stage – Update User's Data
Physical Activity Assessment
Service Registration
Questionnaire: Exercise & Physical Activity
Generate User's Profile – Physical Activity
Assessment during service
DietPic (Activity)
Mid-term Questn.
Services/ Algorithm/ Logic
Display check-up form for health status and lifestyle pattern
Exercise and Physical Activity Questionnaire Interface
User's Profile DB initial value creation
User's Profile DB updates
Retrieve "Current" value for the Field on request
No | Classification | Variable Name | Measure | Score Direction | Question Reference | Answer Choice and Measure Example (Ans to Ref. Question) | Questionnaire routine/Comments
1 | PA-Exercise Screening | Regular Exercise: General | Performing Exercise Regularly | DQ_RE.REPf | YES/NO | Positive HIGH | Do you currently perform regular, intended exercise? | 1=Yes 0=No | If DQ_RE.REPf=0, skip #2~#5 questions (go to #6 question)
Check-up Health Status & Lifestyle Pattern
[NEXT]
Physiological Status Assessment
Lifestyle Assessment
Health Risks Assessment
Physical Activity Assessment
Dietary Habits Assessment
Miscellanies
Physical Activity Check
1. Do you perform Regular Exercise?
YES
NO
Sub-Rule
6. Do you Have experience on RE?
....
Intermediate DB for questionnaire
User's Profile DB
DQ_RE.REPf = 1
PAHabits.EP.PEPf
Service Run-time Updates
Field Name | Measure Code | Date/Time | SCORE | Conversion Formula | Stored | Measure Counts | Current | Certified by
PAHabits.EP.Regularity | GCH-Q.002 | 170315:1430 | 1 | NAN | 1 | 1 | | System
PAHabits.EP.Regularity | GCH-WS.002 | 170430:1730 | 1 | NAN | 1 | 1 | * | E13123
33



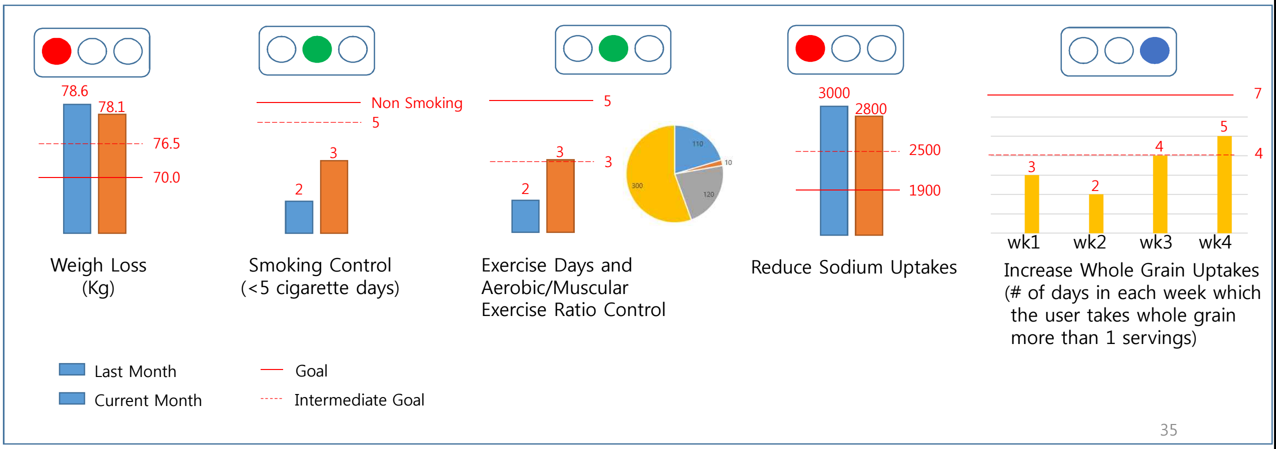
Service Stage – Intervention Goal Assignment
<Intervention Process - example>
User's Status Type (GCH)
Dietary Habit Score (GCH)
User PA Type (GCH)
NBG.NW.MHBP
+
Moderate Eating Habits
+
B-2
Recommendation Guidelines(General/Edu.) SERVICE OBJECTIVES
Weight Loss (0.5)
Smoking Control (0.3)
Increase PA (0.2)
Recommendation Guidelines(Nutrition) LIFESTYLE CHANGE GOAL
Reduce Sodium Uptakes (0.5)
Regular Eating (0.2)
Whole Grain Uptakes (0.2)
Protein uptakes (0.1)
Recommendation Guidelines(PA) LIFESTYLE CHANGE GOAL
2.5+hrs/wk PA (0.7)
Aerobic/Muscular exercise ratio control (0.3)
Compare dietary habits and recommendation
if DietaryHabits.NT.NaUtk = 1 ~ 2, wReducedSodiumUptakes = 0;
= 3 ~ 5, wReducedSodiumUptakes = 1
if DietaryHabits.EP.Regularity = 4 ~ 5, wRegularEating = 0;
= 3, wRegularEating = 0.5
= 1 ~ 2, wRegularEating = 1
if DietaryHabits.NT.WGUtk = 4 ~ 5, wWholeGrainUptakes = 0;
= 3, wWholeGrainUptakes = 0.5
= 1 ~ 2, wWholeGrainUptakes = 1
if DietaryHabits.NT.PtUtk = 4 ~ 5, wProteinUptakes = 0;
= 3, wProteinUptakes = 0.5
= 1 ~ 2, wProteinUptakes = 1
<Example>
DietaryHabits.NT.NaUtk = 3
DietaryHabits.EP.Regularity = 5
DietaryHabits.NT.WGUtk = 1
DietaryHabits.NT.PtUtk = 4
Reduce Sodium Uptakes (0.5) * wReducedSodiumUptakes (1) = 0.5
Regular Eating (0.2) * wRegularEating (0) = 0
Whole Grain Uptakes (0.2) * wWholeGrainUptakes (1) = 0.2
Protein uptakes (0.1) * wProteinUptakes (0) = 0
[Recommendation/GOAL]
Reduce Sodium Uptakes (0.5)
Increase Whole Grain Uptakes (0.2)
~~Regular Eating~~
~~Protein Uptakes~~
#AmountSodiumLimit = OOO mg/day (calculation rule/table) *measure: DietPic Meal Analysis Calculation
#WholeGrainUptakesGoal = O servings/day *measure: DietPic Expert's FG Analysis
Personalized Lifestyle Change Goals
PA Recommendation
[Recommendation/GOAL]
~~2.5+ hrs/wk PA (0.7)~~
Aerobic/Muscular Exercise Ratio Control (0.3)
Dietary Recommendation
[Recommendation/GOAL]
Reduce Sodium Uptakes (0.5)
Increase Whole Grain Uptakes (0.2)
~~Regular Eating~~
~~Protein Uptakes~~
Lifestyle Habits Recommendation
[Recommendation/GOAL]
Weigh Loss (0.5)
Smoking Control (0.3)
~~Increase PA (0.2)~~
34

Service Stage – Service Evaluation



Weigh Loss
Smoking Control

Aerobic/Muscular Exercise Ratio Control
Reduce Sodium Uptakes
Increase Whole Grain Uptakes



Section 5

Mining Minds Version 3.0 Requirement Specification

Section 5.1

Data Curation Layer(DCL) Requirement Specification

1. Data Curation Layer (DCL)

1.1 Functional Requirements

FR ID#	Description
DCL-FR-01	The platform shall read the raw sensory data of the user from his/her data source
DCL-FR-02	The platform shall provide permanent persistence to the user generated raw sensory data
DCL-FR-03	The platform shall provide raw sensory data for context determination of the user
DCL-FR-04	The platform shall maintain user profile data
DCL-FR-05	The platform shall maintain user timeline as a lifelog of daily behaviors
DCL-FR-06	The platform shall provide read, write, delete, and update access to the subscribers of lifelog data
DCL-FR-07	The platform shall provide read access to the subscribers of raw sensory data
DCL-FR-08	The platform shall monitor the lifelog of a user for notify-able situations
DCL-FR-09	The platform shall persist user feedback regarding generated recommendations and identified context

1.1.1 Non-Functional Requirements

FR ID#	Description
DCL-NFR-01	The platform shall read the raw sensory data of the user from his/her personal device in real-time with delay no later than 3 seconds
DCL-NFR-02	The platform shall provide raw sensory data for low level activities determination in real-time with delay no later than 3 seconds
DCL-NFR-03	The platform shall only read the raw sensory data from verified personal device
DCL-NFR-04	The platform shall maintain the consistency, integrity, and reliability of raw sensory data in non-volatile storage

1.1.2 Terms and Definitions

Term	Definition
DCL	Data Curation Layer
ICL	Information Curation Layer
KCL	Knowledge Curation Layer
SCL	Service Curation Layer
SL	Supporting Layer
Lifelog	Information associated to the user's life-events over time
Lifelog schema	Lifelog schema represents the structure and associated semantics of user profile and lifelog data.
User profile	Information describing the user characteristics (i.e., age, gender, etc.)

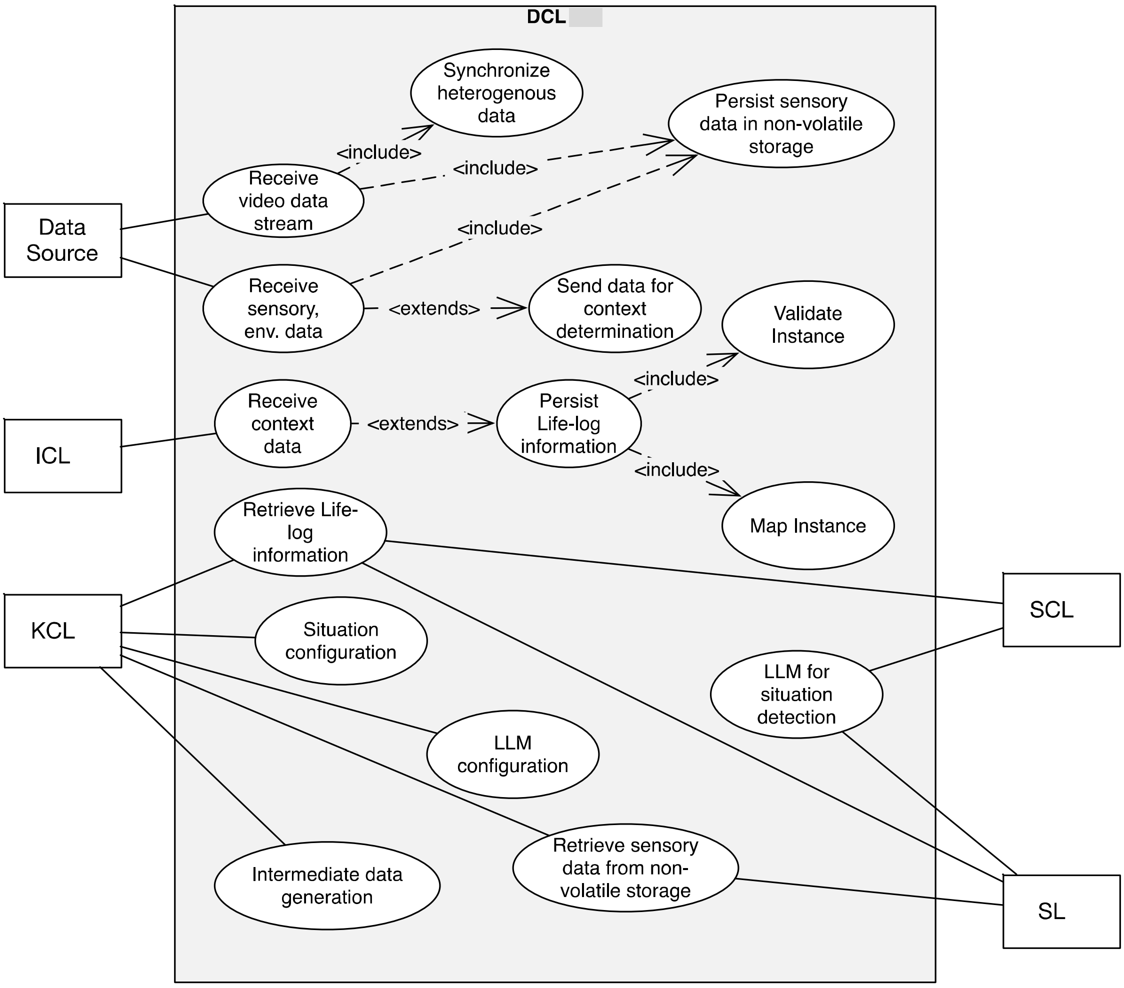
Data source	User devices sending the required data, i.e., smartphone, video camera
Raw sensory data	Numerical values describing a physical phenomenon such as human body motion (e.g., acceleration)
Sensory metadata	Information that describes, at least, the source of data (e.g., video), the user to which the raw sensory data belongs (e.g., user ID) and the time in which the raw sensory data was registered (e.g., timestamp)
Sensory data	Raw sensory data plus sensory metadata
SNS data	Data from social networks (i.e., twitter, Facebook)
Context	General concept to refer either to low-level context and/or high-level context
Situation	An abnormal status of a subject caused by unhealthy behaviors
Rule verification and validation	Verification ensures that rule created is consistent with requirements and validation ensures that the rule created is correctly working on real data
Unresolved case	A new case for which the existing knowledge is insufficient to solve
Recommendation	An actionable statement provided to the subject for healthy habit induction
Fact	An informative statement provided to the subject for education

1.1.3 Use-cases

1.1.3.1 List

Use case ID#	Name
DCL-UC-01	Receive sensory and environmental data from data source
DCL-UC-02	Receive video data stream from data source
DCL-UC-03	Synchronize heterogeneous user data
DCL-UC-04	Send data for context determination
DCL-UC-05	Receive context data
DCL-UC-06	Retrieve Lifelog Information
DCL-UC-07	Persist Lifelog Information
DCL-UC-08	Map Instances
DCL-UC-09	Validate Instances
DCL-UC-10	Situation configuration
DCL-UC-11	LLM configuration for target variables
DCL-UC-12	LLM for situation detection
DCL-UC-13	Retrieve sensory data from non-volatile storage for intermediate data generation (offline)
DCL-UC-14	Retrieve sensory data from non-volatile storage (online)
DCL-UC-15	Persist sensory data in non-volatile storage

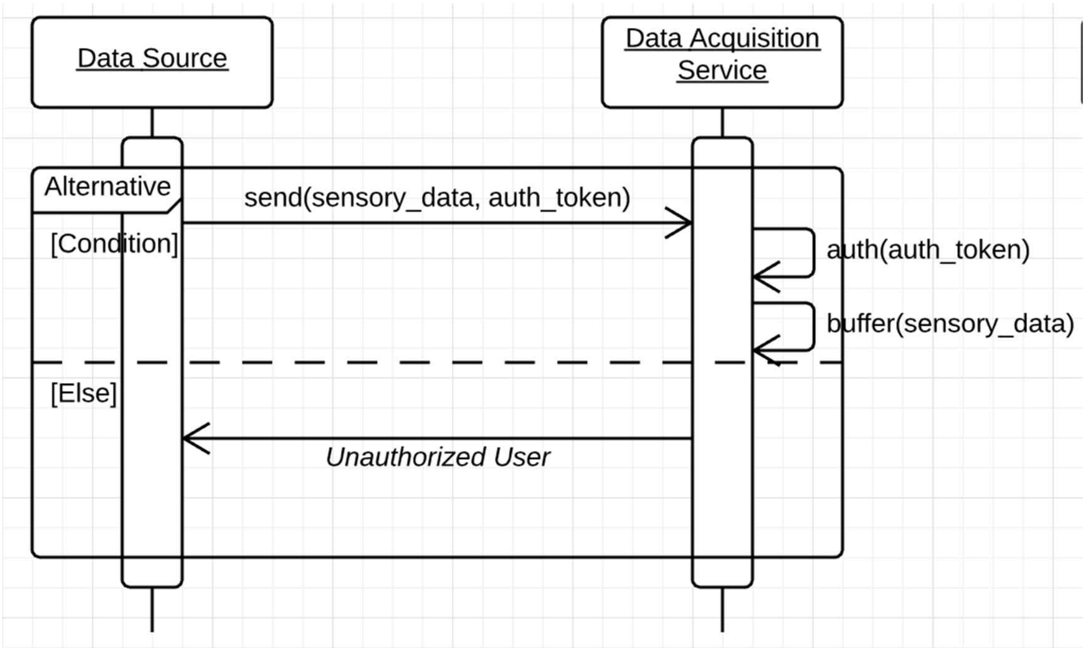
1.1.3.2 Diagram



1.1.3.3 Description

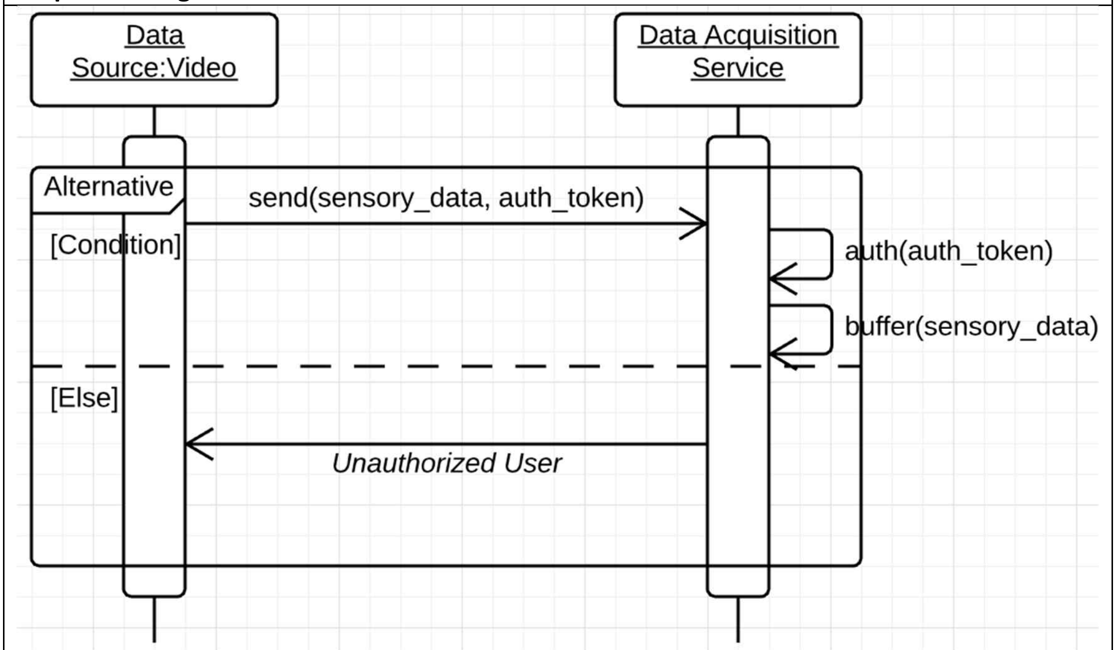
Use Case ID:	**DCL-UC-01**		
Use Case Name:	Receive sensory and environmental data from data source		
High Level Use case ID:			
Created By:	Bilal Amin	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	Data source		
Description:	User sensory data and environmental is received and buffered from data source in real time		
Trigger:	User activity of at least 3 seconds		
Pre-conditions:	User is a registered client of MM platform		
Post-conditions:	Sensory and environmental data is persisted in the buffer		
Normal Flow:	1. Sensory and environmental data is received by a data acquisition component 2. Data source is authenticated and contents of the data are verified 3. Data is temporary buffered for context determination		

Alternative Flows:	N/A
Exceptions:	2a. In step 2 of the normal flow, if the user is detected to be un-authorized or contents are un-verifiable 1. Data acquisition component destroys the data
Includes:	N/A
Frequency of Use:	Very frequent: every 3 second
NFR ID:	
Assumptions:	Communication contract is defined between data source and data acquisition component
Notes and Issues:	NA
Sequence Diagram:	



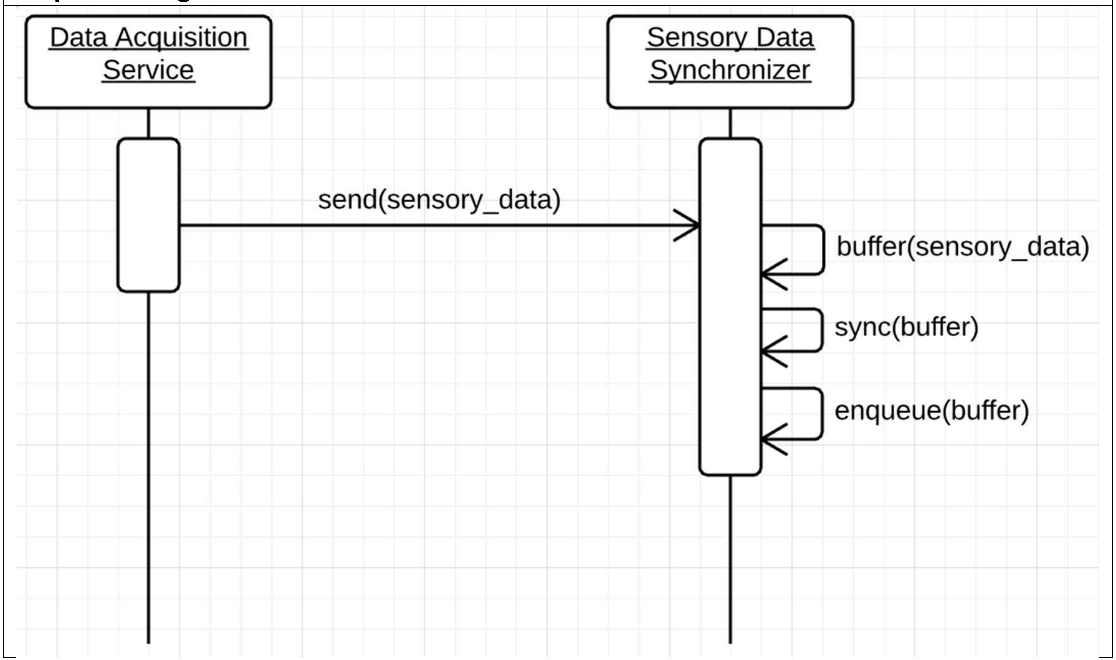
Use Case ID:	**DCL-UC-02**		
Use Case Name:	Receive video data stream from data source		
High Level Use case ID:			
Created By:	Bilal Amin	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	Data source		
Description:	User video data stream is received and buffered from data source in real time		
Trigger:	Video camera is streaming user feed		
Pre-conditions:	User is a registered client of MM platform		

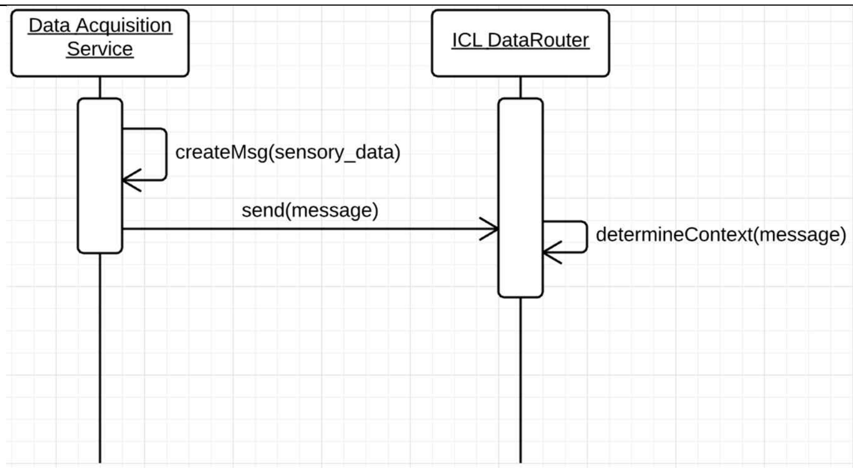
Post-conditions:	User video data stream is persisted in the video stream buffer
Normal Flow:	1. Video data stream is received by a data acquisition component 2. Data source is authenticated and contents of the data stream are verified 3. Video data stream is temporary buffered for context determination
Alternative Flows:	N/A
Exceptions:	2a. In step 2 of the normal flow, if the user is detected to be un-authorized or contents are un-verifiable 2. Data acquisition component destroys the data
Includes:	N/A
Frequency of Use:	Less frequent: If video streaming based data source is available
NFR ID:	
Assumptions:	Video data streaming communication contract is defined between data source and data acquisition component
Notes and Issues:	NA
Sequence Diagram:	



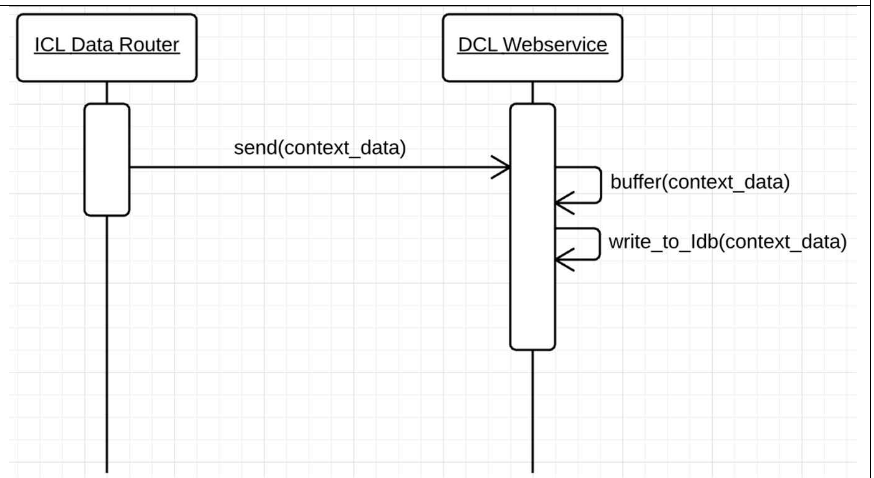
Use Case ID:	**DCL-UC-03**		
Use Case Name:	Synchronize heterogeneous user data		
High Level Use case ID:			
Created By:	Bilal Amin	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016

Actors:	DCL
Description:	User video data stream is synchronized with the corresponding sensory data
Trigger:	Video camera is streaming user feed
Pre-conditions:	Video data stream is persisted in the video stream buffer
Post-conditions:	Video data stream is synchronized with its corresponding sensory data
Normal Flow:	1. Time stamp and user id of video data stream is read from the video stream buffer 2. Sensory data is searched and retrieved from buffer based on the time stamp and user id 3. Retrieved sensory data is concatenated with the video stream data and stored back in the sensory data buffer for context determination
Alternative Flows:	2a. In step 2 of the normal flow, if the sensory data is not found 1. Data acquisition component deletes the video data stream from the video data stream buffer
Exceptions:	NA
Includes:	NA
Frequency of Use:	Less frequent: If video streaming based data source is available
NFR ID:	
Assumptions:	4. Video data streaming communication contract is defined between data source and data acquisition component
Notes and Issues:	NA
Sequence Diagram:	



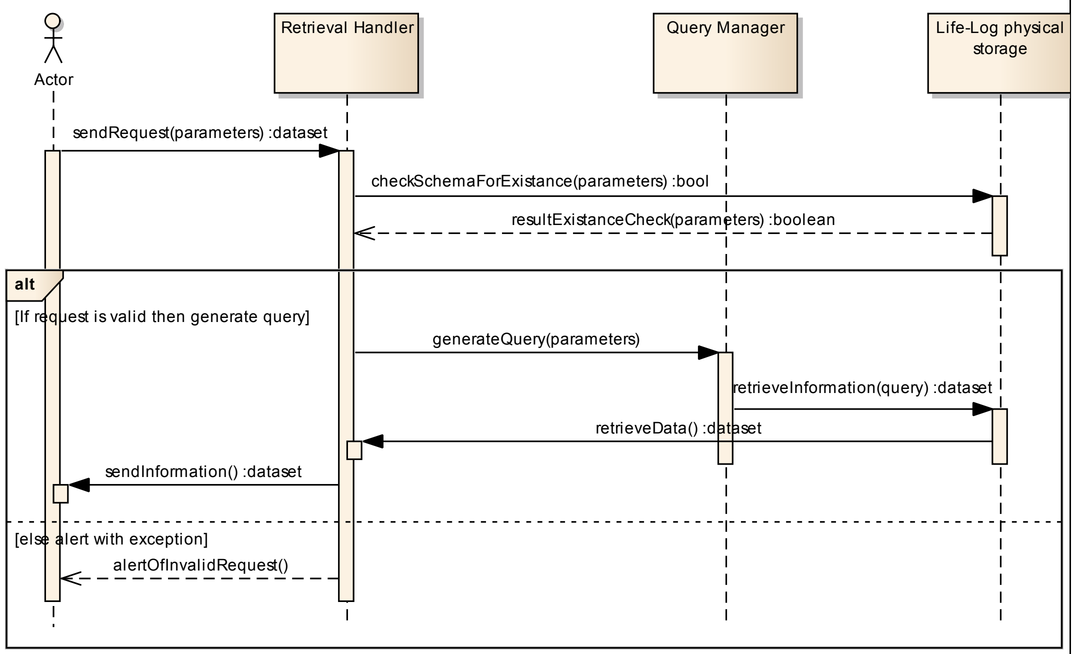
<table>
<tr><td>Use Case ID:</td><td colspan="3">DCL-UC-04</td></tr>
<tr><td>Use Case Name:</td><td colspan="3">Send data for context determination</td></tr>
<tr><td>High Level Use case ID:</td><td colspan="3"></td></tr>
<tr><td>Created By:</td><td>Bilal Amin</td><td>Last Updated By:</td><td>Bilal Amin</td></tr>
<tr><td>Date Created:</td><td>15 July 2015</td><td>Last Revision Date:</td><td>20 Dec 2016</td></tr>
<tr><td>Actors:</td><td colspan="3">Context sender</td></tr>
<tr><td>Description:</td><td colspan="3">Sensory data buffer is sent to ICL for context determination</td></tr>
<tr><td>Trigger:</td><td colspan="3">Sensory data is available for context determination</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">Sensory data is persisted in the buffer</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">Sensory data is sent for context determination</td></tr>
<tr><td>Normal Flow:</td><td colspan="3">1. Context sender reads sensory data from the sensory data buffer
2. Context sender creates communication object by serialization
3. Communication object is sent to the ICL server</td></tr>
<tr><td>Alternative Flows:</td><td colspan="3">NA</td></tr>
<tr><td>Exceptions:</td><td colspan="3">NA</td></tr>
<tr><td>Includes:</td><td colspan="3">NA</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Frequent: whenever context need to be determined</td></tr>
<tr><td>NFR ID:</td><td colspan="3">Leave it blank</td></tr>
<tr><td>Assumptions:</td><td colspan="3">NA</td></tr>
<tr><td>Notes and Issues:</td><td colspan="3">NA</td></tr>
<tr><td>Sequence Diagram:</td><td colspan="3">
</td></tr>
</table>

Use Case ID:	**DCL-UC-05**		
Use Case Name:	Receive context data		
Created By:	Bilal Amin	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	ICL server		

Description:	After the determination by ICL, context is received by context receiver component and forwarded for non-volatile storage
Trigger:	New context or change in previous context is determined by ICL
Pre-conditions:	Context data is available
Post-conditions:	Context data is sent for non-volatile storage
Normal Flow:	1. Context receiver receives context object 2. Context receiver de-serializes context object 3. Context object is sent for non-volatile persistence (async) 4. Context object is sent for lifelog mapping
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Frequent: whenever context is determined
NFR ID:	
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	

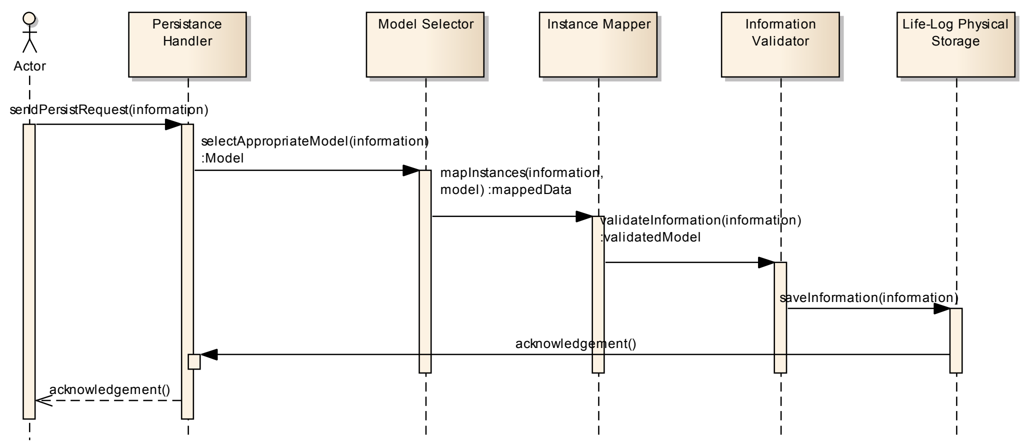
Use Case ID:	**DCL-UC-06**		
Use Case Name:	Retrieve Lifelog Information		
FR ID:			
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	SCL, KCL, SL		
Description:	Each actor needs information from life log for further processing. All actors shall request their related and desired Lifelog information from physical storage.		
Trigger:	On request of an actor to access required information		
Pre-conditions:	The actor shall be authorized with full access on the Lifelog data.		
Post-conditions:	Provide the required data to layer		

Normal Flow:	1. Actor sends request for desired Lifelog information. 2. The desired request shall be checked for information existence. If request is valid a. Prepare the query for desired information based on request b. Load the requested information from physical storage c. Send back the loaded information to the actor
Alternative Flows:	2b. The desired data is not existing in the schema, invalid request 1. Acknowledge the actor with exception of invalid request.
Exceptions:	NA
Includes:	NA
Frequency of Use:	Whenever Lifelog information is required.
NFR ID:	
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	



Use Case ID:	**DCL-UC-07**		
Use Case Name:	Persist Lifelog Information		
FR ID:			
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	ICL		
Description:	Each actor performed some specific operations on incoming data from external resources or on already existing information in Lifelog repository.		

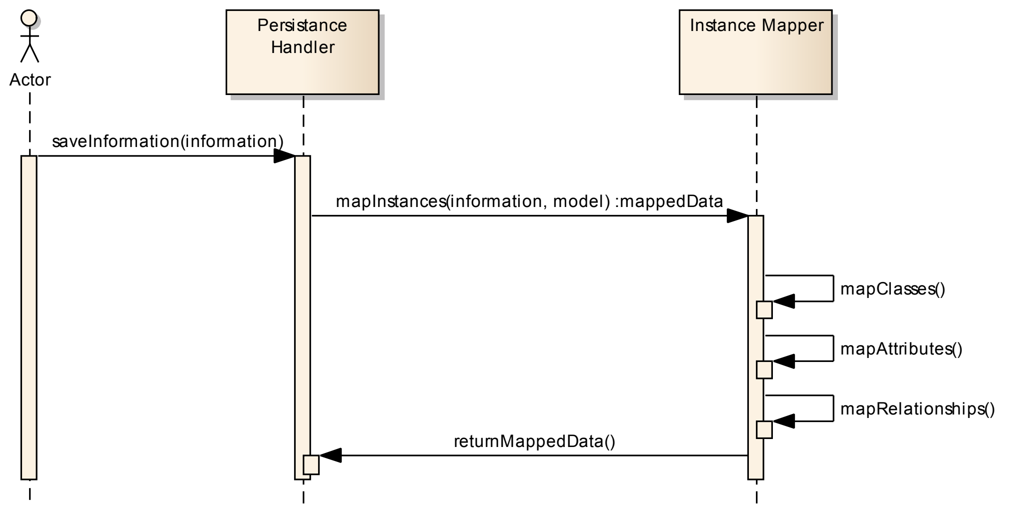
	In both cases the information shall be updated and stored in Lifelog repository.
Trigger:	On request of a actor to persist required information
Pre-conditions:	The actor shall be authorized with full access on the Lifelog data.
Post-conditions:	Successfully stored the created Lifelog information
Normal Flow:	1. Actor sends request to persist new generated Lifelog information. 2. Passes the new created information to check the appropriate hierarchical structure. 3. The appropriate selected hierarchical structure with input information passes to find the information instances. 4. Check the consistency among the records and their relationship. 5. Store the validated and structured information into physical storage
Alternative Flows:	NA
Exceptions:	NA
Includes:	Map Instances, Validate Instances, save information to physical storage.
Frequency of Use:	Whenever new information is generated.
NFR ID:	
Assumptions:	NA
Notes and Issues:	Capability to process multiple actor's requests for storage.
Sequence Diagram:	



Use Case ID:	**DCL-UC-08**		
Use Case Name:	Map Instances		
FR ID:			
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali

Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	DCL, ICL, SCL, SL, and Lifelog physical storage		
Description:	The information produced by each actor shall be mapped against the hierarchical structure of storage.		
Trigger:	On request of an actor to persist required information		
Pre-conditions:	The actor shall be authorized with full access on the Lifelog data.		
Post-conditions:	Successfully mapped the instances with correct Lifelog information schema		
Normal Flow:	1. Actor sends request to persist new generated information. 2. System searches each information records against hierarchical structure. 3. System finds appropriate classes of the instances 4. System extracts attributes in the instances. 5. Find the relationship among the information records. 6. Pass the annotated information for validation.		
Alternative Flows:			
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Whenever new information is persisted.		
NFR ID:			
Assumptions:	NA		
Notes and Issues:	Capability to process multiple actors' requests for storage.		

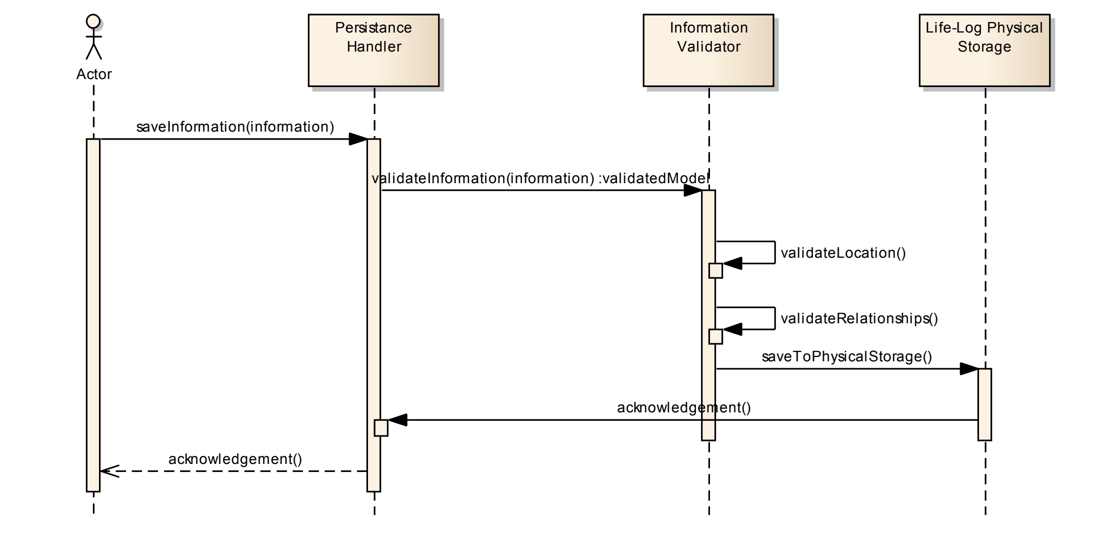
Sequence Diagram:



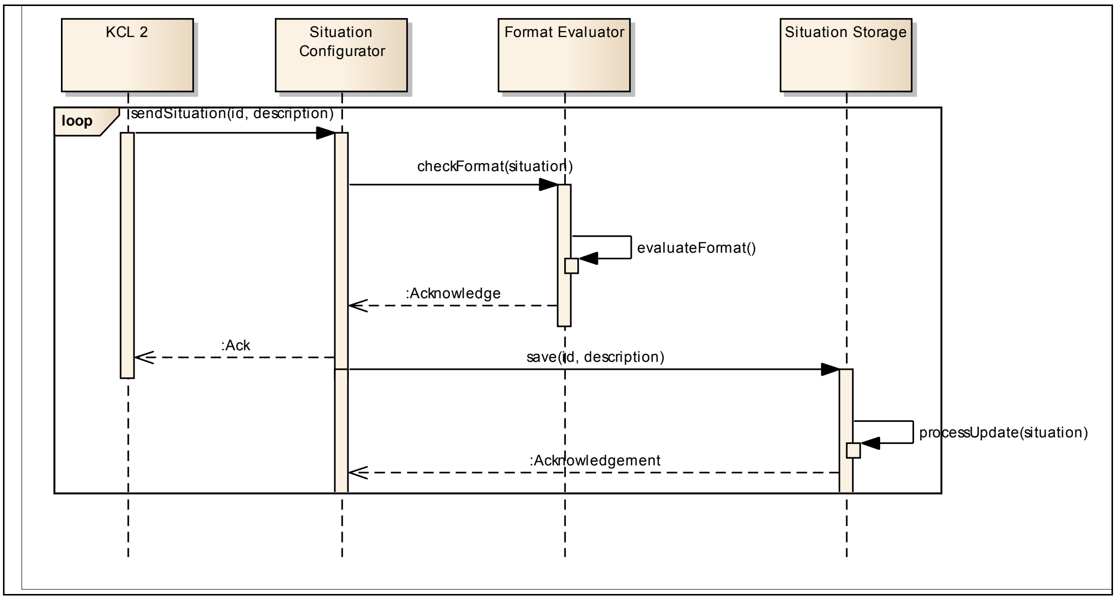
Use Case ID:	**DCL-UC-09**
Use Case Name:	Validate Instances

FR ID:			
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	DCL, ICL, SCL, SL, and Lifelog physical storage		
Description:	The mapped information in previous use case shall be checked for consistency among the existing information.		
Trigger:	On request of an actor to persist required information		
Pre-conditions:	The actor shall be authorized with full access on the Lifelog data.		
Post-conditions:	Successfully validate the instances with correct Lifelog information schema		
Normal Flow:	1. Actor send request to persist new generated information. 2. DCL passes the mapped information for validation of information and their relationships. 3. The system checks the information according to the specific location in the hierarchy. 4. The system checks and builds the relationship among concepts. 5. The verified information shall be passed for persistence.		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Whenever new information is persisted.		
NFR ID:			
Assumptions:	NA		
Notes and Issues:	Capability to process multiple actors' requests for storage.		

Sequence Diagram:

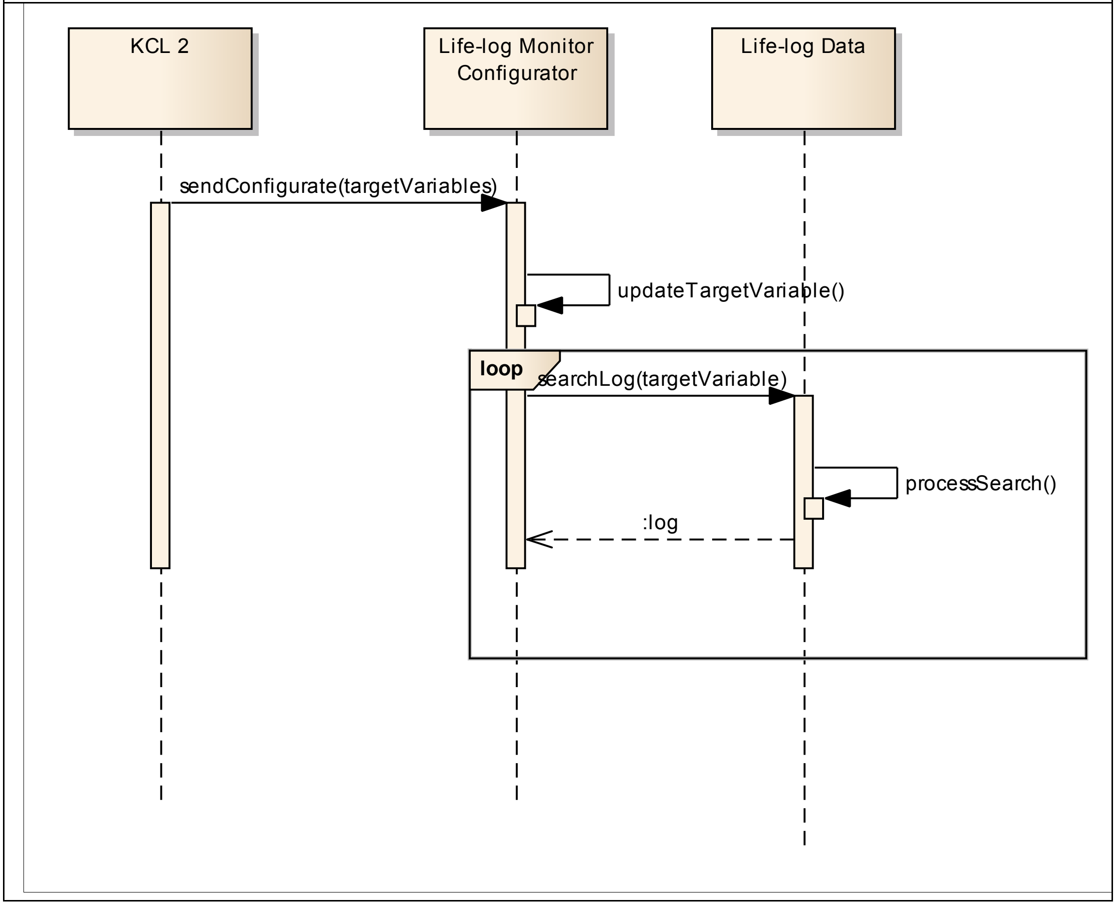


<table>
<tr><td>Use Case ID:</td><td colspan="3">DCL-UC-10</td></tr>
<tr><td>Use Case Name:</td><td colspan="3">Situation Configuration</td></tr>
<tr><td>FR ID:</td><td colspan="3"></td></tr>
<tr><td>Created By:</td><td>Bilal Ali</td><td>Last Updated By:</td><td>Bilal Ali</td></tr>
<tr><td>Date Created:</td><td>15 July 2015</td><td>Last Revision Date:</td><td>20 Dec 2016</td></tr>
<tr><td>Actors:</td><td colspan="3">KCL</td></tr>
<tr><td>Description:</td><td colspan="3">Situation is determined by experts and is communicated to DCL for monitoring the Lifelog.</td></tr>
<tr><td>Trigger:</td><td colspan="3">Creation of new rule to capture a situation.</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">KCL and DCL should agree on common representation of sharing information of Situation configuration</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">1. Situation is stored against a specific category.
2. Situation is available for monitoring the Lifelog.</td></tr>
<tr><td>Normal Flow:</td><td colspan="3">1. KCL connects to DCL and send the newly created situation in common configuration format.
2. DCL evaluates the format of received situation configuration.
3. DCL responds with acknowledgement message.
4. Situation will be parsed into components.
5. Parsed components are updated in persistent storage as per categories.</td></tr>
<tr><td>Alternative Flows:</td><td colspan="3">NA</td></tr>
<tr><td>Exceptions:</td><td colspan="3">Format of situation is not according the agreement.</td></tr>
<tr><td>Includes:</td><td colspan="3">NA</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Invoked per situation creation by the expert.</td></tr>
<tr><td>NFR ID:</td><td colspan="3"></td></tr>
<tr><td>Assumptions:</td><td colspan="3">Well defined schema is available to store situation persistently</td></tr>
<tr><td>Notes and Issues:</td><td colspan="3">Standardize situation format is a challenging task</td></tr>
<tr><td colspan="4">Sequence Diagram:</td></tr>
</table>



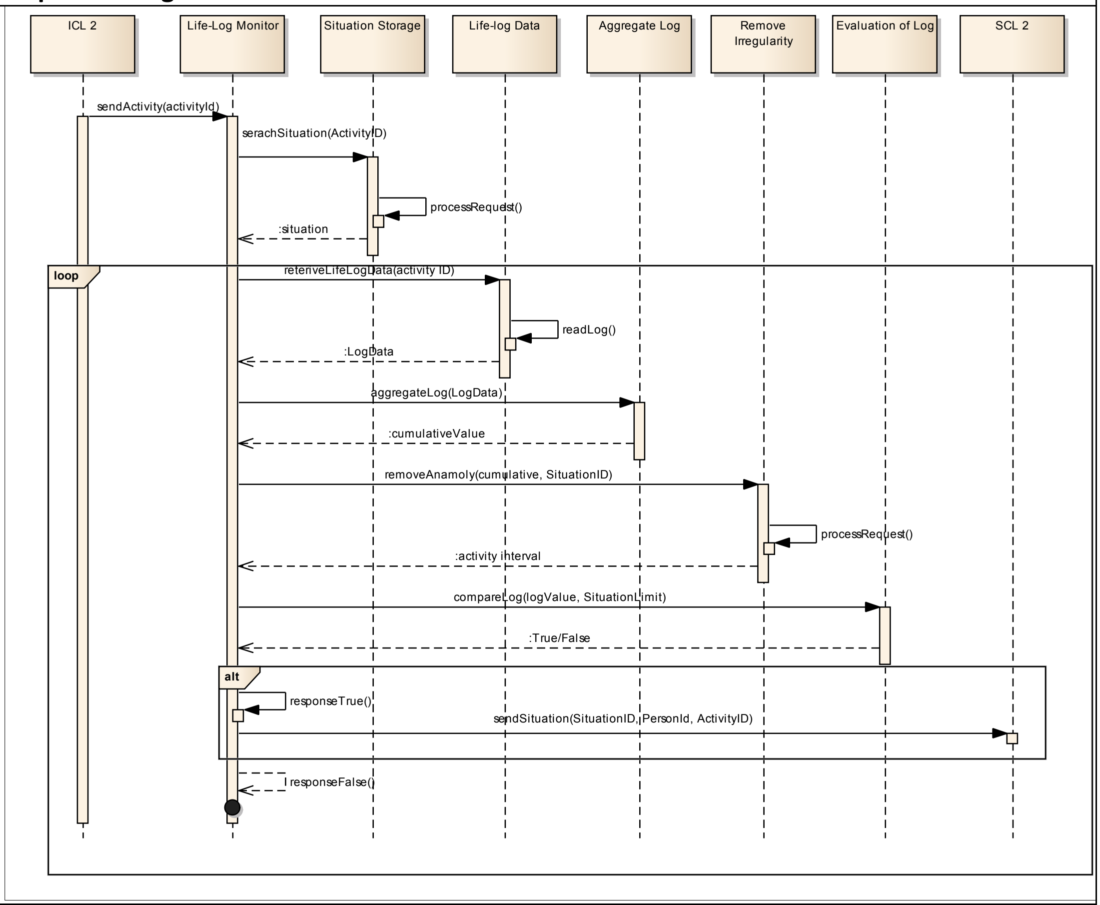
Use Case ID:	**DCL-UC-11**		
Use Case Name:	LLM Configuration for target Variables		
FR ID:			
Created By:	Bilal Ali	**Last Updated By:**	Bilal Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	KCL, Experts		
Description:	Configure the Lifelog monitor for the screening of the target variable from Lifelog data.		
Trigger:	On start of user's monitored activity		
Pre-conditions:	1. Expert defines target variable in common configuration format. 2. Access to Lifelog.		
Post-conditions:	Targeted log is retrieved from Lifelog data as per target variable requirements.		
Normal Flow:	1. KCL will share the target variables in common configured format created by expert. 2. Lifelog monitor is configured based on the shared target variable. 3. Lifelog monitor retrieve log data from Lifelog against the target variables.		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	On update of common configuration format.		
NFR ID:			
Assumptions:			

Notes and Issues:	Real time accommodation of update in common configuration format is challenging task.
Sequence Diagram:	

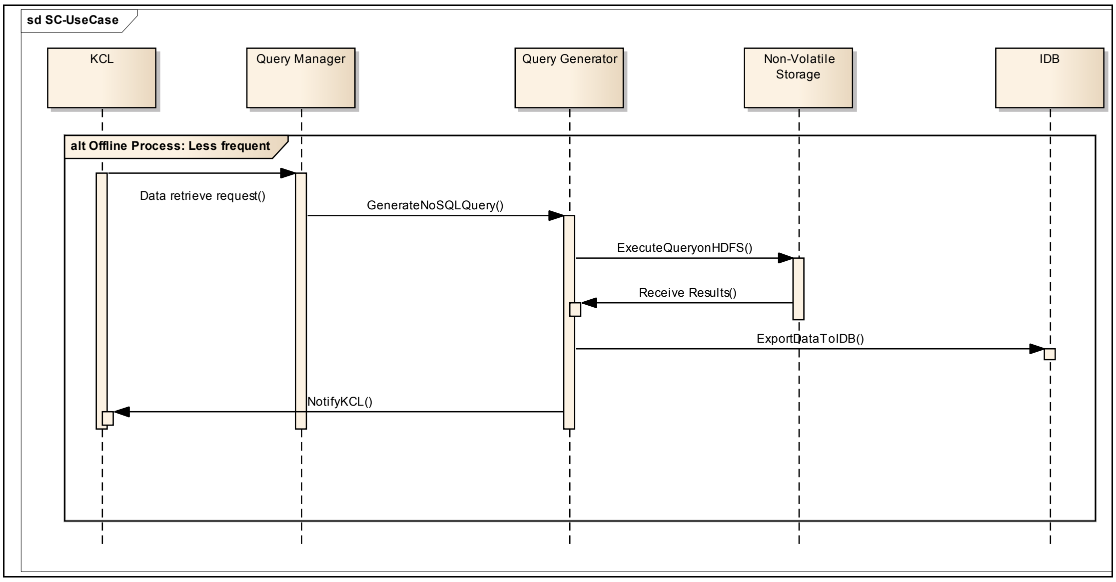


Use Case ID:	**DCL-UC-12**		
Use Case Name:	LLM for Situation Detection		
FR ID:			
Created By:	Bilal Ali	**Last Updated By:**	Bilal Ali
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	Lifelog Data, SCL, ICL		
Description:	Identification of the existence of a condition in user activities to highlight the alarming situation as per experts' understanding.		
Trigger:	On start of user's monitored activity		
Pre-conditions:	1. Activity is identified. 2. Situation is configured. 3. Access to Lifelog.		
Post-conditions:	Alarming situation is detected and triggered the SCL with situation and user.		
Normal Flow:	1. ICL recognizes activity and sends to Lifelog. 2. Lifelog monitor identify the target activity. 3. Retrieve associated situation with the activity.		

	4. Continuous access that activity log. 5. Aggregate the interval/duration of activity. 6. Remove the irregularity in activity as per situation. 7. Evaluate the duration of activity against the situation. a. If situation condition meets then send message to SCL to inform about the occurrence of a situation along with user information. b. If situation condition does not occur, don't send message to SCL.
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	For every activity with configured situation.
NFR ID:	
Assumptions:	
Notes and Issues:	Management of irregularity in activity is a challenging task.
Sequence Diagram:	

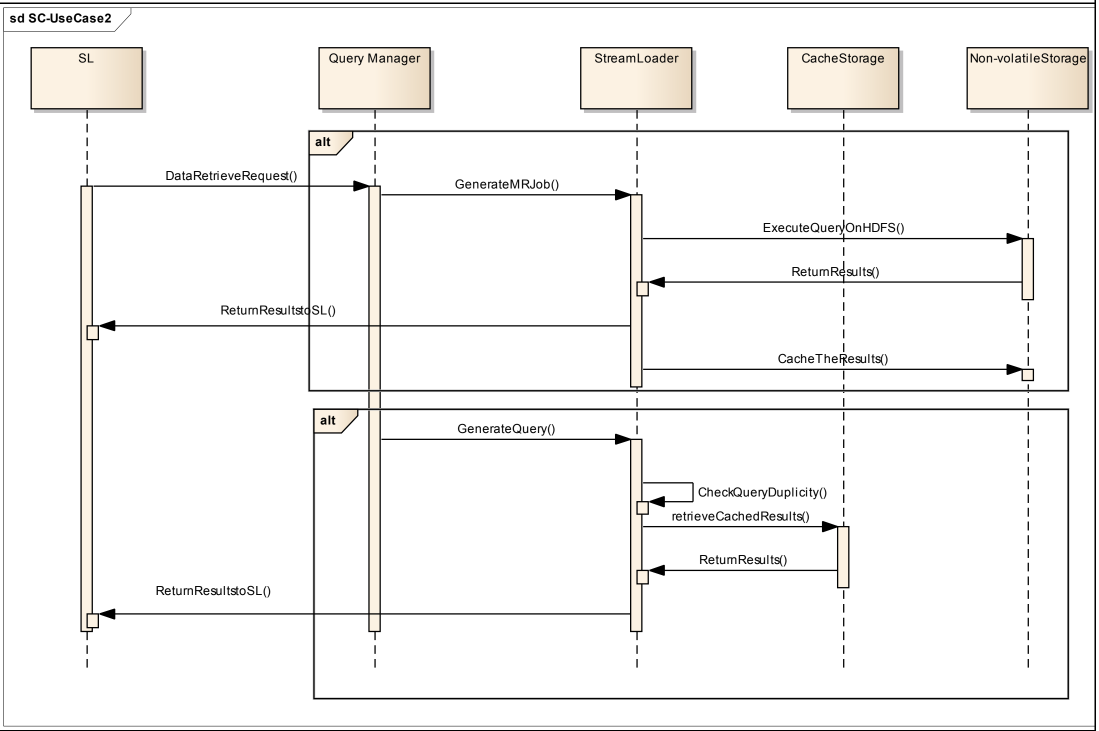


Use Case ID:	**DCL-UC-13**		
Use Case Name:	Retrieve sensory data from non-volatile storage for intermediate data generation (offline)		
FR ID:			
Created By:	Idris	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	KCL		
Description:	Raw data in HDFS is retrieved based on the request from KCL and converted to relational format and stored in IDB.		
Trigger:	KCL requests the data		
Pre-conditions:	1. Relational IDB schema has already been described and shared 2. The data exists in HDFS		
Post-conditions:	1. The data has been transformed and exported to IDB 2. The KCL is informed		
Normal Flow:	1. SCL receives requests from KCL and creates a NoSQL query 2. NoSQL query is executed using Apache Hive on HDFS to retrieve the data 3. Retrieved data is processed and transformed to relational format based on IDB schema 4. Transformed data is exported to IDB		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Less frequent, offline process, may be executed once or on change in the IDB schema		
NFR ID:			
Assumptions:	NoSQL query results are easy to transform to relational format		
Notes and Issues:	NA		
Sequence Diagram:			



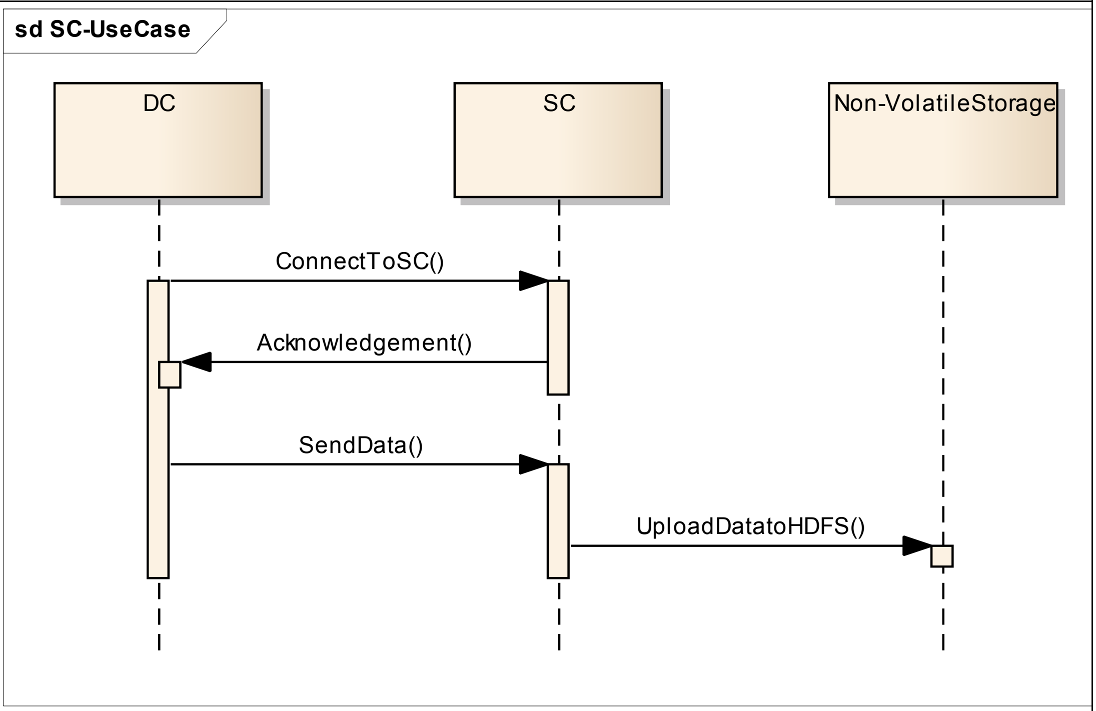
Use Case ID:	**DCL-UC-14**		
Use Case Name:	Retrieve sensory data from non-volatile storage (online)		
FR ID:			
Created By:	Idris	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	SL		
Description:	Raw sensory data in HDFS is retrieved and provided to analytics component of SL in an online process		
Trigger:	Request for data from descriptive analytics is received		
Pre-conditions:	Raw sensory exists and persisted in HDFS (non-volatile storage)		
Post-conditions:	1. Required data is retrieved from HDFS 2. Data is cached locally in SCL 3. Required data is communicated to descriptive analytics in SL directly (online)		
Normal Flow:	1. Descriptive analytics in SL requests data from SCL 2. SCL receives the request and maintains its log. 3. SCL transforms request to a MapReduce Job. 4. MapReduce job is executed on HDFS and results are retrieved. 5. Retrieved results are cached locally in volatile storage 6. Results are forwarded directly to descriptive analytics		
Alternative Flows:	1. Descriptive analytics requests data from SCL 2. SCL receives the request and examines it for duplicity(a request coming more than one time) 3. Duplicate request results are directly forwarded from the cached results to descriptive analytics 4. First time requests follow the normal flow.		
Exceptions:	NA		
Includes:	NA		

Frequency of Use:	Frequent requests from descriptive analytics in SL
NFR ID:	
Assumptions:	Request format contract already defined between SL(descriptive analytics) and DCL (SCL)
Notes and Issues:	NA
Sequence Diagram:	



Use Case ID:	**DCL-UC-15**		
Use Case Name:	Persist sensory data in non-volatile storage		
FR ID:			
Created By:	Idris	**Last Updated By:**	Bilal Amin
Date Created:	15 July 2015	**Last Revision Date:**	20 Dec 2016
Actors:	DCL		
Description:	Receive and persist raw sensory data from DCL in to HDFS		
Trigger:	DCL request to upload data, every 3 seconds		
Pre-conditions:	1. Data storage structure, directory structure in HDFS defined 2. File formats and data formats in HDFS known 3. Raw sensory data is received from DCL 4. Big Data server is already running		
Post-conditions:	1. Raw sensory data is persisted in HDFS non-volatile storage 2. Data is available for processing and access by SL and KCL		

Normal Flow:	1. Big Data server listening for data requests from DCL 2. DCL connects and sends data to SCL 3. SCL uploads received data to HDFS
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Very frequent, every 3 seconds
NFR ID:	
Assumptions:	Data format and specifications already defined between DCL and SCL
Notes and Issues:	NA
Sequence Diagram:	



Section 5.2

Information Curation Layer(ICL) Requirement Specification

Information Curation Layer

1. ICL Requirement Specification

1.1. ICL 3 Function Requirements

Requirements #ID	Description
ICL3-FR-01	The platform shall provide each low-level context recognizer with the appropriate raw sensory data for recognition
ICL3-FR-02	The platform shall identify the user's low-level context
ICL3-FR-03	The platform shall identify the user's high-level context
ICL3-FR-04	The platform shall provide low-level context information for the generation of the life-log
ICL3-FR-05	The platform shall provide high-level context information for the generation of the life-log

1.2. ICL 3 Non-Function Requirements

Requirements #ID	Description
ICL3-NFR-01	Overall low-level context recognition accuracy of the platform shall be be greater than or equal to 85%
ICL3-NFR-02	Overall high-level context inference accuracy of the platform shall be greater than or equal to 80%

1.3. SCL 2 Terms and Definitions

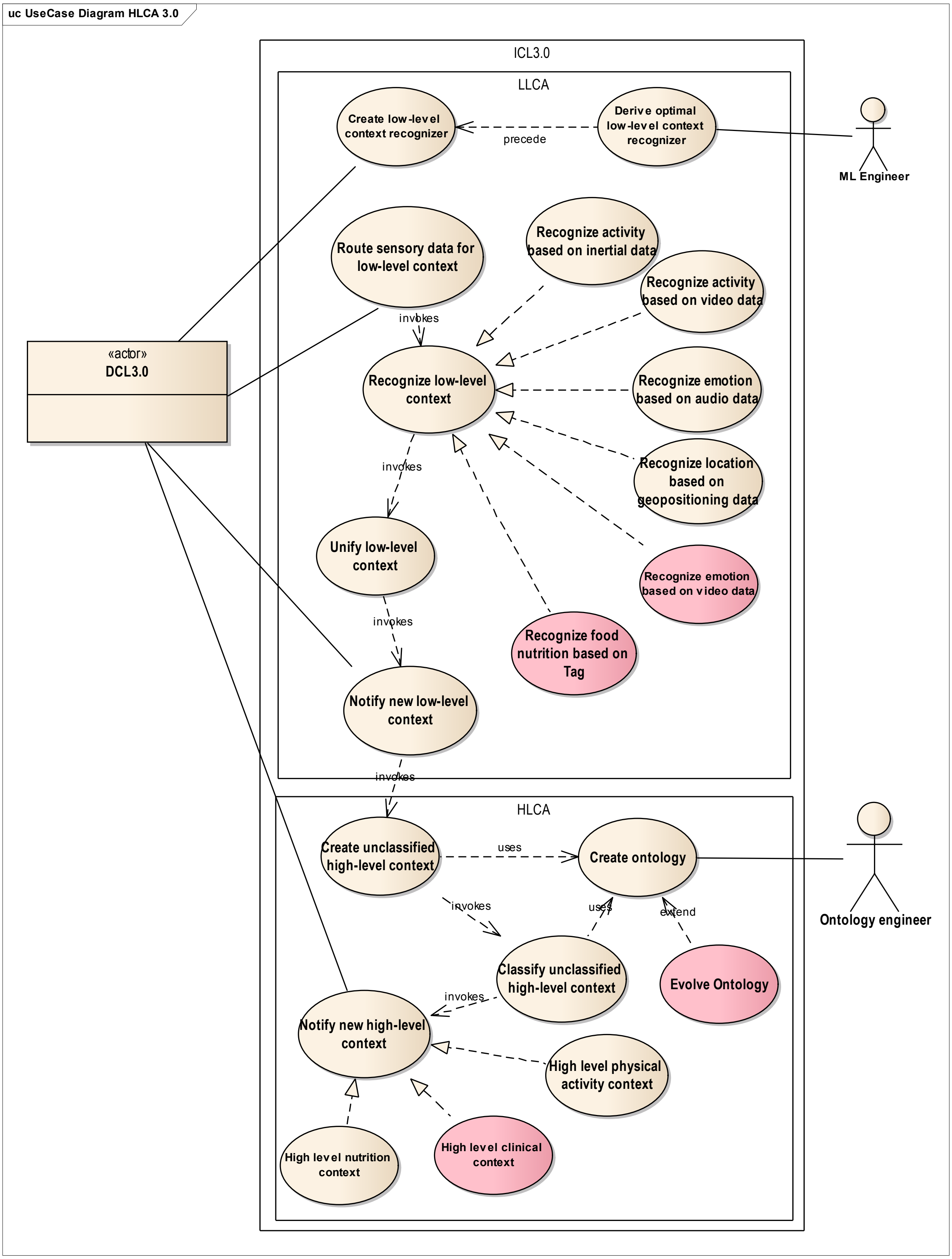
Terms	Description
DCL3	Data Curation Layer Ver. 3
KCL3	Knowledge Curation Layer Ver. 3
Reasoning	The process of producing recommendations
Low-level context	Information describing the user activities (e.g., sitting), user locations (e.g., restaurant) and user emotions (e.g., happy)
High-level context	Information describing the situation of the user (e.g., lunch)
Context	General concept to refer either to low-level context and/or high-level context

2. Use Case Diagram and Details

2.1. Use Case List

Use Case #ID	Description
ICL3-SUC-01	Derive optimal low-level context recognizer
ICL3-SUC-02	Create low-level context recognizers
ICL3-SUC-03	Route sensory data for the low-level context identification
ICL3-SUC-04	Recognize user low-level context
ICL3-SUC-05	Recognize user activity based on inertial raw sensory data
ICL3-SUC-06	Recognize user activity based on video raw sensory data
ICL3-SUC-07	Recognize user location based on geopositioning raw sensory data
ICL3-SUC-08	Recognize user emotion based on audio raw sensory data
ICL3-SUC-09	Recognize user emotion based on video raw sensory data
ICL3-SUC-10	Recognize food based on tag data
ICL3-SUC-11	Unify low-level contexts
ICL3-SUC-12	Notify new low-level context
ICL2-SUC- 13	Create unclassified high-level context instance
ICL2-SUC- 14	Classify high-level context instance
ICL2-SUC-15	Notify new high-level context
ICL2-SUC-16	Load context ontology model
ICL2-SUC-17	Store context instance
ICL2-SUC-18	Retrieve context instance
ICL3.0-SUC-19	Evolve Ontology
ICL2.5-SUC-20	High Level Physical Activity Context
ICL2.5-SUC-21	High Level Nutrition Context
ICL3.0-SUC-22	High Level Clinical Context

2.2. Use case diagram



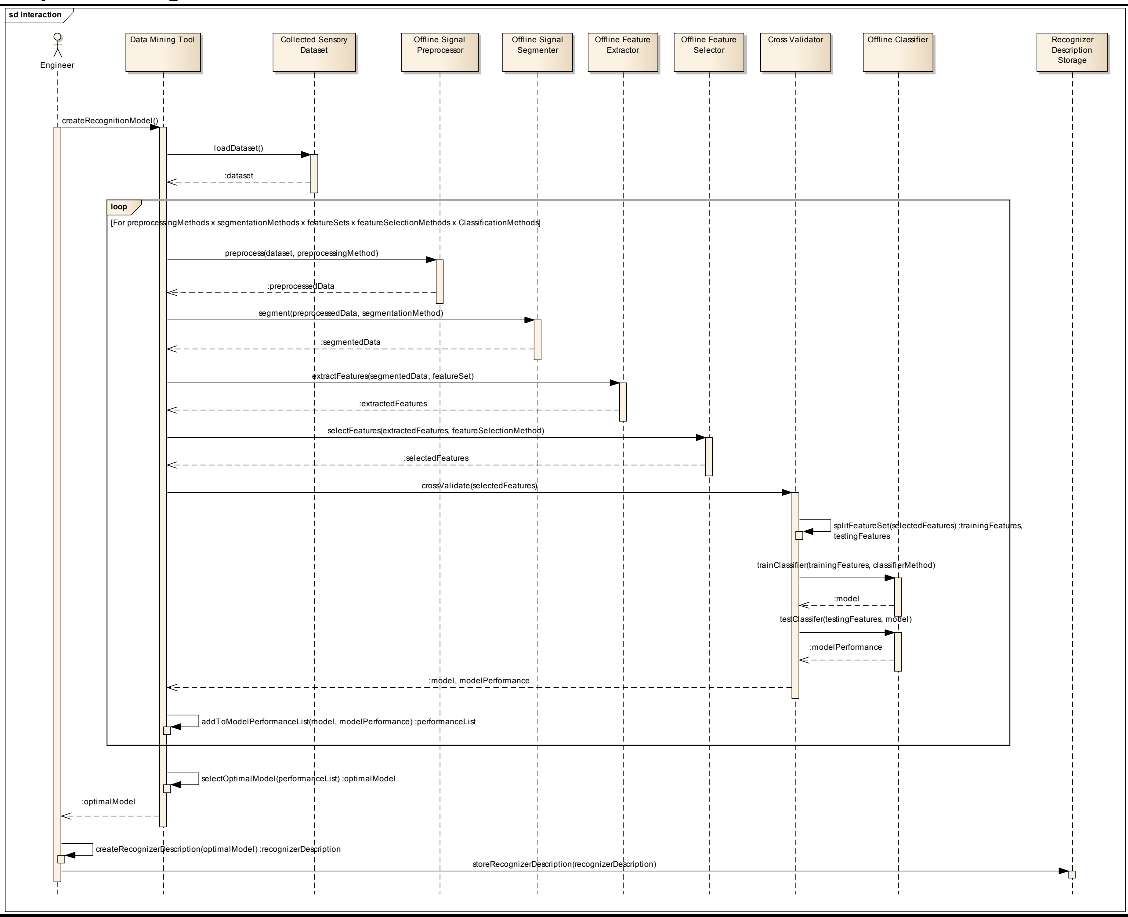
2.3.

2.4. Use Case Details

<table>
<tr><td>Use Case ID:</td><td colspan="3">ICL3-SUC-01</td></tr>
<tr><td>Use Case Name:</td><td colspan="3">Derive optimal low-level context recognizer</td></tr>
<tr><td>FR ID:</td><td colspan="3">MM-FR-11</td></tr>
<tr><td>Created By:</td><td>Oresti Banos</td><td>Last Updated By:</td><td>Jaehun Bang</td></tr>
<tr><td>Date Created:</td><td>22 July 2015</td><td>Last Revision Date:</td><td>9 March 2017</td></tr>
<tr><td>Actors:</td><td colspan="3">Engineer</td></tr>
<tr><td>Description:</td><td colspan="3">Create an optimal recognition model through the evaluation of multiple recognition model candidates (offline process).</td></tr>
<tr><td>Trigger:</td><td colspan="3">Engineer initiates the process for creating an optimal recognition model</td></tr>
<tr><td>Pre-conditions:</td><td colspan="3">A human expert or engineer sets up the experimental setup for the evaluation process</td></tr>
<tr><td>Post-conditions:</td><td colspan="3">The optimal recognition model among considered is delivered to the expert
A recognizer descriptor containing the characteristics of the optimal model is stored</td></tr>
<tr><td>Normal Flow:</td><td colspan="3">Load dataset
For each combination of preprocessing methods, segmentation methods, feature sets, feature selection methods, and classification methods, the dataset is preprocessed (e.g., filtered)
The preprocessed dataset is segmented (e.g., partitioned into windows)
Features (e.g., mean, variance) are extracted from each segment of the dataset
The best features are selected
Cross validation is applied to the selected features<ol type="a">
The feature set is split into training and testing
The classifier is trained using the training features
The classifier is tested in order to determine the model performance
The model performance is stored
Once the model performance has been calculated for all the possible combinations, the optimal model is selected
A recognizer descriptor is generated according to the characteristics of the model (e.g., median filtering, 3 sec window size, etc.)
The generated recognizer descriptor is stored</td></tr>
<tr><td>Alternative Flows:</td><td colspan="3">NA</td></tr>
<tr><td>Exceptions:</td><td colspan="3">NA</td></tr>
<tr><td>Includes:</td><td colspan="3">NA</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Infrequent</td></tr>
<tr><td>NFR ID:</td><td colspan="3">MM-NFR-05</td></tr>
<tr><td>Assumptions:</td><td colspan="3">NA</td></tr>
</table>

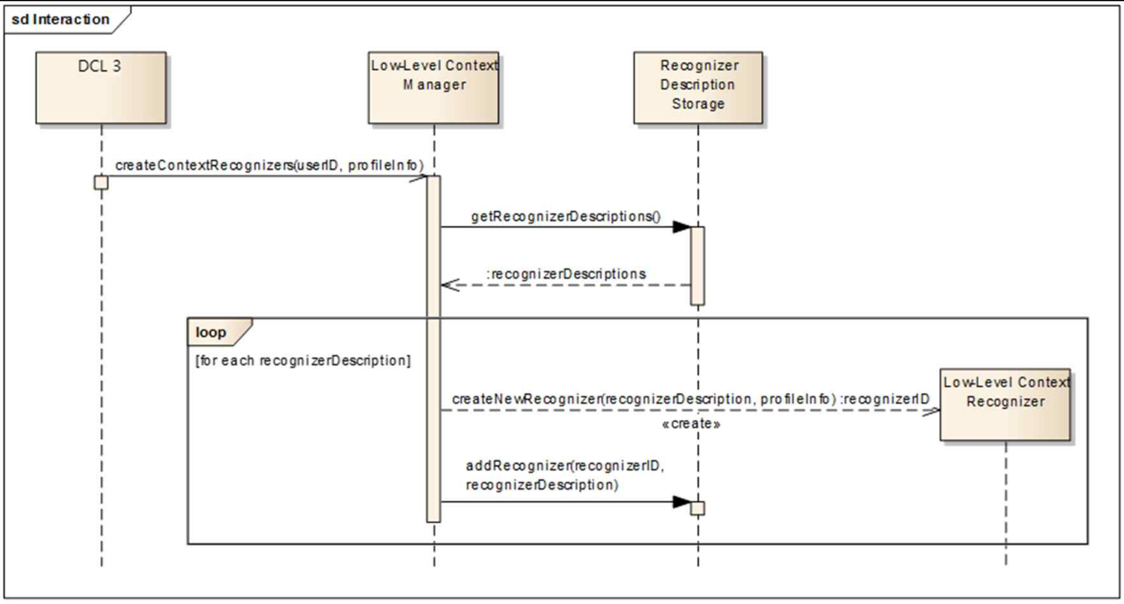
Notes and Issues:	Matlab and Weka tools will be used for this task. A multimodal dataset must be collected for the training and evaluation of the candidates models.

Sequence Diagram:



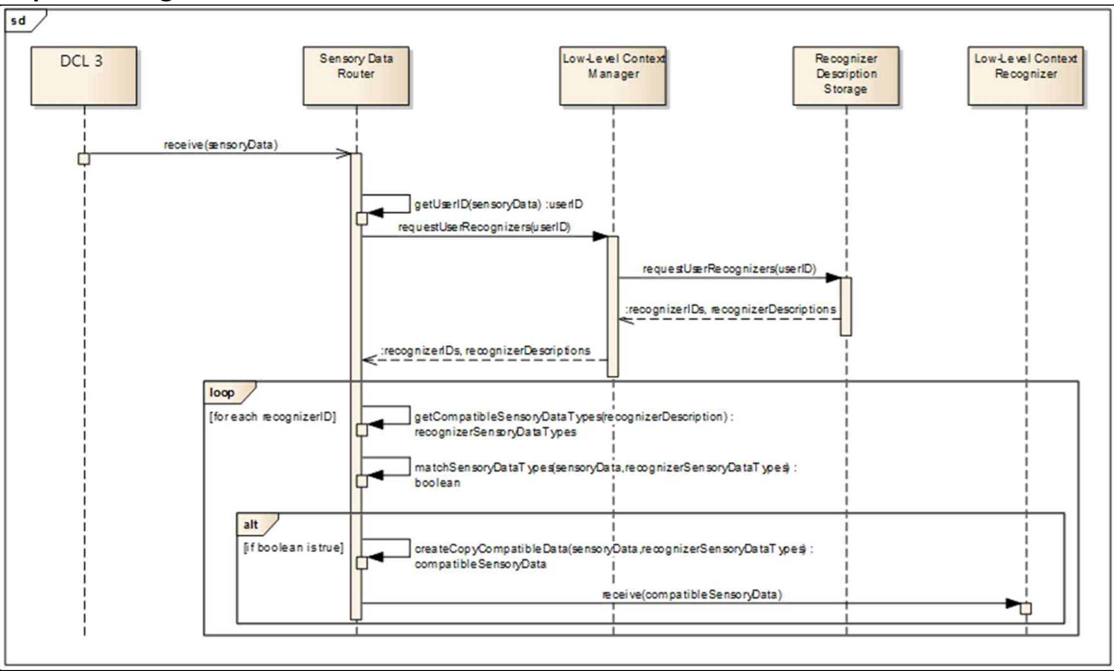
Use Case ID:	**ICL3-SUC-02**		
Use Case Name:	Create low-level context recognizers		
FR ID:	MM-FR-11		
Created By:	Oresti Banos	**Last Updated By:**	Jaehun Bang
Date Created:	22 July 2015	**Last Revision Date:**	9 March 2017
Actors:	DCL 3		
Description:	Sensory data is received from DCL 3 and it is distributed to the corresponding low-level context recognizer based on the data type(s).		
Trigger:	Receive userID and part of the user profile information send by DCL 3 to ICL3		
Pre-conditions:	• DCL 3 sends the userID and part of the user profile information to ICL 3 whenever a new user is registered in the platform		
Post-conditions:	• Low-level context recognizers are generated for the new user for all context types		

Normal Flow:	1. Receive UserID and (part of the) user profile information 2. Load the recognizer descriptions containing the low-level context model types (e.g., emotion recognizer) and characteristics (e.g., median filtering, 3 sec window size, etc.) 3. Create a new recognizer for each recognizer description 4. Create a recognizer identifier for the generated recognizer 5. Save the recognizer identifier in a persistent storage
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Less frequent
NFR ID:	NA
Assumptions:	• DCL 3 will send the required user profile information together with the userID only the first time a user is registered • No user profile updates are considered in this version
Notes and Issues:	NA
Sequence Diagram:	

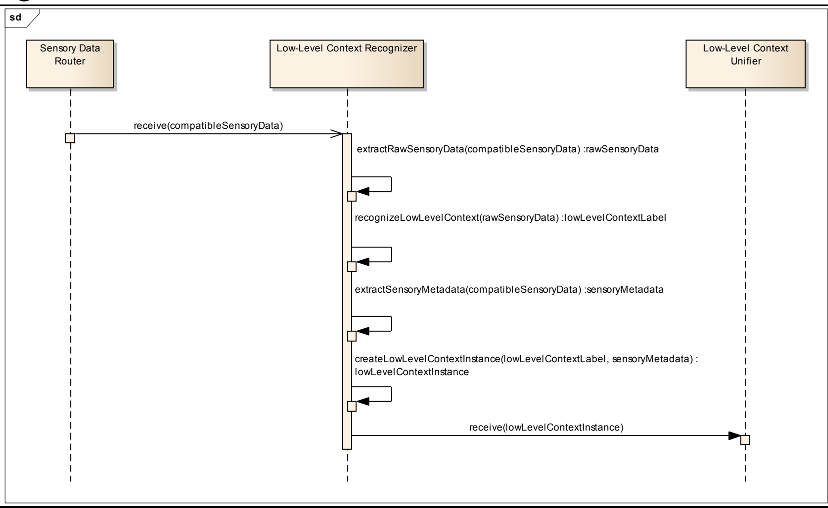


Use Case ID:	**LLCA-UC-03**		
Use Case Name:	Route sensory data for the low-level context identification		
Created By:	Tae Ho Hur	**Last Updated By:**	Tae Ho Hur
Date Created:	06 Dec 2015	**Last Revision Date:**	06 Dec 2015
Actors:	Sensor Devices		
Description:	Sensory data is received and it is distributed to the corresponding low-level context recognizer based on the data type(s).		
Trigger:	Receive sensory data send by each sensor device		

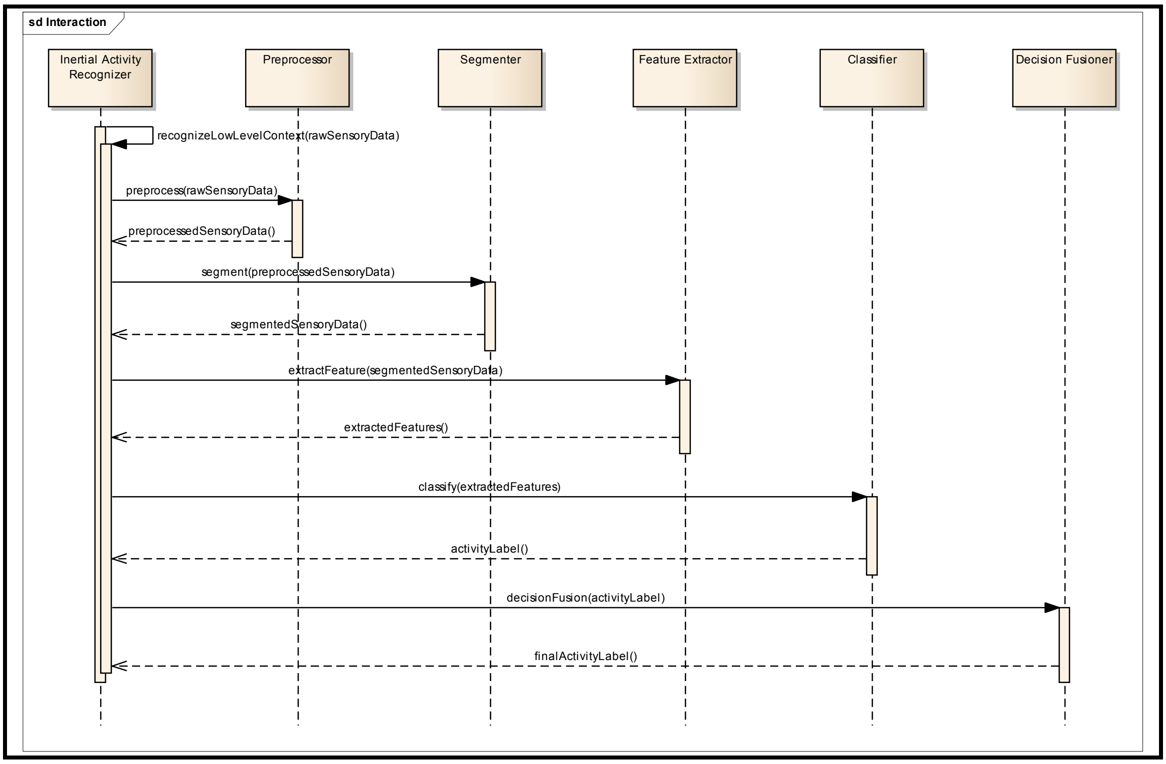
Pre-conditions:	• Device sends sensory data, i.e., raw sensory data plus sensory metadata (e.g., data type, time stamp, device ID, device type, and user ID)
Post-conditions:	• The adequate raw sensory data is sent to each low-level context recognizer in order to perform the recognition process
Normal Flow:	1. Receive sensory data 2. Get the user identifier to which the sensory data belongs 3. Load the low-level context recognizers identifiers for the given user 4. For each low-level context recognizer identifier, get the sensory data type(s) it requires 5. Match the received sensory data with the sensory data type(s) required by the low-level context recognizer 6. Create a copy with the compatible data required by the low-level context recognizer 7. Distribute the data to the corresponding low-level context recognizer
Alternative Flows:	NA
Exceptions:	5a. If no compatible data types are identified for the given low-level context recognizer 1. Go to step 3
Includes:	NA
Frequency of Use:	Very frequent
Assumptions:	• There is an established communication between Sensor Devices and the Sensory Data Router module • Incoming sensory data is already preprocessed (i.e., without missing samples and with synchronized streams)
Notes and Issues:	NA
Sequence Diagram:	



Use Case ID:	ICL3-SUC-04		
Use Case Name:	Recognize user low-level context		
FR ID:	MM-FR-11		
Created By:	Oresti Banos	**Last Updated By:**	Oresti Banos
Date Created:	14 July 2015	**Last Revision Date:**	20 July 2015
Actors:	Sensory Data Router, Low-Level Context Unifier		
Description:	The low-level context associated to a given user is identified based on the received compatible sensory data. The low level context recognition may be of diverse nature depending upon the data types, thus this use case defines an abstract representation of the process to be followed.		
Trigger:	Receive compatible sensory data		
Pre-conditions:	• Compatible sensory data is sent to a given low-level context recognizer		
Post-conditions:	• The recognized low-level context instance is provided to the Low-Level Context Unifier		
Normal Flow:	1. Compatible sensory data is received by a given low-level context recognizer 2. The raw sensory data is extracted from the sensory data 3. The low-level context label is recognized 4. The sensory metadata is extracted from the sensory data 5. A low-level context instance is generated by combining the low-level context label and the sensory metadata 6. The generated low-level context instance is provided to the Low-Level Context Unifier		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Frequent: at every reception of sensory data		
NFR ID:	MM-NFR-05		
Assumptions:	• Only compatible sensory data is received by each corresponding low-level context recognizer		
Notes and Issues:	NA		
Sequence Diagram:			

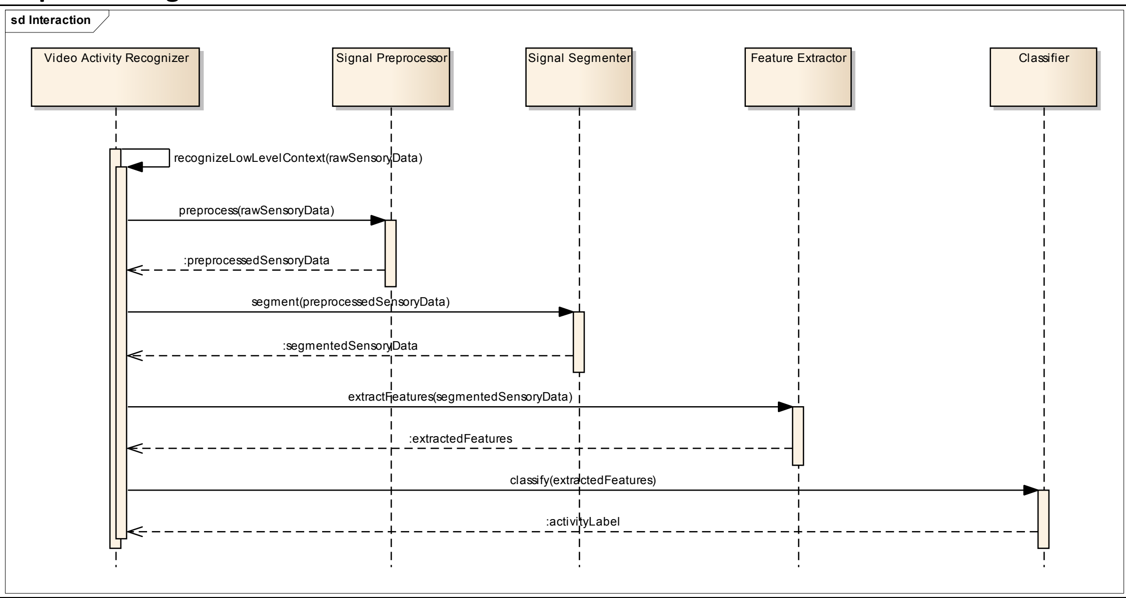


Use Case ID:	ICL3-SUC-05		
Use Case Name:	Recognize user activity based on inertial raw sensory data		
FR ID:	MM-FR-11		
Created By:	Taeho Hur	Last Updated By:	Taeho Hur
Date Created:	09 March 2017	Last Revision Date:	09 March 2017
Actors:	ICL3-SUC-04		
Description:	Identification of the user physical activity (e.g., “sitting”) based on the processing of the body-motion raw sensory data collected from an inertial sensor. The body-motion raw sensory data consists of triaxial acceleration, triaxial rate of turn and triaxial magnetic field data.		
Trigger:	Request for the recognition of the user activity based on a given inertial raw sensory data		
Pre-conditions:	• Raw sensory data is extracted from compatible sensory data (inertial sensory data)		
Post-conditions:	• A label corresponding to the recognized activity is generated		
Normal Flow:	1. Inertial raw sensory data is received for analysis 2. The raw sensory data is preprocessed (e.g., filtered) 3. The preprocessed raw sensory data is segmented (e.g., partitioned into windows) 4. Features (e.g., mean, variance) are extracted from each segment of raw sensory data 5. The extracted features are classified 6. A label identifying the corresponding user activity is generated 7. Decision fusion of activities is made based on majority voting 8. Final activity is chosen		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Frequent: at every reception of inertial raw sensory data		
NFR ID:	MM-NFR-05		
Assumptions:	The raw sensory data is of the nature required by the inertial activity recognizer		
Notes and Issues:	NA		
Sequence Diagram:			



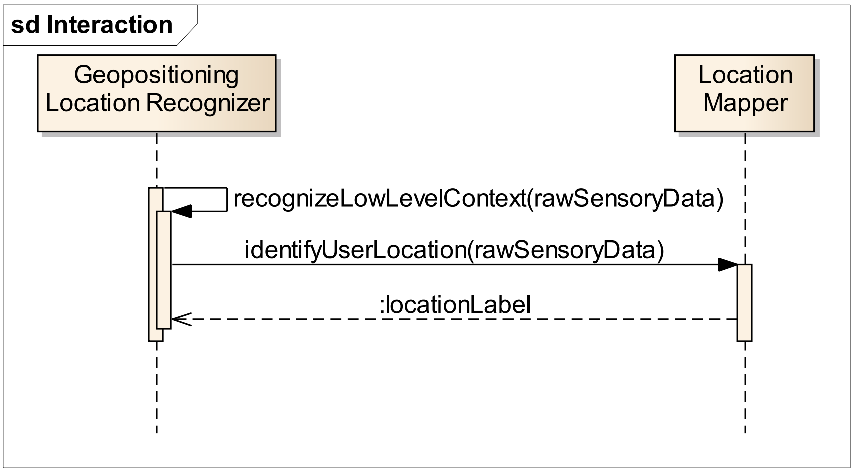
Use Case ID:	**ICL3-SUC-06**		
Use Case Name:	Recognize user activity based on video raw sensory data		
FR ID:	MM-FR-11		
Created By:	Oresti Banos	**Last Updated By:**	Oresti Banos
Date Created:	17 July 2015	**Last Revision Date:**	20 July 2015
Actors:	ICL3-SUC-04		
Description:	Identification of the user physical activity (e.g., “standing”) based on the processing of the body-motion raw sensory data collected through a video camera. The body-motion raw sensory data consists of RGB and depth video.		
Trigger:	Request for the recognition of the user activity based on a given video raw sensory data		
Pre-conditions:	• Raw sensory data is extracted from compatible sensory data (video sensory data)		
Post-conditions:	• A label corresponding to the recognized activity is generated		
Normal Flow:	1. Video raw sensory data is received for analysis 2. The raw sensory data is preprocessed (e.g., filtered) 3. The preprocessed raw sensory data is segmented (e.g., partitioned into windows) 4. Features (e.g., SIFT, HOG) are extracted from each segment of raw sensory data 5. The extracted features are classified		

	6. A label identifying the corresponding user activity is generated
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Frequent: at every reception of video raw sensory data
NFR ID:	MM-NFR-05
Assumptions:	The raw sensory data is of the nature required by the video activity recognizer
Notes and Issues:	NA
Sequence Diagram:	



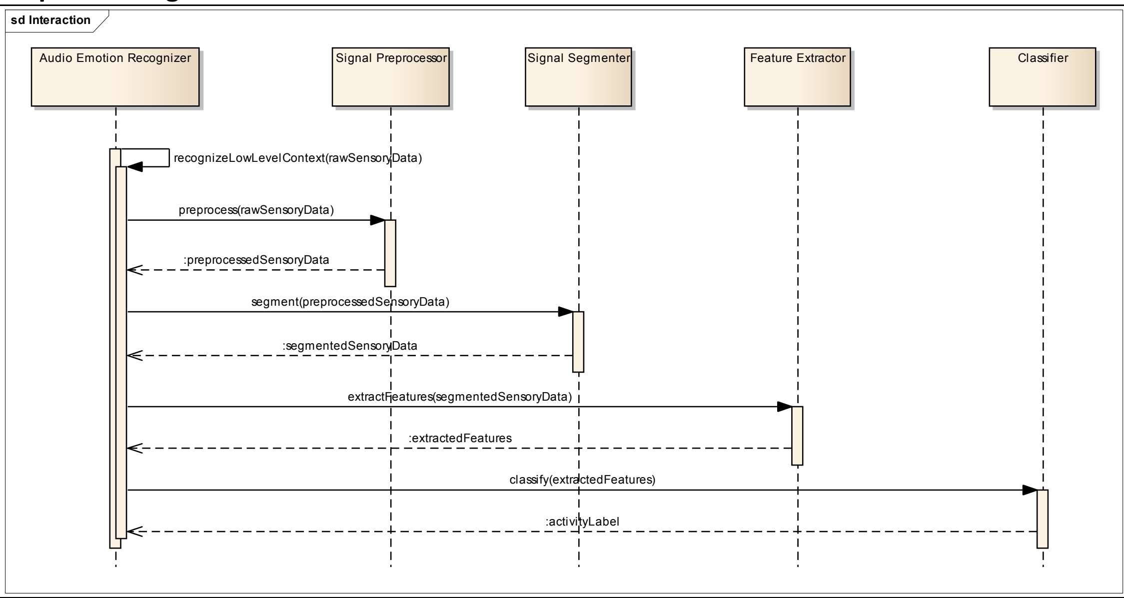
Use Case ID:	**ICL3-SUC-07**		
Use Case Name:	Recognize user location based on geopositioning raw sensory data		
FR ID:	MM-FR-11		
Created By:	Oresti Banos	**Last Updated By:**	Oresti Banos
Date Created:	17 July 2015	**Last Revision Date:**	20 July 2015
Actors:	ICL3-SUC-04		
Description:	Identification of the user location (e.g., “restaurant”) based on the processing of the geopositioning raw sensory data collected from a portable GPS sensor. The body-motion raw sensory data consists of latitude, longitude and speed data.		
Trigger:	Request for the recognition of the user location based on a given geopositioning raw sensory data		

Pre-conditions:	• Raw sensory data is extracted from compatible sensory data (geopositioning sensory data)
Post-conditions:	• A label corresponding to the recognized location is generated
Normal Flow:	1. Geopositioning raw sensory data is received for analysis 2. The geopositioning raw sensory data is compared with the predefined map coordinates 3. A label identifying the corresponding user location is generated
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Frequent: at every reception of geopositioning raw sensory data
NFR ID:	MM-NFR-05
Assumptions:	The raw sensory data is of the nature required by the geopositioning location recognizer
Notes and Issues:	NA
Sequence Diagram:	



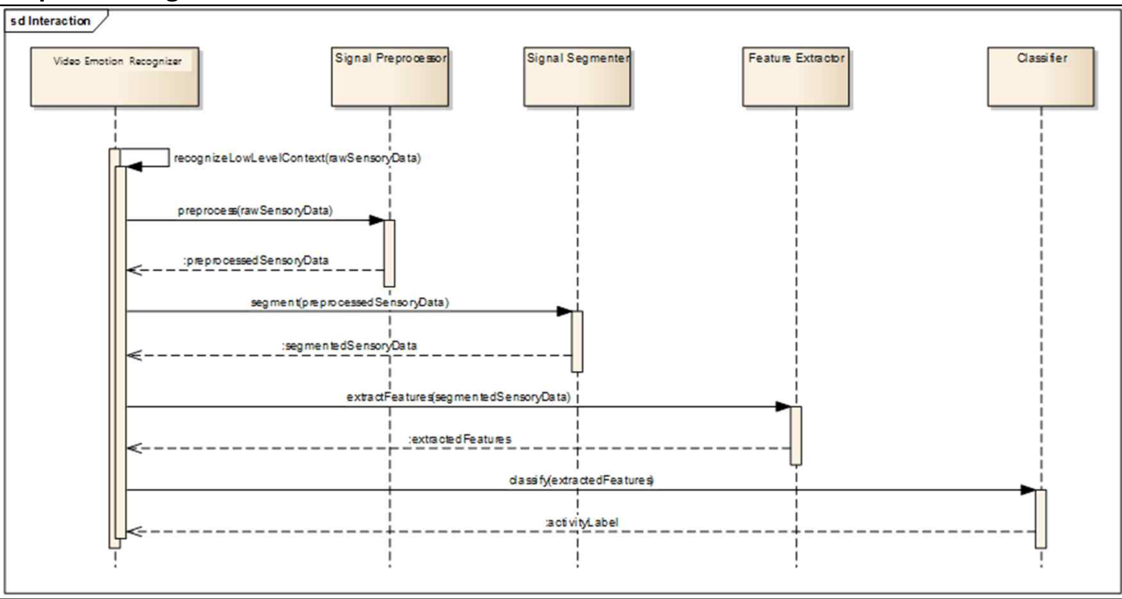
Use Case ID:	**ICL3-SUC-08**		
Use Case Name:	Recognize user emotion based on audio raw sensory data		
FR ID:	MM-FR-11		
Created By:	Oresti Banos	**Last Updated By:**	Jaehun Bang
Date Created:	17 July 2015	**Last Revision Date:**	9 March 2017
Actors:	ICL3-SUC-04		
Description:	Identification of the user emotional state (e.g., “happy”) based on the processing of the audio raw sensory data collected from a microphone sensor. The audio raw sensory data consists of the user voice data.		
Trigger:	Request for the recognition of the user emotion based on a given audio raw sensory data		

Pre-conditions:	• Raw sensory data is extracted from compatible sensory data (audio sensory data)
Post-conditions:	• A label corresponding to the recognized emotion is generated
Normal Flow:	1. Audio raw sensory data is received for analysis 2. The raw sensory data is preprocessed (e.g., filtered) 3. The preprocessed raw sensory data is segmented (e.g., partitioned into windows) 4. Features (e.g., LPC, MFCC) are extracted from each segment of raw sensory data 5. The extracted features are classified 6. A label identifying the corresponding user emotion is generated
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Frequent: at every reception of inertial raw sensory data
NFR ID:	MM-NFR-05
Assumptions:	The raw sensory data is of the nature required by the audio emotion recognizer
Notes and Issues:	NA
Sequence Diagram:	



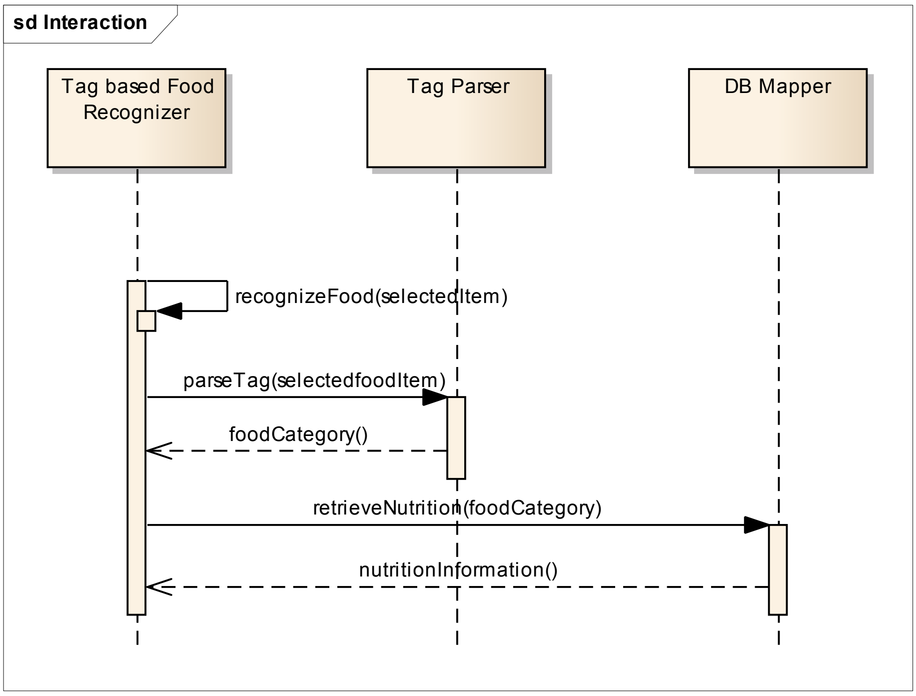
Use Case ID:	**ICL3-SUC-09**		
Use Case Name:	Recognize user emotion based on video raw sensory data		
FR ID:	MM-FR-11		
Created By:	Jaehun Bang	**Last Updated By:**	Jaehun Bang
Date Created:	9 March 2017	**Last Revision Date:**	9 March 2017
Actors:	ICL3-SUC-04		

Description:	Identification of the user emotional state (e.g., "happy") based on the processing of the video raw sensory data collected from a camera sensor. The video raw sensory data consists of the user facial data.
Trigger:	Request for the recognition of the user emotion based on a given video raw sensory data
Pre-conditions:	• Raw sensory data is extracted from compatible sensory data (video sensory data)
Post-conditions:	• A label corresponding to the recognized emotion is generated
Normal Flow:	1. Video raw sensory data is received for analysis 2. The raw sensory data is preprocessed (e.g., filtered and face detection) 3. The preprocessed raw sensory data is segmented (e.g., partitioned into face area) 4. Features (e.g., face factor statistical features) are extracted from each segment of raw sensory data 5. The extracted features are classified 6. A label identifying the corresponding user emotion is generated
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Frequent: at every reception of inertial raw sensory data
NFR ID:	MM-NFR-05
Assumptions:	The raw sensory data is of the nature required by the video emotion recognizer
Notes and Issues:	NA
Sequence Diagram:	

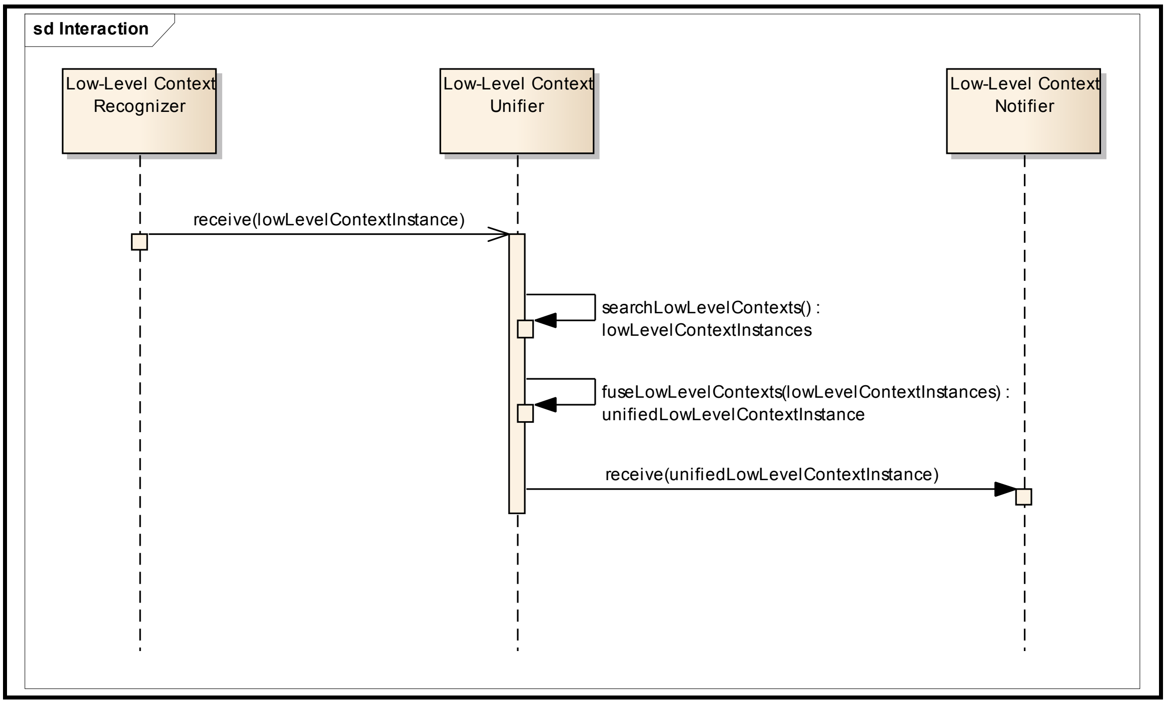


Document ID: MM-Deliverables-01

Use Case ID:	**ICL3-SUC-10**		
Use Case Name:	Recognize food based on tag data		
FR ID:	MM-FR-11		
Created By:	Taeho Hur	**Last Updated By:**	Taeho Hur
Date Created:	9 March 2017	**Last Revision Date:**	9 March 2017
Actors:	ICL3-SUC-04		
Description:	User selects the category of the intaken food (e.g., "grain") from the food DB list from the app. The related nutrition information (e.g, "protein") shall be sent to the HLCA.		
Trigger:	User selects the food category tag		
Pre-conditions:	• User is going to have a meal/snack		
Post-conditions:	• Nutrition information is retrieved based on the selected food		
Normal Flow:	1. User selects the food to intake from the app 2. The selected food item is parsed and conclude the food category 3. The category is mapped to the food DB 4. According nutrition information is retrieved		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Episodic: Only when the user has a meal		
NFR ID:	MM-NFR-05		
Assumptions:	User is having a meal		
Notes and Issues:	NA		
Sequence Diagram:			

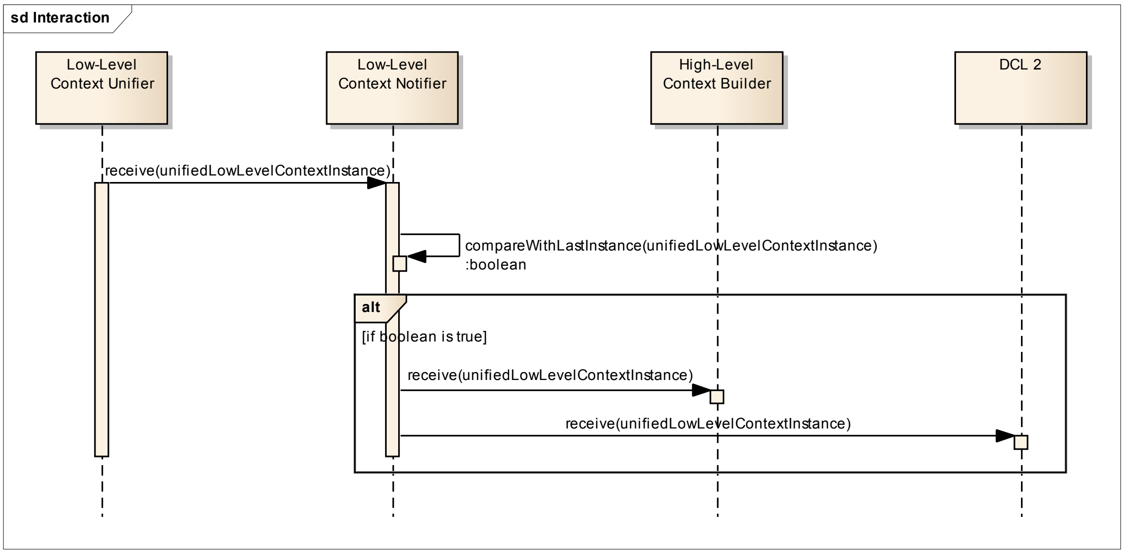


Use Case ID:	**ICL3-SUC-11**		
Use Case Name:	Unify low-level contexts		
FR ID:	MM-FR-11		
Created By:	Oresti Banos	**Last Updated By:**	Oresti Banos
Date Created:	20 July 2015	**Last Revision Date:**	20 July 2015
Actors:	Low-Level Context Recognizer, Low-Level Context Notifier		
Description:	Aggregation of multiple low-level context instances of the same context type (e.g., activity) corresponding to a similar period of time		
Trigger:	Receive low-level context instance		
Pre-conditions:	• Low-level context instances are received from different recognizers of the same context type		
Post-conditions:	• A single low-level context instance is served for notification		
Normal Flow:	1. A low-level context instance is received 2. Search for other low-level context instances of the same type valid at the same time 3. Fuse the identified low-level context instances into a unified low-level context instance 4. Serve the unified low-level context instance for notification		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Frequent: at every reception of a low-level context label		
NFR ID:	MM-NFR-05		
Assumptions:	Identical labels are used to describe the same low-level context for each recognizer of the same context type (e.g., inertial activity recognizer, video activity recognizer)		
Notes and Issues:	NA		
Sequence Diagram:			



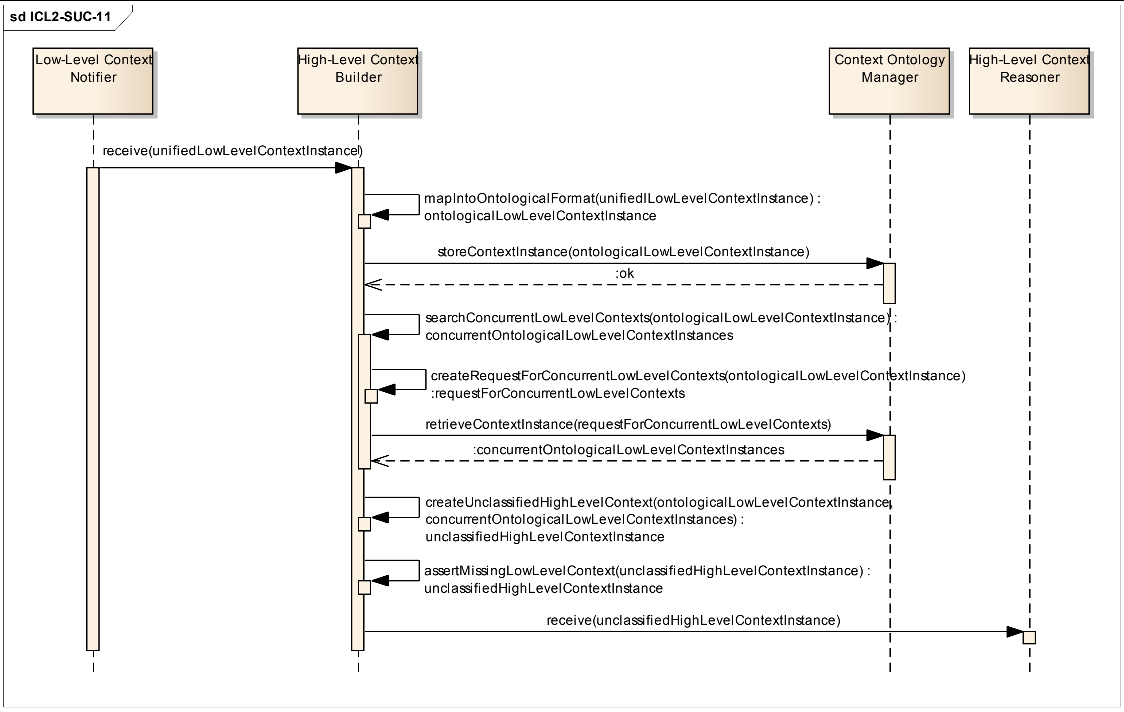
Use Case ID:	**ICL3-SUC-12**		
Use Case Name:	Notify new low-level context		
FR ID:	MM-FR-13		
Created By:	Oresti Banos	**Last Updated By:**	Oresti Banos
Date Created:	14 July 2015	**Last Revision Date:**	20 July 2015
Actors:	Low-Level Context Unifier, High-Level Context Builder, DCL 3		
Description:	Serve the newly recognized low-level context for the identification of high-level context and also communicate it to DCL 3 for persistence.		
Trigger:	New low-level context is identified		
Pre-conditions:	• A unified low-level context instance is received		
Post-conditions:	• The unified low-level context instance is served for the identification of the high-level context(s) • The unified low-level context instance is sent to DCL 3		
Normal Flow:	1. A low-level context instance is received from the low level context unifier 2. The received instance is compared with the last low-level context instance 3. The new low-level context instance is served for the identification of the high-level context 4. The new low-level context instance is sent to DCL 3		
Alternative Flows:	3a. If the received instance contains the same low-level context type as the previous one 1. Finalize		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Frequent: at every reception of a low-level context instance		

NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	



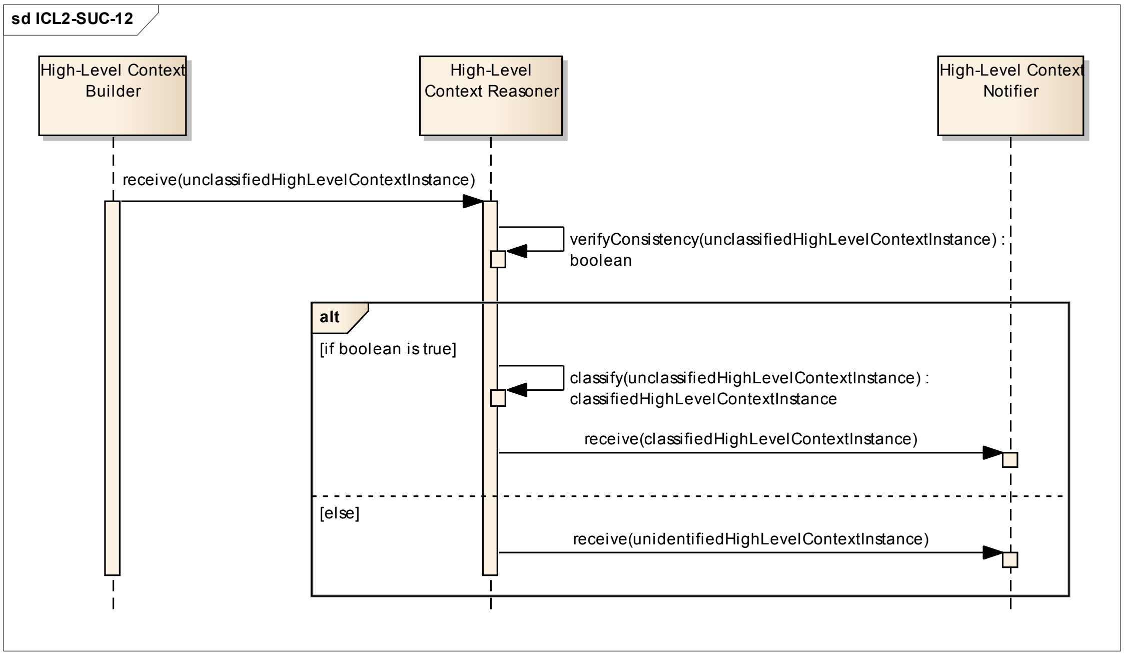
Use Case ID:	**ICL3-SUC-13**		
Use Case Name:	Create unclassified high-level context instance		
FR ID:	MM-FR-12		
Created By:	Claudia Villalonga	**Last Updated By:**	Claudia Villalonga
Date Created:	14 July 2015	**Last Revision Date:**	3 September 2015
Actors:	Low-Level Context Notifier, Context Ontology Manager, High-Level Context Reasoner		
Description:	Build a high-level context instance based on the identified low-level contexts		
Trigger:	Receive low-level context instance (label plus metadata)		
Pre-conditions:	• A new low-level context instance is served to the high-level context builder		
Post-conditions:	• The unclassified high-level context instance is created		
Normal Flow:	1. Map low-level context instance into ontological format 2. Store low-level context instance (ICL3-SUC-17) 3. Search for other low-level context instances of different type valid at the same time 3.1. Create a request for other low-level context instances of different type that are valid at the same time 3.2. Retrieve matching context instances (ICL3-SUC-18) 4. Create new unclassified high-level context instance which links to the available low-level context instance(s)		

	5. Assert on the unclassified high-level context instance that the missing low-level context instances do not exist 6. Serve the unclassified high-level context instance for classification
Alternative Flows:	NA
Exceptions:	NA
Includes:	NA
Frequency of Use:	Less Frequent: whenever a new low-level context is recognized
NFR ID:	NA
Assumptions:	Low-level contexts are interpretable
Notes and Issues:	NA
Sequence Diagram:	



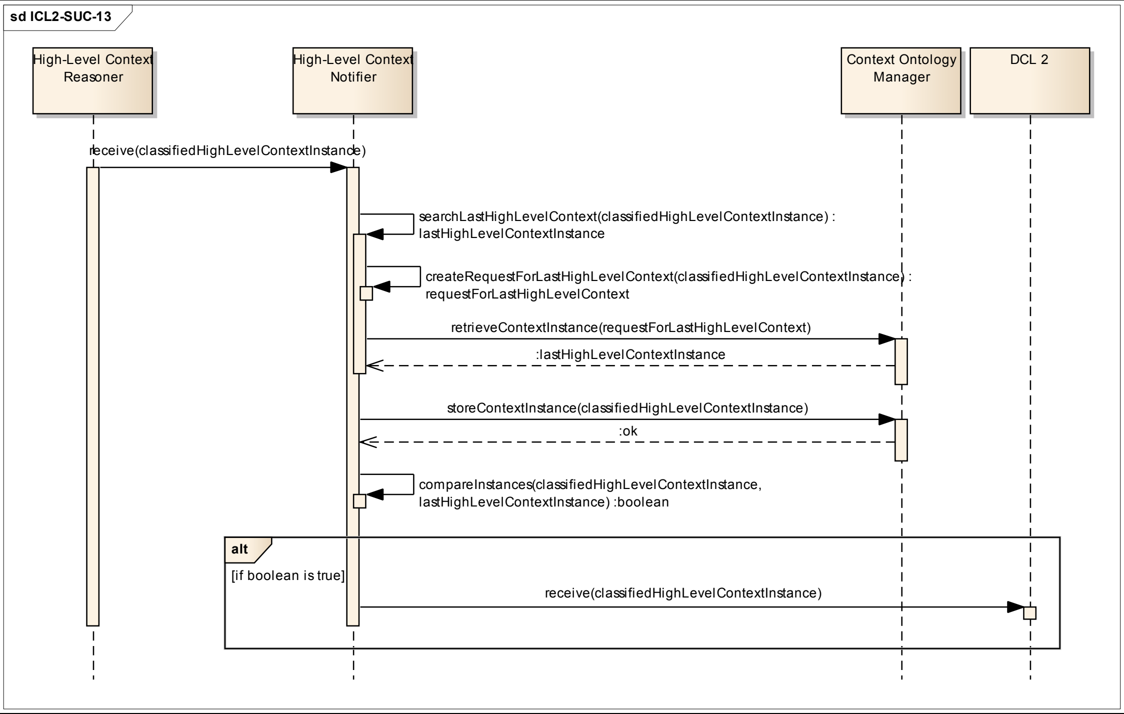
Use Case ID:	**ICL3-SUC-14**		
Use Case Name:	Classify high-level context instance		
FR ID:	MM-FR-12		
Created By:	Claudia Villalonga	**Last Updated By:**	Claudia Villalonga
Date Created:	14 July 2015	**Last Revision Date:**	3 September 2015
Actors:	High-Level Context Builder, High-Level Context Notifier		
Description:	Classify the unclassified high-level context instance into one of the high-level context categories		
Trigger:	Creation of unclassified high-level context instance		
Pre-conditions:	• The unclassified high-level context instance is created		

Post-conditions:	• The classified high-level context instance is served for notification
Normal Flow:	1. Verify the consistency of unclassified high-level context instance 2. Reason on the unclassified high-level context instance to identify the context type to which it belongs 3. Serve the classified high-level context for notification
Alternative Flows:	1a. If the unclassified high-level context instance is not valid 1. Communicate unidentified context
Exceptions:	NA
Includes:	NA
Frequency of Use:	Less Frequent: whenever an unclassified high-level context instance is created
NFR ID:	MM-NFR-06
Assumptions:	Low-level contexts and high-level contexts are interpretable
Notes and Issues:	NA
Sequence Diagram:	

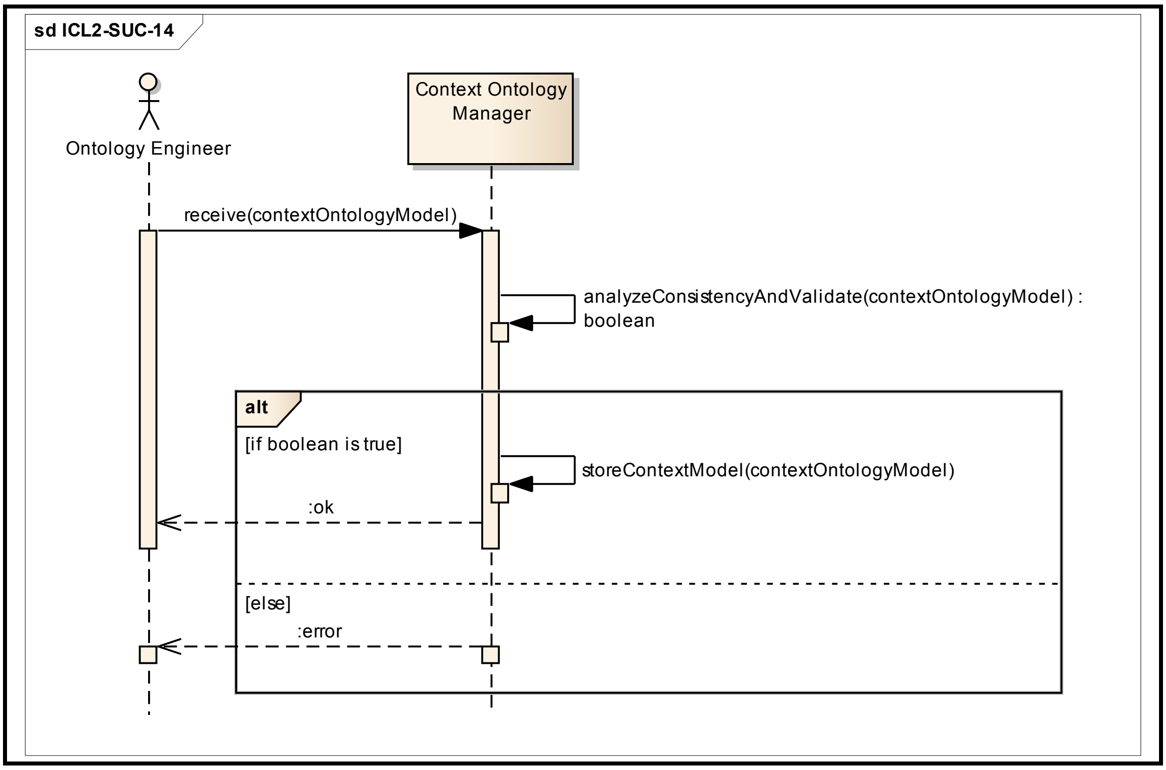


Use Case ID:	**ICL3-SUC-15**		
Use Case Name:	Notify new high-level context		
FR ID:	MM-FR-14		
Created By:	Oresti Banos	**Last Updated By:**	Claudia Villalonga
Date Created:	14 July 2015	**Last Revision Date:**	3 September 2015
Actors:	High-Level Context Reasoner, DCL 3		

Description:	Communicate the newly recognized high-level context to DCL 3 for persistence.
Trigger:	High-level context is identified
Pre-conditions:	• A high-level context instance is received
Post-conditions:	• The new high-level context instance is sent to DCL 3
Normal Flow:	1. A high-level context instance is received from the high-level context classifier 2. Search for the last high-level context instance 2.1. Create a request for the last high-level context instance 2.2. Retrieve the matching context instance (ICL3-SUC-18) 3. Store high-level context instance (ICL3-SUC-17) 4. Compare the high-level context instance with the last high-level context instance 5. The new high-level context instance is sent to DCL 3
Alternative Flows:	5a. If the received instance contains the same high-level context type as the previous one 1. Finalize
Exceptions:	NA
Includes:	NA
Frequency of Use:	Less frequent: at every reception of a high-level context instance
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	

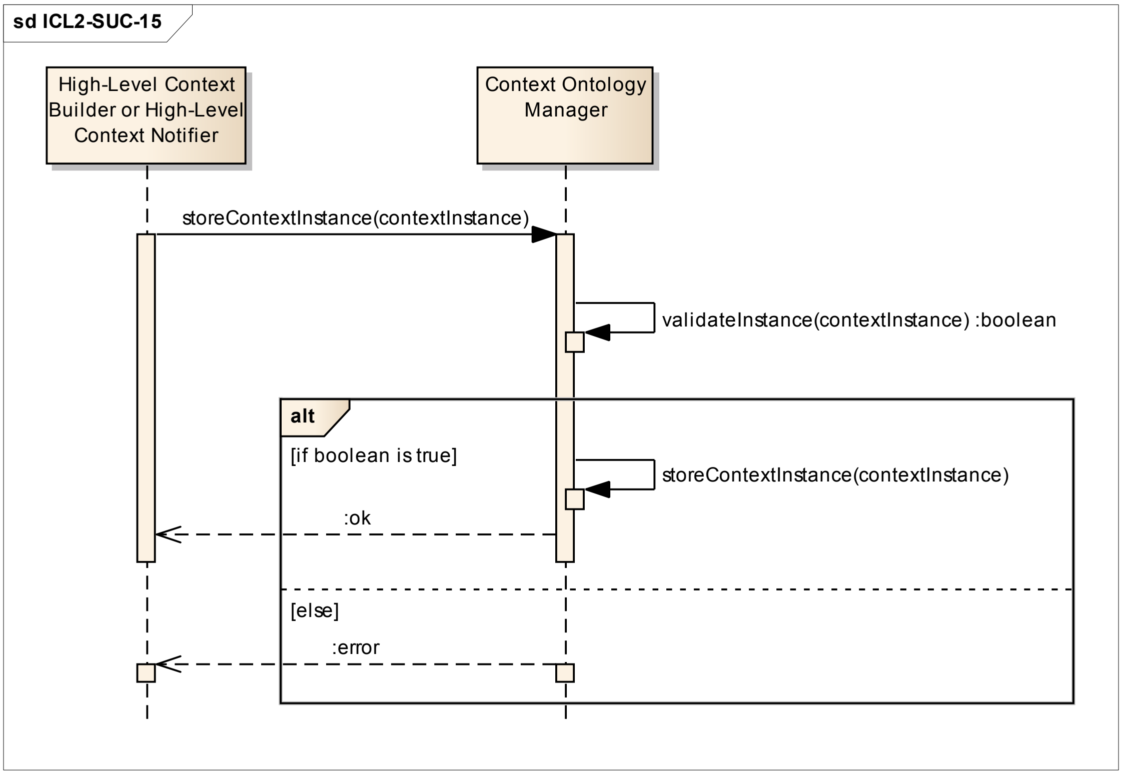


Use Case ID:	ICL3-SUC-16		
Use Case Name:	Load context ontology model		
FR ID:			
Created By:	Claudia Villalonga	Last Updated By:	Claudia Villalonga
Date Created:	31 August 2015	Last Revision Date:	3 September 2015
Actors:	Ontology Engineer		
Description:	Load and store a context ontology model that describes high-level context and its relations to low-level context into the system in order to enable the recognition of high-level context		
Trigger:	The ontology engineer who has created a context ontology model loads it through the interface		
Pre-conditions:	• A context ontology model that describes high-level context and its relations to low-level context has been created		
Post-conditions:	• The context ontology model is stored and available for the recognition of high-level context		
Normal Flow:	1. A context ontology model is received 2. The context ontology model is analyzed for its consistency and validity 3. The context ontology is stored in order to provide persistence 4. Success is notified		
Alternative Flows:	2a. If the context ontology model is not valid or inconsistent 1. Error is notified 3a. If there is an error during storage 1. Repeat step 3		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Infrequent		
NFR ID:	NA		
Assumptions:	NA		
Notes and Issues:	NA		
Sequence Diagram:			



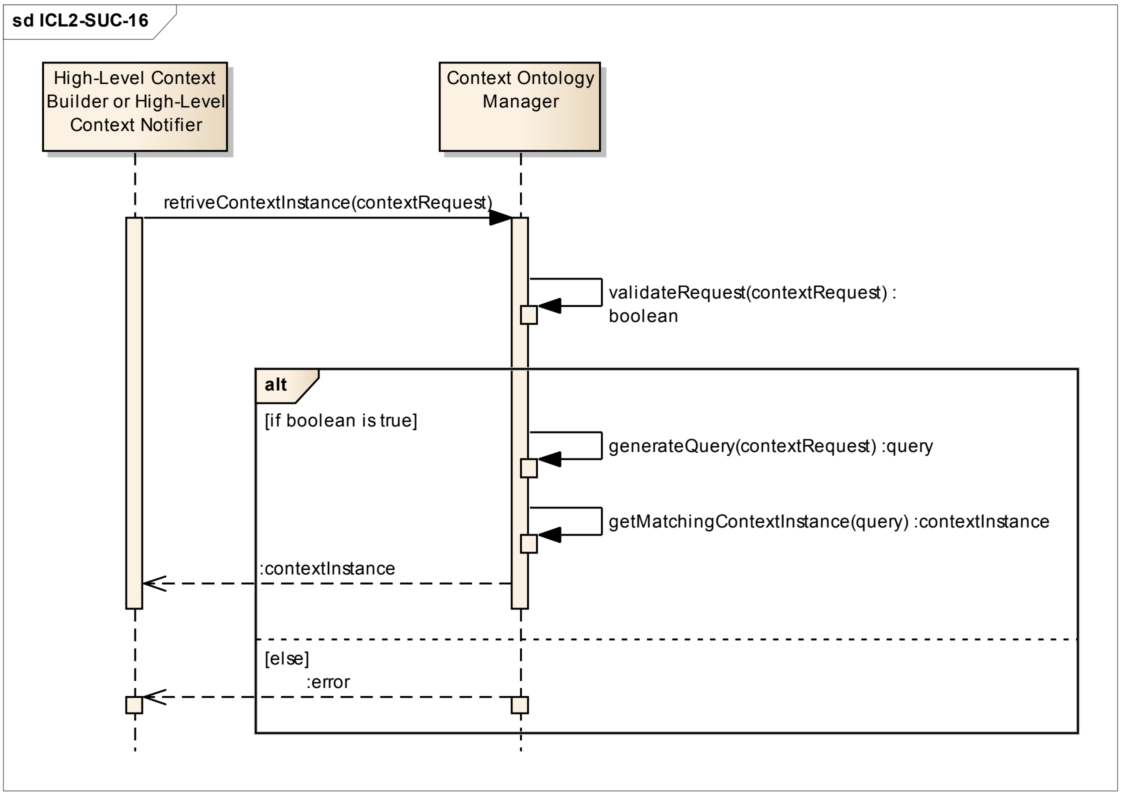
Use Case ID:	ICL3-SUC-17		
Use Case Name:	Store context instance		
FR ID:			
Created By:	Claudia Villalonga	Last Updated By:	Claudia Villalonga
Date Created:	31 August 2015	Last Revision Date:	4 September 2015
Actors:	High-Level Context Builder and High-Level Context Notifier		
Description:	Persist a context instance (high-level context instance or low-level context instance) into the system		
Trigger:	A new context instance has been created or identified		
Pre-conditions:	• A new context instance is received		
Post-conditions:	• The context instance is stored		
Normal Flow:	1. Receive context instance 2. Validate the context instance 3. Store the context instance 4. Notify success		
Alternative Flows:	2a. If the context instance is not valid 1. Error is notified 3a. If there is an error during storage 1. Repeat step 3		
Exceptions:	NA		
Includes:	NA		

Frequency of Use:	Less Frequent: whenever a new low-level context instance is created, a new high-level context instance is created, or a high-level context instance is classified
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	



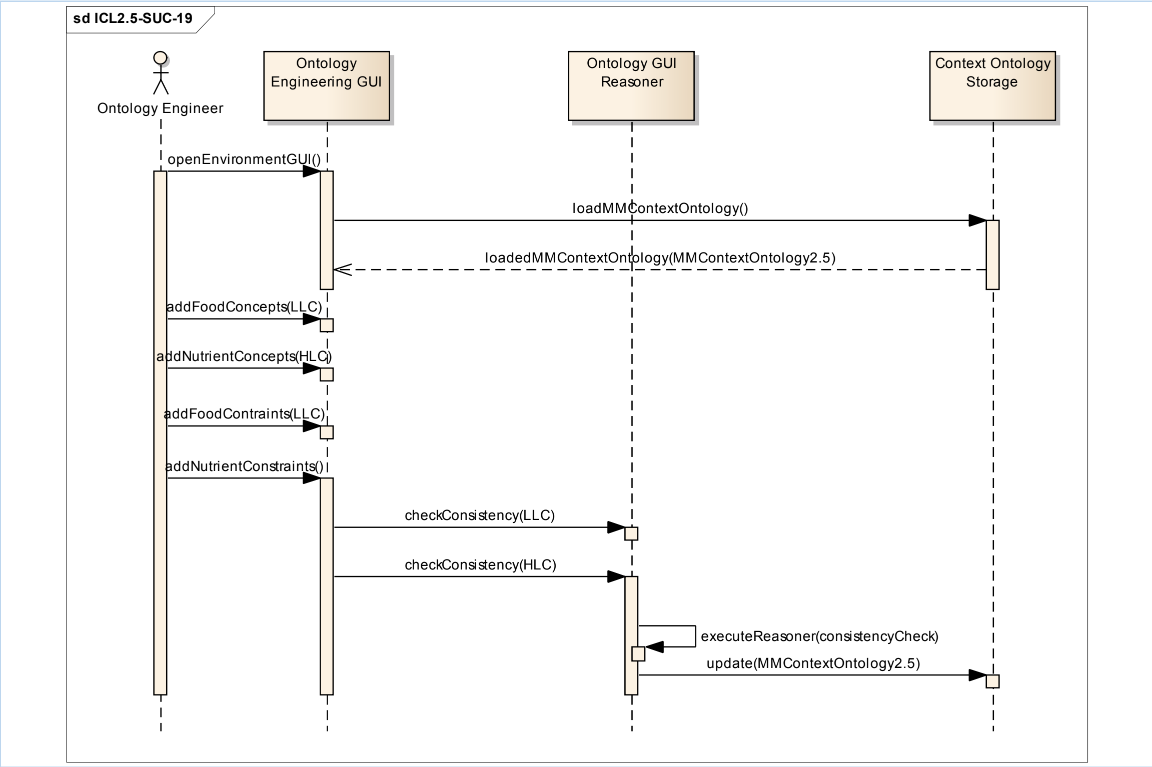
Use Case ID:	**ICL3-SUC-18**		
Use Case Name:	Retrieve context instance		
FR ID:			
Created By:	Claudia Villalonga	**Last Updated By:**	Claudia Villalonga
Date Created:	31 August 2015	**Last Revision Date:**	4 September 2015
Actors:	High-Level Context Builder and High-Level Context Notifier		
Description:	Provide context instances (high-level context instances or low-level context instances) that match a given request		
Trigger:	A requester retrieves context instances		
Pre-conditions:	• A request for context instances is received		
Post-conditions:	• Matching context instances are provided to the requester		
Normal Flow:	1. Receive a request for a specific context 2. Validate the request 3. Generate the query associated to the request		

	4. Match the query to the stored context instances 5. Return the matching context instances
Alternative Flows:	2a. If the request for context is not valid 1. Return error message
Exceptions:	NA
Includes:	NA
Frequency of Use:	Less frequent: whenever context instances are required in order to generate a new high-level context instance or to verify the high-level context
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	



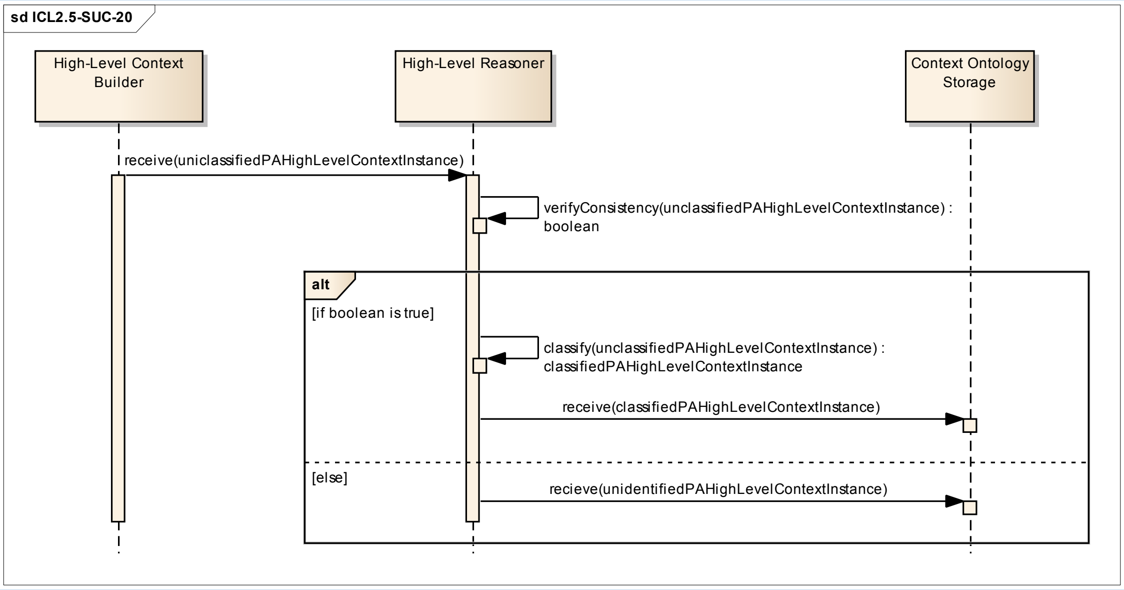
Document ID: MM-Deliverables-01

Use Case ID:	*ICL2.5-SUC-19*		
Use Case Name:	Evolve Ontology		
FR ID:	MM-FR-11		
Created By:	Wajahat Ali Khan	**Last Updated By:**	Muhammad Asif Razzaq
Date Created:	18 Mar 2016	**Last Revision Date:**	10 Mar 2017
Actors:	Ontology Engineer		
Description:	Evolving the already created context ontology based on the clinical service. The low level and high level clinical related contexts are modelled in the context ontology that only included physical activities and nutrition related low level and high level entities.		
Trigger:	The ontology engineer who has created a context ontology model loads it through the interface		
Pre-conditions:	• A context ontology model that describes high-level context and its relations to low-level context has been created with physical activities, nutrition and clinical information related resources		
Post-conditions:	• The modified or evolved context ontology model is stored and available for the recognition of high-level context		
Normal Flow:	1. Ontology engineer defines the low level clinical concepts in addition to the low level physical activities, and nutrition concepts 2. Ontology engineer defines the high level clinical concepts in addition to the high level physical activities and nutrition concepts 3. The constraints are defined on the new concepts added to the ontology 4. Consistency of the ontology is checked with reasoner 5. The modified context ontology is stored for persistence 6. Success is notified		
Alternative Flows:	4a. If the context ontology model is not valid or inconsistent a. Error is notified b. Check step 1,2,3		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Infrequent		
NFR ID:	NA		
Assumptions:	Context Ontology based on physical activities and nutrition service already exists		
Notes and Issues:	NA		
Sequence Diagram:			



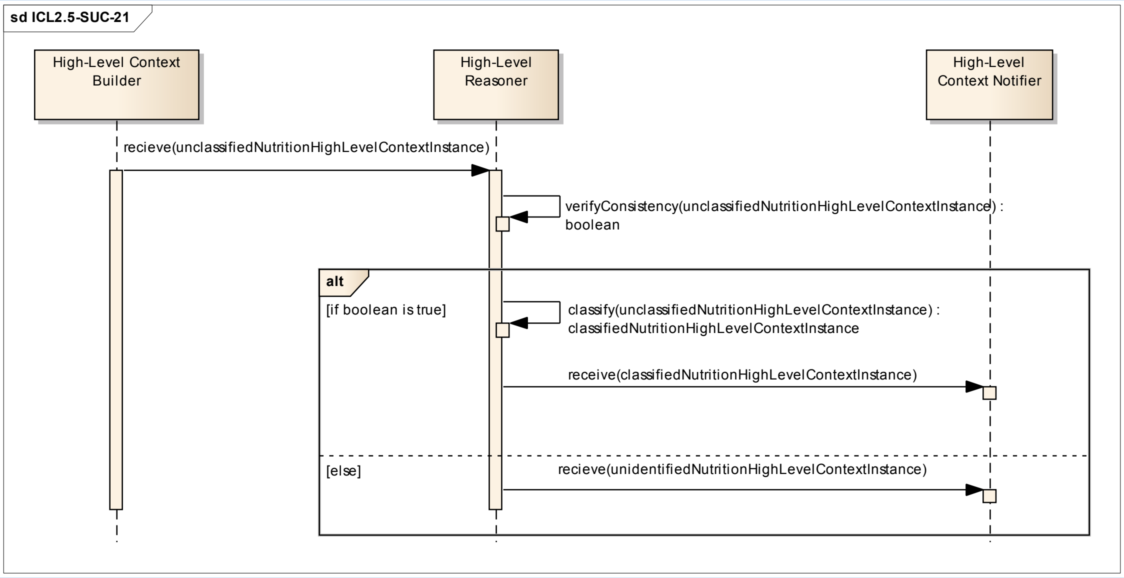
Use Case ID:	*ICL2.5-SUC-20*		
Use Case Name:	High level physical activity context		
FR ID:	MM-FR-14		
Created By:	Wajahat Ali Khan	**Last Updated By:**	Wajahat Ali Khan
Date Created:	18 March 2016	**Last Revision Date:**	18 March 2016
Actors:	High-Level Context Reasoner		
Description:	A classified high level physical activity context is recognized by the reasoner from the unclassified high level context.		
Trigger:	High-level context is identified		
Pre-conditions:	• An unclassified high-level physical activity context instance is received		
Post-conditions:	• The new high-level physical activity context instance is forwarded to Notify DCL		
Normal Flow:	1. A high-level physical activity context instance is processed by high-level context classifier 2. Reasoning is performed by the reasoner to find out classified high level physical activity context 3. The classified high level physical activity context is provided to Notify DCL		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		
Frequency of Use:	Less frequent: at every reception of a unclassified high-level		

	physical activity context instance
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	



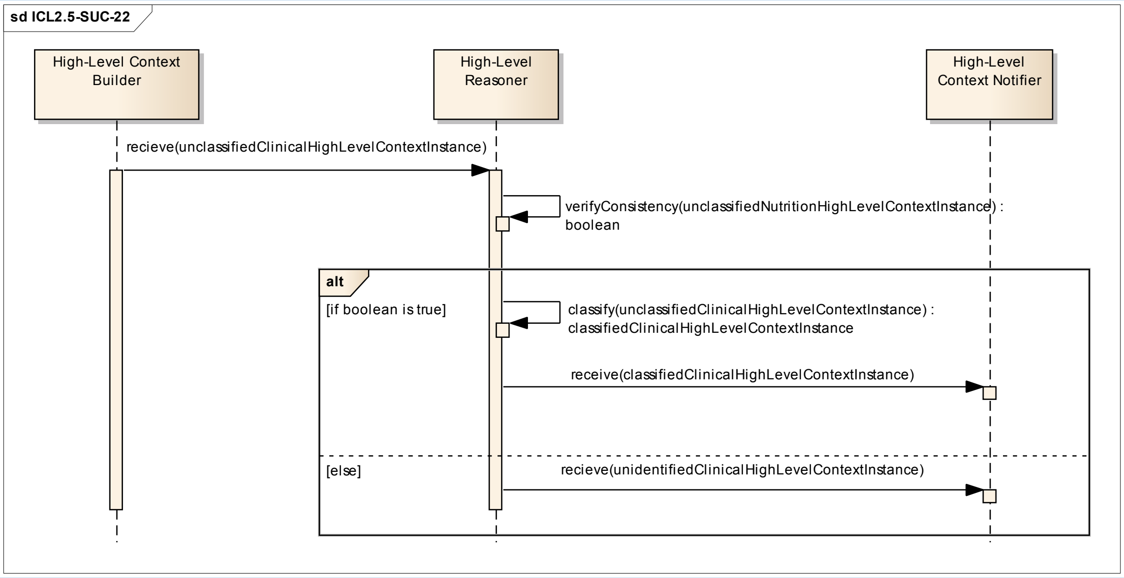
Use Case ID:	*ICL2.5-SUC-21*		
Use Case Name:	High level nutrition context		
FR ID:	MM-FR-14		
Created By:	Wajahat Ali Khan	**Last Updated By:**	Wajahat Ali K han
Date Created:	18 March 2016	**Last Revision Date:**	18 March 2016
Actors:	High-Level Context Reasoner		
Description:	A classified high level nutrition context is recognized by the reasoner from the unclassified high level context.		
Trigger:	High-level context is identified		
Pre-conditions:	• An unclassified high-level nutrition context instance is received		
Post-conditions:	• The new high-level nutrition context instance is forwarded to Notify DCL		
Normal Flow:	1. A high-level nutrition context instance is processed by high-level context classifier 2. Reasoning is performed by the reasoner to find out classified high level nutrition context 3. The classified high level nutrition context is provided to Notify DCL		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		

Frequency of Use:	Less frequent: at every reception of a unclassified high-level nutrition context instance
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	



Use Case ID:	*ICL3.0-SUC-22*		
Use Case Name:	High level clinical context		
FR ID:	MM-FR-14		
Created By:	Muhammad Asif Razzaq	**Last Updated By:**	Muhammad Asif Razzaq
Date Created:	10 March 2017	**Last Revision Date:**	10 March 2017
Actors:	High-Level Context Reasoner		
Description:	A classified high level clinical context is recognized by the reasoner from the unclassified high level context.		
Trigger:	High-level context is identified		
Pre-conditions:	• An unclassified high-level clinical context instance is received		
Post-conditions:	• The new high-level clinical context instance is forwarded to Notify DCL		
Normal Flow:	1. A high-level clinical context instance is processed by high-level context classifier 2. Reasoning is performed by the reasoner to find out classified high level clinical context 3. The classified high level clinical context is provided to Notify DCL		
Alternative Flows:	NA		
Exceptions:	NA		
Includes:	NA		

Frequency of Use:	Less frequent: at every reception of a unclassified high-level clinical context instance
NFR ID:	NA
Assumptions:	NA
Notes and Issues:	NA
Sequence Diagram:	



Section 5.3

Knowledge Curation Layer(KCL) Requirement Specification

Knowledge Curation Layer

1. KCL Requirement Specification

1.1. KCL Function Requirements

Requirements #ID	Description
KCL-FR-01	The data-driven knowledge acquisition shall know schema detail of lifelog and user profile data in order to load the data and extract feature model.
KCL-FR-02	The data-driven knowledge acquisition shall load all related data specified as feature model from Data Curation Layer (DCL). The loaded lifelog and user profile data will be used for classification model creation.
KCL-FR-03	The expert-driven knowledge acquisition shall share the production rules with Service Curation Layer (SCL) for final reasoning.
KCL-FR-04	The expert-driven environment shall create “Situations” and share its configuration with SCL and DCL. Furthermore, all associated rules with “Situation” shall be provided to SCL.

1.2. KCL Non-Function Requirements

Requirements #ID	Description
KCL-NFR-01	The layer shall persist only verified and validated rules into knowledge base.
KCL-NFR-02	The layer shall ensure consistency of distributed copies of knowledge base.

1.3. KCL Terms and Definitions

Terms	Description
KCL	Knowledge Curation Layer Ver. 3
DCL	Data Curation Layer Ver. 3
SCL	Service Curation Layer Ver. 3
Situation	An abnormal status of a subject caused by unhealthy behaviors.
Production Rule	Production rule is ultimate and shareable rule which is used in reasoning to produce recommendation.
Domain Expert	Domain expert is an actor who will interact with system to create knowledge base.
Verification and Validation	Verification ensures that rule created is consistent with requirements

	and validation ensures that the rule created is correctly working on real data.
Schema	Schema represents the structure and associated semantics of user profile and life log data.

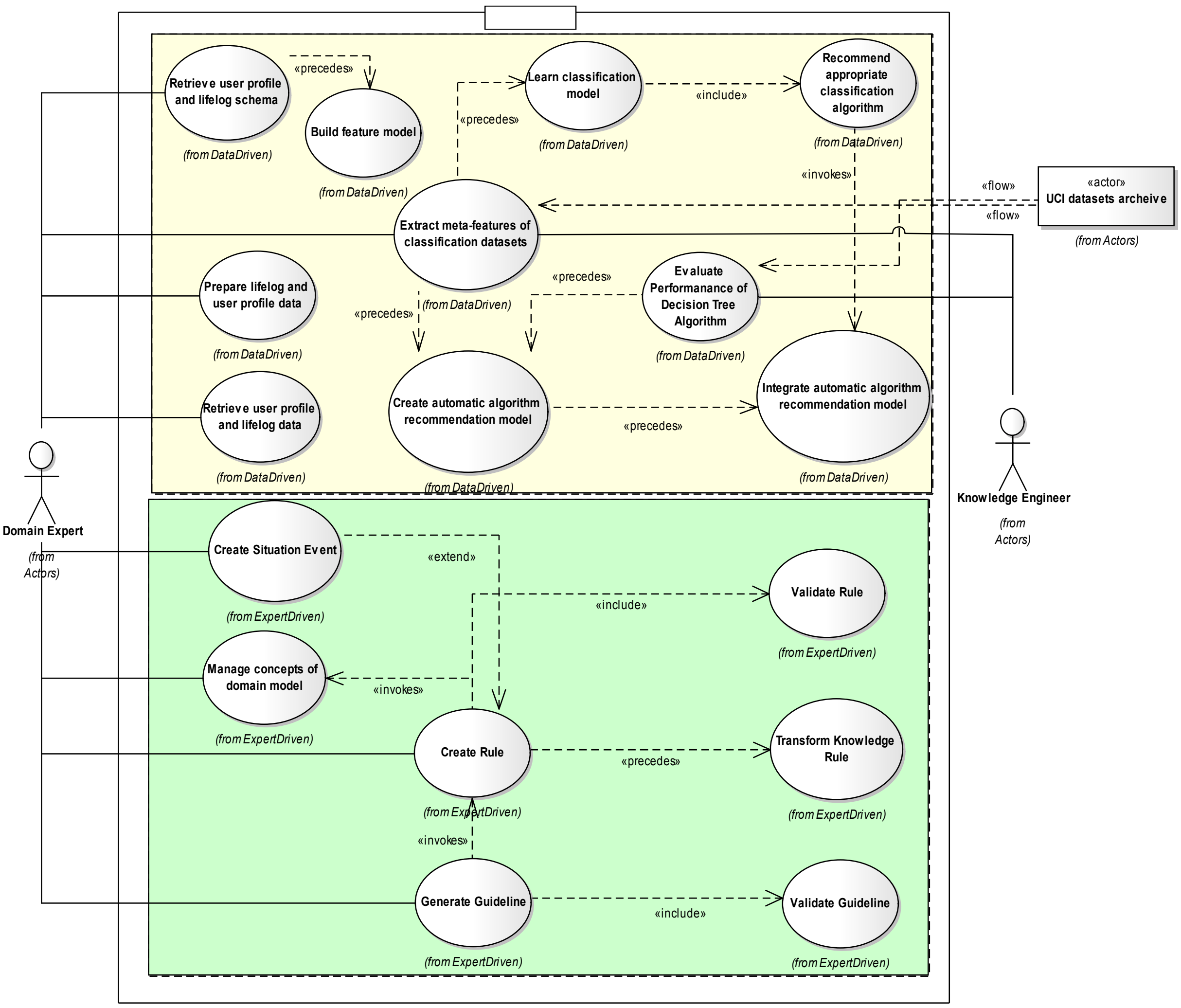
2. Use Case Diagram and Details

2.1. Use Case List

Requirements #ID	**Description**
KCL-UC-01	Select valid combinations of features from lifelog and user profile schema to build feature model for yielding correct classification model.
KCL-UC-02	Apply preprocessing methods on retrieved lifelog and user profile data to prepare the data for classification model learning.
KCL-UC-03	Expert generate guidelines to utilize their practices to create rules in the knowledge bases.
KCL-UC-04	System validate the guidelines in tree structure to maintain the rules.
KCL-UC-05	User profile and lifelog schema is needed to be known before feature modeling and creation of classification model.
KCL-UC-06	Retrieve user profile and lifelog data for creation of classification model.
KCL-UC-07	Extract meta-features of classification datasets.
KCL-UC-08	Evaluate performance of decision tree algorithms (i.e, f-measure)
KCL-UC-09	Create automatic algorithm recommendation model (AARM) from offline datasets. AARM will be used as recommendation model for algorithm selection.
KCL-UC-10	Create rules to enhance the knowledge base of the system to generate recommendations in easy manner.
KCL-UC-11	Rule validation avoid the duplication of rules in the knowledge base and enhance the maintainability of knowledge base.
KCL-UC-12	It integrates AARM dataset in Mining Minds Data Driven knowledge acquisition approach for recommendation of automatic algorithm on given dataset.
KCL-UC-13	It generates classification model from user profile lifelog data that can be explored by model learning mechanism with the help of learning method as well as processed data.
KCL-UC-14	The integrated AARM shall automatically recommend appropriate classification algorithm. Or domain expert can select any algorithm from

	available set of decision tree algorithms.
KCL-UC-15	Domain model is used in creation of rule. It manages the domain model for creating rule.
KCL-UC-16	It transforms the rules or guidelines into executable knowledge representation.
KCL-UC-17	It creates situation event and index the rule based on situation event.

2.2. Use case diagram



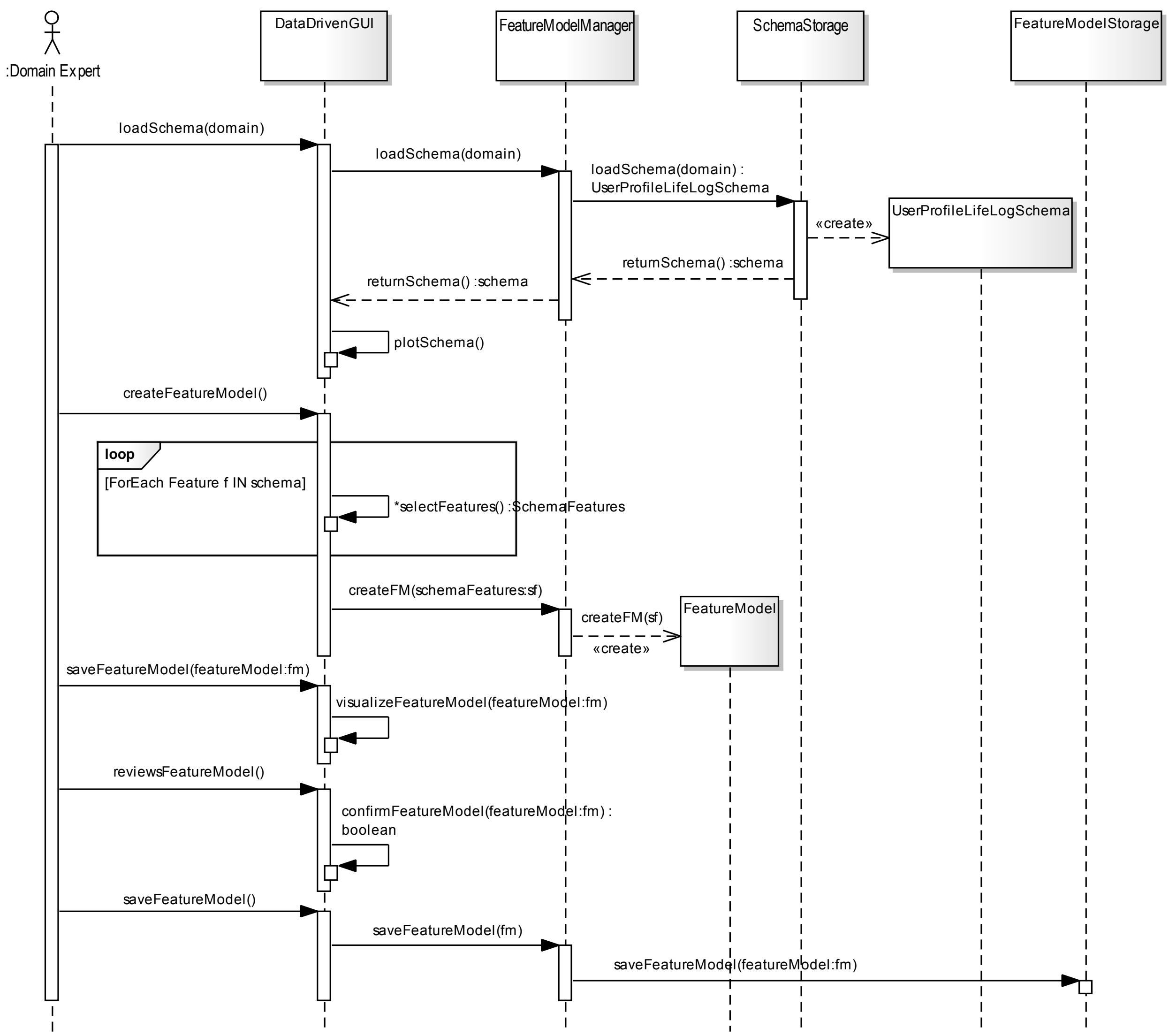
2.3. Use Case Details

Use Case ID:	**KCL-UC-01**

Use Case Name:	Build feature model		
Created By:	Maqbool Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	11-07-2015	**Last Revision Date:**	27-07-2015
Actors:	Domain Expert		
Description:	A feature model defines the valid combinations of features in a domain that enables capturing feature variability and interdependencies. For building feature model and its reusability, domain expert uses selected domain schema (i.e. lifelog and user profile schema) and selects the related features for final feature model.		
Trigger:	Prior to classification model creation needed for required domain		
Preconditions:	• System has retrieved the schema from DCL. • Domain expert has selected domain under consideration (e.g. nutrition).		
Postconditions:	System will build the feature model		
Normal Flow:	1. Domain expert retrieves the schema from schema storage. 2. System loads and plots the schema 3. Domain expert builds the feature models as follows: a. Select the required features for corresponding domain b. Verify the consistency of the selected features (UCh as concept hierarchy) c. Save the feature model 4. System creates the feature model based on selected features and visualizes it in hierarchical form 5. Domain expert reviews the feature model and confirms it for saving into repository 6. System persists the feature model into repository.		
Alternative Flows:	N/A		
Exceptions:			
Includes:	N/A		
Frequency of Use:	When new service is required and mining mind have sufficient data for classification model creation		
Special Requirements:	N/A		

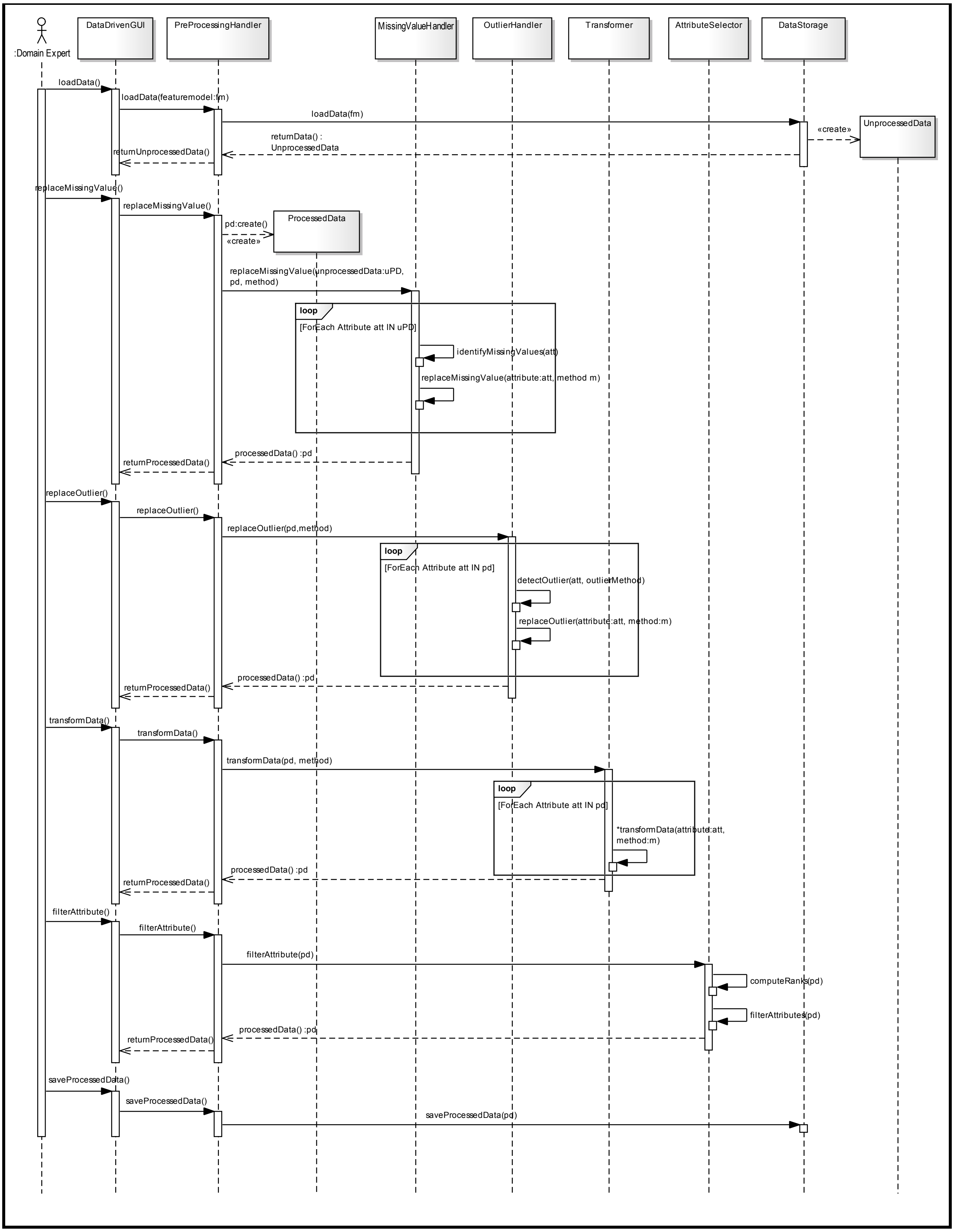
Assumptions:	Initially we assume that feature model is valid
Notes and Issues:	One possible candidate representation for feature model is XML.

Sequence Diagram:



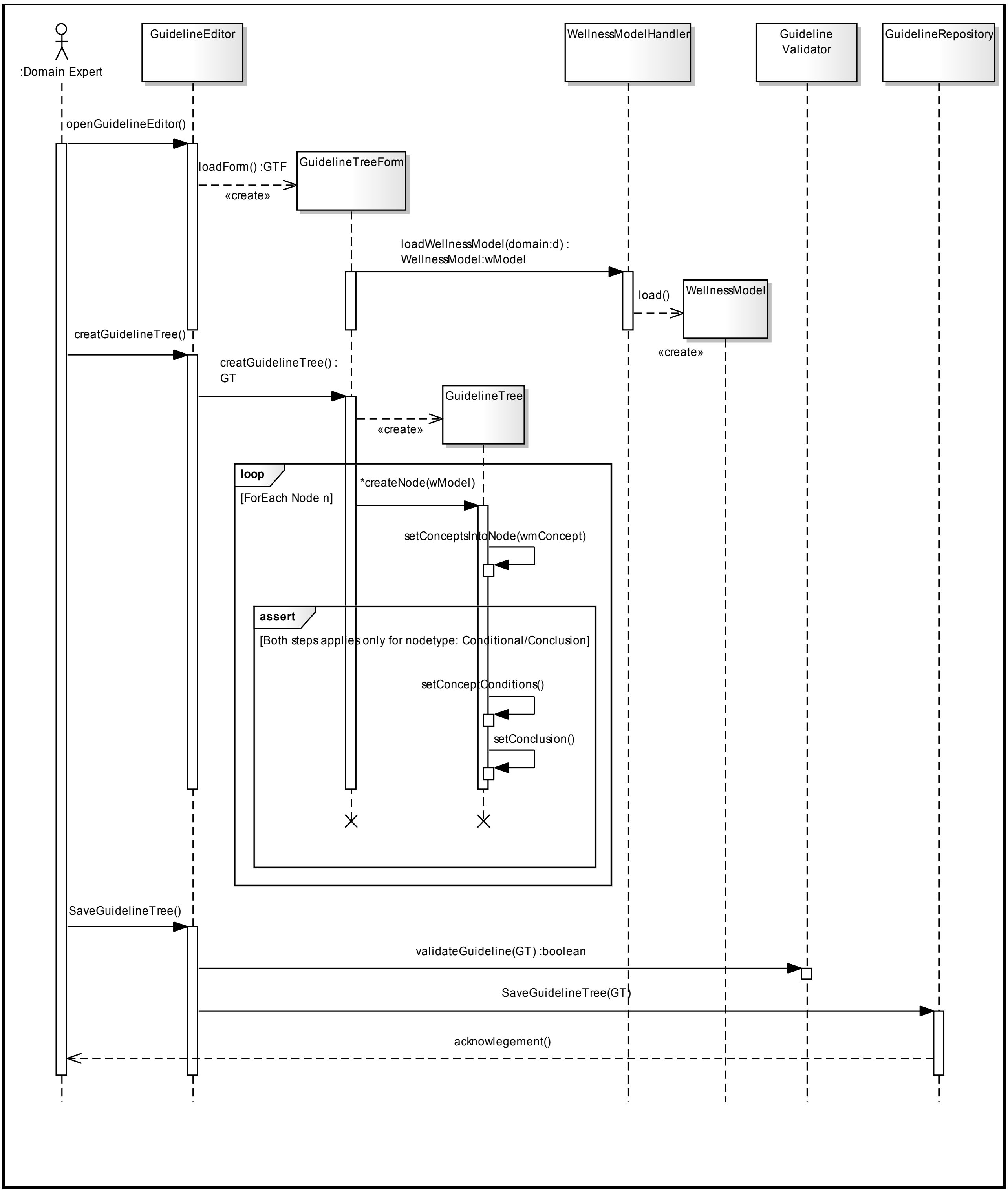
Use Case ID:	**KCL-UC-02**		
Use Case Name:	Prepare lifelog and user profile data		
Created By:	Maqbool Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	10-07-2015	**Last Revision Date:**	15-07-2015
Actors:	Domain Expert		
Description:	It is important to pre-process the data (i.e. lifelog and user profile data) to generate models with high accuracy. '*Prepare lifelog and user profile data*' use case apply various pre-processing techniques such as missing value handling, outlier detection, transformation, and features selections to convert unprocessed data into processed data.		
Trigger:	Prior to classification model creation needed for high accuracy of model learning		
Preconditions:	• System has retrieved the data from DCL, which is unprocessed data.		
Postconditions:	System will prepare and store the data		
Normal Flow:	1. Domain expert loads the unprocessed data 2. System displays the retrieved data 3. For each attribute: a. Domain expert identifies the missing values and select appropriate method from following options for missing value replacement. ▪ Default value ▪ Mean ▪ Mode b. System replaces the missing values using selected method. 4. For each attribute: a. Domain expert apply outlier detection method UCh as interquartile range and scatterplot. b. System display the outliers c. Domain expert select appropriate method from following options for outlier replacement. ▪ Mean ▪ Mode d. System replace the outlier using selected method. 5. For each attribute: a. Domain expert identifies, normalizes the non-transformed values, and updates the dataset. b. System modifies the values set and update the dataset 6. Domain expert applies the attributes filtration techniques (i.e ranking) 7. System computes the ranks for all attributes and displays to expert		

	8. Domain expert select the highly ranked attributes (i.e. rank value >= 0.8) 9. System filters the attributes based on selected attributes and displays to domain expert 10. Domain expert saves the processed data into repository 11. System persists the processed data into repository
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	When new service is required and mining mind have sufficient data for classification model creation
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	• An **outlier** is any value that is numerically distant from most of the other data points in a set of data. It can be detected by histograms, scatterplots, or interquartile range techniques. • **Data transformation** is the process to convert and normalize the data from one format to another. It can be done by Log, square root, or arcsine transformation techniques.
Sequence Diagram:	


:Domain Expert
DataDrivenGUI
PreProcessingHandler
MissingValueHandler
OutlierHandler
Transformer
AttributeSelector
DataStorage
UnprocessedData
ProcessedData
loadData()
loadData(featuremodel:fm)
loadData(fm)
«create»
returnData() : UnprocessedData
returnUnprocessedData()
replaceMissingValue()
replaceMissingValue()
pd:create()
«create»
replaceMissingValue(unprocessedData:uPD, pd, method)
loop
[ForEach Attribute att IN uPD]
identifyMissingValues(att)
replaceMissingValue(attribute:att, method m)
processedData() :pd
returnProcessedData()
replaceOutlier()
replaceOutlier()
replaceOutlier(pd,method)
loop
[ForEach Attribute att IN pd]
detectOutlier(att, outlierMethod)
replaceOutlier(attribute:att, method:m)
processedData() :pd
returnProcessedData()
transformData()
transformData()
transformData(pd, method)
loop
[ForEach Attribute att IN pd]
*transformData(attribute:att, method:m)
processedData() :pd
returnProcessedData()
filterAttribute()
filterAttribute()
filterAttribute(pd)
computeRanks(pd)
filterAttributes(pd)
processedData() :pd
returnProcessedData()
saveProcessedData()
saveProcessedData()
saveProcessedData(pd)

Use Case ID:	**KCL-UC-03**		
Use Case Name:	Generate Guideline		
Created By:	Taqdir Ali and Maqbool Hussain	**Last Updated By:**	Taqdir Ali, Maqbool Hussain
Date Created:	11-07-2015	**Last Revision Date:**	15-07-2015
Actors:	Domain Expert		
Description:	Guidelines are the combination of one or multiple rules in form of decision tree. The tree format guidelines are understandable to the domain experts and it can easily interpret and execute by computer.		
Trigger:	Whenever domain expert wants to generate new guideline or update the existing one.		
Preconditions:	1. The domain expert shall be authenticated with full access on the guideline management. 2. Domain expert shall have existing guidelines as reference for generating guideline tree.		
Postconditions:	• The expert shall generate guidelines to acquire their knowledge into the system.		
Normal Flow:	1. Domain expert opens the guidelines editor. 2. System displays new guideline tree form and load the wellness model. 3. Domain expert selects/drags tree node into editor form. 4. System display the node and open the corresponding properties window, which includes; a. Loads wellness model tree. b. Displays operators, relationships and node type (conditional, conclusion or both) artifacts. 5. Domain expert selects concepts for the node using any of the following methods and confirm to save the node. a. Using wellness model, dragging concepts and facts into node conditional or conclusion part. b. Using auto pop-up Intelli-sense window to select concepts and facts into conditional or conclusion part. 6. System saves the tree node and displays as part of the guideline tree. 7. Domain expert add other nodes to guideline tree by using step repeating step 3 on ward. After completion, (s)he saves the guideline tree.		

	8. System validates the guideline tree using “KCL-UC-04” and save into guideline repository.
Alternative Flows:	2a. System loads existing guideline tree for modification (modifying existing or adding new node). a. Domain expert selects existing node in guideline tree or drag new node to appropriate place in guideline tree. b. To modify node, step 3 onward will be invoked in Normal Flow.
Exceptions:	N/A
Includes:	Validate Guideline
Frequency of Use:	Whenever domain expert want to create new guideline tree or update existing guideline tree.
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	• Guidelines tree created will base on existing guidelines of corresponding domain and domain expert shall interpret textual guidelines into tree format. • Appropriate modelling of guideline is challenging task.
Sequence Diagram:	

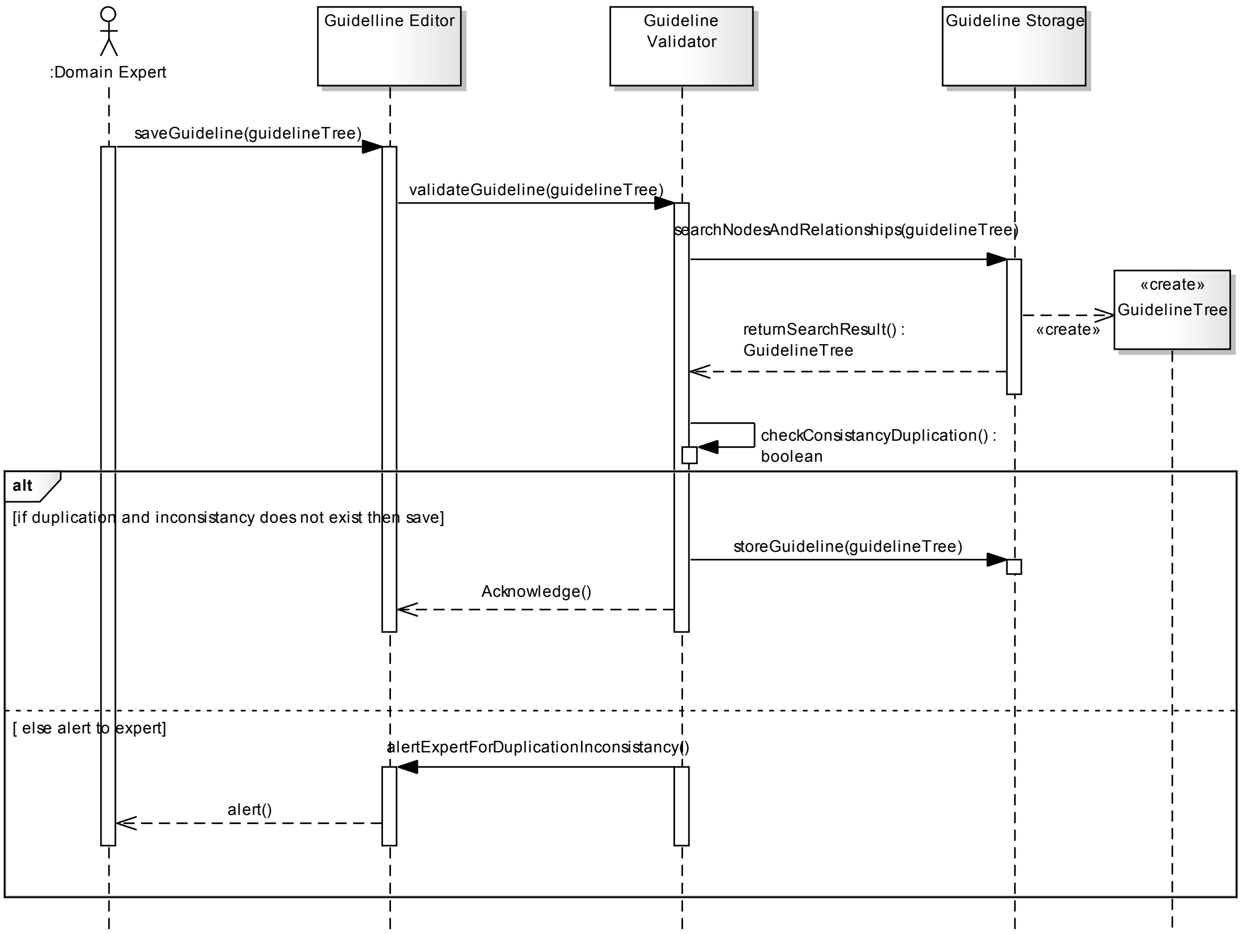


Use Case ID:	KCL-UC-04

Use Case Name:	Validate Guideline		
Created By:	Taqdir Ali and Maqbool Hussain	**Last Updated By:**	Taqdir Ali, Maqbool Hussain
Date Created:	11-07-2015	**Last Revision Date:**	15-07-2015
Actors:	Domain Expert		
Description:	Guidelines have different facts and conclusions in form of nodes related with different relationships. The guidelines tree shall be validated for the possible duplication.		
Trigger:	Whenever domain expert wants to generate new guideline or update the existing one.		
Preconditions:	1. System shall be running 2. The domain expert shall be authenticated with full access on the guideline management.		
Postconditions:	• Validated guideline tree		
Normal Flow:	1. Domain Expert save the new guideline or update the existing guideline. 2. The system validate guideline for inconsistency and duplication as follows. a. Fetch the existing guidelines and process each node and relationship b. Guideline Tree is approved for having no inconsistency and duplication of new nodes and relationships of the facts and conclusion with the existing guidelines. c. Guideline Tree is stored into guidelines repository. d. Acknowledge the expert to save guideline successfully.		
Alternative Flows:	2b. Guideline Tree is found having inconsistency or duplication with existing guideline tree a. The system produces alert the inconsistency or duplication in guideline tree b. Domain expert review the alert message and correct the guideline tree. c. Step 1 and Step 2 of normal flow is executed.		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Whenever domain expert want to create new guideline or update existing guideline.		

Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Appropriate modelling of guideline and validation is challenging task.

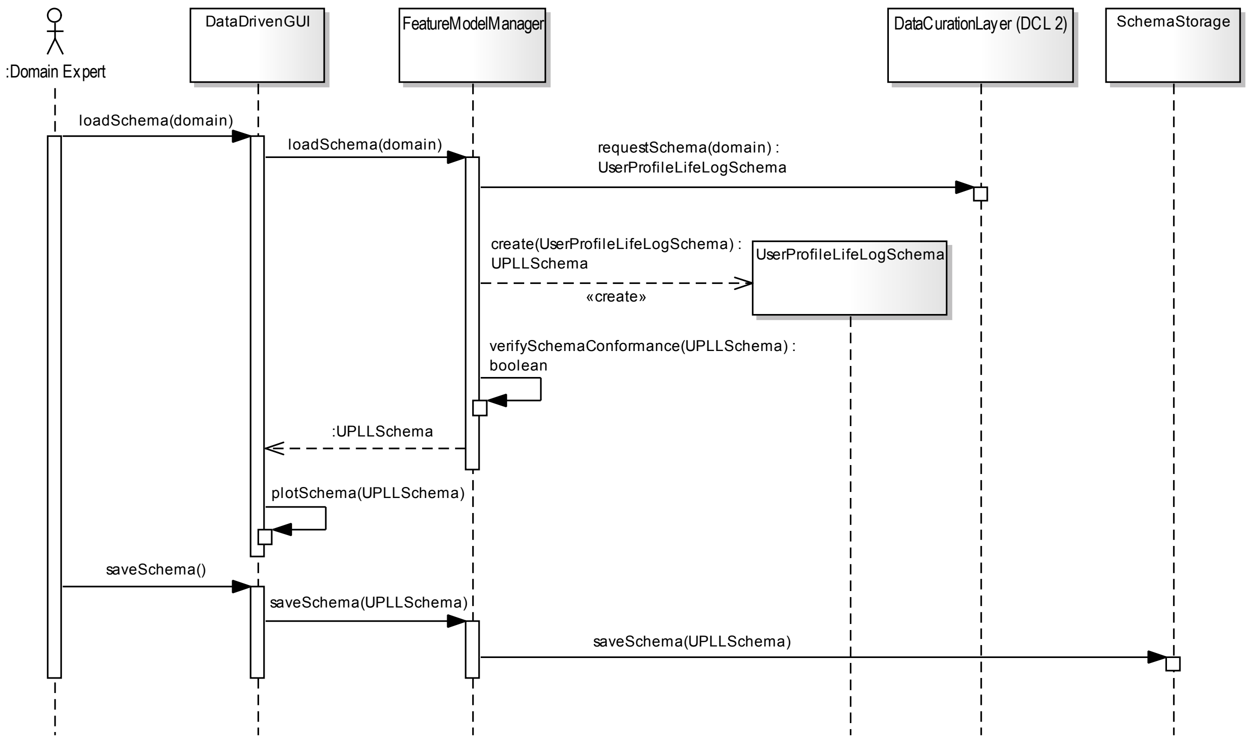
Sequence Diagram:



Use Case ID:	**KCL-UC-05**		
Use Case Name:	Retrieve user profile and lifelog schema		
Created By:	Maqbool Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	10-07-2015	**Last Revision Date:**	27-07-2015
Actors:	Domain Expert, DCL		
Description:	User profile and lifelog schema retrieval help domain expert to view all available features for building feature model.		
Trigger:	Prior to classification model creation needed for required domain		
Preconditions:	• System has access through service interface to retrieve user profile and lifelog schema from DCL • System and DCL has agreement on common schema representation format • DCL has capability to share user profile and lifelog schema in secure environment.		
Postconditions:	System will receive user profile and lifelog schema conform to its representation scheme.		
Normal Flow:	1. Domain expert selects the domain and sends requests to DCL for user profile and lifelog schema. 2. DCL shares the user profile and lifelog schema 3. System receives the user profile and lifelog schema 4. Domain expert uses the system and performs the following tasks; a. Verifies the conformance of received schema b. Plots the verified schema c. Saves the verified schema 5. System saves the verified schema		
Alternative Flows:	N/A		
Exceptions:	1a. System unable to connect to DCL a. System connection is failed during retrieving user profile and lifelog schema b. System hold and will retry after sometime to connect to DCL and retrieve the user profile and lifelog schema 4a. System unable to verify lifelog schema conformance a. System fail to conform the schema representation from DCL b. System will send message to DCL about incompatible schema format		

Includes:	N/A
Frequency of Use:	When new service is required and mining mind have sufficient data for classification model creation
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	If DCL is unable to send user profile and lifelog schema in required format, then alternate strategy has to be considered.

Sequence Diagram:

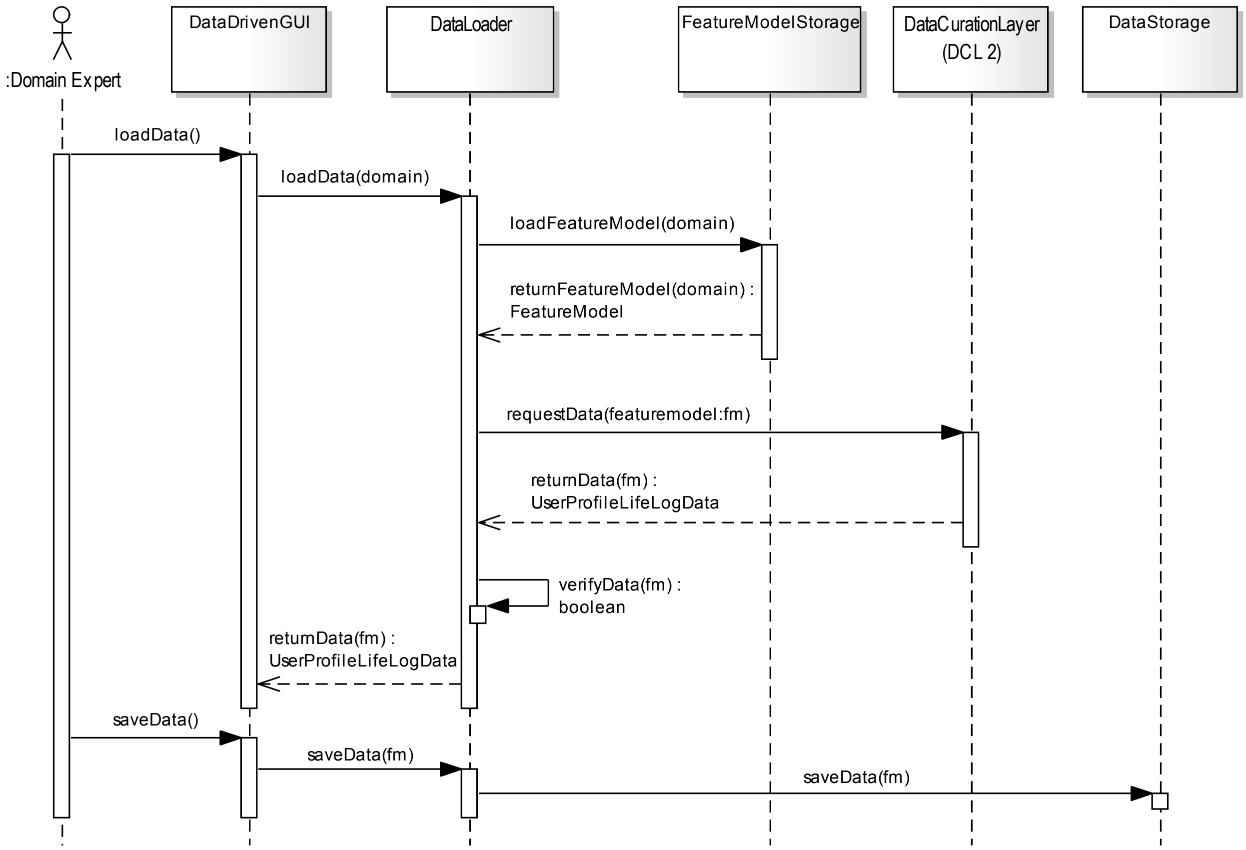


Use Case ID:	**KCL-UC-06**
Use Case Name:	Retrieve user profile and lifelog data

Created By:	Maqbool Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	10-07-2015	**Last Revision Date:**	27-07-2015
Actors:	Domain Expert, DCL		
Description:	User profile and lifelog data has hidden knowledge that can be explored after loading from data storage		
Trigger:	Prior to classification model creation needed for required data		
Preconditions:	• System has access through service interface to retrieve user profile and lifelog data from DCL • DCL has capability to share user profile and lifelog data in secure environment. • System has already loaded the previously imported user profile and lifelog schema		
Postconditions:	System will receive user profile and lifelog data based on selected schema		
Normal Flow:	1. Domain expert loads the feature model for selected domain 2. System loads the corresponding feature model 3. Domain expert sends request to DCL for user profile and lifelog data based on loaded feature model 4. DCL shares the user profile and lifelog data 5. System receives the user profile and lifelog data 6. Domain expert uses the system and performs the following tasks; o Verifies the user profile and lifelog data o Saves the data after verification. 7. System saves the verified data		
Alternative Flows:			
Exceptions:	3a. System unable to connect to DCL a. System connection is failed during retrieving user profile and lifelog data b. System hold and will retry after some time to connect to DCL and retrieve the user profile and lifelog data 6a. System receives irrelevant data a. System detects the irrelevant data sent by DCL. b. System request again DCL to make sure that data received is according to feature selected.		
Includes:			
Frequency of Use:	When new service is required and mining mind have sufficient data for classification model		

	creation
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	If DCL is unable to send data based on dynamic feature selection from schema, then alternate strategy has to be considered.

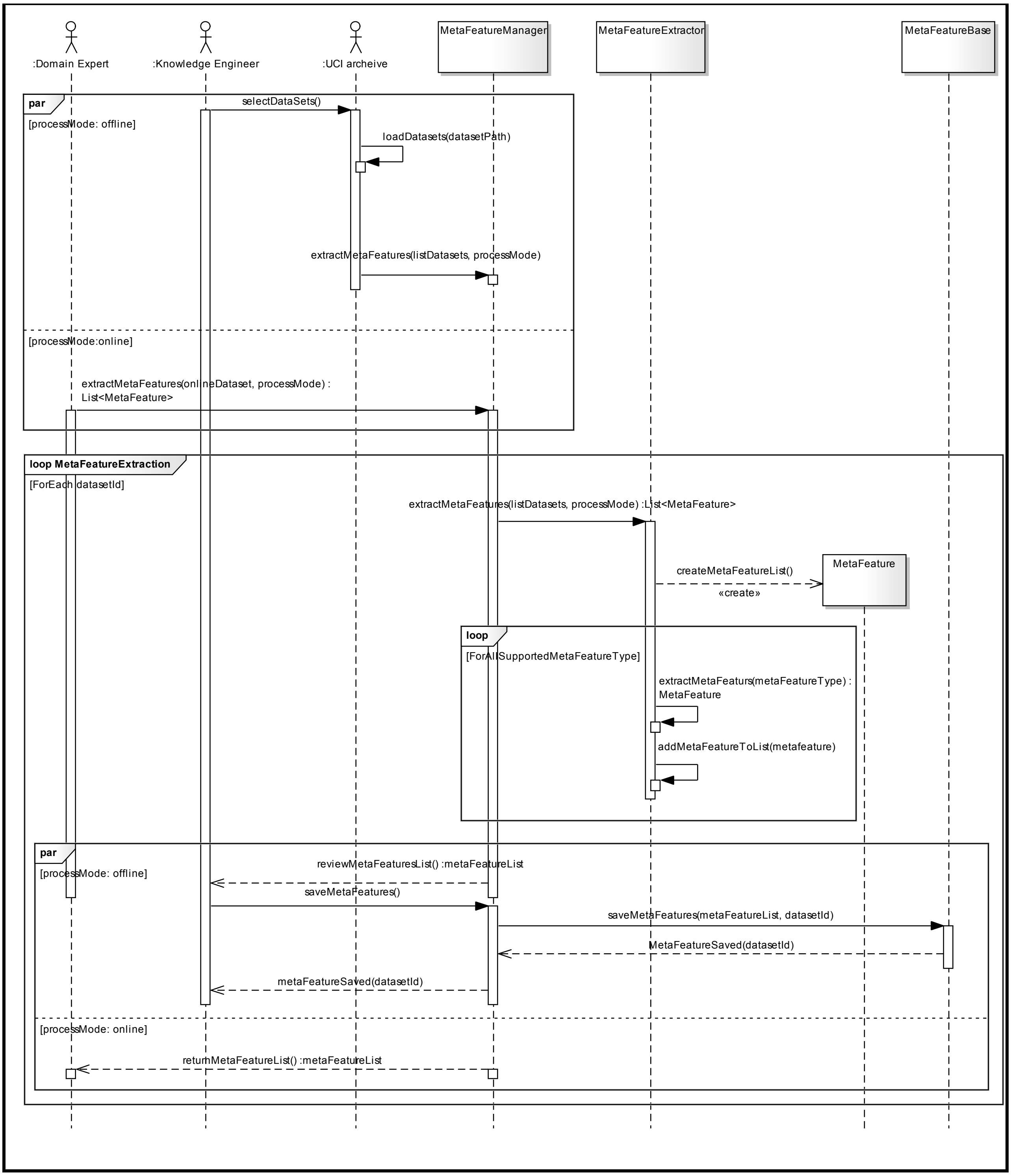
Sequence Diagram:



Use Case ID:	**KCL-UC-07**
Use Case Name:	Extract meta-features of classification datasets

Created By:	Rahman Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	16-07-2015	**Last Revision Date:**	20-07-2015
Actors:	Knowledge Engineer/Domain Expert, UCI archive		
Description:	Datasets have simple, statistical, information theory and landmarking meta-features that can best describe nature of a dataset. These features can best used for building an algorithm selection model.		
Trigger:	In the offline process, when the algorithm selection model is build, and in the online process, when an appropriate algorithm is needed to be identified for a new query dataset		
Preconditions:	UCI archive datasets are available and are in refined format		
Postconditions:	The meta-features are ready for being used in building algorithm selection model.		
Normal Flow:	1. Knowledge Engineer selects one dataset from UCI archive. 2. System retrieves selected dataset. 3. Knowledge Engineer provides dataset to meta-feature extractor for extracting meta-features. 4. System extracts following meta-features set for selected dataset. a. basic meta-features b. statistical meta-features c. information theory meta-features d. extract landmark features 5. Knowledge Engineer reviews the extracted meta-features and saves it into meta-features base (MFB). 6. System saves meta-features into a MFB. 7. Knowledge Engineer repeats step 1-6 for each intended dataset.		
Alternative Flows:	1a. Meta-feature extraction for online dataset a. Domain Expert provides new dataset used for classification model creation. b. Step 3-4 of Normal Flow is executed for Domain expert interactions with system.		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Frequently, whenever a new dataset arrives as a query dataset.		
Special Requirements:	Minimum availability of classification datasets > 60 for a reasonable accuracy		

Assumptions:	• The archived datasets are available and are in refined .arff format • The meta-feature space is defined in advance
Notes and Issues:	Meta-feature extraction is time consuming task for offline process as we have to take into account more than 60 datasets.
Sequence Diagram:	

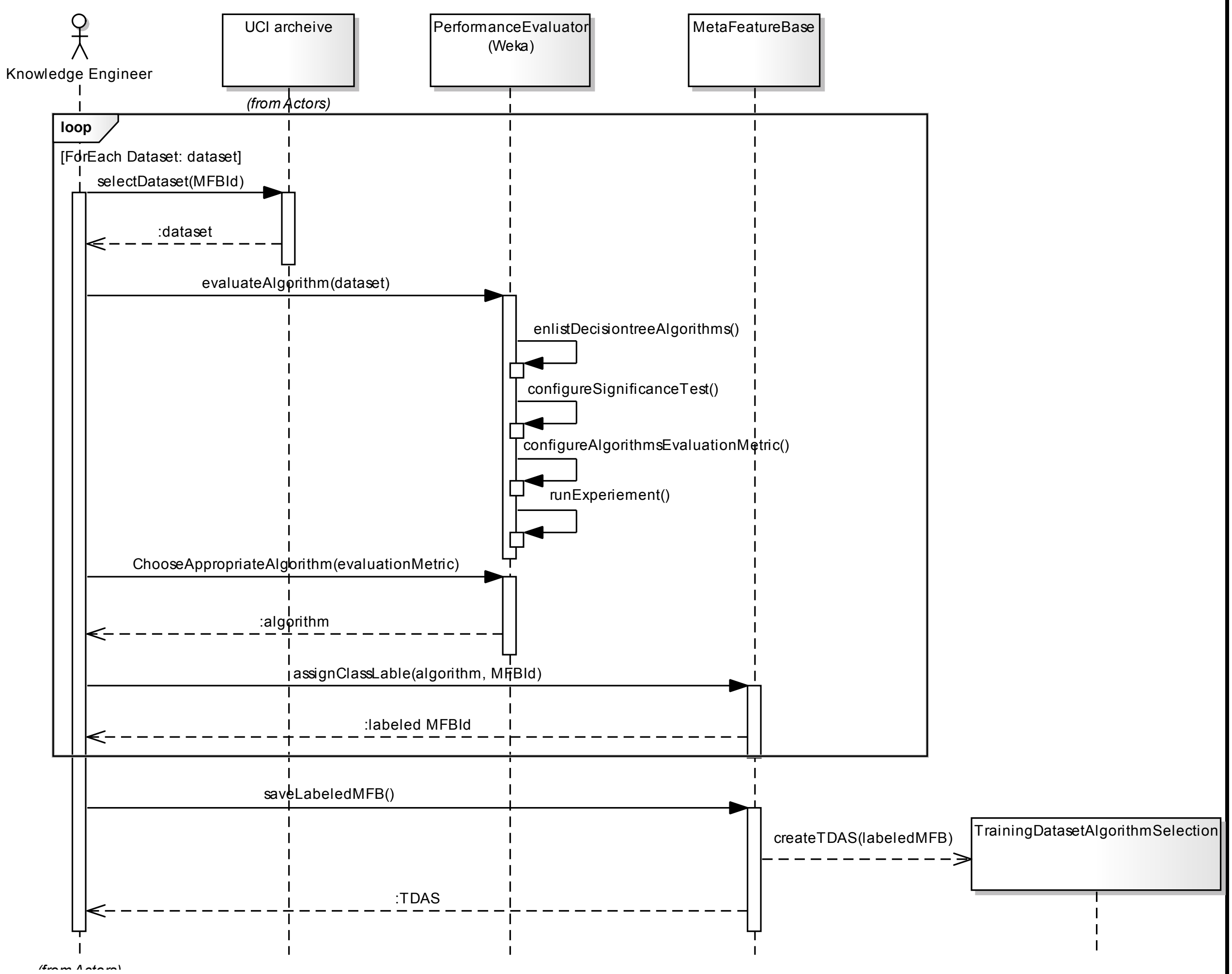


Use Case ID:	**KCL-UC-08**

Use Case Name:	Evaluate performance of decision tree algorithm		
Created By:	Rahman Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	16-07-2015	**Last Revision Date:**	20-07-2015
Actors:	Knowledge Engineer, UCI archive		
Description:	Different algorithms have different performance score for the same dataset. To build an algorithm selection model, performance score of each algorithm needs to evaluate for choosing an appropriate one.		
Trigger:	In the offline process, when the algorithm selection model is to build the first time.		
Preconditions:	• UCI archive datasets are available and are in .arff format • The algorithm to be considered is specified in advanced (Decision Tree algorithms implemented in Weka) • The evaluation metric is specified (F-measure)		
Postconditions:	All datasets records in Meta-Feature Base (MFB) will be assigned with optimal decision tree algorithm class label.		
Normal Flow:	1. Knowledge Engineer selects UCI archive dataset, mentioned in MFB, for finding optimal decision tree algorithm. 2. System (Weka) loads selected datasets. 3. Knowledge Engineer setups experiment; a. Enlists all the decision tree algorithms available in system (Weka) b. Configure significance test (alpha=0.5) c. Configure algorithms comparison metric (f-measure) 4. System runs experiment and produces detailed f-score for all selected algorithms. 5. Knowledge Engineer performs following tasks; a. Records evaluation matrix. b. Chooses algorithm with the highest f-score. c. Assigns chosen algorithm as class label in MFB. d. Step 1-5 are repeated for other non-labeled datasets in MFB. e. After finishing labeling all records in MFB, saves the updated MFB as training dataset for algorithm selection (TDAS). 6. System saves the updated records in MFB as final TDAS.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	N/A		

Frequency of Use:	• Rarely, once enough new datasets are added to the system
Special Requirements:	N/A
Assumptions:	• The decision tree-based algorithms are implemented in Weka.
Notes and Issues:	Evaluating performance of decision tree algorithms over a large number (min 60) classification dataset is a computationally complex task. For minimum 60 datasets and at least 5 decision tree algorithms, a minimum of 300 experiments are required. Moreover, 60 additional significance test experiments are needed. On average, each experiment takes times in minutes ranging from 2 minutes to 30 minutes, depending on the complexity of the dataset.

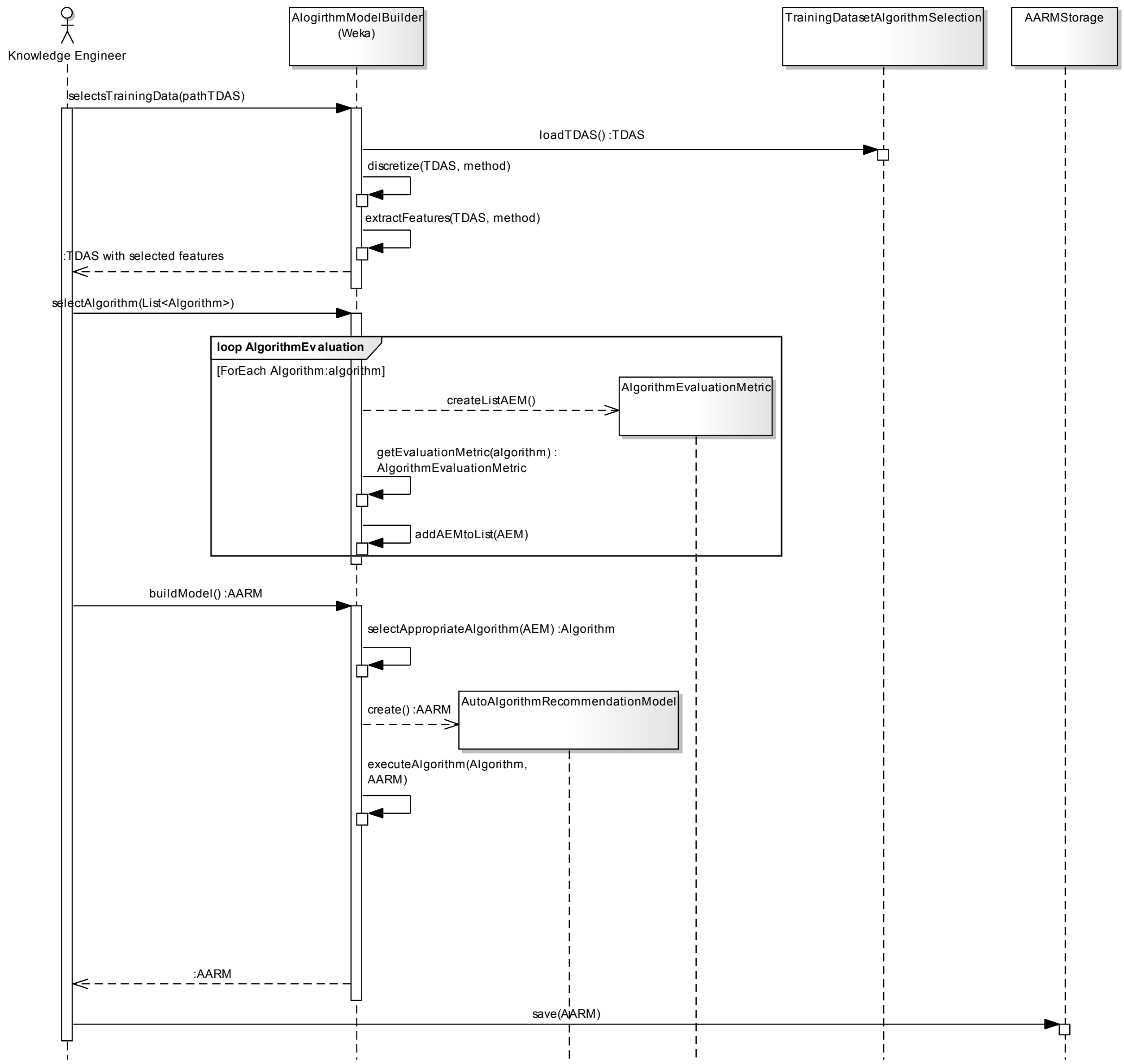
Sequence Diagram:



Use Case ID:	**KCL-UC-09**		
Use Case Name:	Create automatic algorithm recommendation model		
Created By:	Rahman Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	10-07-2015	**Last Revision Date:**	22-07-2015
Actors:	Knowledge Engineer/Domain Expert		
Description:	An automatic algorithm selection model enables knowledge engineer to automatically select appropriate algorithm for building classification model for his new dataset		
Trigger:	When the training dataset comprising datasets meta-features and algorithms performance evaluation are made available.		
Preconditions:	The datasets meta-features and algorithms performance training datasets is made available.		
Postconditions:	The automatic algorithm recommendation model (AARM) is ready to integrate in Mining Minds for real time algorithm selection.		
Normal Flow:	1. Knowledge Engineer selects TDAS. 2. System (Weka) loads TDAS. 3. Knowledge Engineer performs preprocessing of the TDAS (i.e., discretization, and features selection). 4. System refines the datasets with appropriate features. 5. Knowledge Engineer select an algorithm from decision tree (DT) or Rules learner (RL) for building AARM. 6. System executes selected algorithm on TDAS and produces evaluation matrix. 7. Knowledge Engineer records performs following tasks; a. Records evaluation matrix and repeats step 5-6 till all algorithms finished. b. Knowledge Engineer selects the appropriate algorithm. 8. System builds AARM model using the selected algorithm. 9. Knowledge Engineer saves AARM. 10. System saves the AARM into AARM storage.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	N/A		

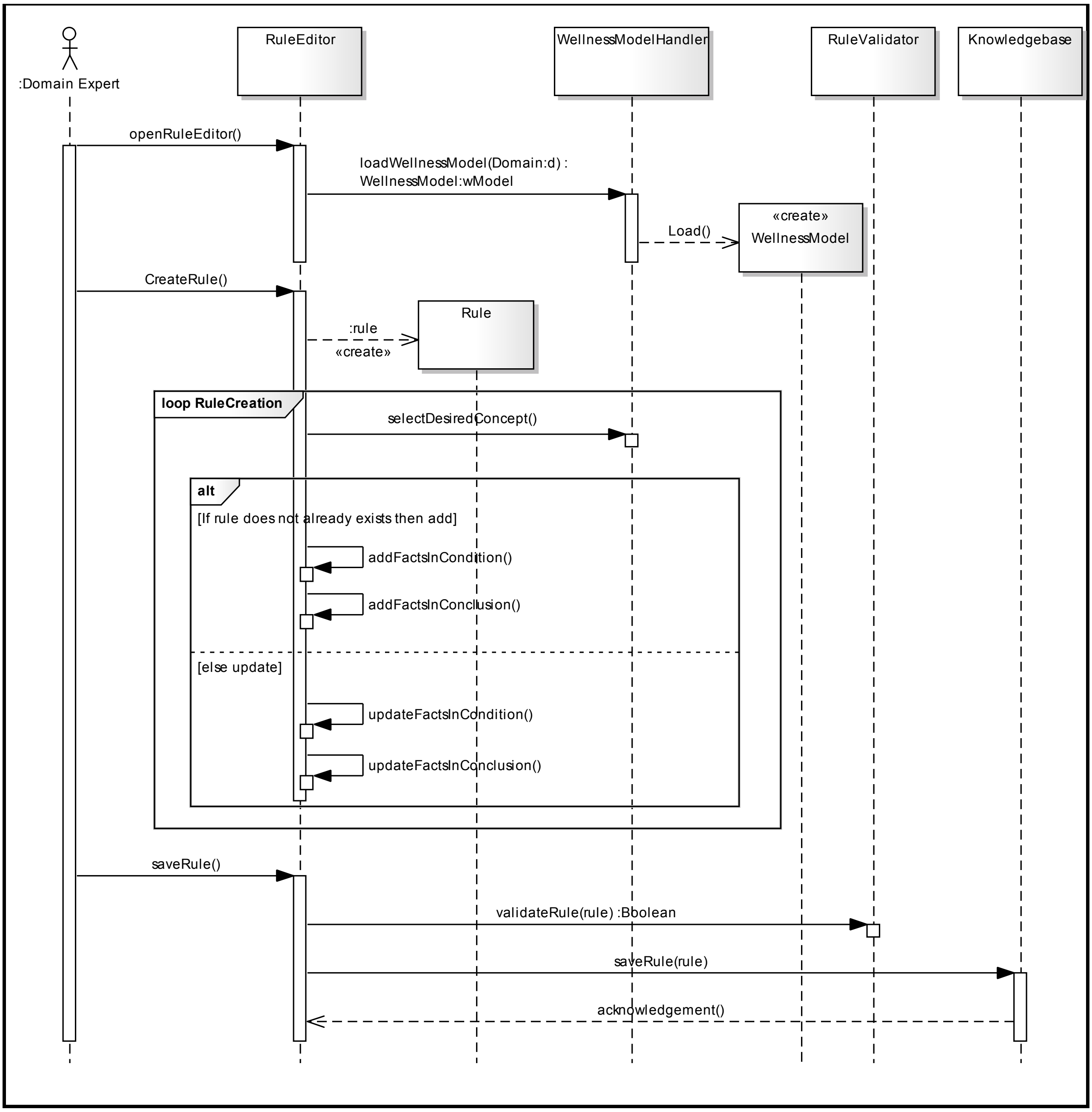
Frequency of Use:	Rarely, when TDAS is updated with new datasets or algorithms
Special Requirements:	Availability of records > 60 for minimum acceptable accuracy
Assumptions:	N/A
Notes and Issues:	N/A

Sequence Diagram:



Use Case ID:	**KCL-UC-10**		
Use Case Name:	Create Rule		
Created By:	Taqdir Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	11-07-2015	**Last Revision Date:**	29-07-2015
Actors:	Domain Expert		
Description:	Knowledge bases need to enhance with up-to-date knowledge for correct recommendation. The expert shall create rules in the knowledge base to transform their practices into executable knowledge in form of rules.		
Trigger:	Domain experts trigger it for rule creation/updating when needed.		
Preconditions:	• The domain expert shall be authenticated with full access of rule management in the knowledge base.		
Postconditions:	The correct rule shall be saved into the knowledge base.		
Normal Flow:	1. Domain expert opens the rule editor. 2. System loads wellness domain model in form of concepts tree. 3. Domain expert selects concepts for the rule conditions and conclusion using any of the following methods; o Using wellness model, dragging concepts and facts into conditional or conclusion part of the rule editor. o Using auto pop-up Intelli-sense window to select concepts and facts into conditional or conclusion part of the rule editor. 4. System checks the existing rules to add/update the rule o Add new facts of the rule in condition. o Add new conclusion according to rule facts. 5. Step 3-4 will be repeated for each new/updated concept added to rule, and domain expert finally saves the rule. 6. System save rule as follows; o The system validates the rule using "KCL-UC-11". o The system stores the validated rule into the knowledge base		
Alternative Flows:	6a. System founds the rule is already exists in rule repository o Domain expert review the existing facts and conclusion. o Step 5-6 will be followed to change the rule.		
Exceptions:	N/A		

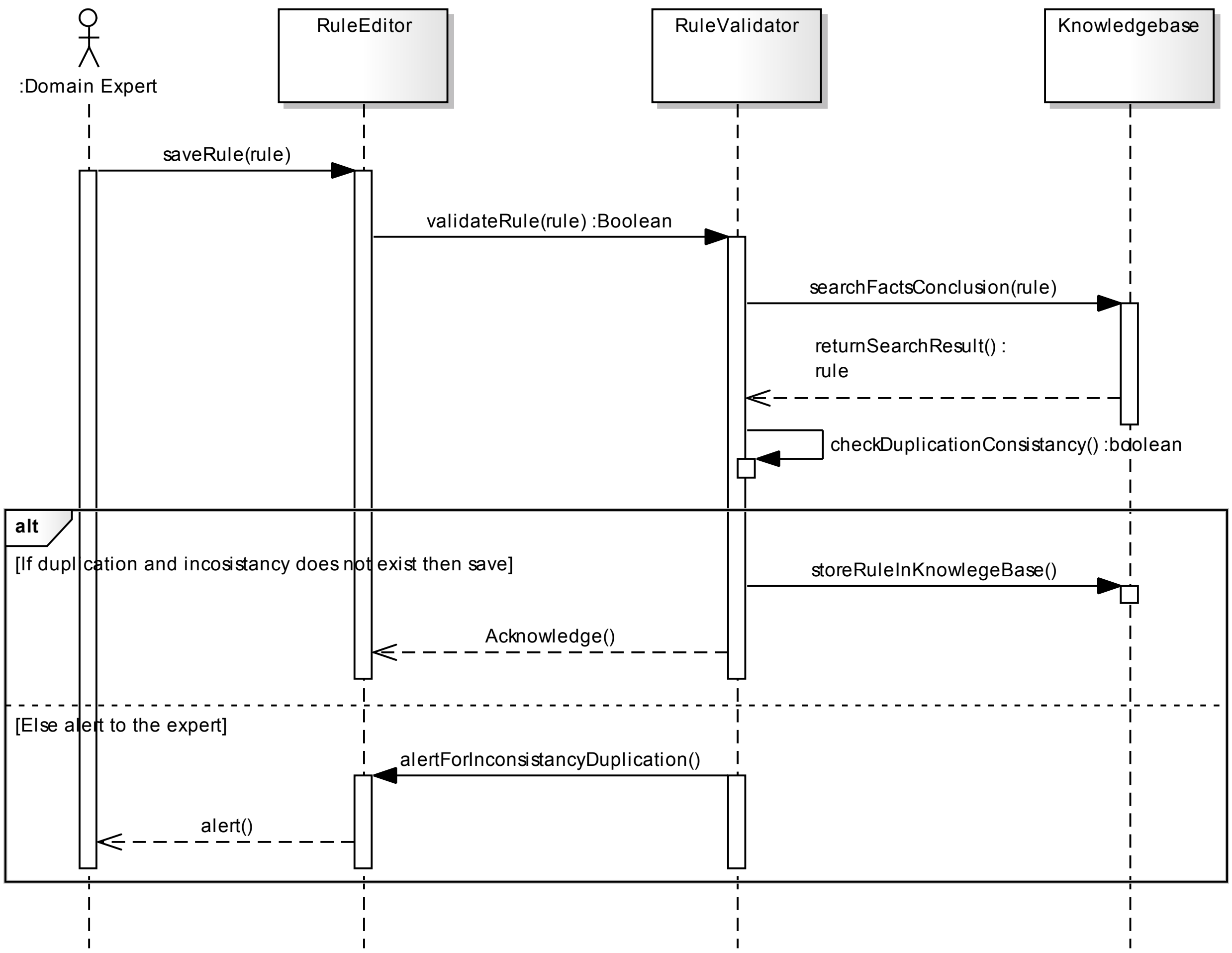
Includes:	Validate Rule
Frequency of Use:	Whenever domain expert want to add rule or edit the existing rule.
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	• If knowledge base does not exist in system the administrator shall build the knowledge base first and configure with system. • After investigation, we may use unify representation for rules and guidelines.
Sequence Diagram:	


:Domain Expert
RuleEditor
WellnessModelHandler
RuleValidator
Knowledgebase
openRuleEditor()
loadWellnessModel(Domain:d) :
WellnessModel:wModel
«create»
WellnessModel
Load()
CreateRule()
Rule
:rule
«create»
loop RuleCreation
selectDesiredConcept()
alt
[If rule does not already exists then add]
addFactsInCondition()
addFactsInConclusion()
[else update]
updateFactsInCondition()
updateFactsInConclusion()
saveRule()
validateRule(rule) :Bboolean
saveRule(rule)
acknowledgement()

Use Case ID:	**KCL-UC-11**
Use Case Name:	Validate Rule
Created By:	Taqdir Ali
Last Updated By:	Maqbool Hussain
Date Created:	11-07-2015
Last Revision Date:	29-07-2015
Actors:	Domain Expert
Description:	In new rules creation and editing existing rules, duplication and inconsistency may occur. The validation is needed to validate and find the duplication and inconsistency among the rules.
Trigger:	• When new rule is going to create. • When existing rule is going to update
Preconditions:	The rule creation and editing process completed by physician successfully.
Postconditions:	The validated rule shall be saved into knowledge base.
Normal Flow:	1. Domain expert saves the created rule. 2. System validate rule for inconsistency and duplication as follows a. Fetch the facts and conclusion of existing rules. b. The new or updated rule approved for having no inconsistency and duplication. c. Created rule stores into the rules repository. d. Acknowledge the expert to save the rule successfully.
Alternative Flows:	2b. The created rule is found having inconsistency or duplication with existing rules in the rules repository a. The system produces alert the inconsistency or duplication of the rule with existing rules in repository. b. Domain expert review the alert message and correct the created rule. c. Step 1 and Step 2 of normal flow is executed.
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever domain expert want to save the rule
Special Requirements:	N/A

Assumptions:	N/A
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.

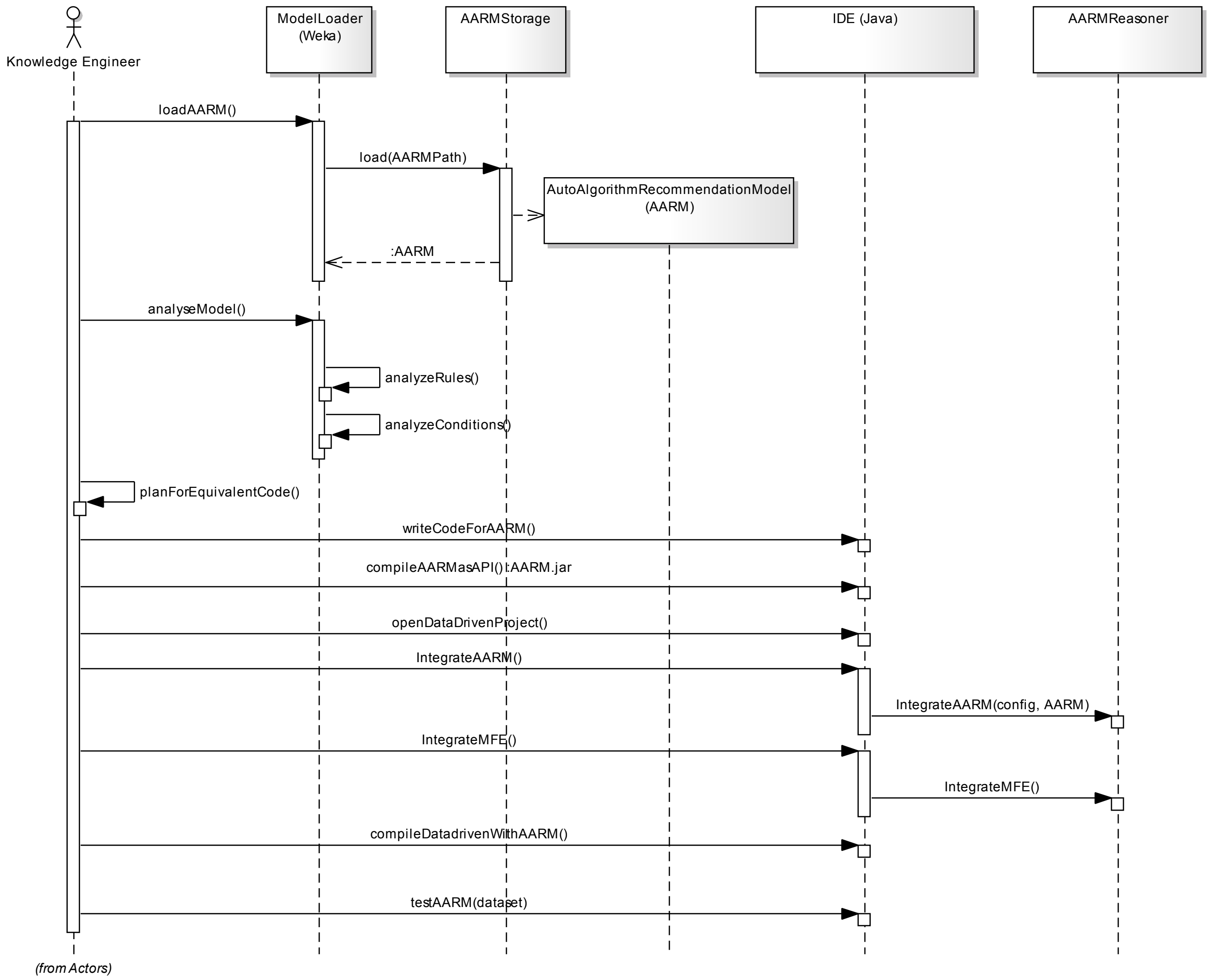
Sequence Diagram:



<table>
<tr><td>Use Case ID:</td><td colspan="3">KCL-UC-12</td></tr>
<tr><td>Use Case Name:</td><td colspan="3">Integrate automatic algorithm recommendation model</td></tr>
<tr><td>Created By:</td><td>Rahman Ali</td><td>Last Updated By:</td><td>Maqbool Hussain</td></tr>
<tr><td>Date Created:</td><td>10-07-2015</td><td>Last Revision Date:</td><td>28-07-2015</td></tr>
<tr><td>Actors:</td><td colspan="3">Knowledge Engineer</td></tr>
<tr><td>Description:</td><td colspan="3">For real time recommendation of classification algorithm for a new dataset, AARM need to be integrated in Mining Minds Data Driven knowledge acquisition approach.</td></tr>
<tr><td>Trigger:</td><td colspan="3">When AARM is built.</td></tr>
<tr><td>Preconditions:</td><td colspan="3">• AARM is available
• Data driven approach has a unified interface to support AARM as plugin
• Data driven has unified interface for accessing Meta-Feature Extractor.</td></tr>
<tr><td>Postconditions:</td><td colspan="3">AARM is plugged into data driven environment and readily available for real time recommendation of appropriate classification algorithm.</td></tr>
<tr><td>Normal Flow:</td><td colspan="3">1. Knowledge engineer selects AARM and performs the following tasks;
• Analyses number of rules in the AARM
• Analyses condition attributes used in each rule of AARM
• Transforms rules into executable classes (using any IDE of Java).
2. Knowledge engineer integrates the executable AARM into data driven as follows;
a. Write integration code (following unified interface) into data driven source code
b. Update possible configuration for newly added AARM plugin.
c. Update possible configuration for accessing Meta-Feature Extractor.
3. Knowledge Engineer compile the AARM as integral part with data driven code.
4. Knowledge engineer tests AARM with sample dataset.</td></tr>
<tr><td>Alternative Flows:</td><td colspan="3">N/A</td></tr>
<tr><td>Exceptions:</td><td colspan="3">N/A</td></tr>
<tr><td>Includes:</td><td colspan="3">N/A</td></tr>
</table>

Frequency of Use:	Rarely, once the AARM is updated
Special Requirements:	AARM has acceptable accuracy (60%)
Assumptions:	AARM is created
Notes and Issues:	N/A

Sequence Diagram:

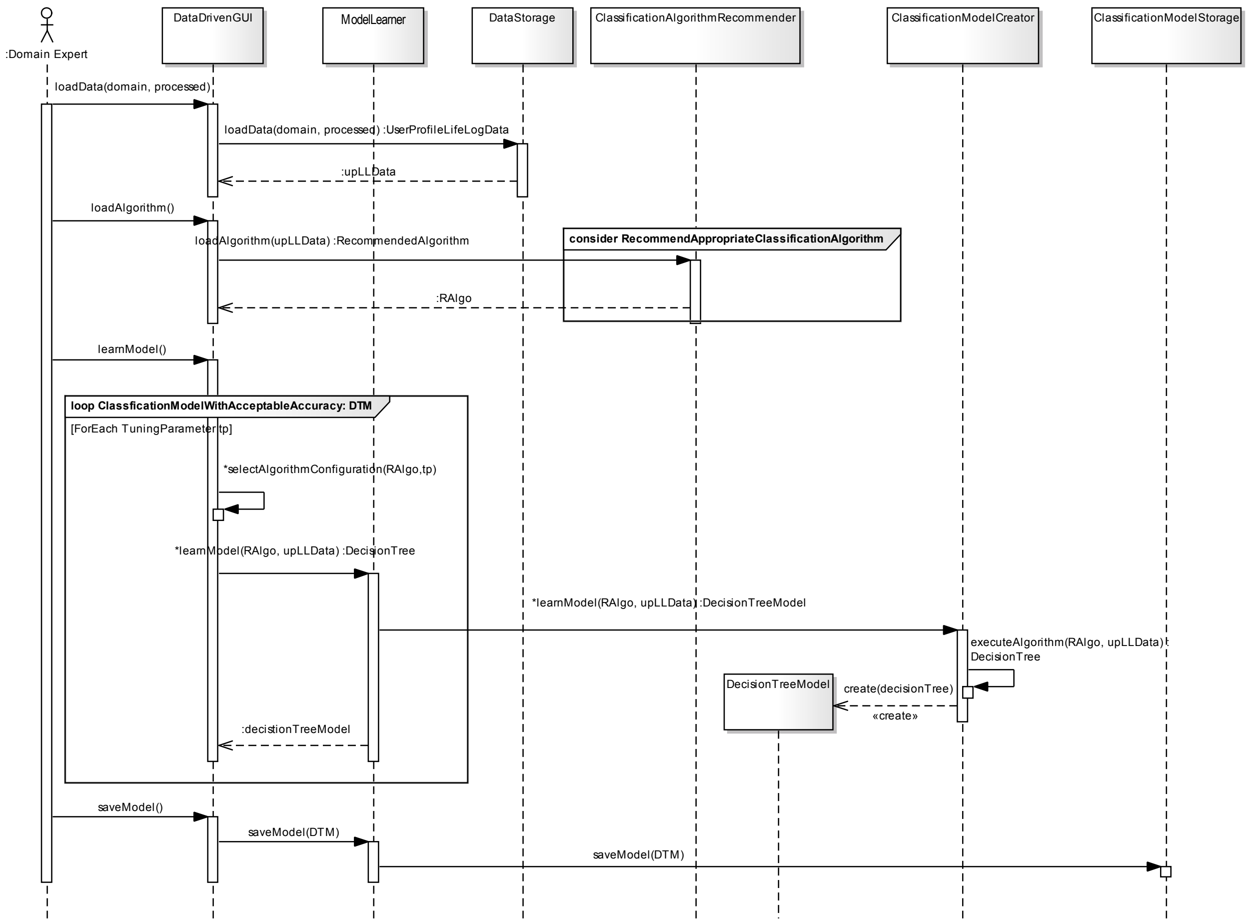


Use Case ID:	**KCL-UC-13**

Use Case Name:	Learn classification model		
Created By:	Maqbool Ali	**Last Updated By:**	Maqbool Hussain
Date Created:	10-07-2015	**Last Revision Date:**	27-07-2015
Actors:	Domain Expert		
Description:	An expert wants to see hidden knowledge from user profile lifelog data that can be explored by model learning mechanism with the help of learning method as well as processed data.		
Trigger:	Learn model required to explore hidden knowledge		
Preconditions:	• System has loaded the prepared user profile lifelog data		
Postconditions:	System will build the classification model (decision tree)		
Normal Flow:	1. Domain expert loads the user profile lifelog processed data for selected domain 2. System loads the corresponding processed data 3. Domain expert invokes the "Recommend appropriate classification algorithm" use case by providing processed data to load the appropriate learning algorithm 4. System loads the appropriate decision tree learning algorithm 5. Domain expert select the algorithm tuning parameters of selected algorithm for further improving the results 6. System applies the tuning parameters on selected algorithm and computes the learning accuracy after learning the user profile lifelog processed data 7. Repeat the step 5-6 until required learning accuracy is achieved. 8. Domain expert finalizes the classification model with acceptable accuracy and saves the model. 9. System saves the decision tree learning model.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	KCL-UC-14 (Recommend appropriate classification algorithm)		
Frequency of Use:	When new service is required and mining mind have sufficient data for classification model creation		
Special Requirements:	N/A		

Assumptions:	N/A
Notes and Issues:	Acceptable Accuracy: Depends on criticality of the domain. For example, classification model for clinical domain needs high accuracy, while for the case of nutrition domain, high accuracy is not critical.

Sequence Diagram:

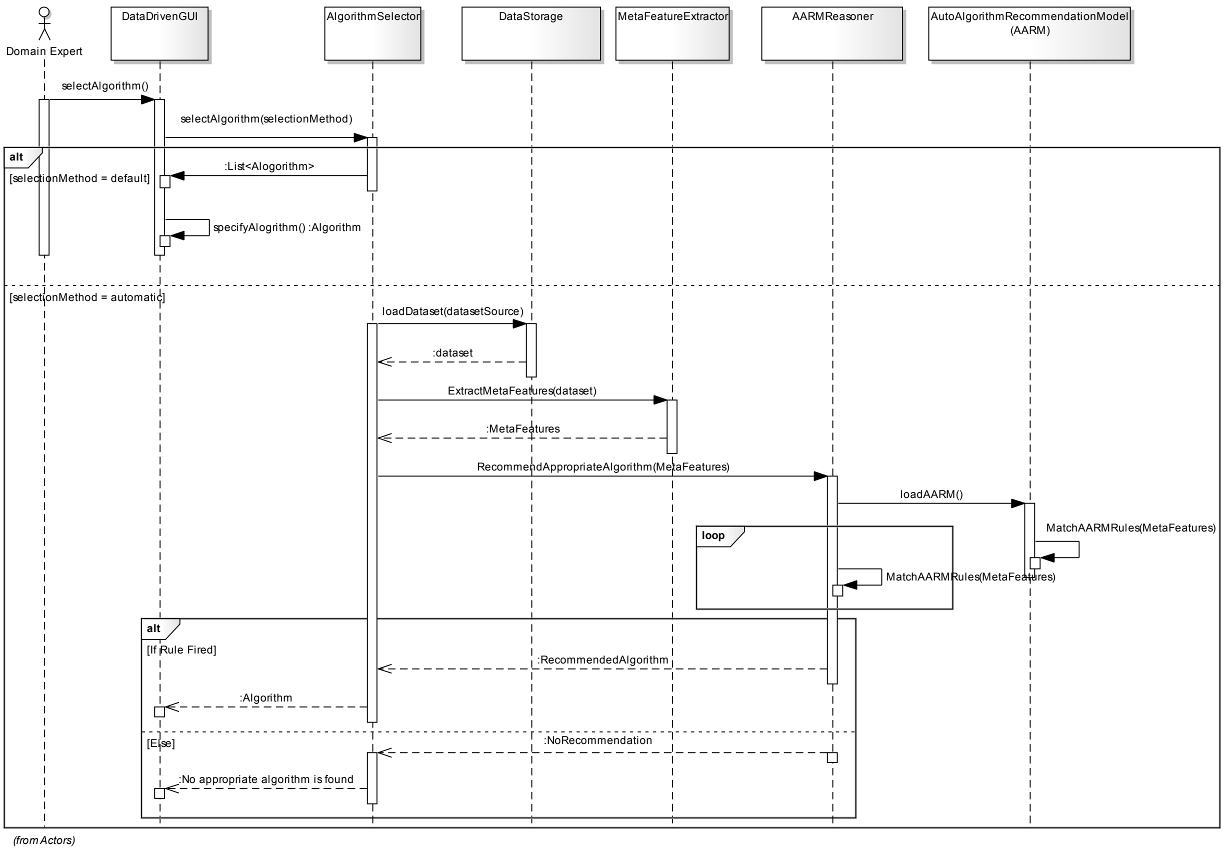


Use Case ID:	**KCL-UC-14**
Use Case Name:	Recommend appropriate classification algorithm

<table>
<tr><td>Created By:</td><td>Rahman Ali</td><td>Last Updated By:</td><td>Rahman Ali</td></tr>
<tr><td>Date Created:</td><td>16-07-2015</td><td>Last Revision Date:</td><td>27-07-2015</td></tr>
<tr><td>Actors:</td><td colspan="3">Domain expert</td></tr>
<tr><td>Description:</td><td colspan="3">For building classification model for the user new dataset, AARM shall automatically recommend appropriate classification algorithm.</td></tr>
<tr><td>Trigger:</td><td colspan="3">When domain expert wants to build a classification model.</td></tr>
<tr><td>Preconditions:</td><td colspan="3">• AARM is plugged into the data driven environment
• Meta-features extractor is plugged into the data driven environment
• New dataset is stored in local machine, structured in .arff file format</td></tr>
<tr><td>Postconditions:</td><td colspan="3">The recommended appropriate classification algorithm can be used for building classification model</td></tr>
<tr><td>Normal Flow:</td><td colspan="3">1. Domain expert loads new dataset (.arff file) from the data driven datasets storage using data driven environment.
2. System extracts meta-features of the new dataset by including KCL-UC-07 (alternate flow)
3. Domain experts provides meta-features to system for recommending appropriate classification algorithm
4. System performs meta-reasoning over integrated AARM using the following steps;
a. Starts matching each meta-feature value of the new dataset with condition attributes of each rule
b. If matched, fires the rule, recommend right hand side of the rule as the appropriate algorithm
c. Else, display a message “could not recommend”</td></tr>
<tr><td>Alternative Flows:</td><td colspan="3">1a. 4c(a) If AARM not available or have no acceptable recommendation accuracy, use Weka experimenter.</td></tr>
<tr><td>Exceptions:</td><td colspan="3">N/A</td></tr>
<tr><td>Includes:</td><td colspan="3">KCL-UC-07 (alternate flow)</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Frequently, when domain expert needs to select appropriate algorithm for his/her dataset</td></tr>
<tr><td>Special Requirements:</td><td colspan="3">N/A</td></tr>
<tr><td>Assumptions:</td><td colspan="3">N/A</td></tr>
</table>

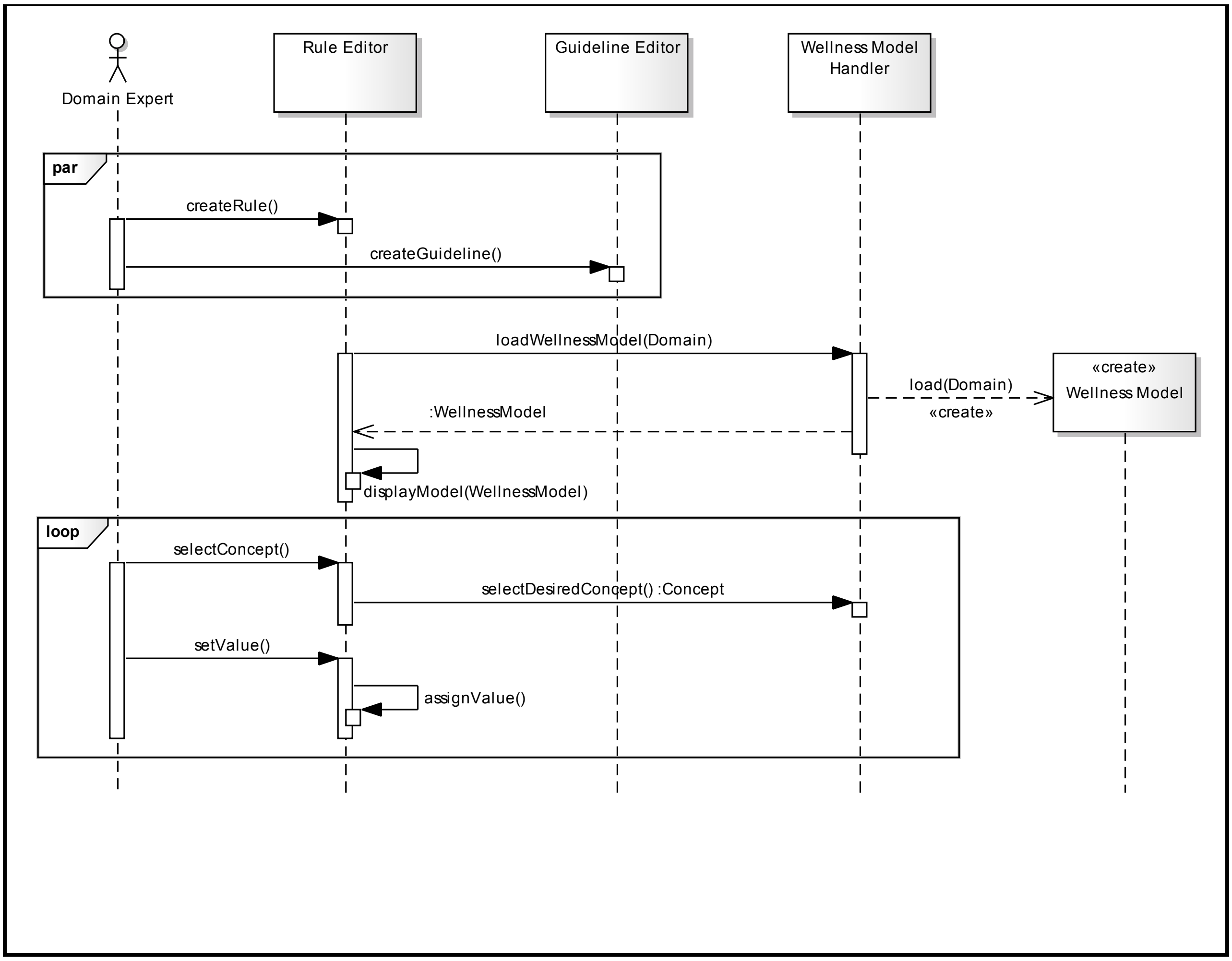
Notes and Issues:	N/A

Sequence Diagram:



Use Case ID:	**KCL-UC-15**		
Use Case Name:	Manage concepts of domain model		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali, Maqbool Hussain
Date Created:	27-07-2015	**Last Revision Date:**	27-07-2015
Actors:	Domain Expert		
Description:	The concepts of wellness domain shall be used in creation of rules and generation of guidelines in tree format. The domain expert shall easily select the required concepts from		

	wellness domain model.
Trigger:	Domain Model will be loaded during rule creation or guideline creation.
Preconditions:	The expert be authenticated with full access of concepts management in domain model
Postconditions:	The right concept shall be added or edited at the right location in wellness model
Normal Flow:	1. Domain expert creates rule (using KCL-UC-10) or creates guideline (using KCL-UC-03). 2. System loads domain model for corresponding domain. 3. Domain expert selects concepts from loaded domain model. 4. System associate domain concept to part of rule or guideline tree. 5. Domain expert assigns value to selected concept. 6. System assigns corresponding value to selected concept and show it in rule or guideline tree. 7. Step 4-6 are repeated till rule or guideline is finished.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever domain expert want to add or edit the concepts in wellness model
Special Requirements:	N/A
Assumptions:	Wellness model repository in the system is exist.
Notes and Issues:	If wellness model storage does not exist in system the administrator shall build the wellness model storage first and configure with system.
Sequence Diagram:	

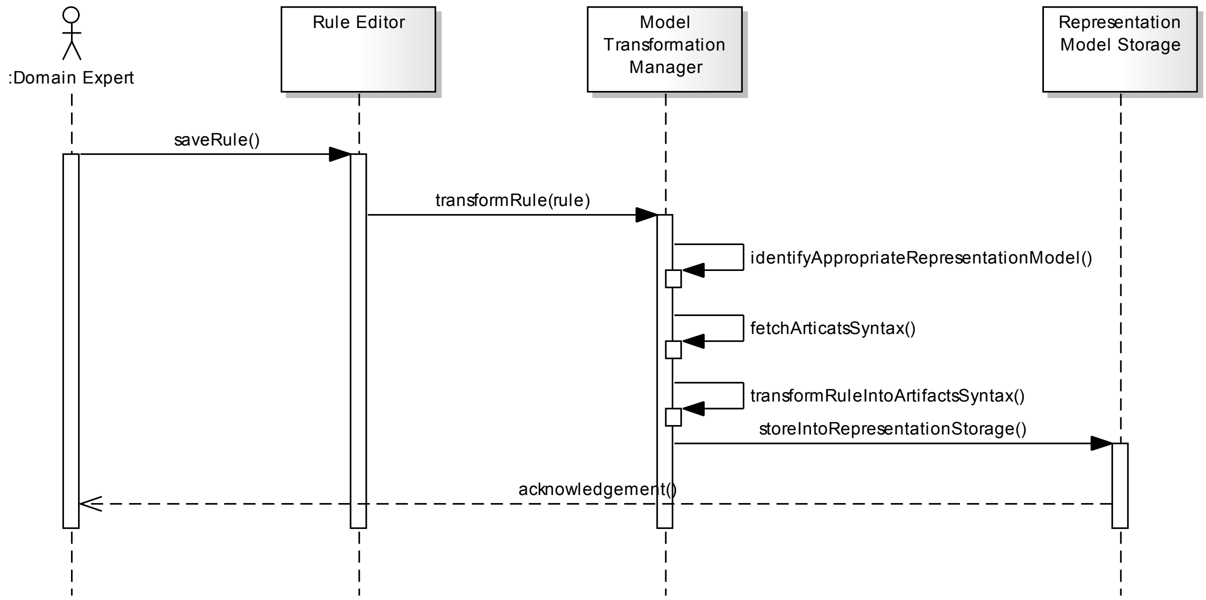


Domain Expert
Rule Editor
Guideline Editor
Wellness Model Handler
par
createRule()
createGuideline()
loadWellnessModel(Domain)
«create»
Wellness Model
load(Domain)
«create»
:WellnessModel
displayModel(WellnessModel)
loop
selectConcept()
selectDesiredConcept() :Concept
setValue()
assignValue()

Use Case ID:	**KCL-UC-16**		
Use Case Name:	Transform Knowledge Rule		
Created By:	Taqdir Ali and Maqbool Hussain	**Last Updated By:**	Maqbool Hussain
Date Created:	27-07-2015	**Last Revision Date:**	27-07-2015
Actors:	Domain Expert		
Description:	The new created rules are needed to transform to some computer interpretable, executable format for execution as well as to shareable, standard format for maintenance and sharing with other organizations.		
Trigger:	Whenever domain expert wants to store the created or updated rule.		
Preconditions:	The expert created rule successfully and the system validated the rule.		
Postconditions:	System shall transform the created and validated rule into appropriate representation.		
Normal Flow:	1. Domain expert save the new created rule or update the existing rule. 2. The system identifies the appropriate representation model 3. Fetch the artifacts of the identified representation model 4. Transforms the rule into the artifacts and syntax of the identified representation model. 5. The rule in the representation model is stored into the repository.		
Alternative Flows:	N/A		
Exceptions:	N/A		
Includes:	N/A		
Frequency of Use:	Whenever domain expert want to save rule		
Special Requirements:	N/A		
Assumptions:	N/A		

Notes and Issues:	Appropriate representation configuration is challenging task.

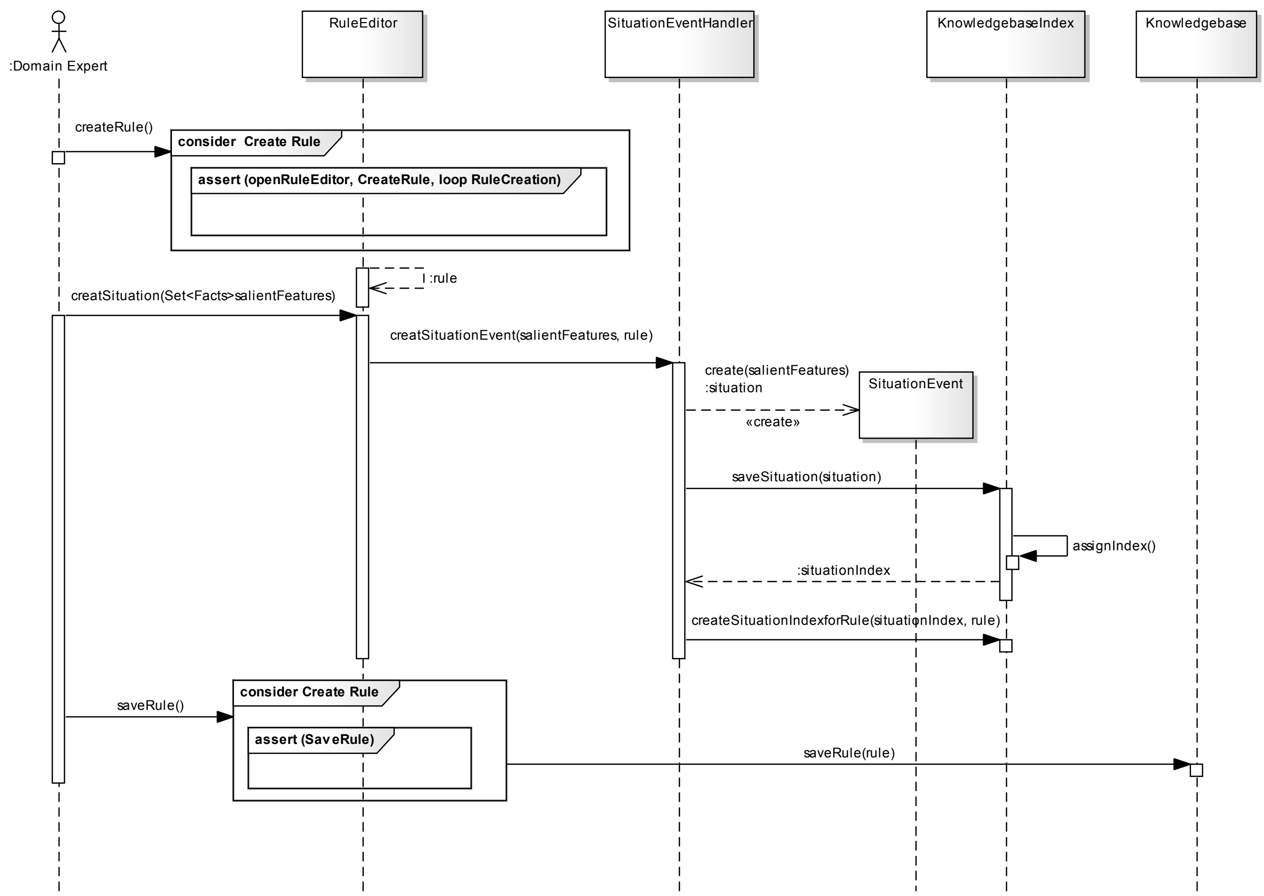
Sequence Diagram:



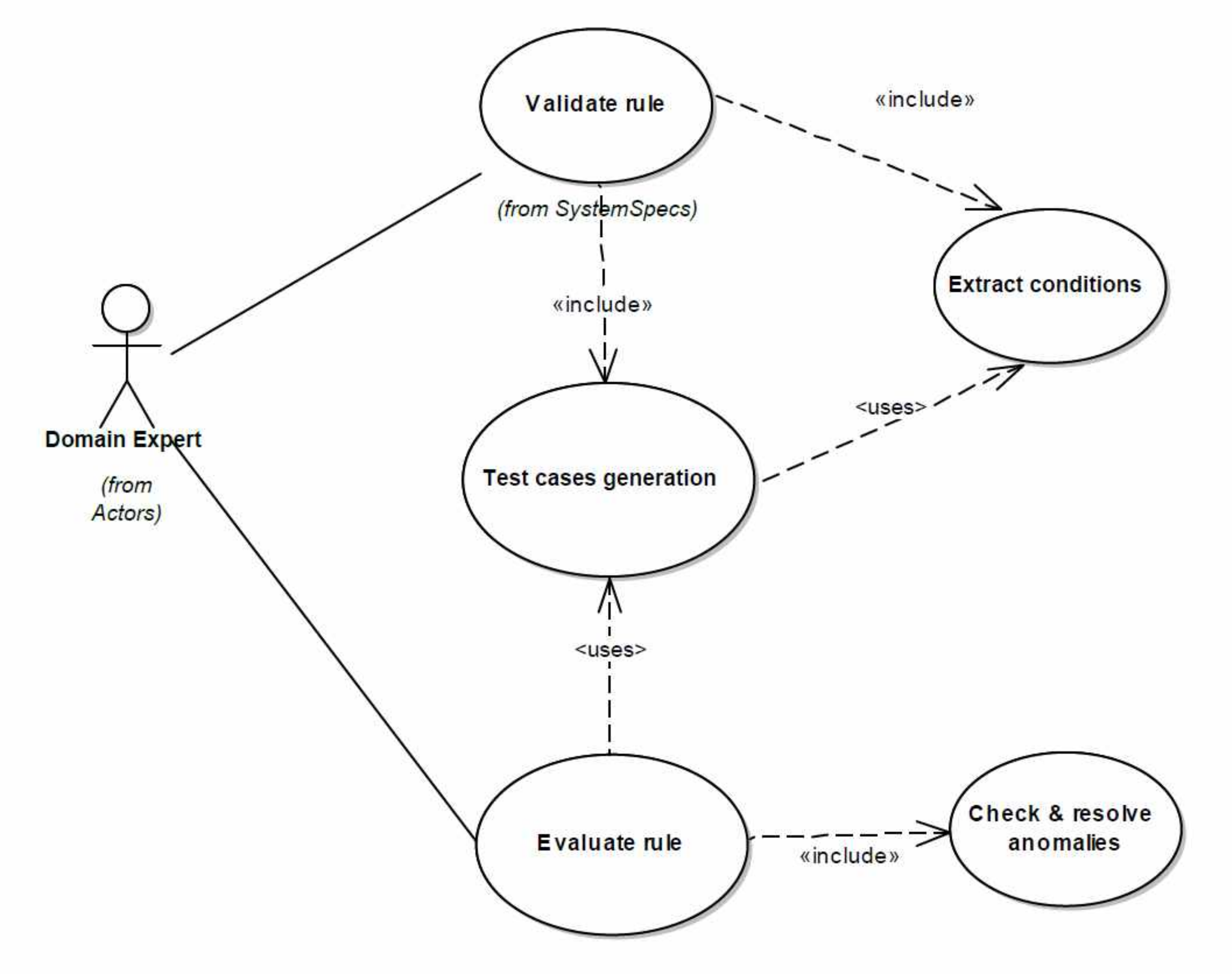
<table>
<tr><td>Use Case ID:</td><td colspan="3">KCL-UC-17</td></tr>
<tr><td>Use Case Name:</td><td colspan="3">Create Situation Event</td></tr>
<tr><td>Created By:</td><td>Maqbool Hussain</td><td>Last Updated By:</td><td>Maqbool Hussain</td></tr>
<tr><td>Date Created:</td><td>29-07-2015</td><td>Last Revision Date:</td><td>29-07-2015</td></tr>
<tr><td>Actors:</td><td colspan="3">Domain Expert</td></tr>
<tr><td>Description:</td><td colspan="3">Situation Event is important features of mining mind which includes set of associated recommendation rules. Situation event is created and the rule is indexed in knowledgebase based on situation event.</td></tr>
<tr><td>Trigger:</td><td colspan="3">Whenever domain expert wants to store the created or updated rule.</td></tr>
<tr><td>Preconditions:</td><td colspan="3">The rule has salient features based on which the rule can be indexed.</td></tr>
<tr><td>Postconditions:</td><td colspan="3">Rule is saved into knowledgebase
Rule is indexed based on the created situation event</td></tr>
<tr><td>Normal Flow:</td><td colspan="3">Domain expert start creating rule;<ol type="a">
Performs steps 1-5 in KCL-UC-10.
Selects salient features (indicating as event) from conditions of the rule.
The system performs following actions;<ol type="a">
Create situation event with salient features.
Saves the situation event and assign index (generate if situation event is not exist in knowledgebase index).
Index the created rule with situation event.
Domain expert saves the rule by performing steps 5-6 in KCL-UC-10.
System saves the rule and index of the rule.</td></tr>
<tr><td>Alternative Flows:</td><td colspan="3">N/A</td></tr>
<tr><td>Exceptions:</td><td colspan="3">N/A</td></tr>
<tr><td>Extends:</td><td colspan="3">Create Rule (KCL-UC-10)</td></tr>
<tr><td>Frequency of Use:</td><td colspan="3">Whenever domain expert want to save rule</td></tr>
</table>

Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	

Sequence Diagram:

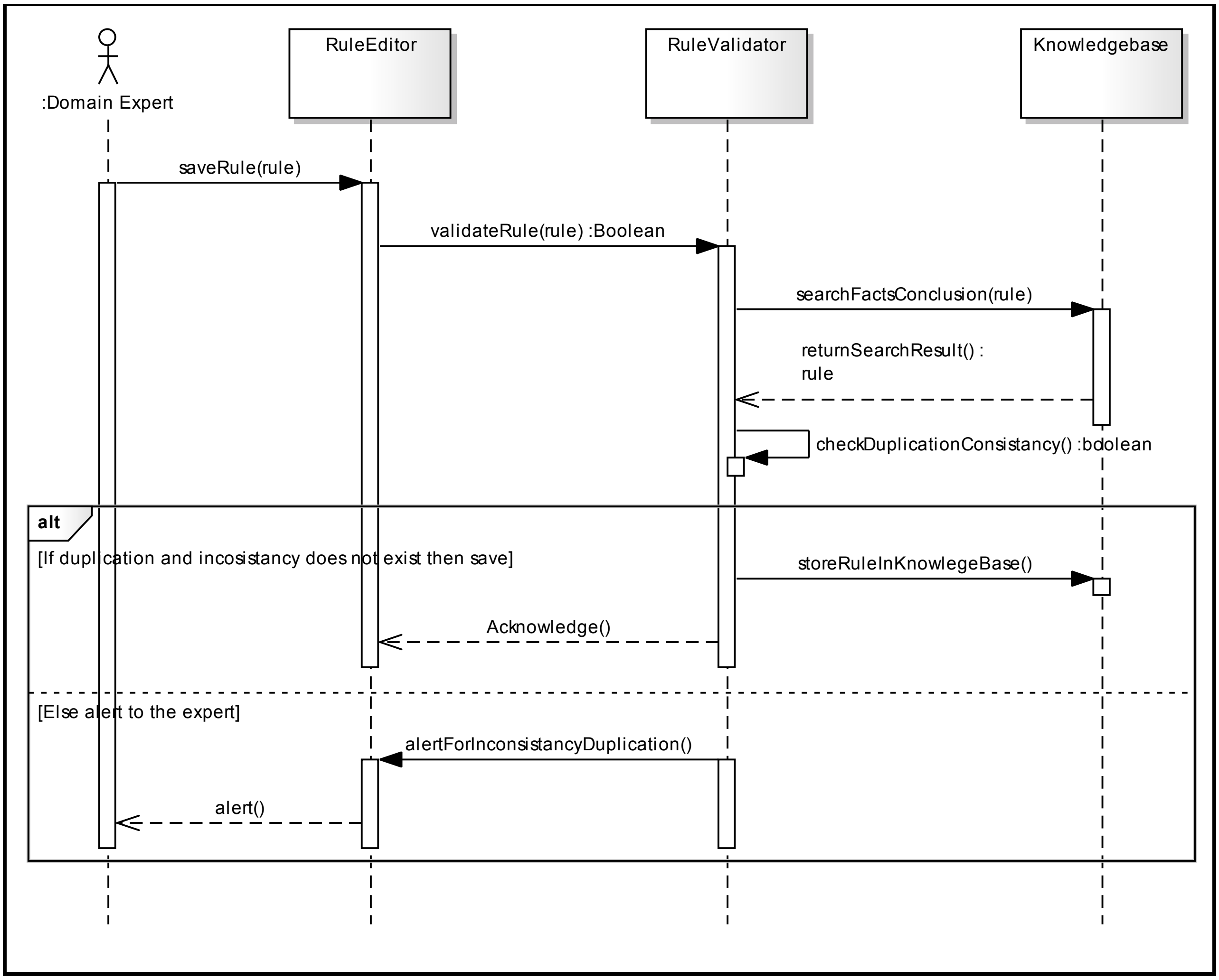


Rule Validator: Use case diagram



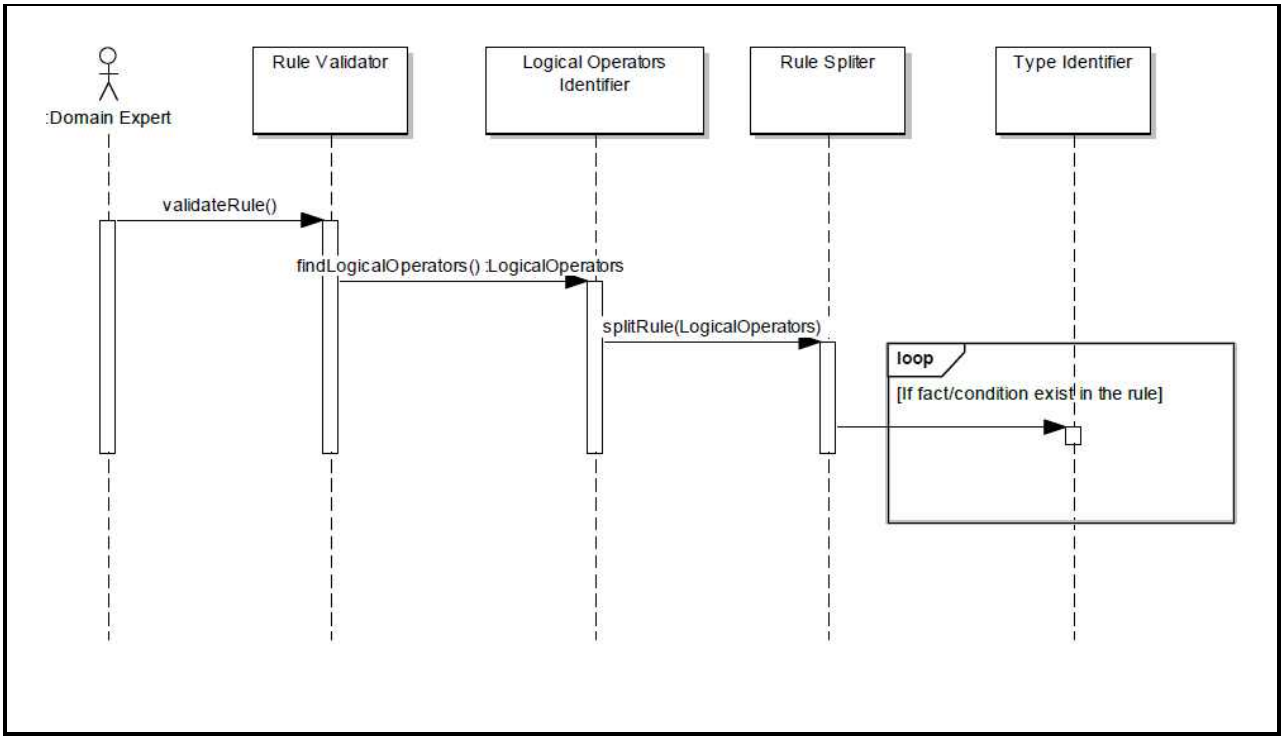
Use Case ID:	**KCL3-UC-18**		
Use Case Name:	Validate Rule		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	04-03-2017	**Last Revision Date:**	05-03-2017
Actors:	Domain Expert		
Description:	In new rules creation and editing existing rules, duplication and inconsistency may occur. The validation is needed to validate and find the duplication and inconsistency among the rules.		
Trigger:	• When new rule is going to create. • When existing rule is going to update		

Preconditions:	The rule creation and editing process completed by physician successfully.
Postconditions:	The validated rule shall be saved into knowledge base.
Normal Flow:	3. Domain expert saves the created rule. 4. System validate rule for inconsistency and duplication as follows a. Fetch the facts and conclusion of existing rules. b. The new or updated rule approved for having no inconsistency and duplication. c. Created rule stores into the rules repository. d. Acknowledge the expert to save the rule successfully.
Alternative Flows:	3b. The created rule is found having inconsistency or duplication with existing rules in the rules repository a. The system produces alert the inconsistency or duplication of the rule with existing rules in repository. b. Domain expert review the alert message and correct the created rule. c. Step 1 and Step 2 of normal flow is executed.
Exceptions:	N/A
Includes:	1. Extract conditions 2. Test cases generation
Frequency of Use:	Whenever domain expert want to save the rule
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.
Sequence Diagram:	



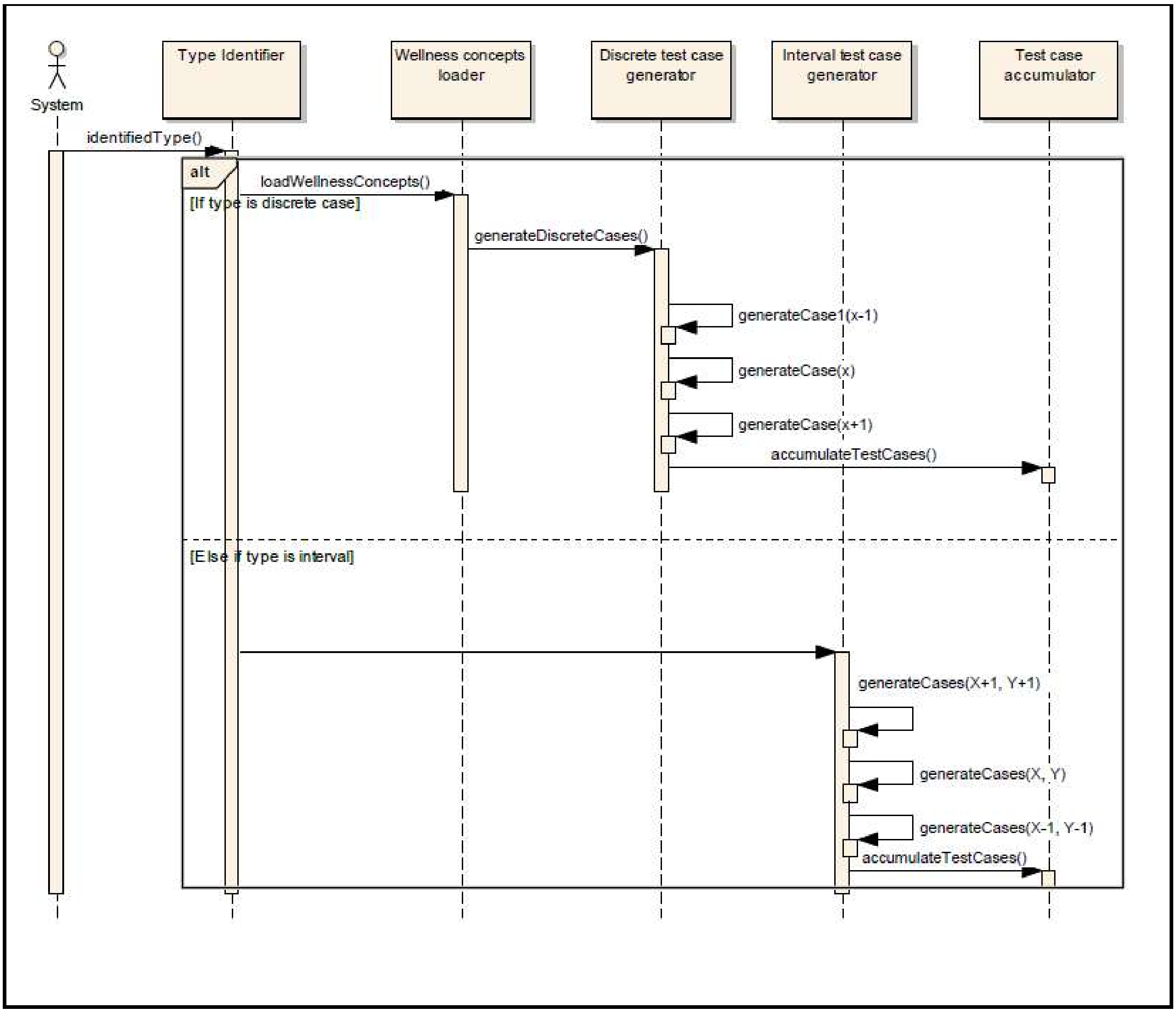
Use Case ID:	**KCL3-UC-19**		
Use Case Name:	Extract conditions		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	04-03-2017	**Last Revision Date:**	05-03-2017
Actors:	Domain Expert		
Description:	Whenever the rule is validated, the test cases are generated, the test cases depends on conditions of the rule. Therefore, all the facts and conditions are needed to extract from rules.		

Trigger:	• When new rule is going to validate. • When existing rule is going to validate.
Preconditions:	The rule creation and editing process completed by physician successfully.
Postconditions:	Multiple pairs of facts and conditions are extracted.
Normal Flow:	1. System fetch the rule, which is being to be added or updated 2. System shall identify the logical operators in the rule. 3. System splits the rule based on the identified logical operators 4. In splat conditions, find the type of keys and their values. a. If type is discrete then execute non-interval test case generation.
Alternative Flows:	4b. If type is interval then a. Execute interval case generator
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever domain expert want to validate the rule
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.
Sequence Diagram:	



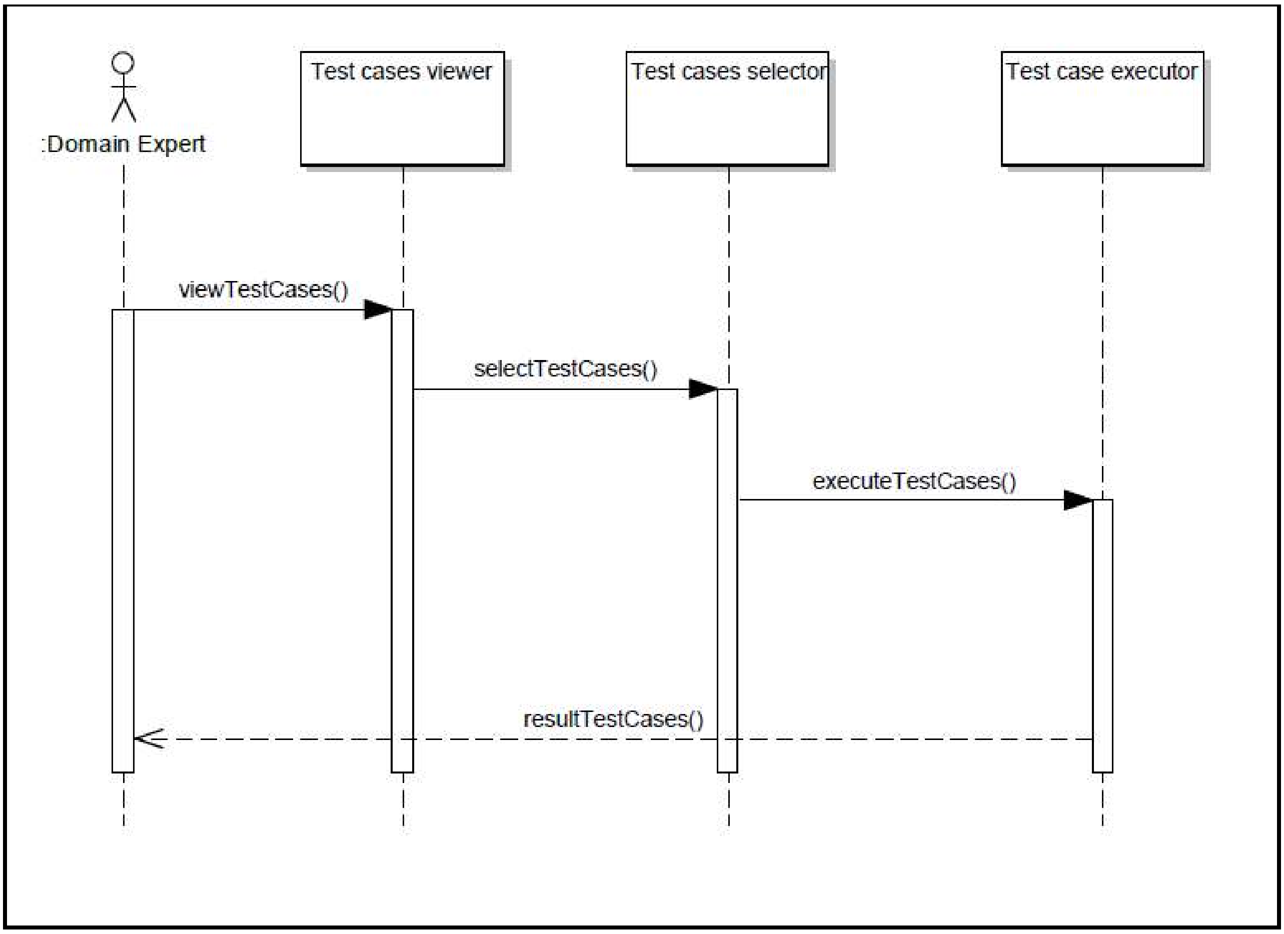
Use Case ID:	**KCL3-UC-20**		
Use Case Name:	Test cases generation		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	04-03-2017	**Last Revision Date:**	05-03-2017
Actors:	Domain Expert		
Description:	Whenever the rule is validated, the test cases are generated, the new created rule or existing modified rule is checking for duplications and conflict using the generated test cases.		
Trigger:	• When new rule is going to validate. • When existing rule is going to validate.		
Preconditions:	The rule conditions and facts are identified and extracted successfully.		
Postconditions:	Multiple test cases shall generate after successful process.		

<table>
<tr><td>Normal Flow:</td><td>1. If type of extracted condition or fact is discrete
a. System shall fetch the possible values from wellness model
b. Generate matched test cases with respect to the extracted condition.
c. Generate unmatched conditions with respect to the extracted condition.
d. Store the generated test cases with corresponding conditions in test case base.</td></tr>
<tr><td>Alternative Flows:</td><td>1. (b) If type of extracted condition or fact is interval
a. System finds the type of interval
b. If interval is single side infinite interval or exact value interval then generate following 3 test cases.
i. Generate case for X-1
ii. Generate case for X
iii. Generate Case for X+1
c. If interval is double sided finite interval then generate following 6 test cases.
i. Generate case for X-1
ii. Generate case for X
iii. Generate Case for X+1
iv. Generate case for Y-1
v. Generate case for Y
vi. Generate Case for Y+1</td></tr>
<tr><td>Exceptions:</td><td>N/A</td></tr>
<tr><td>Includes:</td><td>N/A</td></tr>
<tr><td>Frequency of Use:</td><td>Whenever domain expert want to validate the rule</td></tr>
<tr><td>Special Requirements:</td><td>N/A</td></tr>
<tr><td>Assumptions:</td><td>N/A</td></tr>
<tr><td>Notes and Issues:</td><td>Finding duplication using facts of condition and conclusion in existing rules is challenging task.</td></tr>
<tr><td colspan="2">Sequence Diagram:</td></tr>
</table>



Use Case ID:	**KCL3-UC-21**		
Use Case Name:	Evaluate rule		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	04-03-2017	**Last Revision Date:**	05-03-2017

Actors:	Domain Expert
Description:	Whenever the rule is to be validated, the new created or existing modified rule is needed to test with respect to generated test cases, then evaluate the rule with those test cases
Trigger:	• When new rule is going to validate. • When existing rule is going to validate.
Preconditions:	The test cases are successfully generated and executed for rule which is going to evaluate.
Postconditions:	Find the duplicate and conflicted rules based on executed test cases.
Normal Flow:	1. Domain experts views all the auto generated test cases on viewer. 2. Domain expert selects the test cases to execute and test on the created rule 3. Domain experts executes the selected test cases 4. The system gives results of executed test cases based on the included use case "check and resolve anomalies".
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever domain expert want to validate the rule
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.
Sequence Diagram:	



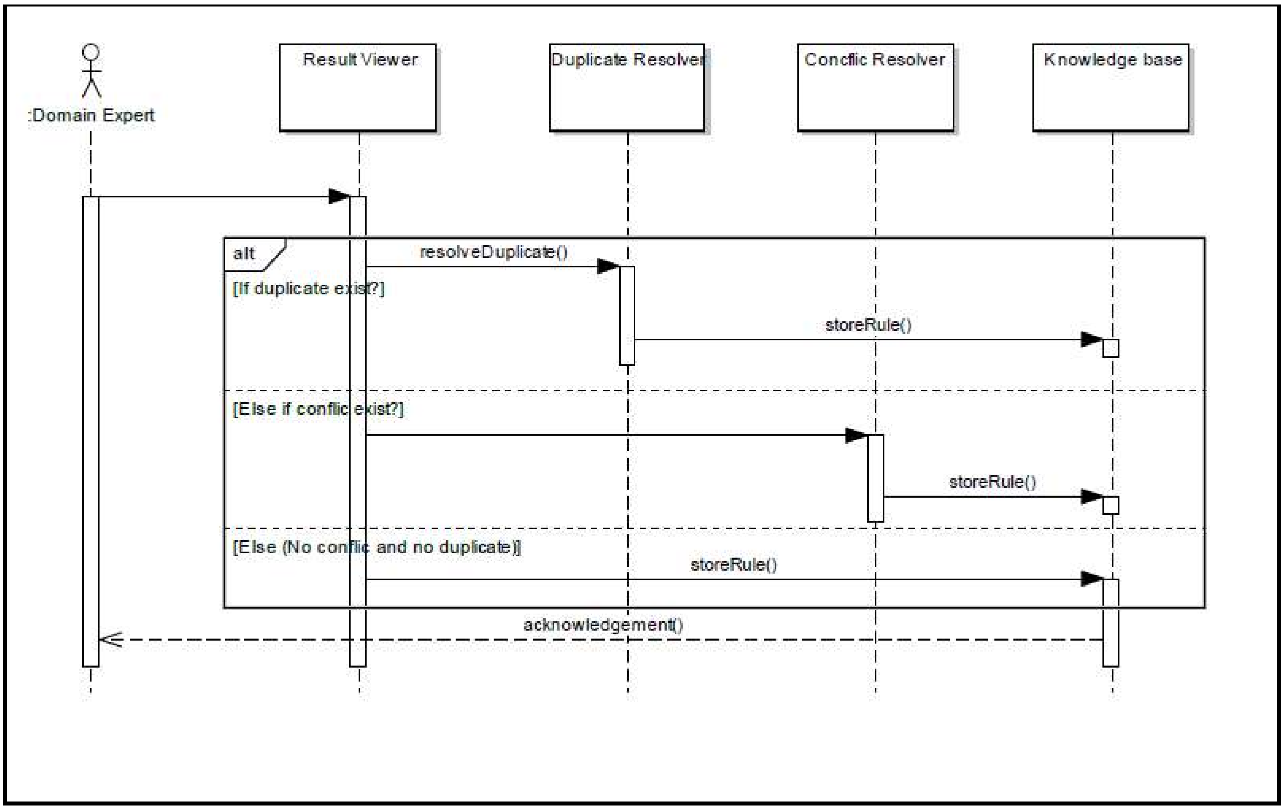
Use Case ID:	**KCL3-UC-22**		
Use Case Name:	Check and resolve anomalies		
Created By:	Taqdir Ali	**Last Updated By:**	Taqdir Ali
Date Created:	04-03-2017	**Last Revision Date:**	05-03-2017
Actors:	Domain Expert		
Description:	Whenever the rule is to be validated, the executed test cases checks and identifies the anomalies in form of duplicate and conflict. The domain experts identified and resolve those anomalies.		

Trigger:	• When new rule is going to validate. • When existing rule is going to validate.
Preconditions:	The test cases are successfully generated and executed for rule which is going to evaluate.
Postconditions:	Find and resolve the duplicate and conflicted rules based on executed test cases.

Normal Flow:	1. System shows the results of execution of test cases on the creation rule or modification rule.
	2. Domain experts views the result messages on result viewer for finding duplication and conflict
	3. If duplicate rules exist
	    a. Discard the new created rule.
	4. If conflict exists in the created rule
	    a. Find the conflicting attributes and facts in the rule.
	    b. Resolve the conflict and gives the priority to rules
	5. Store the rules into the knowledge base.

Alternative Flows:	3 (a) If there is no duplicate
	    a. Check the rule for conflict.
	4 (a) If there is no conflict
	    a. Store the rule into the knowledge base

Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever domain expert want to validate the rule
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	Finding duplication using facts of condition and conclusion in existing rules is challenging task.
Sequence Diagram:	


Result Viewer
Duplicate Resolver
Concflic Resolver
Knowledge base
:Domain Expert
alt
resolveDuplicate()
[If duplicate exist?]
storeRule()
[Else if conflic exist?]
storeRule()
[Else (No conflic and no duplicate)]
storeRule()
acknowledgement()

Section 5.4

Service Curation Layer(SCL) Requirement Specification

1.1 Service Curation Layer

1.1.1 SCL Functional Requirement

Requirements #ID	Description
SCL-FR-01	The layer shall receive the service request from user application, third party application, or mining mind platform generated events
SCL -FR-02	The layer shall retrieve data from intermediate database (user profile, lifelog, and environmental variables)
SCL -FR-03	The layer shall retrieve production knowledge from knowledge base
SCL -FR-04	The layer shall identify the unresolved user requests and notify the corresponding layer for missing knowledge
SCL -FR-05	The layer shall deliver the results to the service requester and to corresponding layer of mining mind for persistence
SCL -FR-06	The layer shall generate goal based recommendation
SCL -FR-07	The layer shall personalize the recommendation based on location and weather information.
SCL -FR-08	The layer shall provide take care of user preference-based recommendation through continuous feedback from the user.

1.1.2 SCL Non-Function Requirements

Requirements #ID	Description
SCL -NFR-01	The service request shall include the user identification required to prepare the data request
SCL -NFR-02	The user application and host layer of the services shall connect to the internet

1.1.3 SCL 3 Terms and Definitions

Terms	Description
SCL	Service Curation Layer
DCL	Data Curation Layer

KCL	Knowledge Curation Layer
SL	Supporting Layer
Reasoning	The process of producing recommendations
RBR	Rule Base Reasoning
Case	A case is collection of data/information that represent a complete state of a subject in a particular time
CBR	Case Base Reasoning
Hybrid Reasoning	A combined reasoning of RBR and CBR
Unresolved Case	A new case for which the existing knowledge is insufficient to solve
Production Knowledge	The knowledge is in production environment and is ready for execution
Recommendation	An actionable statement provided to the subject for healthy habit induction
Fact	An informative statement provided to the subject for education
Goal	A target that a subject is intend to achieve
Achievement	The measurement of the subject status so far towards achieving the goal
Interpreted Recommendation	The recommendation passed through the process of interpretations
Environmental Variables	Environment variables represent the factors related to environment rather than user such as weather, time, season etc.
Service	An outcome of the system in which a subject is interested such as recommendation, facts, alerts, notifications.

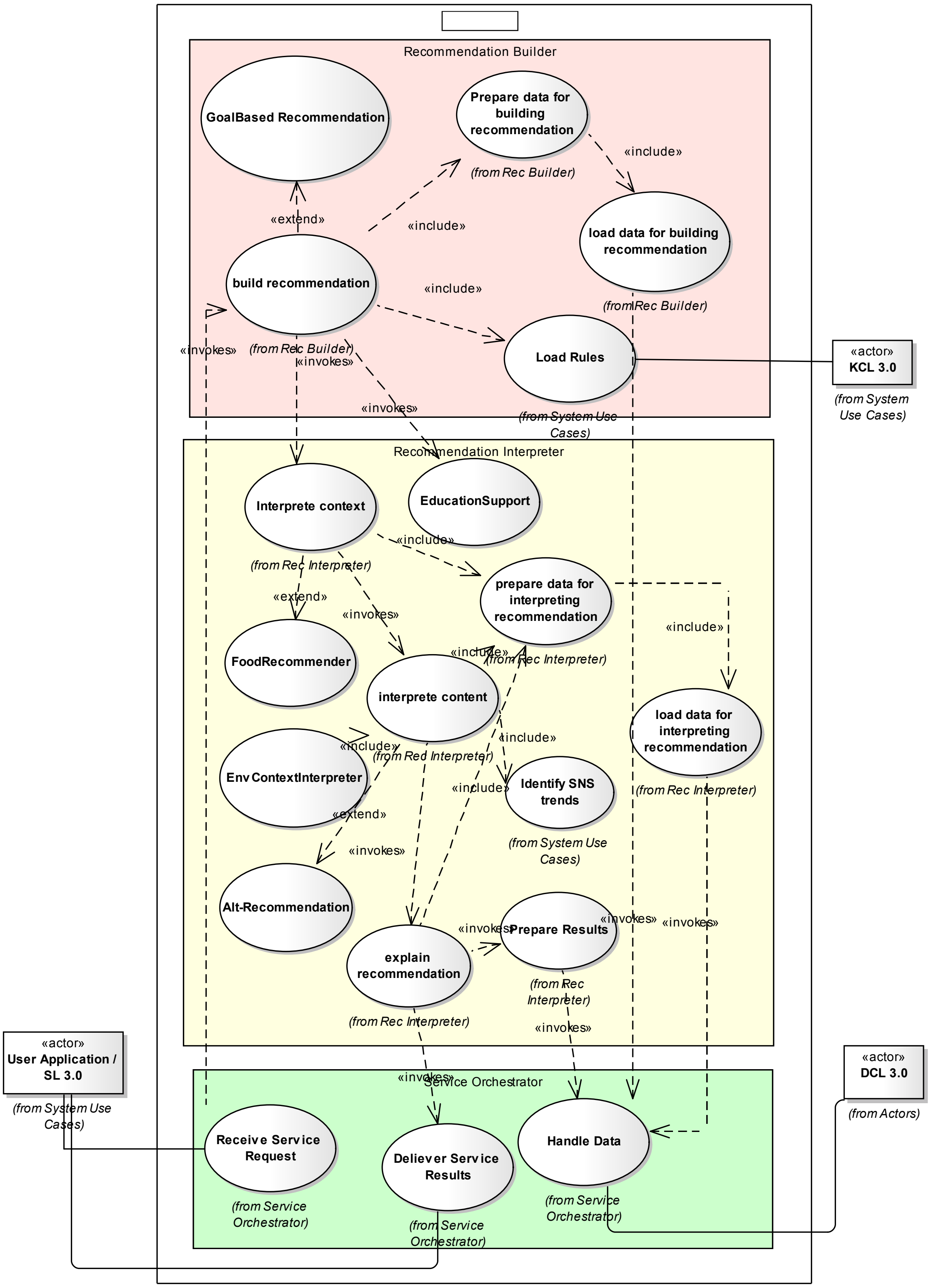
1.1.4 Use Cases

1.1.4.1 Use Case List

Use Case #ID	Description
SCL-UC-01	Load data for building recommendation
SCL-UC-02	Prepare data for building recommendation
SCL-UC-03	Load Rules

SCL-UC-04	Build Recommendation
SCL-UC-05	Receive Production Knowledge
SCL-UC-06	Report Unresolved Case
SCL-UC-07	Load data for interpreting recommendation
SCL-UC-08	Prepare data for interpreting recommendation
SCL-UC-09	Interpret Context
SCL-UC-10	Interpret Content
SCL-UC-11	Explain recommendations
SCL-UC-12	Prepare Results
SCL-UC-13	Receive service request
SCL-UC-14	Handle Data
SCL-UC-15	Deliver service results
SCL-UC-16	Goal-based recommendation Generation
SLC-UC-17	Provide Educational Support for user awareness
SCL-UC-18	Location- and weather-based Personalized recommendation
SCL-UC-19	User Preference-based refined recommendations
SCL-UC-20	Provide Alternative Recommendation

1.1.4.2 Use Case Diagram



Recommendation Builder
GoalBased Recommendation
Prepare data for building recommendation
(from Rec Builder)
«include»
«extend»
«include»
load data for building recommendation
build recommendation
«include»
(from Rec Builder)
Load Rules
«invokes»
(from Rec Builder)
«invokes»
«invokes»
(from System Use Cases)
«actor»
KCL 3.0
(from System Use Cases)
Recommendation Interpreter
Interprete context
EducationSupport
«include»
(from Rec Interpreter)
prepare data for interpreting recommendation
«extend»
«invokes»
«include»
FoodRecommender
«include»
(from Rec Interpreter)
interprete content
load data for interpreting recommendation
«include»
Env ContextInterpreter
«include»
(from Rec Interpreter)
Identify SNS trends
«include»
(from Rec Interpreter)
«extend»
(from System Use Cases)
«invokes»
Alt-Recommendation
«invokes»
Prepare Results
«invokes»
«invokes»
explain recommendation
(from Rec Interpreter)
(from Rec Interpreter)
«invokes»
«actor»
User Application / SL 3.0
«invokes»
Service Orchestrator
«actor»
DCL 3.0
(from System Use Cases)
(from Actors)
Receive Service Request
Handle Data
Deliever Service Results
(from Service Orchestrator)
(from Service Orchestrator)
(from Service Orchestrator)

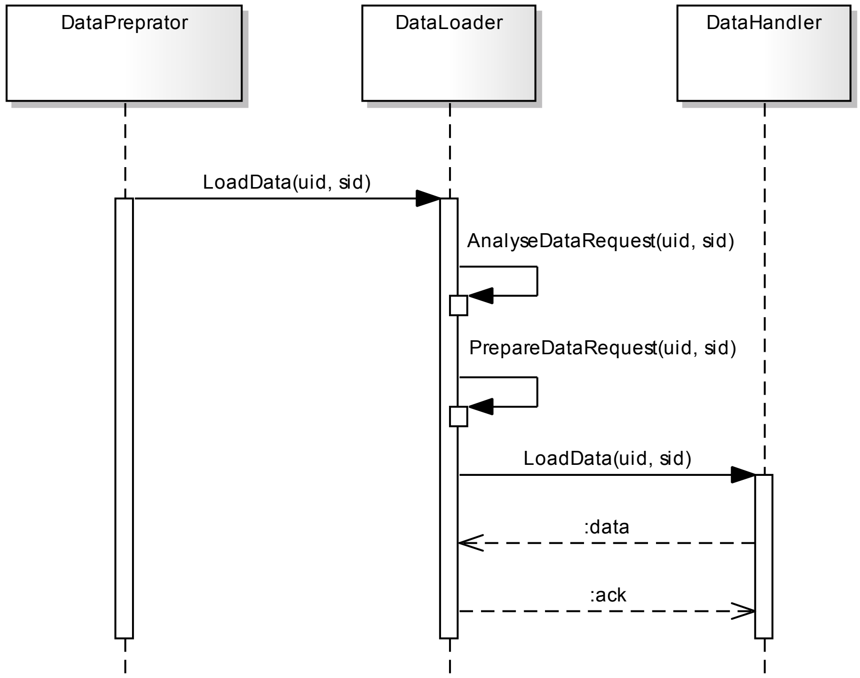
1.1.4.3 Description

Use Case ID:	SCL-UC-01		
Use Case Name:	Load data for building recommendation		
Created By:	Rahman Ali	**Last Updated By:**	Rahman Ali
Date Created:	July 14, 2015	**Last Revision Date:**	July 28, 2015

Actors:	SCL-UC-02 (Prepare Data)
Description:	Retrieving user profile and lifelog data is required for reasoning to generate recommendation. This data is retrieved using Data Handler of the Service Orchestrator.
Trigger:	Triggered when a new service request is received from the user application or DCL.
Preconditions:	User profile and lifelog data is available in user lifelog.
Postconditions:	User profile and lifelog data is successfully retrieved and prepared for reasoner to process.
Normal Flow:	1. Data Preprator sends request for loading data 2. Data Loader receives the request and performs the following tasks; a. Analyses the request and user for the appropriate data loading b. Prepare separate requests for user lifelog data 3. Data Loader sends analyses request to Data Handler 4. Data Handler provides the data to Data Loader
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Very frequent; repeated for every service request

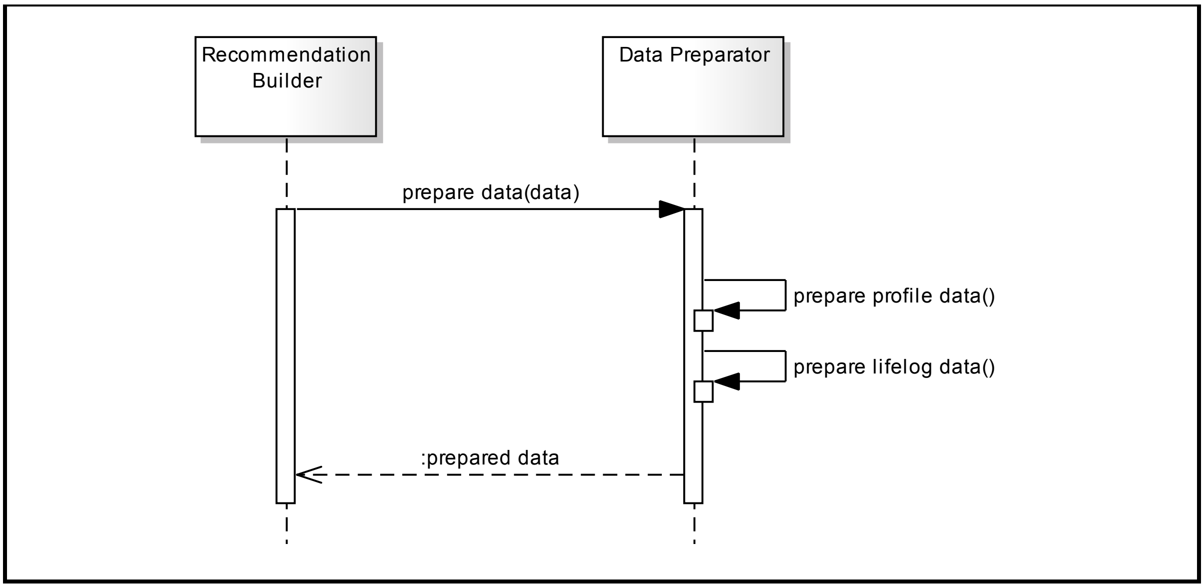
Special Requirements:	N/A
Assumptions:	Service Orchestrator knows the required data for each registered service.
Notes and Issues:	N/A

Sequence Diagram



Use Case ID:	SCL-UC-02		
Use Case Name:	Prepare data for building recommendation		
Created By:	Rahman Ali	**Last Updated By:**	Rahman Ali
Date Created:	July 14, 2015	**Last Revision Date:**	July 28, 2015

Actors:	SCL-UC-04 (Build Recommendation)
Description:	Knowledge based reasoning requires prepared data to execute the rules during the reasoning process.
Trigger:	Triggered when new service request is made for generating recommendations
Preconditions:	User profile and lifelog data is loaded into RB 2
Postconditions:	User prepared data is readily available for reasoner to process.
Normal Flow:	1. Recommendation Builder sends data preparation request to Data Preparator along with the loaded data 2. Data Preparator prepares profile data 3. Data Preparator prepares lifelog data 4. Data Preparator returns prepared data to Recommendation Builder
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-01
Frequency of Use:	Very frequent; for every service request
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	

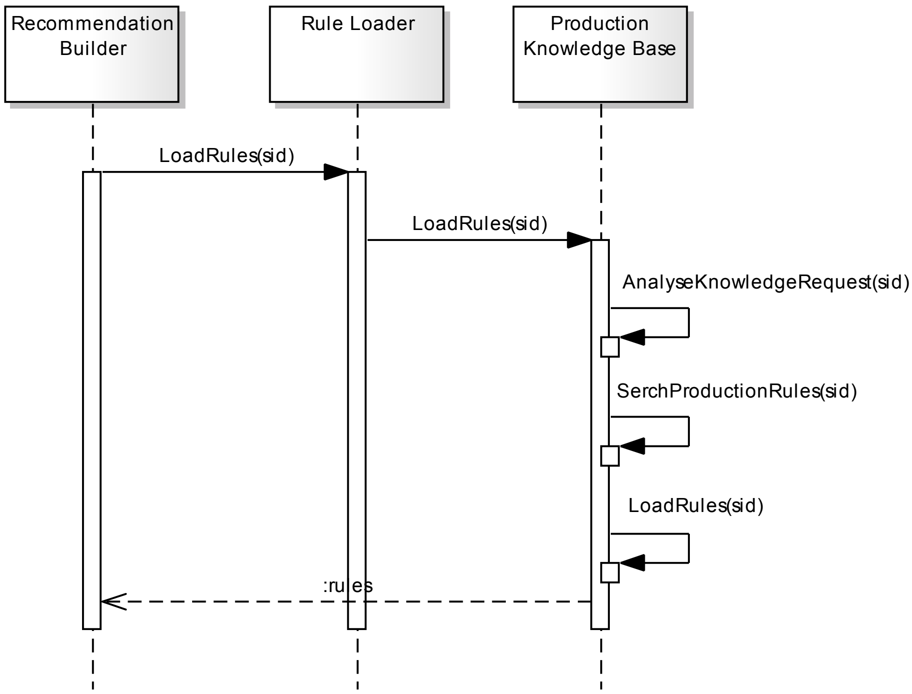


Use Case ID:	SCL-UC-03		
Use Case Name:	Load Rules		
Created By:	Rahman Ali	**Last Updated By:**	Rahman Ali
Date Created:	July 28, 2015	**Last Revision Date:**	July 28, 2015

Actors:	SCL-UC-04 (Build Recommendation)
Description:	Rule-based reasoned needs knowledge rules to perform reasoning using the prepared data to generate recommendations for the service request.
Trigger:	At the time when new service request arrives for recommendation.
Preconditions:	• Updated knowledge is available in Production Knowledge Base. • KCL and RB 2 agree on common format of production rules.
Postconditions:	The reasoned is ready to execute the rules and generate recommendations.
Normal Flow:	1. Recommendation Builders send knowledge load request to Rule Loader 2. Rule Loader sends request to Production Knowledge Base 3. System performs the following tasks; a. Analyses the request knowledge

	b. Search production knowledge base for the requested rules c. Loads the rules d. Provides the rules back to Recommendation Builder
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Frequent: when reasoner is invoked for new service generation.
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Sequence Diagram

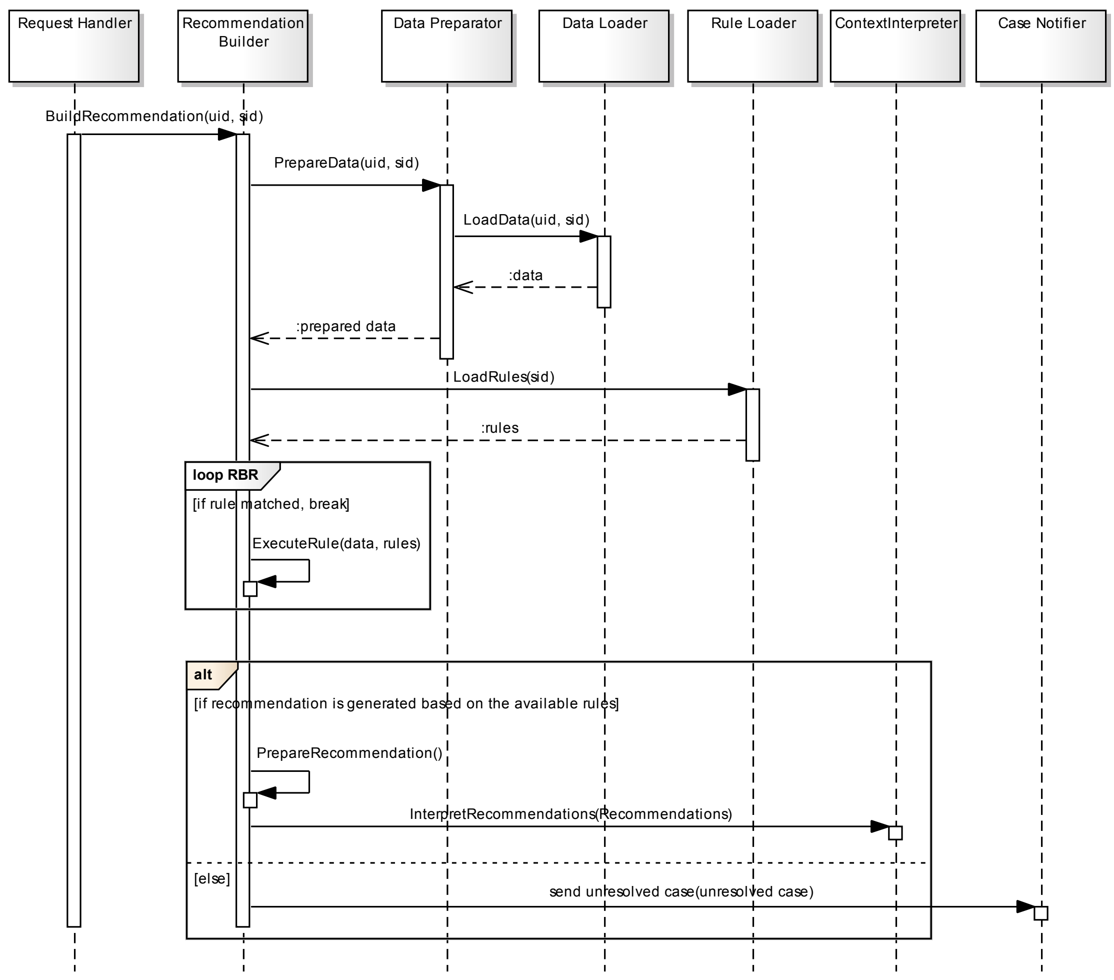


Use Case ID:	SCL-UC-04		
Use Case Name:	Build recommendations		
Created By:	Rahman Ali	**Last Updated By:**	Rahman Ali
Date Created:	July 15, 2015	**Last Revision Date:**	July 15, 2015

Actors:	SCL-UC-13 (Request Handler), SCL-UC-04, SCL-UC-02, SCL-UC-01, SCL-UC-03, SCL-UC-09 (Interpret Context)
Description:	RBR performs rule-based reasoning to generate recommendations using the production rules and prepared data.
Trigger:	At the time when new service request arrives for recommendation.
Preconditions:	Knowledge is available in Production Knowledge Base.
Postconditions:	The recommendation is reported to RI 2, if reasoning is successful, otherwise the new case is provided to Unified Knowledge Interface along with the missing rule message.
Normal Flow:	1. Request Handler invokes recommendation builder for recommendation 2. Recommendation Builder load prepared data 3. Recommendation Builder retrieves loaded rules 4. Recommendation Builder performs rule-based reasoning on the prepared data and loaded rules 5. Recommendation Builder generates recommendation and perform the following tasks; a. Prepare recommendation b. Provides recommendations to Context Interpreter for interpretation
Alternative Flows:	5a. The system could not find rule to execute a. Recommendation Builder sends message along with Unresolved Case to Case Notifier
Exceptions:	N/A
Includes:	SCL-UC-02, SCL-UC-03

Frequency of Use:	Frequent: when recommendation builder is invoked for generating recommendation.
Special Requirements:	N/A
Assumptions:	KCL and RB 2 agree on common format of production rules.
Notes and Issues:	N/A

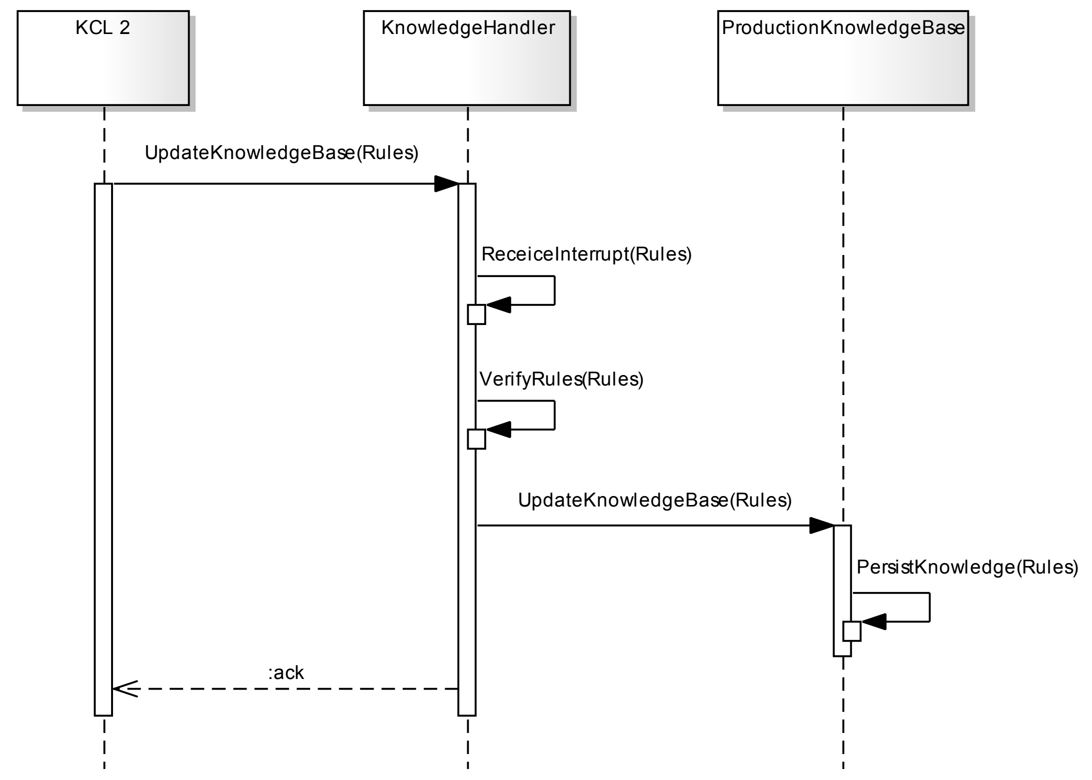
Sequence Diagram



Use Case ID:	SCL-UC-05		
Use Case Name:	Receive Production Knowledge		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 5, 2015	**Last Revision Date:**	July 28, 2015

Actors:	KCL
Description:	The knowledge is originated by KCL and is transfer to SCL 2 to keep a local copy of the production knowledge.
Trigger:	At knowledge creation/update time
Preconditions:	SCL 2 and KCL has a common representation agreement
Postconditions:	The SCL 2 copy of knowledge is updated and is synchronized with KCL
Normal Flow:	1. KCL interrupt Knowledge Handler for new knowledge 2. Knowledge Handler verifies the knowledge 3. Knowledge Handler make a local of the received knowledge in the Production Knowledge Base 4. Knowledge is persisted in Production Knowledge Base 5. KCL is acknowledged of the knowledge receipt
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Less frequent: at knowledge creation/update time
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

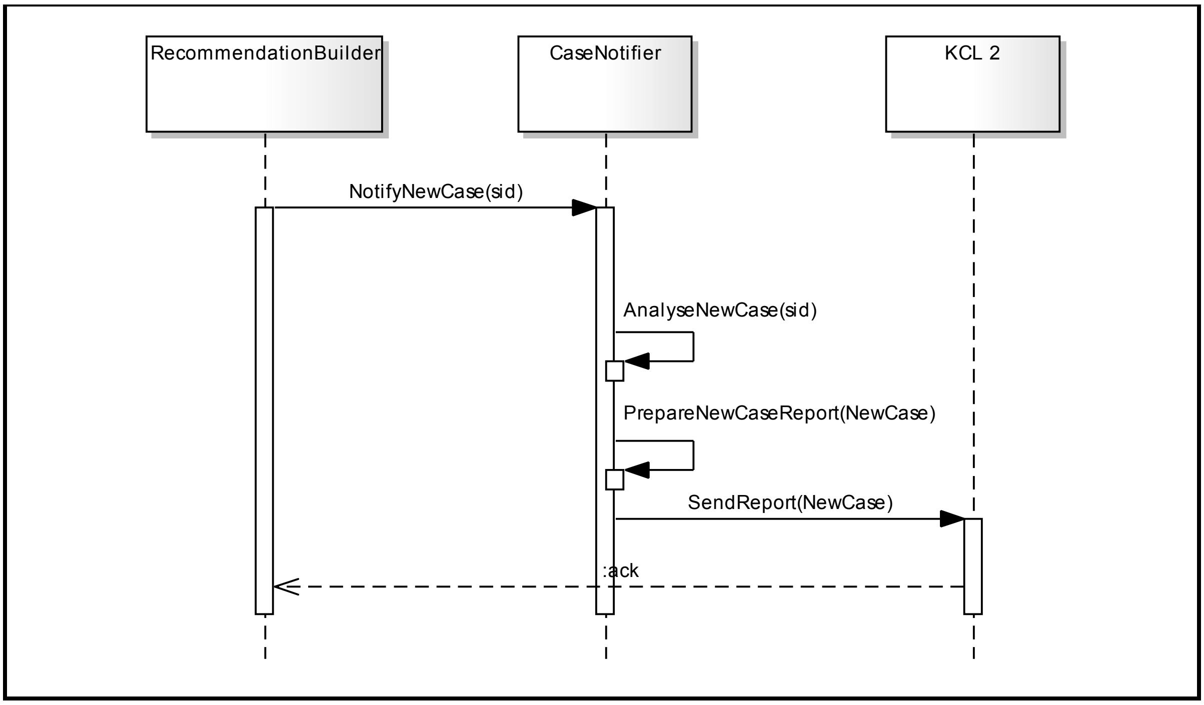
Sequence Diagram



Use Case ID:	SCL-UC-06		
Use Case Name:	Report unresolved case		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 5, 2015	**Last Revision Date:**	July 28, 2015

Actors:	SCL-UC-04 (Build Recommendation), KCL
Description:	Notifying KCL that reasoner is incapable to generate recommendation for the service request. KCL may be able to acquired new knowledge for such service request to handle in future.
Trigger:	At the time when reasoner is not capable to generate recommendation because of insufficient knowledge in the KB.

Preconditions:	Reasoner has completed the reasoning process
Postconditions:	The message with reason is successfully reported to KCL
Normal Flow:	1. Recommendation Builder notify unresolved case as new case prepare the report 2. Case Notifier analyses the new case 3. Case Notifier prepare the new case report 4. Case Notifier sends new case report to KCL 5. KCL acknowledges the new case receipt
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Less frequent: when reasoner detects new case not handled with existing knowledge.
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	

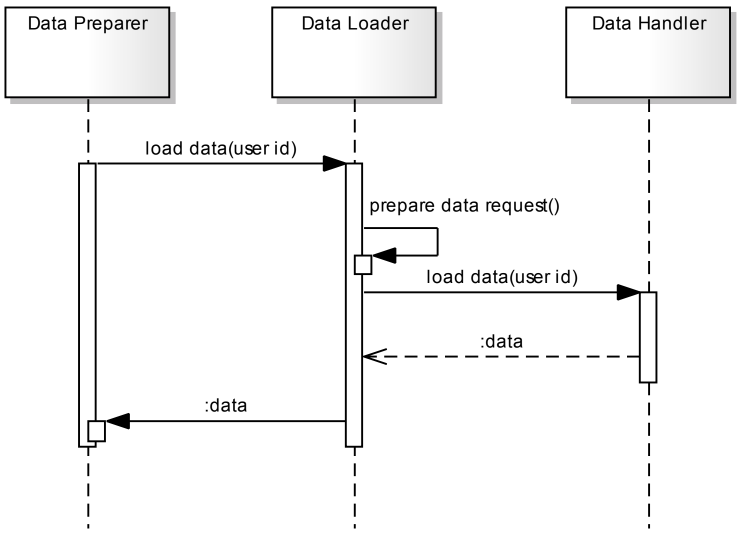


Use Case ID:	SCL-UC-07		
Use Case Name:	Load data for interpreting recommendations		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 15, 2015	**Last Revision Date:**	July 28, 2015

Actors:	Data Preparer
Description:	The data is loaded from DCL through Service Orchestrator in order to interpret the recommendations
Trigger:	After recommendation is built
Preconditions:	• Recommendation are built • User profile is stored in lifelog • Context is recognized
Postconditions:	The user profile, lifelog, and environmental variable data is available for preparation.

Normal Flow:	1. Data Loader receives interrupt from Data Preparer 2. Data loader prepare data request 3. Data loader send request to Data Handler 4. Data loader receives data from Data Handler
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Very frequent; at every service request
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

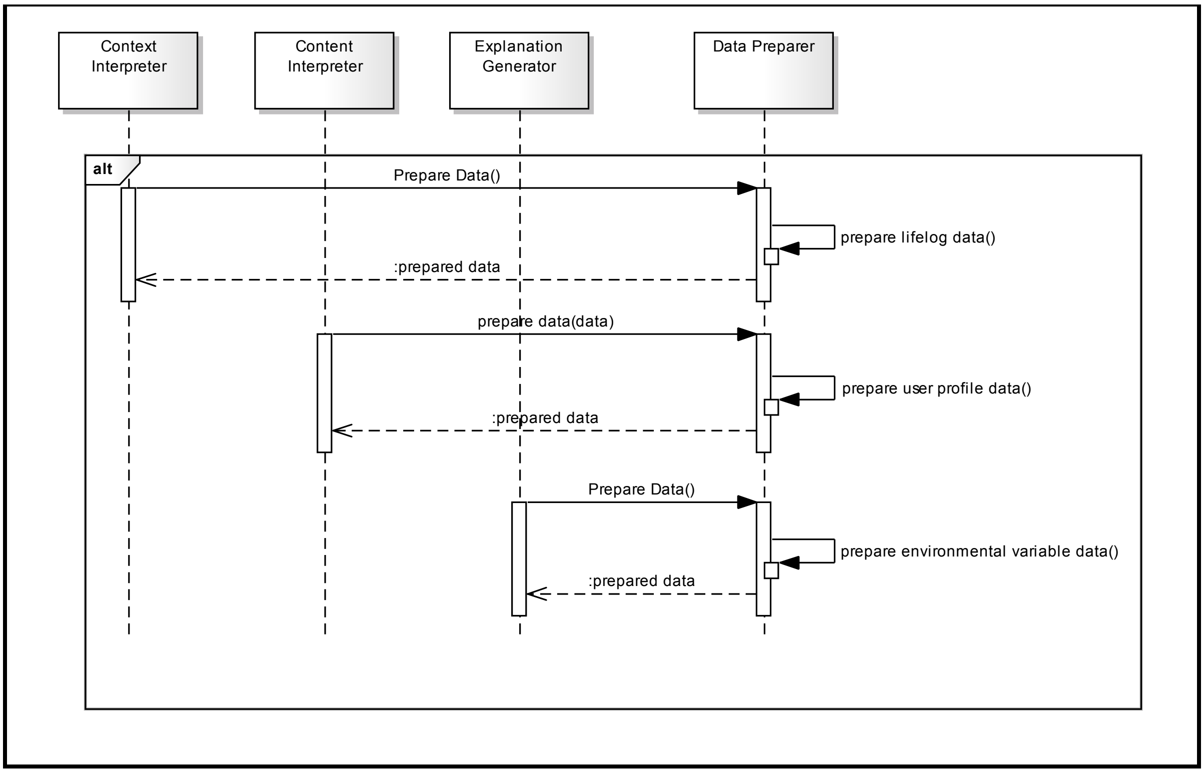
Sequence Diagram



Use Case ID:	SCL-UC-08
Use Case Name:	Prepare data for interpreting recommendations

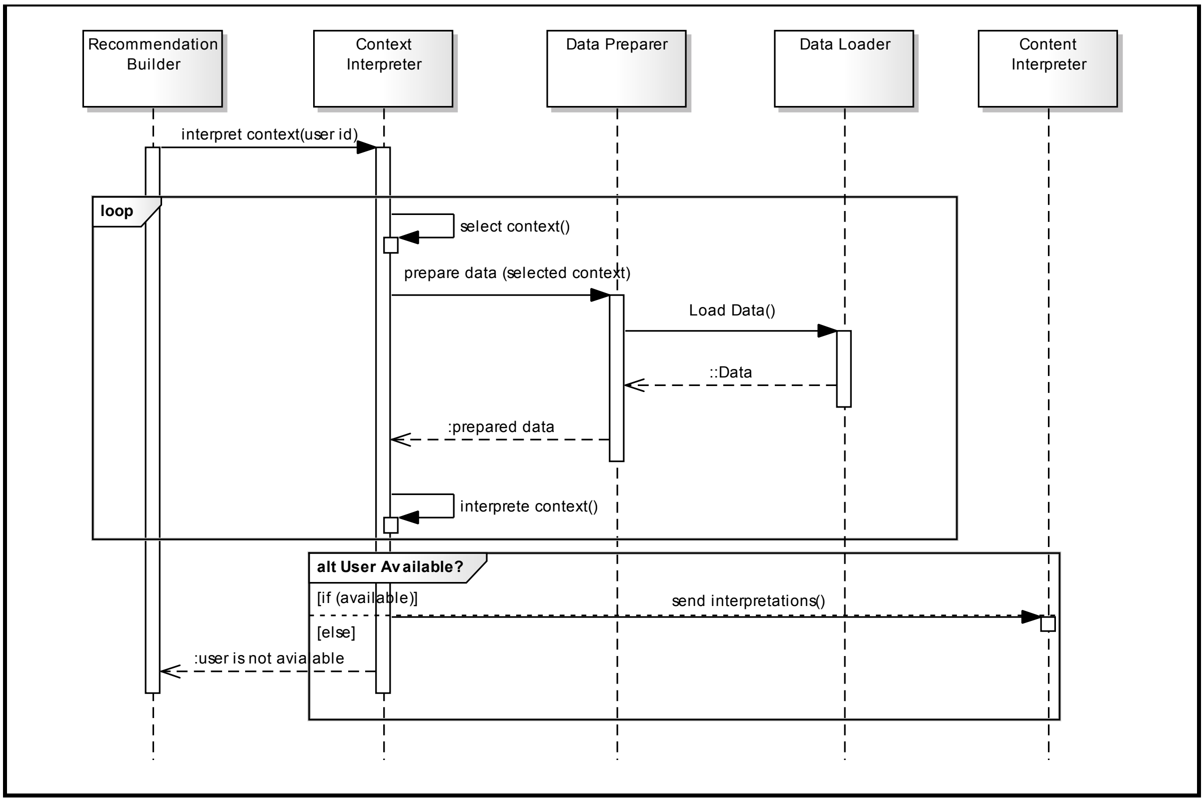
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 15, 2015	**Last Revision Date:**	July 28, 2015

Actors:	Context Interpreter
Description:	The loaded data is prepared for interpretations according to different functions such as lifelog for contextual interpretations, user profile for content interpretations, and environmental variables for explanations.
Trigger:	After loading data for interpretations
Preconditions:	• Recommendation are built • Data is loaded
Postconditions:	The user profile, lifelog, and environmental variable data is prepared and is available for interpretations
Normal Flow:	1. Context Interpreter sends data to Data Preparer for preparations 2. Data Preparer prepares lifelog data 3. Content Interpreter sends request to Data Preparer for preparing profile data 4. Data Preparer prepares profile data 5. Data Preparer prepares environmental variable data for the Explanation Generator
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-07
Frequency of Use:	Very frequent: at every service request
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	
Sequence Diagram	



Use Case ID:	SCL-UC-09		
Use Case Name:	Interpret context		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 15, 2015	**Last Revision Date:**	July 28, 2015
Actors:	Build Recommendation (SCL-UC-4)		
Description:	The loaded data is prepared for interpretations according to different functions such as lifelog for contextual interpretations, user profile for content interpretations, and environmental variables for explanations.		

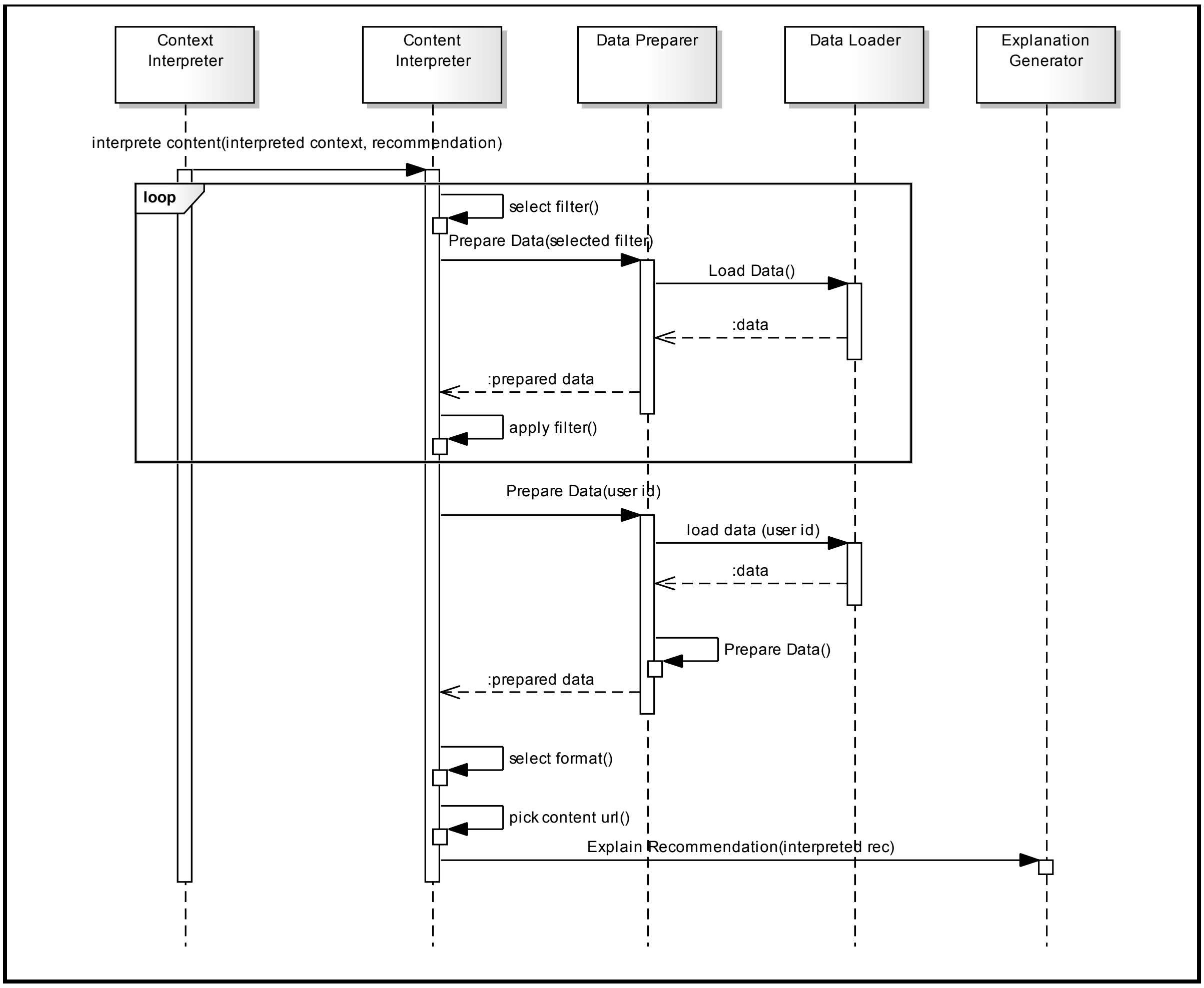
Trigger:	After loading data for interpretations
Preconditions:	Recommendation are built and data is loaded
Postconditions:	The user profile, lifelog, and environmental variable data is prepared and is available for interpretations
Normal Flow:	1. Context Interpreter receives request for from Recommendation Interpreter for context interpretation 2. Context Interpreter load and prepare data lifelog data (contextual data) for interpretations. 3. Context interpreter select a context 4. Context interpreter interprets the context 5. Repeat 2-4 until all applicable contexts interpreted 6. Context Interpreter receives the interpreted context 7. Context Interpreter sends the recommendations to content interpreter for interpreting the contents
Alternative Flows:	7a. if user is not available then the process is halt and message is sent to Recommendation Builder.
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent; at every service request
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SCL-UC-10		
Use Case Name:	Interpret contents		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali
Date Created:	July 15, 2015	**Last Revision Date:**	July 28, 2015

Actors:	Interpret Context (SCL-UC-09)
Description:	The recommended contents of recommendations are difficult for user to understand. These contents needs to be interpreted with support of multimedia contents.
Trigger:	After interpretation of the context
Preconditions:	Recommendations are generated and context is interpreted

Postconditions:	Recommendations are ready for explanation
Normal Flow:	1. Context Interpreter sends the contextually interpreted recommendations to the content filterer. 2. Content interpreter perform the following tasks; a. Select appropriate filter b. Applies the filter 3. Step 2 is repeated for all filters 4. Content interpreter selects the appropriate format 5. Content interpreter adds the relevant url 6. Content interpreter forwards the format and filtered contents to explanation generator
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent: when recommendation are generated
Special Requirements:	The format should be defined in advanced based on the user special conditions
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	

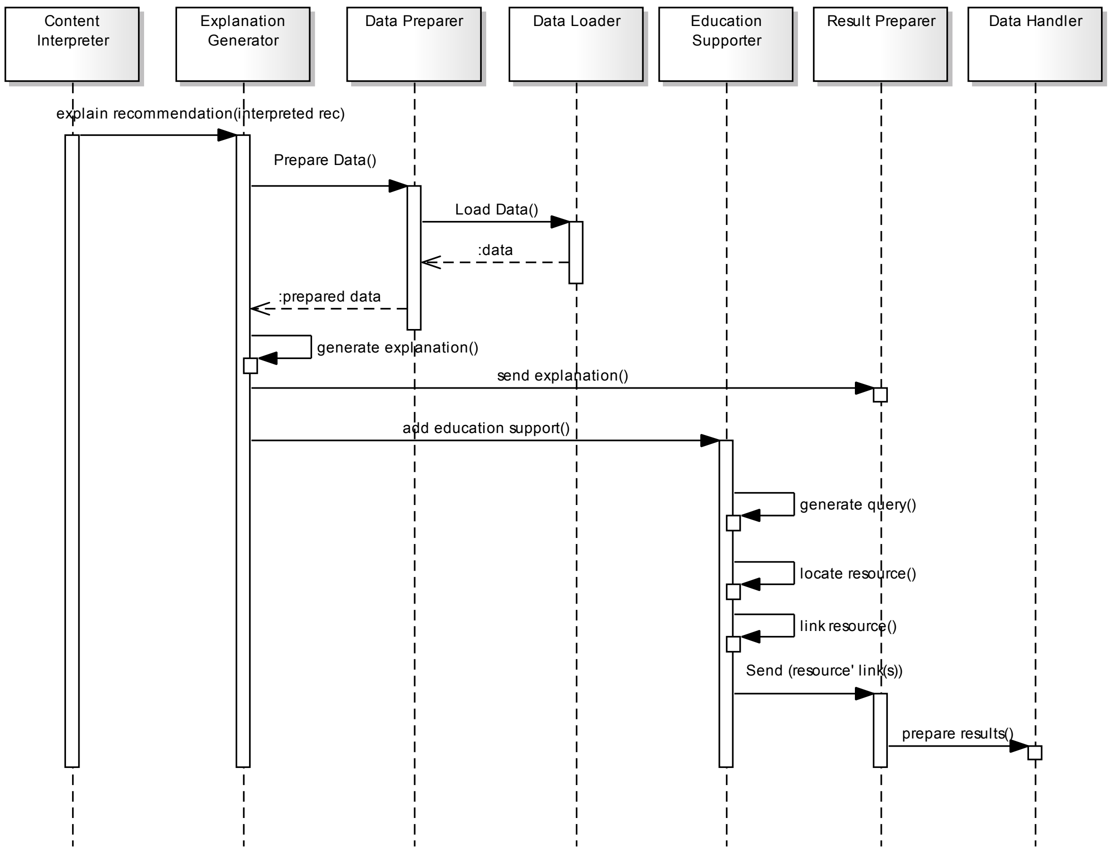

Context Interpreter
Content Interpreter
Data Preparer
Data Loader
Explanation Generator
interprete content(interpreted context, recommendation)
loop
select filter()
Prepare Data(selected filter)
Load Data()
:data
:prepared data
apply filter()
Prepare Data(user id)
load data (user id)
:data
Prepare Data()
:prepared data
select format()
pick content url()
Explain Recommendation(interpreted rec)

Use Case ID:	SCL-UC-11		
Use Case Name:	Explain Recommendations		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali
Date Created:	July 15, 2015	**Last Revision Date:**	July 20, 2015

Actors:	Interpret content (SCL-UC-10)
Description:	Usually user don't understand the contents of recommendations. To make them understandable the interpreted recommendations needs to be explained based on the user understandability.
Trigger:	When contents are interpreted
Preconditions:	Recommendations are interpreted
Postconditions:	Recommendations are ready to deliver to the user
Normal Flow:	1. Explanation generator receives the interpreted recommendations from content interpreter. 2. Explanation generator performs the following tasks; a. Select environment variable b. Generate explanation 3. Explanation Generator sends explained recommendation to educational support handler 4. Educational support handler performs the following tasks a. generate query b. locate resource c. link resource d. send resource link to interpreter 5. System sends explanation and educational resource links to result preparer.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent: when recommendations are interpreted

Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

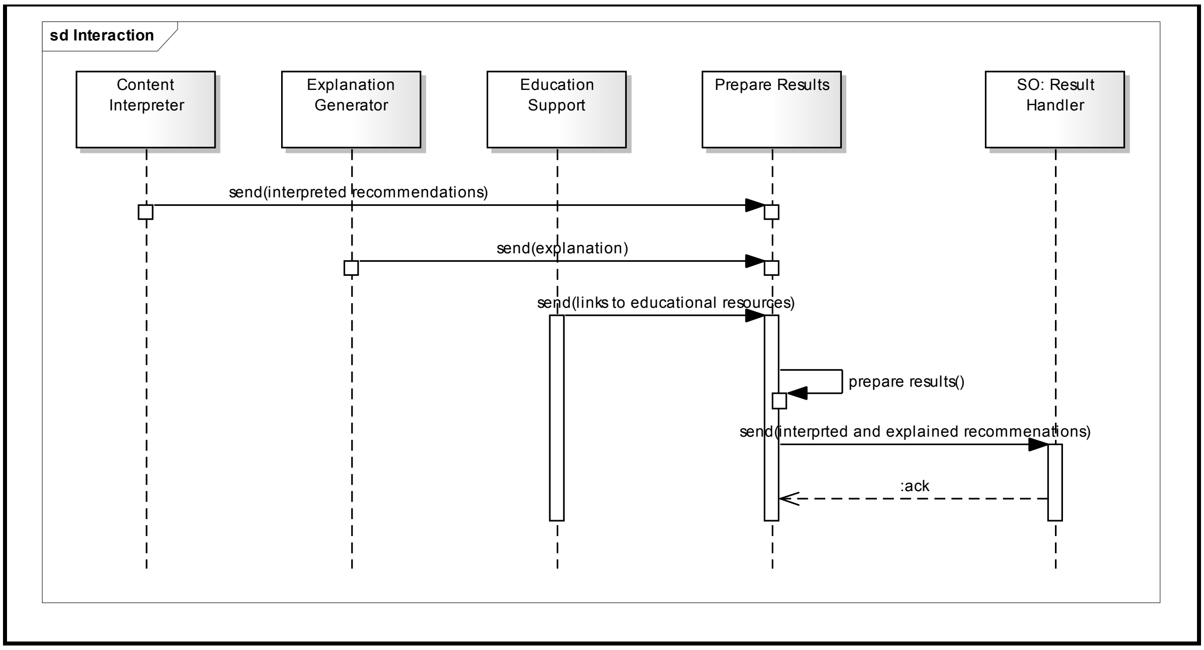
Sequence Diagram



Use Case ID:	SCL-UC-12		
Use Case Name:	Prepare Results		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal

Date Created:	July 5, 2015	Last Revision Date:	July 20, 2015

Actors:	Explain Recommendation (SCL-UC-11), Interpret Contents (SCL-UC-10)
Description:	This use case prepare the results accumulated from explanation generator and content interpreter and forwards to results handler of service orchestrator.
Trigger:	When recommendation are interpreted and explained
Preconditions:	The recommendation are interpreted and explained
Postconditions:	The results are forwarded to service orchestrator.
Normal Flow:	1. Result Preparer receives outputs from content interpreter and/or explanation generator as well as education support. 2. Result Preparer combines the received results 3. Result Preparer sends the results to result handler of service orchestrator
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Frequent
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	

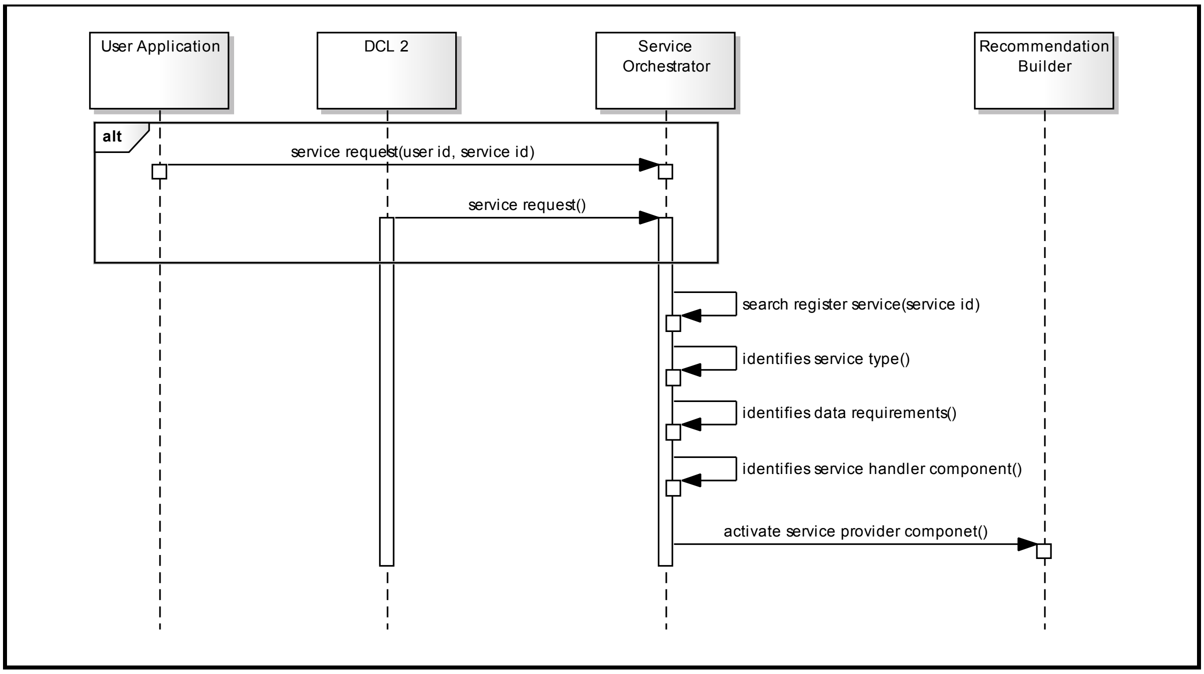


Use Case ID:	SCL-UC-13		
Use Case Name:	Receive service request		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 5, 2015	**Last Revision Date:**	July 20, 2015

Actors:	User Application / SL
Description:	Service orchestrator receives request from user application, or DCL for recommendation. Orchestrator parses the request and invokes required service of Mining Mind for responding.
Trigger:	At the time of a request from the user application, or from mining mind generated events.
Preconditions:	• User is registered with Mining Minds • Service is registered as mining Minds valid service • Service-data binding is specified in advance
Postconditions:	The request is received and recommendations are generated

Normal Flow:	1. Service orchestrator receives the service request from user application
	2. System parses the request
	    a. Search for the registered service
	    b. Identifies the service type
	    c. Identifies data requirements of the service
	    d. Identifies the appropriate handling module
	3. Service Orchestrator passes the request to recommendation builder of SCL 2 to build the recommendation

Alternative Flows:	1.a Event handler of service orchestrator receives the request as an interrupt from DCL, whenever a situation occurs 4.a Step 2-4 of the normal flow are executed.
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Very frequent: at every service request
Special Requirements:	N/A
Assumptions:	Service orchestrator and DCL agreed on service contract.
Notes and Issues:	
Sequence Diagram	

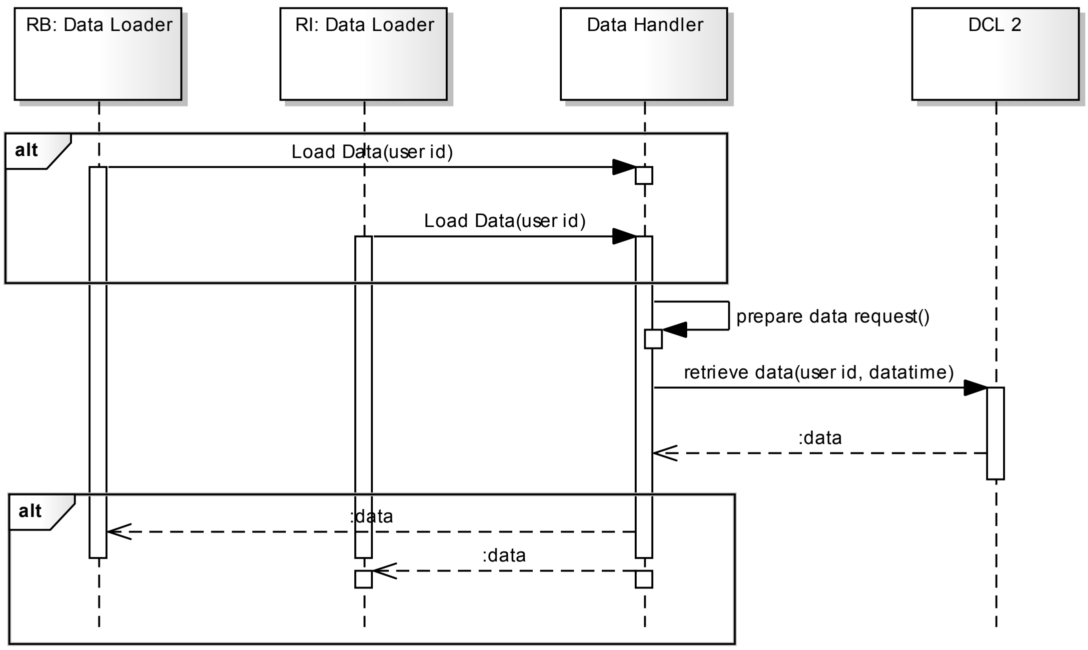


Use Case ID:	SCL-UC-14		
Use Case Name:	Handle Data		
Created By:	Muhammad Afzal	**Last Updated By:**	Muhammad Afzal
Date Created:	July 28, 2015	**Last Revision Date:**	July 28, 2015

Actors:	DCL
Description:	This use case receives data request from recommendation builder and recommendation interpreter. It makes request from DCL to get the data for requester.
Trigger:	At data request time
Preconditions:	Service Request has been received to service orchestrator
Postconditions:	Data has been provided to requester
Normal Flow:	1. Data Handler in Service Orchestrator received data loading request from recommendation builder 2. Prepare data request

	3. Retrieve data from DCL 4. Send data to RB: Data Loader
Alternative Flows:	1a. Data Handler in Service Orchestrator received data loading request from recommendation interpreter Step 2-3 of normal flow 4a. Send data to RI: Data Loader
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Very Frequent: At every service request
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A

Sequence Diagram

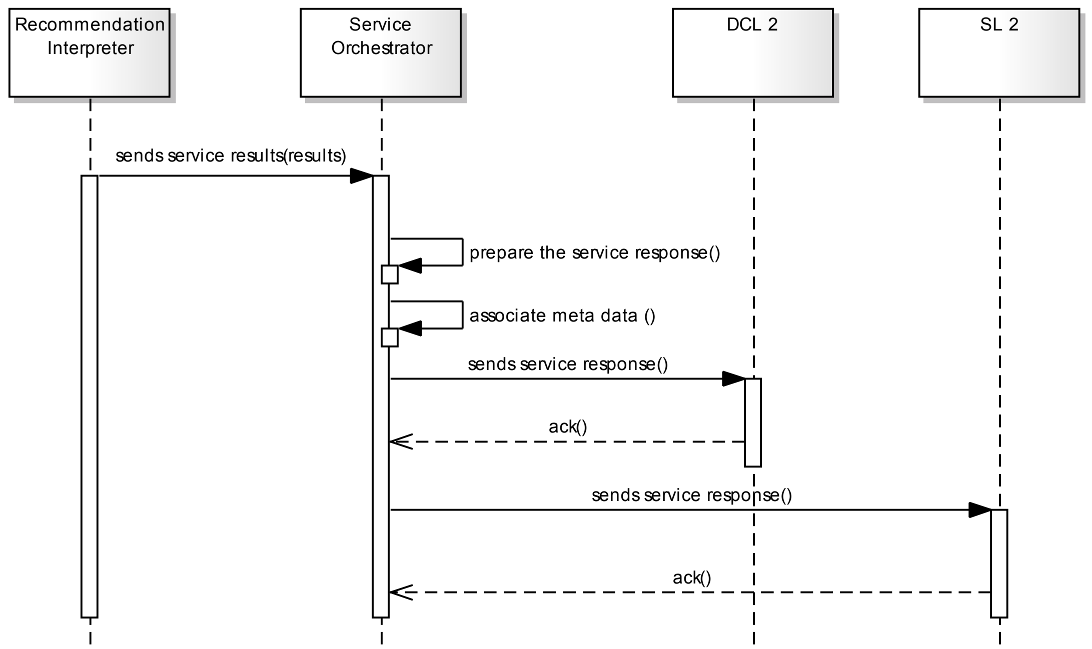


Use Case ID:	SCL-UC-15		
Use Case Name:	Deliver service results		
Created By:	Muhammad Afzal	**Last Updated By:**	Rahman Ali, Muhammad Afzal
Date Created:	July 5, 2015	**Last Revision Date:**	July 20, 2015

Actors:	Application / SL
Description:	It is required to send request response to the service requester in the form of recommendation. Service orchestrator delivers the interpreted recommendation to user.
Trigger:	At the time of completion of interpretations
Preconditions:	Recommendations are generated and interpreted
Postconditions:	Service results are successfully delivered to the requester and DCL for persistence
Normal Flow:	1. Service orchestrator receives results from recommendation interpreter 2. System perform the following tasks; a. Prepares the response message b. Associate recommendations with service meta-data 3. Service Orchestrator sends recommendations to DCL for persistence 4. Service Orchestrator receives acknowledgement of storage 5. Service Orchestrator sends interpreted recommendations to SL 6. Service Orchestrator receives acknowledgement of receipt
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Very frequent: at every service request completion

Special Requirements:	N/A
Assumptions:	Service orchestrator agreed on service contract with DCL and user application.
Notes and Issues:	N/A

Sequence Diagram

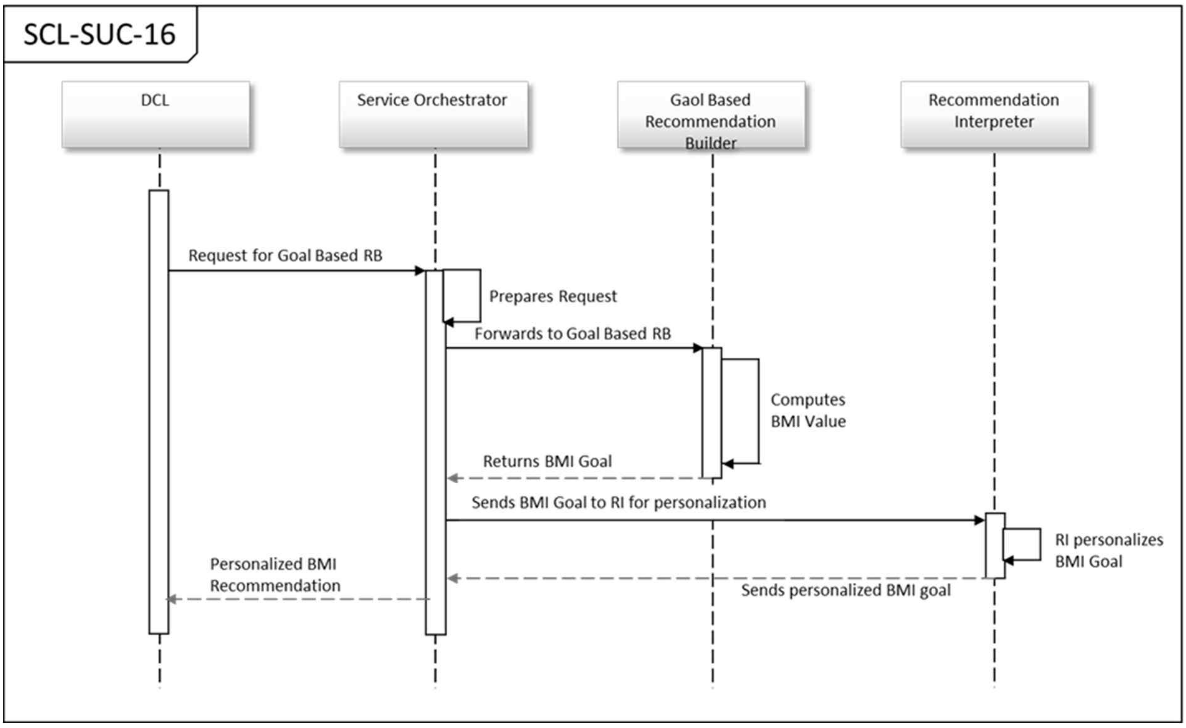


Goal Based Recommendation Generation

Use Case ID:	SCL-UC-16		
Use Case Name:	Goal-based recommendation Generation		
Created By:	Muhammad Sadiq	**Last Updated By:**	Muhammad Sadiq
Date Created:	March 16, 2017	**Last Revision Date:**	March 16, 2017

Actors:	DCL, Serivice Orchestrator, Recommendation Interpreter
Description:	GoalBased recommender generates goal based recommendation based on the user values
Trigger:	DCL/User invokes goal based recommender through service orchestrator for goal based personalized recommendation generation
Preconditions:	Prepares request
Postconditions:	Goal based recommendation is ready to be sent
Normal Flow:	1. User characteristic information e.g. height, weight etc. is received 2. GoalBased Reasoner computes user BMI 3. Computed BMI is compared with Standard BMI 4. The difference of computed and standard BMI is generated 5. Categorize user based on the computed BMI difference 6. Generate recommendation based on respective BMI status 7. Return recommended goal and associated information to service orchestrator 8. Service orchestrator sends the recommendation to RI for personalization of the recommendation 9. RI personalizes the generated recommendation and sends it back to service orchestrator 10. Service orchestrator return the generated personalized goal based recommendation to DCL
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	
Frequency of Use:	Depending on situation.
Special Requirements:	N/A
Assumptions:	KCL 3.0 and RB 2 agree on common format of production rules.
Notes and Issues:	N/A

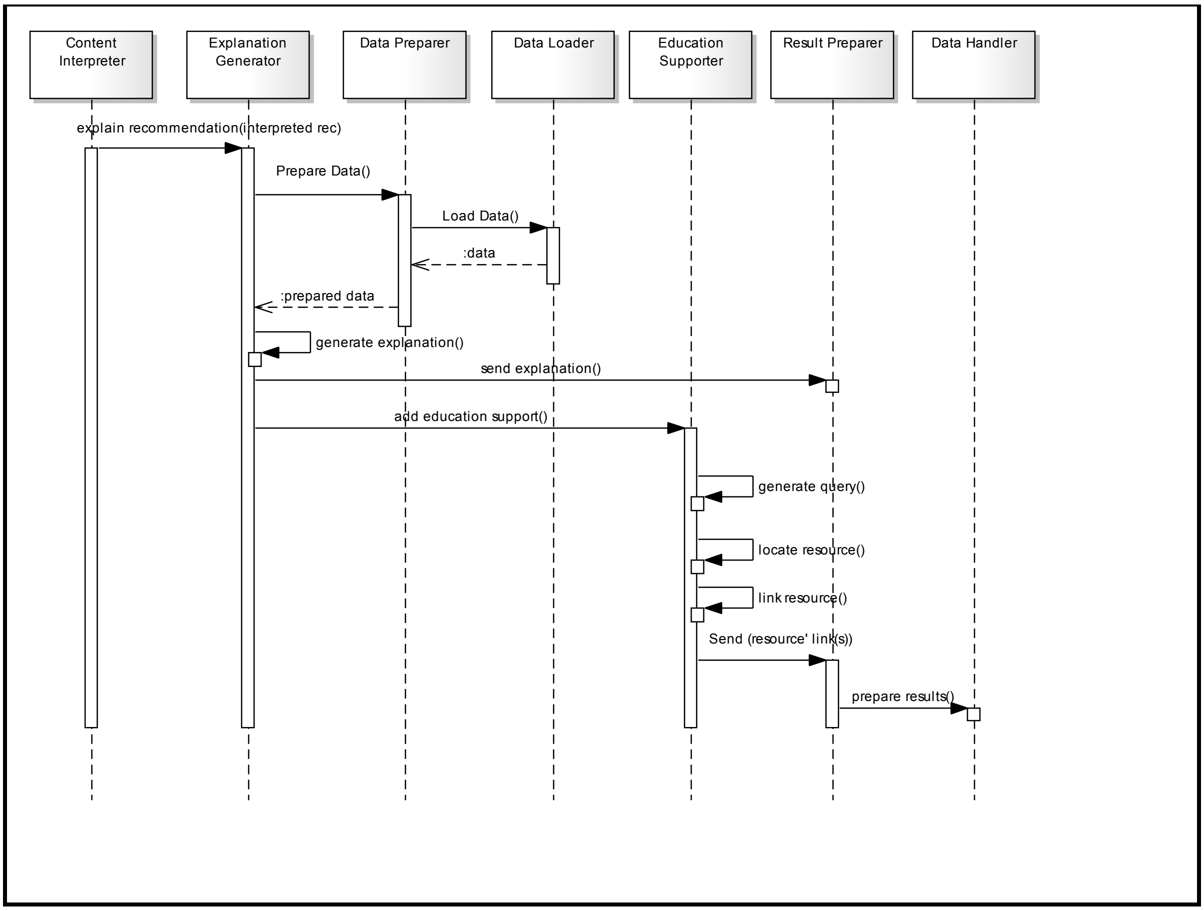
Sequence Diagram



Use Case ID:	SCL-UC-17		
Use Case Name:	Provide Educational Support for user awareness		
Created By:	Imran Ali	**Last Updated By:**	Imran Ali
Date Created:	March 16, 2017	**Last Revision Date:**	March 16, 2017

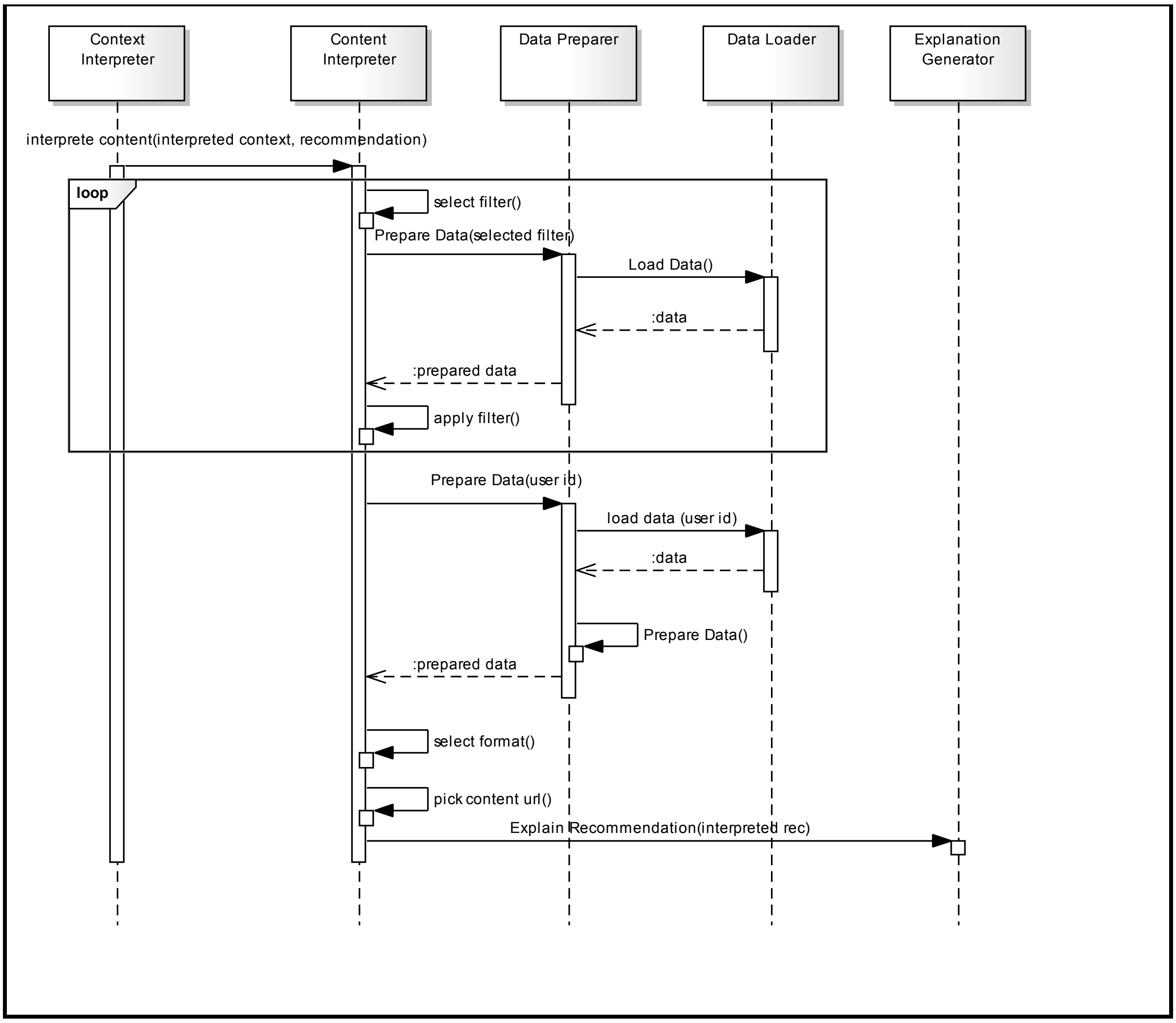
Actors:	Build Recommendation (SCL-UC-4)
Description:	Provide educational support to the user for first 4 weeks of the subscription period. Educational support consists of audio/visual aids and informative web resources
Trigger:	After loading data for interpretations
Preconditions:	Recommendation are built and data is loaded

Postconditions:	The user profile, lifelog, and environmental variable data is prepared and is available for interpretations
Normal Flow:	5. Recommendation Interpreter receives request from Recommendation Builder 6. Recommendation Interpreter extracts users information from lifelog 7. Recommendation Interpreter evaluates the matching educational contents for the target user audience 8. Recommendation Interpreter prepares the required educational contents 9. Recommendation Interpreter sends the required educational content unit to result preparer
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Thrice a day: Time-based
Special Requirements:	N/A
Assumptions:	Service Orchestrator knows the required data for each registered service.
Notes and Issues:	N/A
Sequence Diagram	



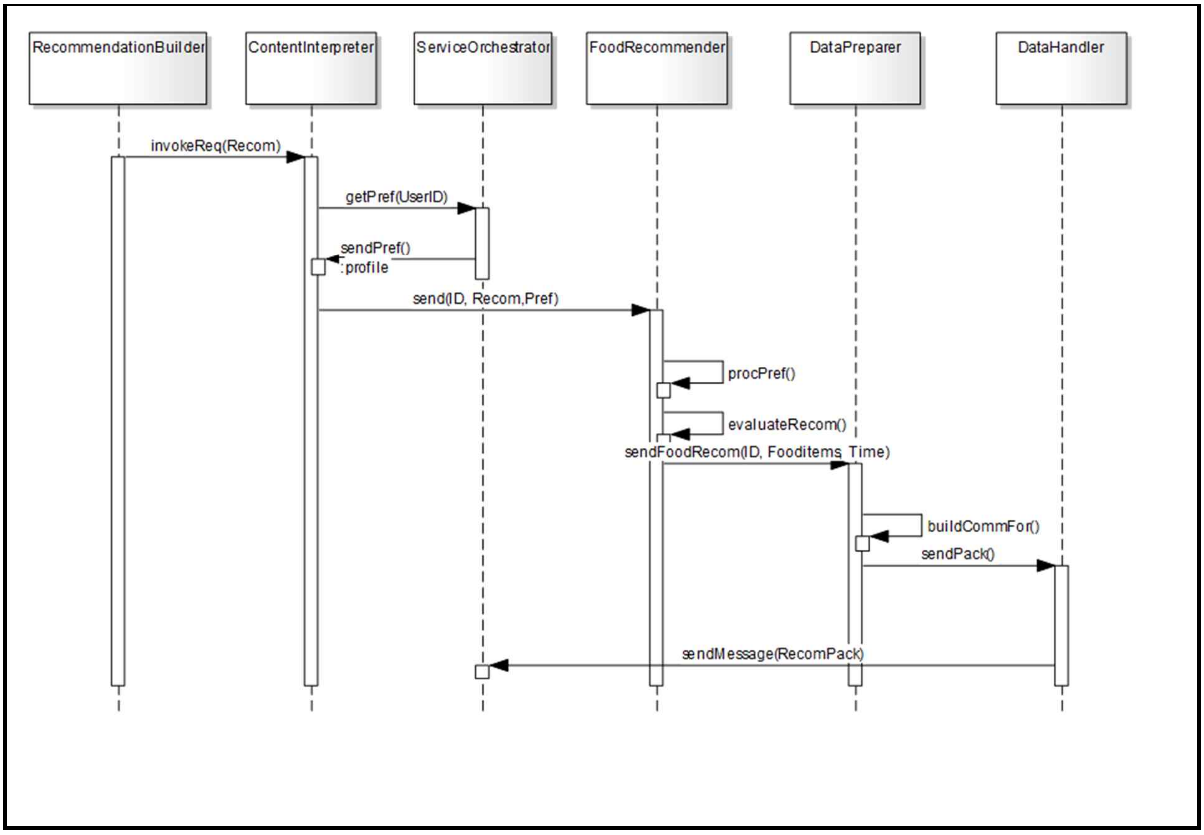
Use Case ID:	SCL-UC-18		
Use Case Name:	Location- and weather-based Personalized recommendation		
Created By:	Imran Ali	**Last Updated By:**	Imran Ali
Date Created:	March 16, 2017	**Last Revision Date:**	March 16, 2017
Actors:	Build Recommendation (SCL-UC-4)		
Description:	Interpret recommendations for environmental contexts and provide the required information in an structured form to SL		
Trigger:	After loading data for interpretations		

Preconditions:	Recommendation are built and data is loaded
Postconditions:	The user profile, lifelog, and environmental variable data is prepared and is available for interpretations
Normal Flow:	1. Recommendation Interpreter receives physical activity based recommendation 2. Recommendation Interpreter assess the user interruptibility 3. In case of availability Recommendation Interpreter evaluates the contextual viability of the recommendation 4. Recommendation Interpreter sends the recommendation for education support 5. Recommendation Interpreter prepares educational unit for the target user is prepared 6. Final recommendation package is sent to Result Preparer
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent; at every service request
Special Requirements:	N/A
Assumptions:	Service Orchestrator knows the required data for each registered service.
Notes and Issues:	N/A
Sequence Diagram	



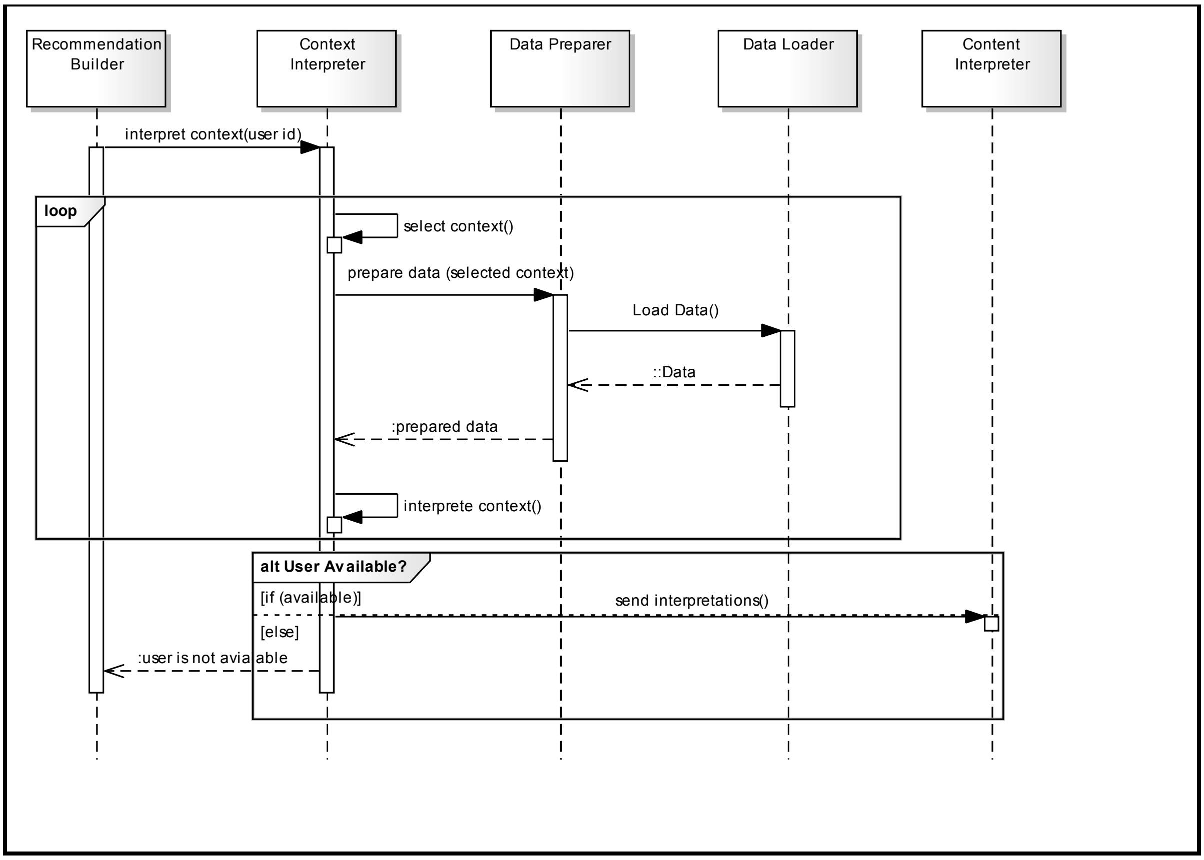
Use Case ID:	SCL-UC-19		
Use Case Name:	User Preference-based refined recommendations		
Created By:	Imran Ali	**Last Updated By:**	Imran Ali
Date Created:	March 16, 2017	**Last Revision Date:**	March 16, 2017
Actors:	Build Recommendation (SCL-UC-4)		

Description:	Interpret Recommendations based on User's Preferences for Food-based Recommendation
Trigger:	After loading data for interpretations
Preconditions:	Recommendation are built and data is loaded
Postconditions:	The user profile and lifelog data is prepared and is available for interpretations
Normal Flow:	1. Recommendation Interpreter receives food-based recommendation 2. Recommendation Interpreter evaluates the targeted user group 3. Recommendation Interpreter retrieves preferences of the user 4. Recommendation Interpreter matches user preferences with the targeted food-categorizes (menu-sets) 5. Recommendation Interpreter finalizes the menu-set selection 6. Final recommendation package is sent to Result Preparer
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent; at every service request
Special Requirements:	N/A
Assumptions:	Service Orchestrator knows the required data for each registered service.
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SCL-UC-20		
Use Case Name:	Provide Alternative Recommendation		
Created By:	Imran Ali	**Last Updated By:**	Imran Ali
Date Created:	March 16, 2017	**Last Revision Date:**	March 16, 2017
Actors:	Build Recommendation (SCL-UC-4)		
Description:	Interpret Recommendation for special condition. An alternative recommendation is generated when the provided recommendation is not deemed viable for the user		

Trigger:	After loading data for interpretations
Preconditions:	Recommendation are built and data is loaded
Postconditions:	The user profile, lifelog, and environmental variable data is prepared and is available for interpretations
Normal Flow:	1. Recommendation Interpreter receives physical activity based recommendation 2. Recommendation Interpreter evaluates the availability of the user 3. Recommendation Interpreter evaluates the special user conditions 4. Recommendation Interpreter decides of the need for alternative recommendation 5. Educational contents are prepared for the final recommendation 6. Final recommendation package is sent to Result Preparer
Alternative Flows:	1.1 Stall recommendation is user is not available
Exceptions:	N/A
Includes:	SCL-UC-08
Frequency of Use:	Very frequent; at every service request
Special Requirements:	N/A
Assumptions:	Service Orchestrator knows the required data for each registered service.
Notes and Issues:	N/A
Sequence Diagram	


Recommendation Builder
Context Interpreter
Data Preparer
Data Loader
Content Interpreter
interpret context(user id)
loop
select context()
prepare data (selected context)
Load Data()
::Data
:prepared data
interprete context()
alt User Available?
[if (available)]
send interpretations()
[else]
:user is not aviailable

Section 5.5

Supporting Layer(SL) Requirement Specification

Supporting Layer

1. SL Requirement Specification

1.1. SL Terms and Definitions

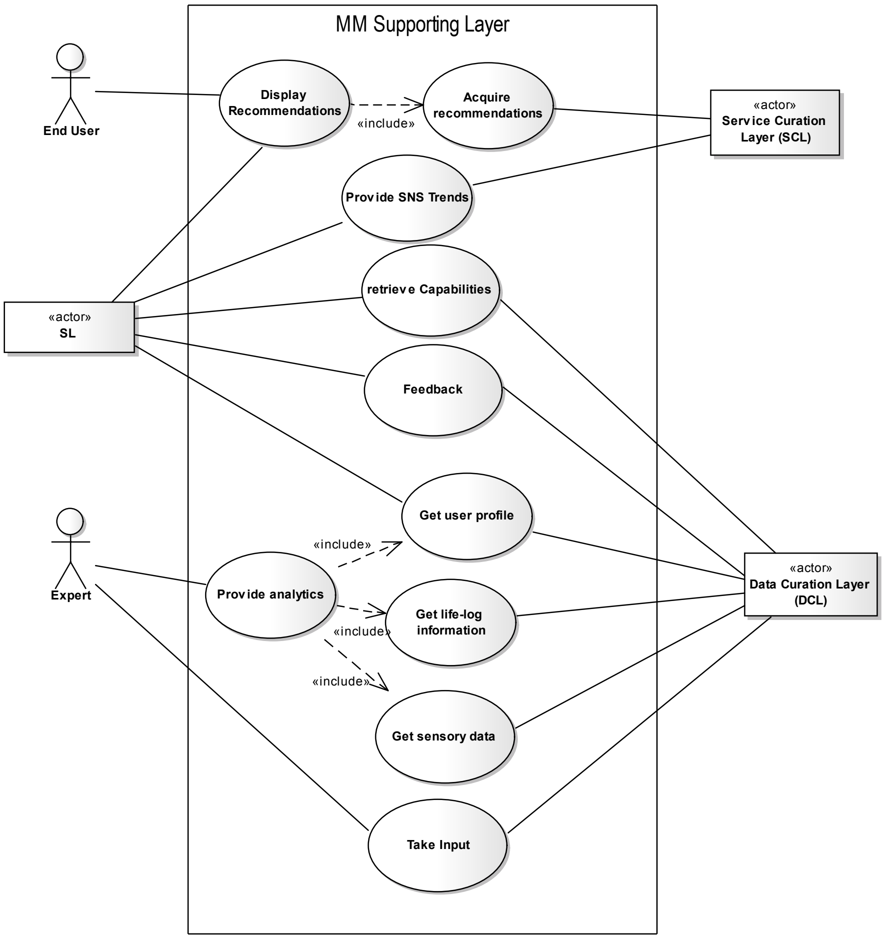
Terms	Description
DCL	Data Curation Layer Ver. 3
SCL	Service Curation Layer Ver. 3
ICL	Information Curation Layer Ver. 3
KCL	Knowledge Curation Layer Ver. 3
SL	Supporting Layer Ver. 3
Self-report	A method which involves asking a participant about their feelings, attitudes, beliefs and so on
User Capabilities	User cognitive, perceptual and physical characteristics
Context of use	Environmental variables and low level context (location)
Device Characteristics	Screen size, resolution, memory, and battery
UI Adaptation	The changes in user interface
Observational Data	User interaction data with the user interface
User Experience	User perception, satisfaction about the user interfaces
User Profile	User related data that do not change frequently

2. Use Case Diagram and Details

3.2 **Use Case List**

Requirements #ID	Description
SL-UC-01	User registration
SL-UC-02	Retrieve capabilities for user interface adaptation
SL-UC-03	Mapping the user capability information into model
SL-UC-04	Adapt user interface based on user profile, context and device
SL-UC-05	User capabilities collection
SL-UC-06	Self-reporting user experience measurement
SL-UC-07	Collect and analyze observational data
SL-UC-08	Acquire Recommendations for displaying to end user
SL-UC-09	Feedback Collection
SL-UC-10	Retrieve capabilities for context based services
SL-UC-11	Map Request to Query
SL-UC-12	Transform Data
SL-UC-13	Classify Data
SL-UC-14	Analyze Data
SL-UC-15	Display Analytics
SL-UC-16	Take Expert Input

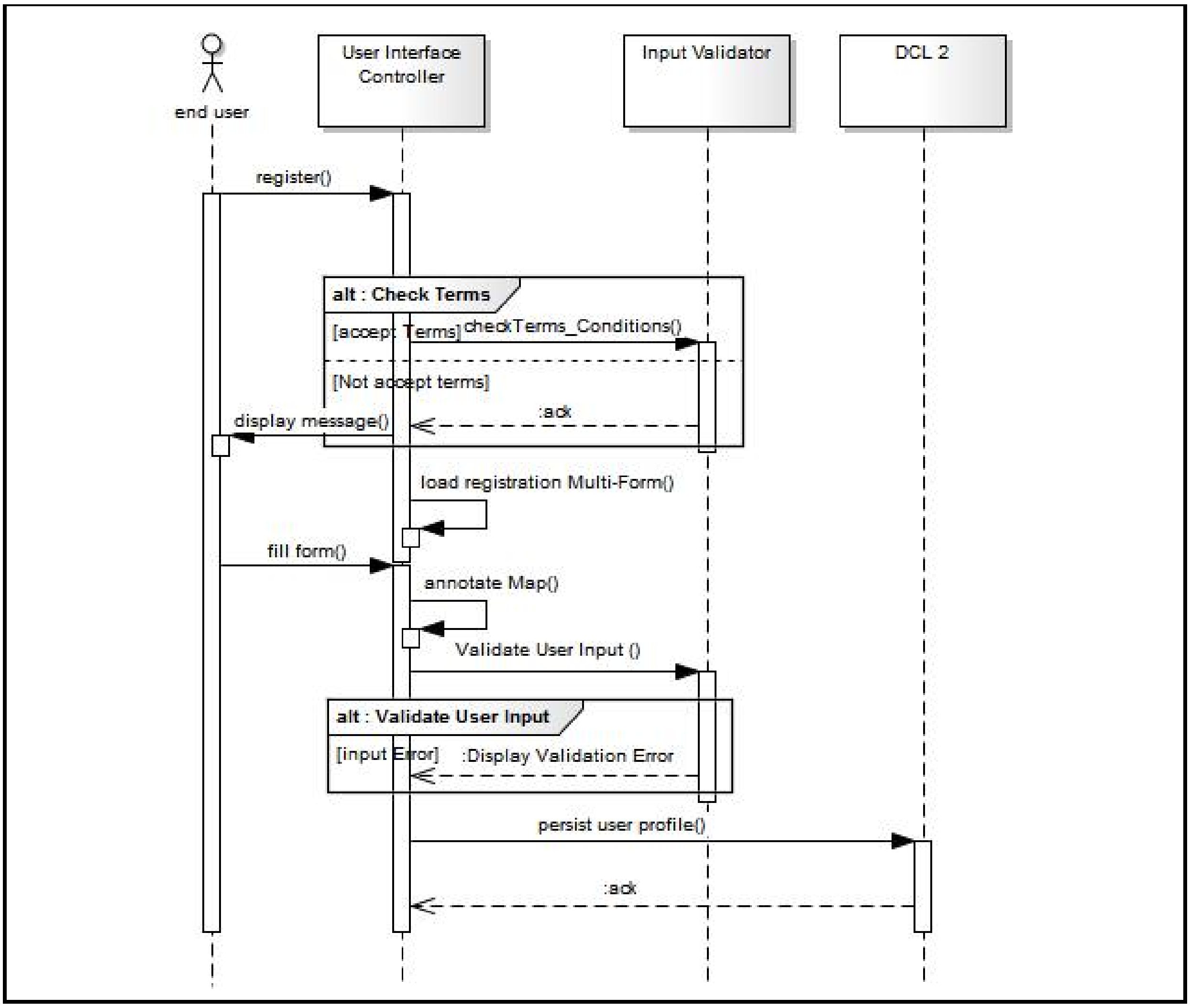
Use case diagram



3.2 Use Case Details

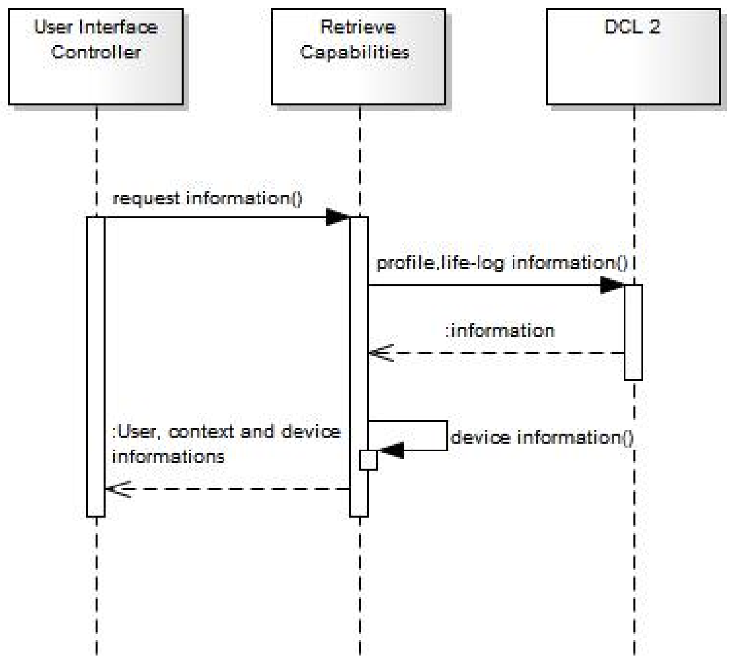
Use Case ID:	SL-UC-01		
Use Case Name:	User registration		
Created By:	Jamil Hussain	**Last Updated By:**	Jamil Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017
Actors:	Primary: End-user Secondary: DCL		
Description:	This use case is for the user registration. A user must register with the MM app before they are able to use it. Registration primarily consists of entering an email address for verification and creating a password. All		

	basic demographics, account, activity level, user interest information, and personalized map information shall be collect from user and persist in DCL.
Trigger:	End user
Preconditions:	The non-register user asked the application to register to it.
Postconditions:	The user successfully registered to the application and can access its functionality
Normal Flow:	1. The user start registration of the new account by pressing the "Sig up" button on the application first screen. 2. Then **Terms & Conditions** page displayed 3. The application will display the multi-step registration form with empty fields for the account and user profile. 4. The user annotates the map for personalization by selecting different l ocations of his interest. 5. Validate User Input • The application will automatically validate all the user input fo r all the required fields • The user cannot proceed until providing the correct data. 6. The user can press "Submit" button and the new account data will be p ersisted to the DCL.
Alternative Flows:	2a. In step 2. If the user Agree with Terms & Conditions • User is allowed to the next step by click on agreed term & conditio n checkbox. 2b. In step 2. If user not agree with terms & conditions then • User is redirect to the first screen. 3a. in step 3. Display Validation Error • If the validation failed, then the validation icon will be displayed ne arby the wrong field and there will be validation message.
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	When user first time use the system [low]
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SL-UC-02		
Use Case Name:	Retrieve capabilities for user interface adaptation		
Created By:	Jamil Hussain	**Last Updated By:**	Jamil Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017
Actors:	DCL		
Description:	This use case focuses on the retrieval of the capabilities for user interface adaptation. The capabilities includes user profile information, context information and device information. It is utilized for adaptation based on changes or observational data.		
Trigger:			

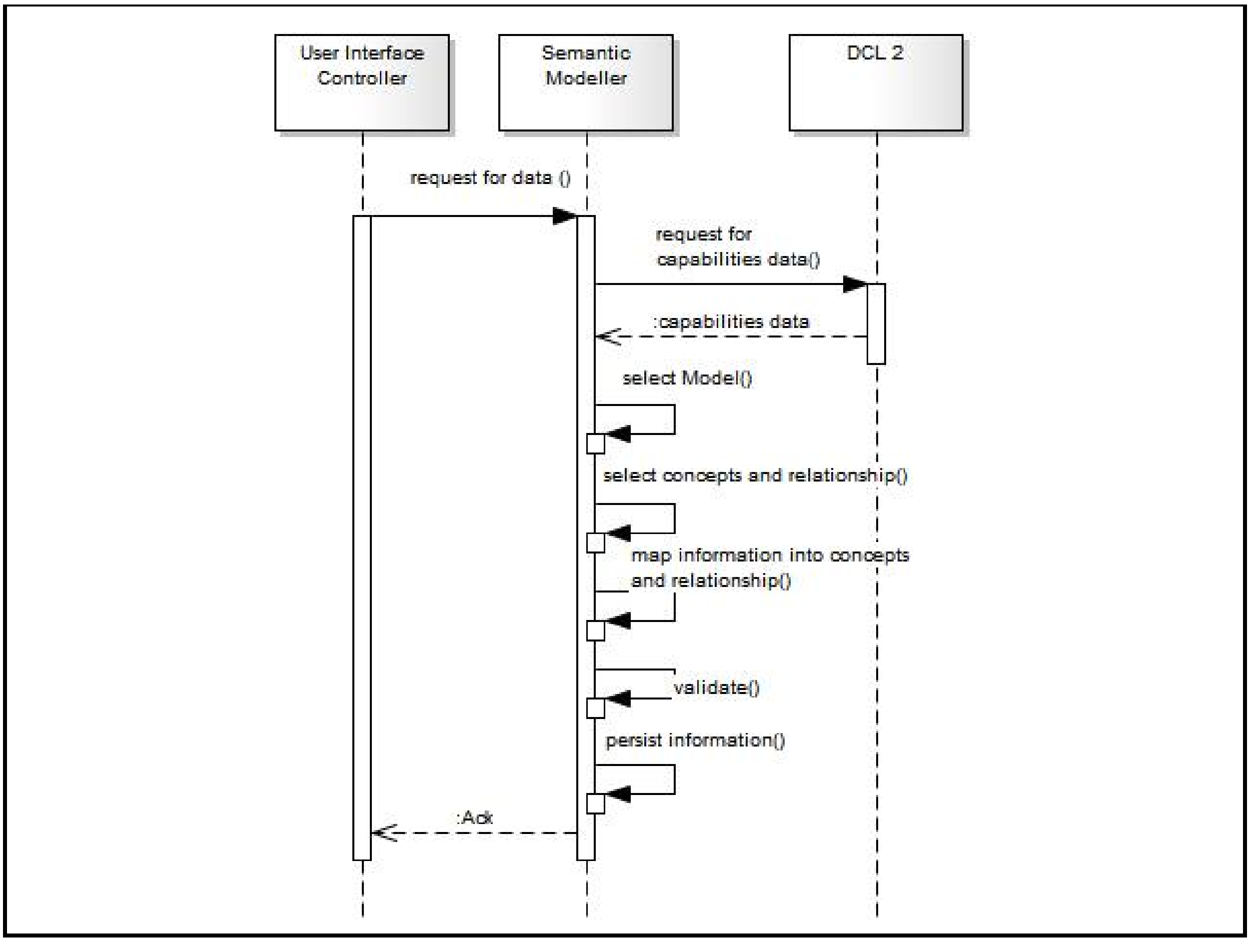
Preconditions:	The DCL provide the access to required information
Postconditions:	All required capabilities are successfully collected.
Normal Flow:	1. SL generates request for user, device, and context information colle ction from DCL 2. This information is utilized for the adaptation of the user interfaces 3. The adaptation is based on changes in user profile, context informat ion or collected observational data
Alternative Flows:	N/A
Exceptions:	If there is not capabilities information then the default user interfaces will be displayed.
Includes:	N/A
Frequency of Use:	Always when the application is running [High]
Special Requirements:	N/A
Assumptions:	The capabilities information should be available with the DCL.
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SL-UC-03

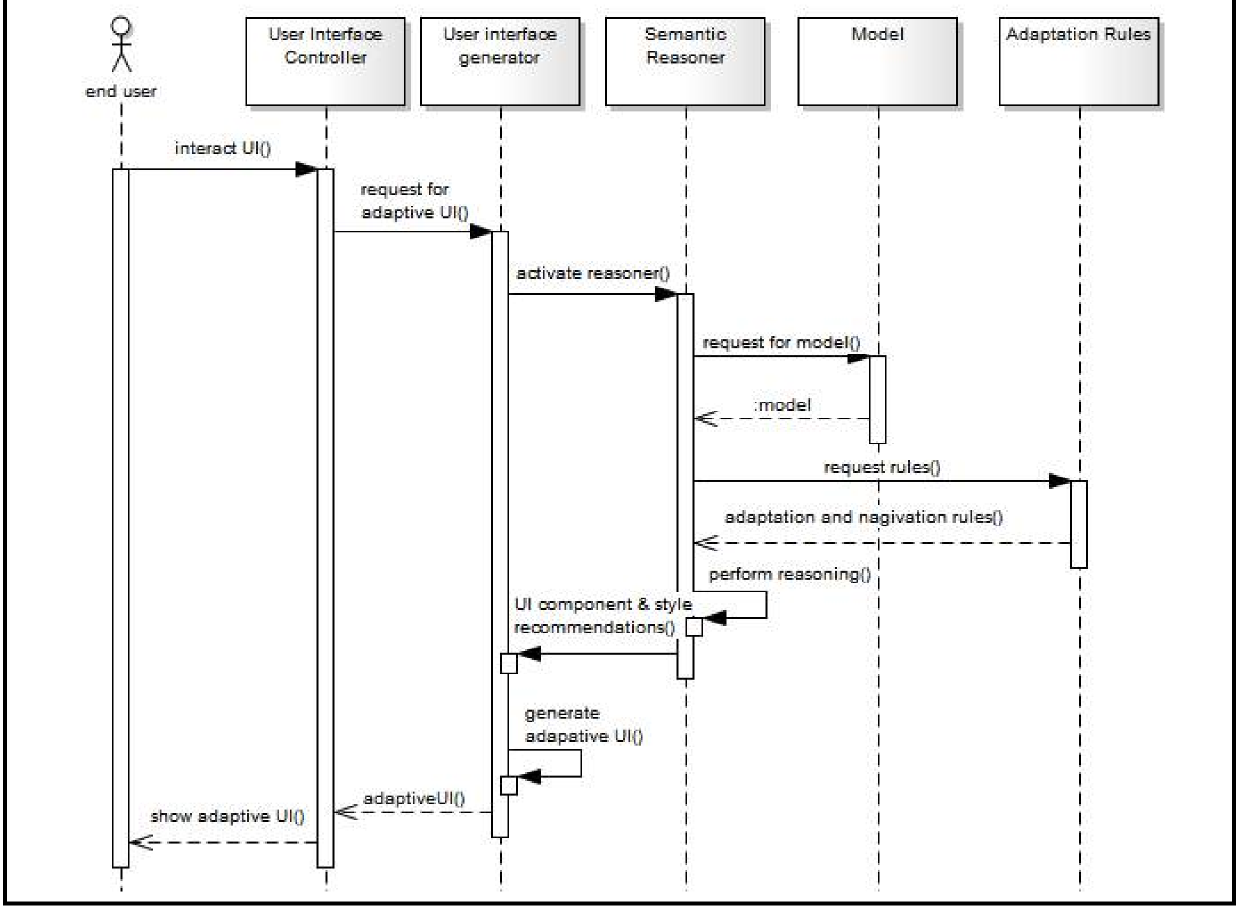
Use Case Name:	Mapping the user capability information into model		
Created By:	Jamil Hussain	**Last Updated By:**	Jamil Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017

Actors:	DCL
Description:	The collected capabilities information from DCL.0 shall be mapped against the hierarchical structure of the model
Trigger:	SL initiate communication with DCL.
Pre-conditions:	1. User is a registered client of MM platform 2. Updated user profile must be available
Post-conditions:	1. User profile and environmental variables are received by UI/UX 2. All collected variables are successfully mapped and validated
Normal Flow:	1. UI/UX send request to DCL.0 for environmental variables (e.g., tem perature, weather, time, noise, light level etc.) and user profile vari ables (e.g. uid, name, age, perceptual information) 2. DCL sent back the requested variables. 3. The semantic modeller maps the data to model 4. The mapped information is persisted in model.
Alternative Flows:	N/A
Exceptions:	
Includes:	**SL3-UCS-01**
Frequency of Use:	Always when the application is running [High]
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



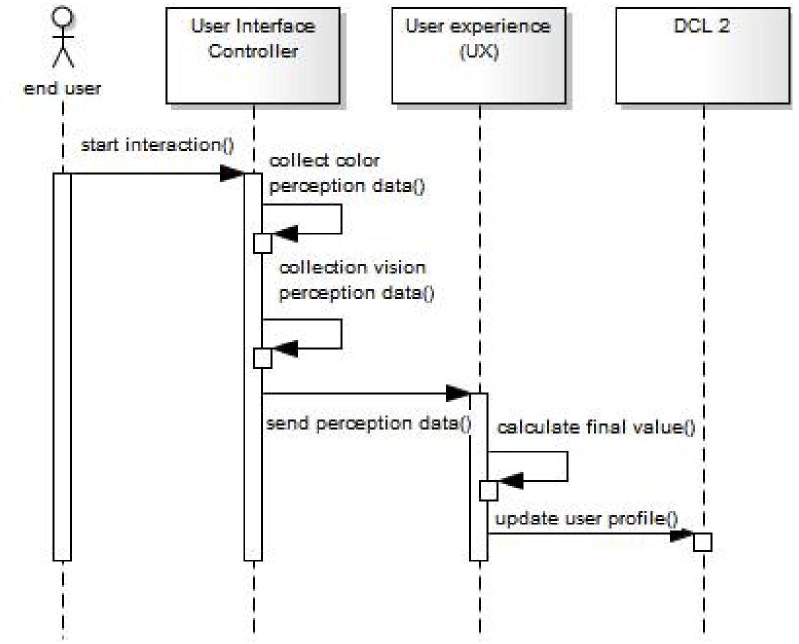
Use Case ID:	SL-UC-04		
Use Case Name:	Adapt user interface based on user profile, context and device		
Created By:	Jamil Hussain	**Last Updated By:**	Jamil Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017
Actors:	DCL, end user		
Description:	The collected information of user profile, context of use and device characteristics from DCL results in adaption of the user interface		
Trigger:	End user start interacting with user interface		
Pre-conditions:	• The user profile and context of use and device data has been colle cted by UI/UX Authoring tool		
Post-conditions:	• Adaptive UI rendered/generated based on collected information		

Normal Flow:	1. User request for personalize user interface to UI/UX. 2. UI/UX send request to adaptation engine for generating the personalized user interface based on user, context and device information. 3. Adaptation engine perform reasoning based on pre-defined adaption and navigations rules. 4. Reasoner recommend the user interface components and its styles to generate the UI. 5. Then personalized generated UI is displayed to end user.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	When user interact with the system
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SL-UC-05		
Use Case Name:	User capabilities collection		
Created By:	Jamil Hussain	**Last Updated By:**	Jamil Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017

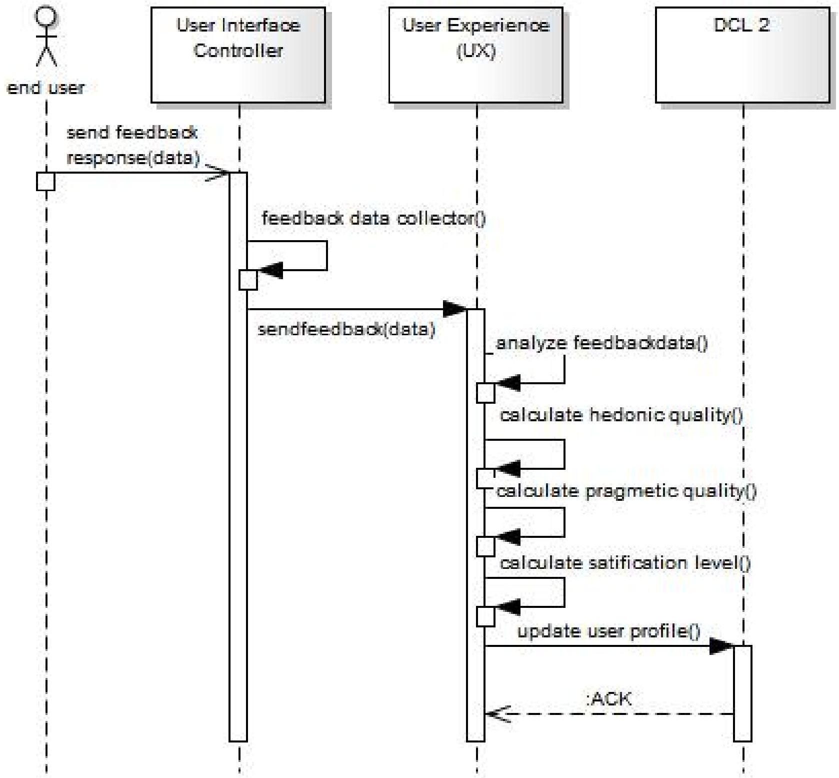
Actors:	End user, DCL
Description:	The UI/UX collects the user capabilities information's by analyzing the user perception such as user visual and color perception
Trigger:	User uses the tools for collection
Pre-conditions:	• Perception collection tools are installed
Post-conditions:	• User perceptual information successfully collected and update info rmation in user profile DCL
Normal Flow:	1. User select the tools for color and visual perception and interact w ith it accordingly. 2. Tools acquire its interaction data in order to find the user percepti ons 3. User experience calculate its final value. 4. Final values are stored in user profile DCL.
Alternative Flows:	
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	When user interact with the system
Special Requirements:	
Assumptions:	Service contract between SL and DCL is defined
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SL-UC-06		
Use Case Name:	Self-reporting user experience measurement		
Created By:	Jamil Hussain	**Last Updated By:**	Jamil Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017

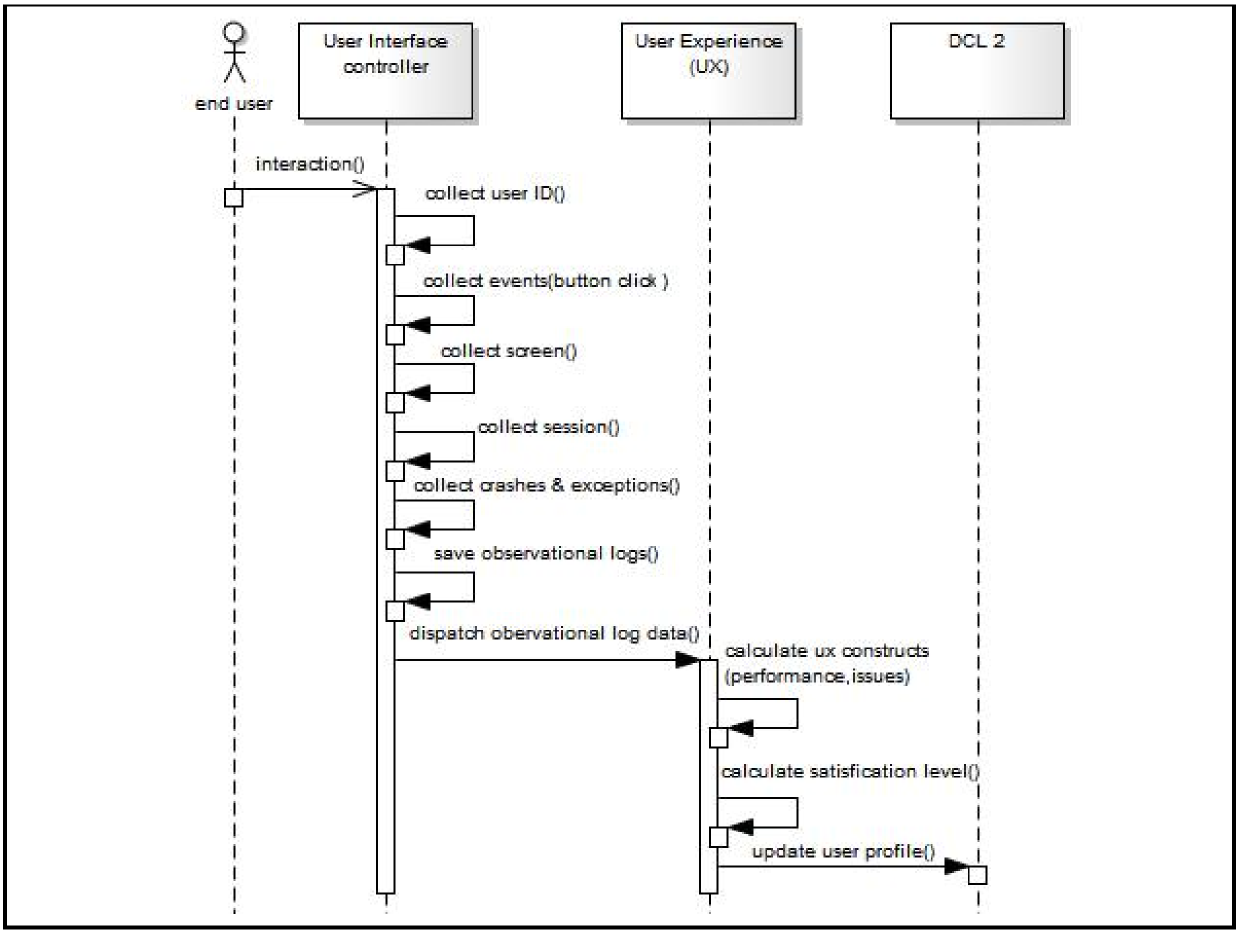
Actors:	End user, DCL
Description:	The UI/UX shall collect feedback about how users feel about the system during or after use by self-reporting method.
Trigger:	End user
Pre-conditions:	• Self-reporting questionnaire already exist
Post-conditions:	• The feedback is successfully collected
Normal Flow:	1. The end user provide feedback using the questionnaire. 2. The feedback is sent to user experience in order to evaluate the us er response. 3. user experience variables such as usability, pleasure, beauty are ca lculated based on filled questions 4. The UI/UX update the calculated variables values in user profile by sending request to DCL

Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	When user interact with the system
Special Requirements:	N/A
Assumptions:	N/A
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SL-UC-07		
Use Case Name:	Collect and analyze observational data		
Created By:	Jamil Hussain	**Last Updated By:**	Jamil Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017
Actors:	End user, DCL		

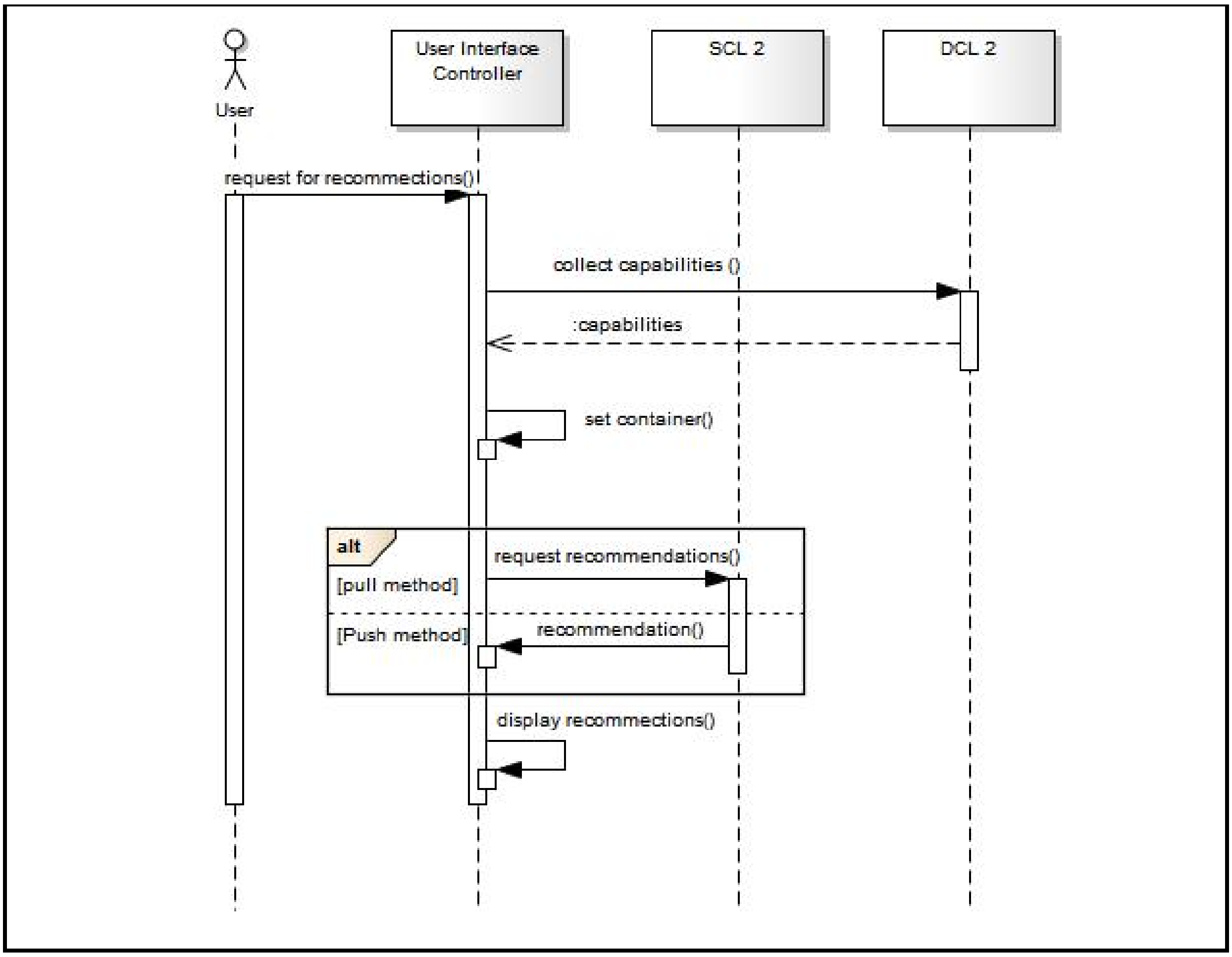
Description:	The UI/UX shall identify areas of improvement and maximize the user interaction by analyzing the user interaction with app.
Trigger:	Initiated by end user
Pre-conditions:	• Analytics tracker is already installed
Post-conditions:	• Observational data are successful collected and analyzed for user experience measurement
Normal Flow:	1. Analytics collector collect the user interaction data such as user ID, event, session, screen, crashes & exceptions, and user timings 2. The collected data is stored locally before being dispatched 3. Data is dispatched for user experience measurement from the app for every 30 minutes 4. the pragmatic quality such as usability-(e.g. performance, issues) are calculated in order to find the user experience (UX) 5. UX quality variables are sent to DCL2 for storage/updating in user profile.
Alternative Flows:	N/A
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Frequent, request by SL
Special Requirements:	N/A
Assumptions:	Service contract between DCL and SL is defined
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SL-UC-08		
Use Case Name:	Acquire Recommendations for displaying to end user		
Created By:	Jamil Hussain	**Last Updated By:**	Jamil Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017

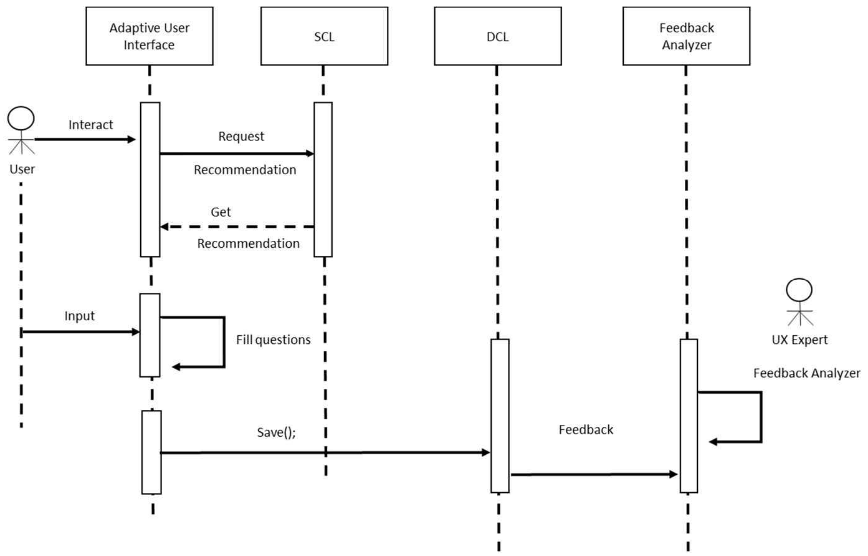
Actors:	Primary: End-user Secondary: SCL, DCL
Description:	This use case collects the recommendations generated by SCL and displays it on the user interface for the end users. The provided recommendations are displayed according to user capabilities, context of use, and device characteristics. This information is obtained from the DCL.
Trigger:	SCL push the recommendations to the App or end-user send request for recommendations
Preconditions:	End-user subscribes to particular services

Postconditions:	All recommendations are successfully displayed according to user capabilities, context, and device characteristics.
Normal Flow:	1. SCL generate the recommendations and provide it to user interface 2. The SCL recommendations are acquired by the SL 3. SL investigates the user capabilities, context of use, and device charact eristics by obtaining from DCL 4. The recommendation are displayed in graphical user interface based o n collected capabilities of user, context and device information.
Alternative Flows:	2a. In step 2. The SCL recommendations are acquired by the SL 1. user request for recommendations (pull method) 2b. In step 2. The SCL recommendations are acquired by the SL 1. SL push recommendations to App based on situations
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Whenever the recommendations are generated by SCL [Medium]
Special Requirements:	N/A
Assumptions:	The user profile data and context information should exist in the DCL
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SL-UC-09		
Use Case Name:	Feedback Collection		
Created By:	Jamil Hussain	**Last Updated By:**	Jamil Hussain
Date Created:	March 09 2017	**Last Revision Date:**	09 March 2017

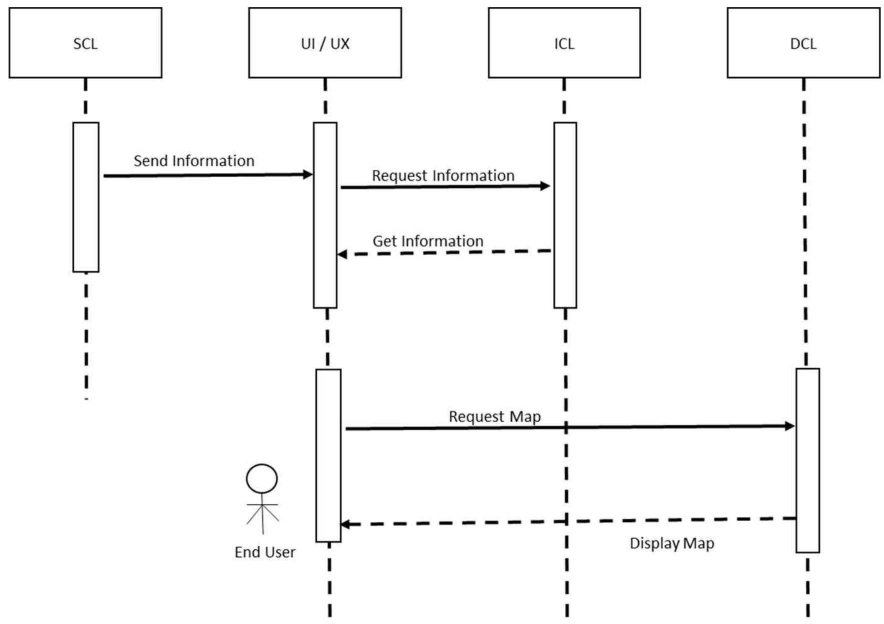
Actors:	Primary: End-user Secondary: DCL
Description:	The UI/UX shall collect feedback about how users feel about recommendations and additional questionnaires in a predefined time period
Trigger:	End user
Preconditions:	Self-reporting questionnaire already exist
Postconditions:	The feedback is successfully collected

Normal Flow:	1. The end user provide feedback using the questionnaires and feedback about recommendation. 2. The DCL stores the feedback in the lifelog for user preferences in the case of feedback. 3. The questionnaires are stored in the lifelog for further analysis from the expert
Alternative Flows:	
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Every 3 weeks
Special Requirements:	N/A
Assumptions:	The feedback data is saved in DCL 3.0
Notes and Issues:	N/A
Sequence Diagram	

sa

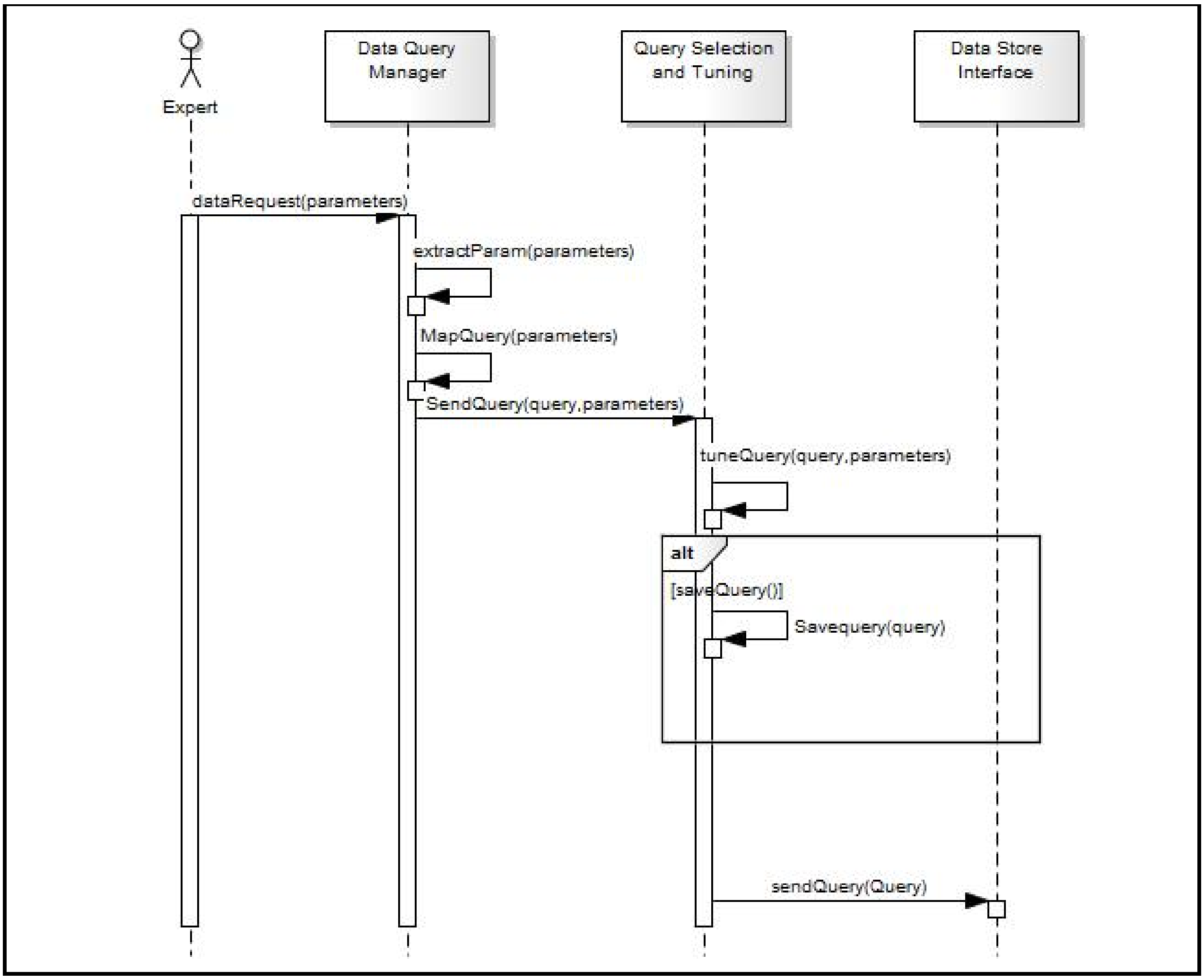
Use Case ID:	SL-UC-10		
Use Case Name:	Retrieve capabilities for context based services		
Created By:	Jamil Hussain	**Last Updated By:**	Jamil Hussain
Date Created:	March 09 2017	**Last Revision Date:**	March 09 2017

Actors:	DCL, SCL
Description:	The UI/UX shall collect feedback about how users feel about recommendations and additional questionnaires in a predefined time period
Trigger:	End user
Preconditions:	Self-reporting questionnaire already exist
Postconditions:	The feedback is successfully collected
Normal Flow:	1. The end user provide feedback using the questionnaires and feedback about recommendation. 2. The DCL stores the feedback in the lifelog for user preferences in the case of feedback. 3. The questionnaires are stored in the lifelog for further analysis from the expert
Alternative Flows:	
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	Every 3 weeks
Special Requirements:	N/A
Assumptions:	The feedback data is saved in DCL
Notes and Issues:	N/A
Sequence Diagram	



Use Case ID:	SL-UC-11		
Use Case Name:	Map Request to Query		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017
Actors:	Primary: Expert		
Description:	This use case focuses on mapping the expert request to the query library for data store interface.		
Trigger:	The request from the expert panel for analytics		
Preconditions:	A predefined query library for retrieving the big data		
Postconditions:	The query is sent to the data store interface and the data is fetched.		

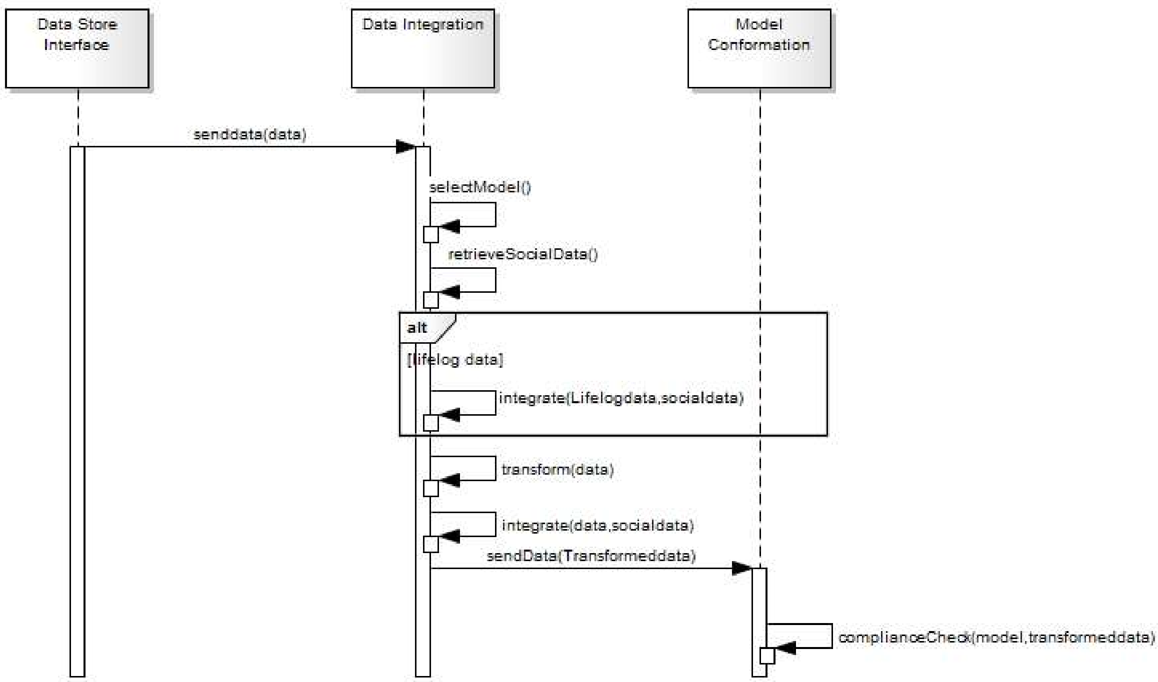
Normal Flow:	1. The expert requests the analytics for a specific context. 2. The parameters of the request is extracted and sent to query manager. 3. The query manager matches the parameters with the predefined queries in the query library. 4. The Query is selected and tuned according to the duration of the data to be extracted.
Alternative Flows:	4a. In step 4 of the normal flow, if there is a more tuning done than the query 1. The query is saved in the library for future calls.
Exceptions:	N/A
Includes:	N/A
Frequency of Use:	This use case can be used by the domain expert about 5-10 times based on the volume of data. [Low]
Special Requirements:	N/A
Assumptions:	For this use case the assumption is a query library.
Notes and Issues:	1. How many queries are there in the query library?
Sequence Diagram	



Use Case ID:	SL-UC-12		
Use Case Name:	Transform Data		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017

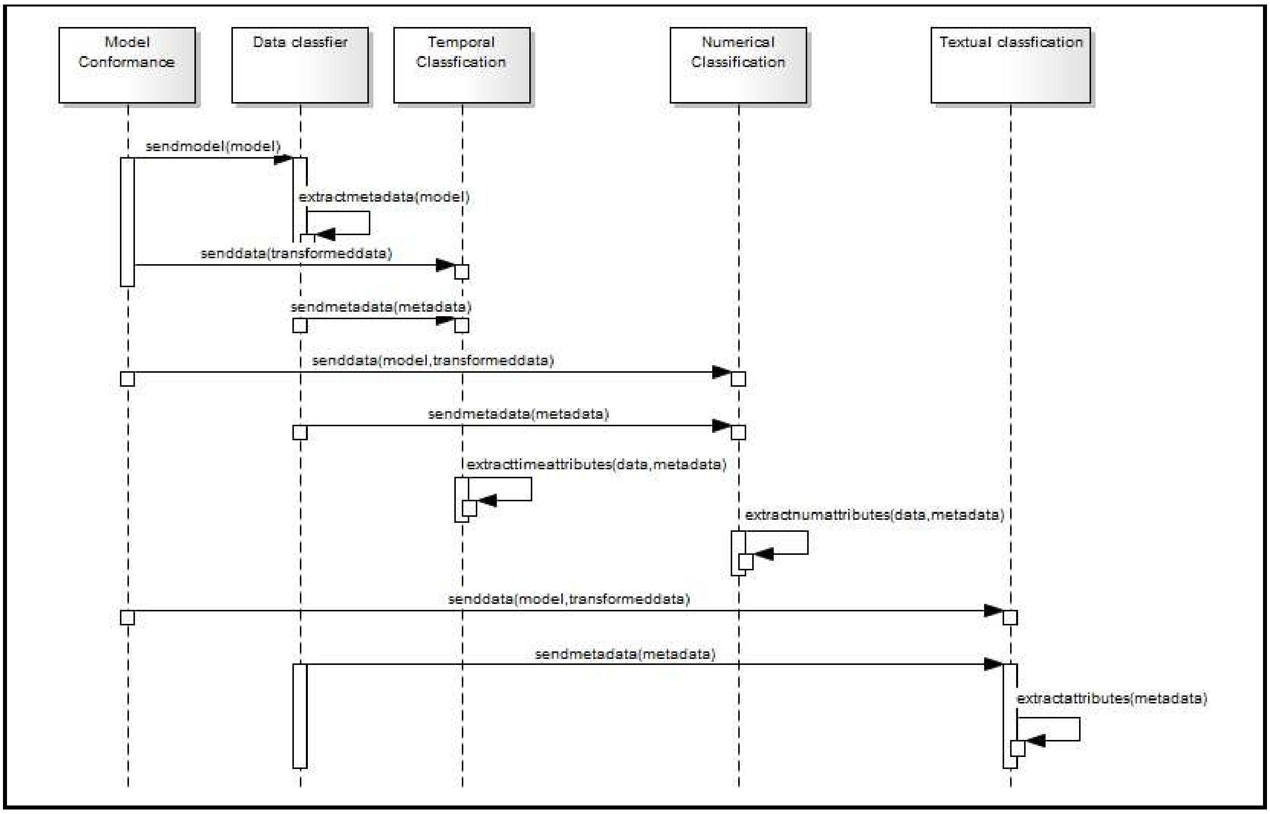
Actors:	Primary: Expert
Description:	The mapping query is transformed to specific model structure for trend analysis.
Trigger:	The data store interface initiates the data transformation process
Preconditions:	The data is sent from the data store interface.

Postconditions:	The transformed data is sent to trend analyzer.
Normal Flow:	1. The unstructured data from the big data repository is sent to the data inte gration component. 2. The data is transformed in an object model or a table depending on the re quirements. 3. The social network data is than additionally integrated which is retrieved t hrough a web service. 4. The transformed data is then checked for compliance with the model tem plate.
Alternative Flows:	2a. In step 2 of the normal flow, if the data is retrieved from the life log then it is sent directly to the integration component.
Exceptions:	If the transformed data does not pass the compliance check, step 2 is started again.
Includes:	N/A
Frequency of Use:	This use case is used when the data comes from the big data and requires social data integration. [Low]
Special Requirements:	N/A
Assumptions:	
Notes and Issues:	1. How many models can the data be transformed in?
Sequence Diagram	



Use Case ID:	SL-UC-13		
Use Case Name:	Classify Data		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017

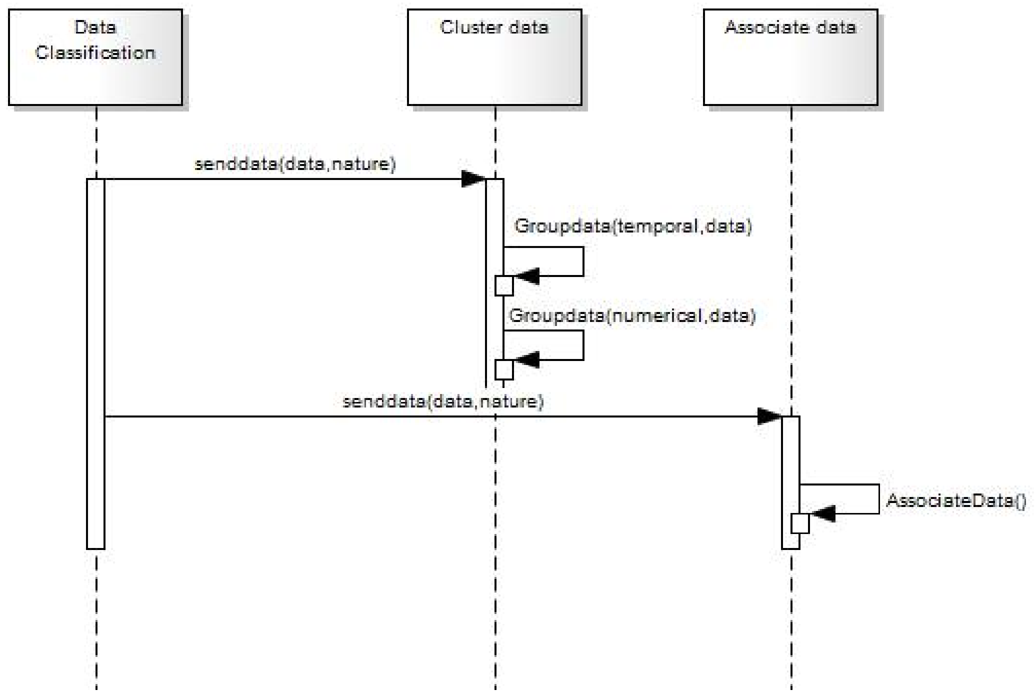
Actors:	Primary: Expert
Description:	The transformed data is further classified and clustered to identify and analyze trends.
Trigger:	The transformed data is sent for trend analysis.
Preconditions:	The data is structured into a particular model.
Postconditions:	The data is classified into temporal, numerical and textual categories
Normal Flow:	1. The model is passed for the classification. 2. Metadata is extracted from the model. 3. The data is categorized based on the extracted metadata. 4. The temporal, numerical and textual data is extracted from the transform ed data.
Alternative Flows:	
Exceptions:	1. There is no temporal data to be classified. 2. There is no numerical data to be classified. 3. There is no textual data to be classified.
Includes:	
Frequency of Use:	This use case is used when the transformed data comes from model transformation module. [Low]
Special Requirements:	
Assumptions:	
Notes and Issues:	
Sequence Diagram	



Use Case ID:	SL-UC-14		
Use Case Name:	Analyze Data		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017

Actors:	Primary: Expert
Description:	The classified data is analyzed through association and clustering techniques for visualization and analytics.
Trigger:	The classified data is passed to association clustering for finding analytics and trends.
Preconditions:	The numerical, temporal and textual data is classified separately so that association could be applied.
Postconditions:	The association is done with the data for analytics and data to be plotted is sent for visualization.

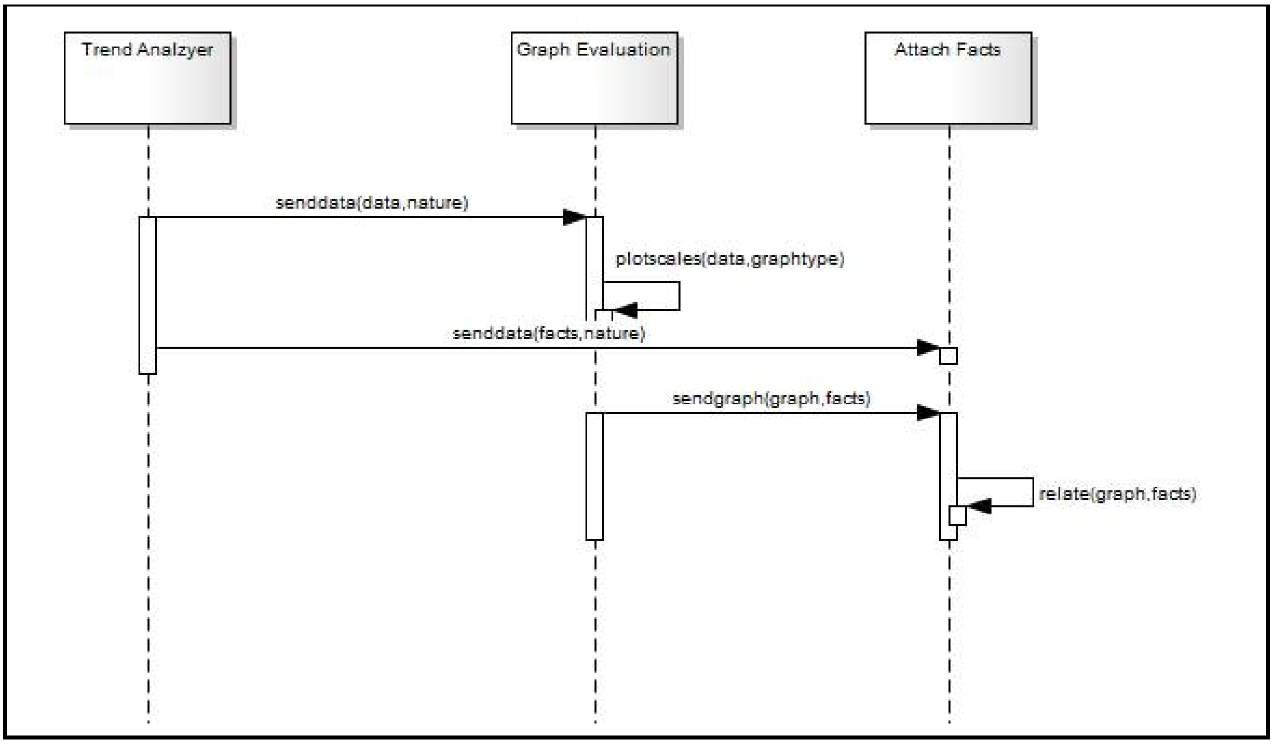
Normal Flow:	1. The data classifier passes the data for association clustering. 2. The temporal and numerical data is analyzed for clustering. 3. The data is clustered into a group for graph plotting. 4. The textual data is associated with each other to create analytics based on the textual attribute and their corresponding facts.
Alternative Flows:	
Exceptions:	1. Clustering could not be done due to multiple outliers.
Includes:	
Frequency of Use:	This use case is used when the classified data is sent for grouping and association. [Low]
Special Requirements:	
Assumptions:	
Notes and Issues:	
Sequence Diagram	



Use Case ID:	SL-UC-15
Use Case Name:	Display Analytics

Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017

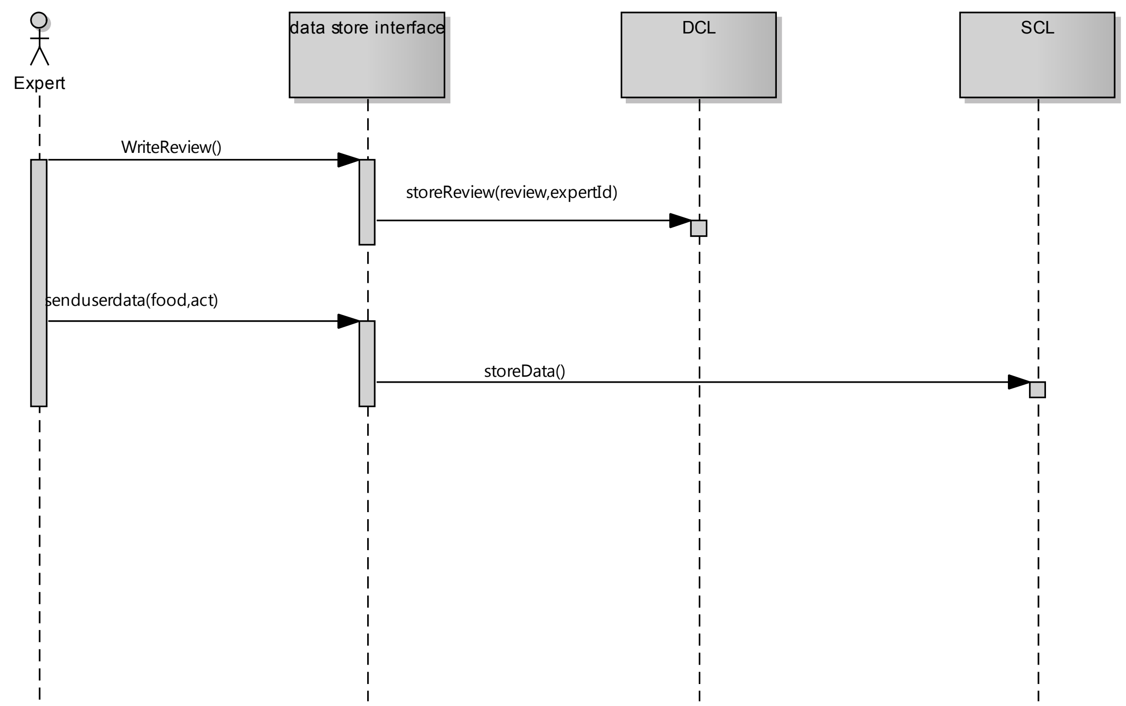
Actors:	Primary: Expert
Description:	The grouped data and relevant analytics is passed to visualization enabler so that the graphs are plotted and displayed.
Trigger:	The trend analyzer sends the data for graph visualization and plotting.
Preconditions:	The data is sent to visualization enabler distinguishable by their attributes and association.
Postconditions:	The analytics and relevant visualization is sent to the user interface.
Normal Flow:	1. The data is categorized according to the graph templates for visualization. 2. The scales are defined for the grouped data to be plotted on the coordina tes. 3. The association text and the relevant facts about the data is also attached to the graph as analytics.
Alternative Flows:	
Exceptions:	
Includes:	
Frequency of Use:	This use case is used when the grouped data is sent for display in graph and analytics form. [Low]
Special Requirements:	
Assumptions:	
Notes and Issues:	
Sequence Diagram	



Use Case ID:	SL-UC-16		
Use Case Name:	Take Expert Input		
Created By:	Shujaat Hussain	**Last Updated By:**	Shujaat Hussain
Date Created:	14 July 2015	**Last Revision Date:**	09 March 2017

Actors:	Primary: Expert
Description:	This use case focuses on taking expert input on activity and nutrition for the user's analytics
Trigger:	The request from the expert panel for input
Preconditions:	The user meal and activity information is stored
Postconditions:	The query is sent to the data store interface and the data is stored.

Normal Flow:	1. The expert requests the input panel for a specific user. 2. The parameters of the request is extracted and sent to query manager. 3. The query manager matches the parameters with the predefined queries in the query library. 4. The interface is shown for expert to input activity and meal information 5. It is stored in the life log
Alternative Flows:	4a. In step 4 of the normal flow, if there is a more tuning done than the query 1. The query is saved in the library for future calls.
Exceptions:	
Includes:	
Frequency of Use:	
Special Requirements:	
Assumptions:	
Notes and Issues:	
Sequence Diagram	



Section 6

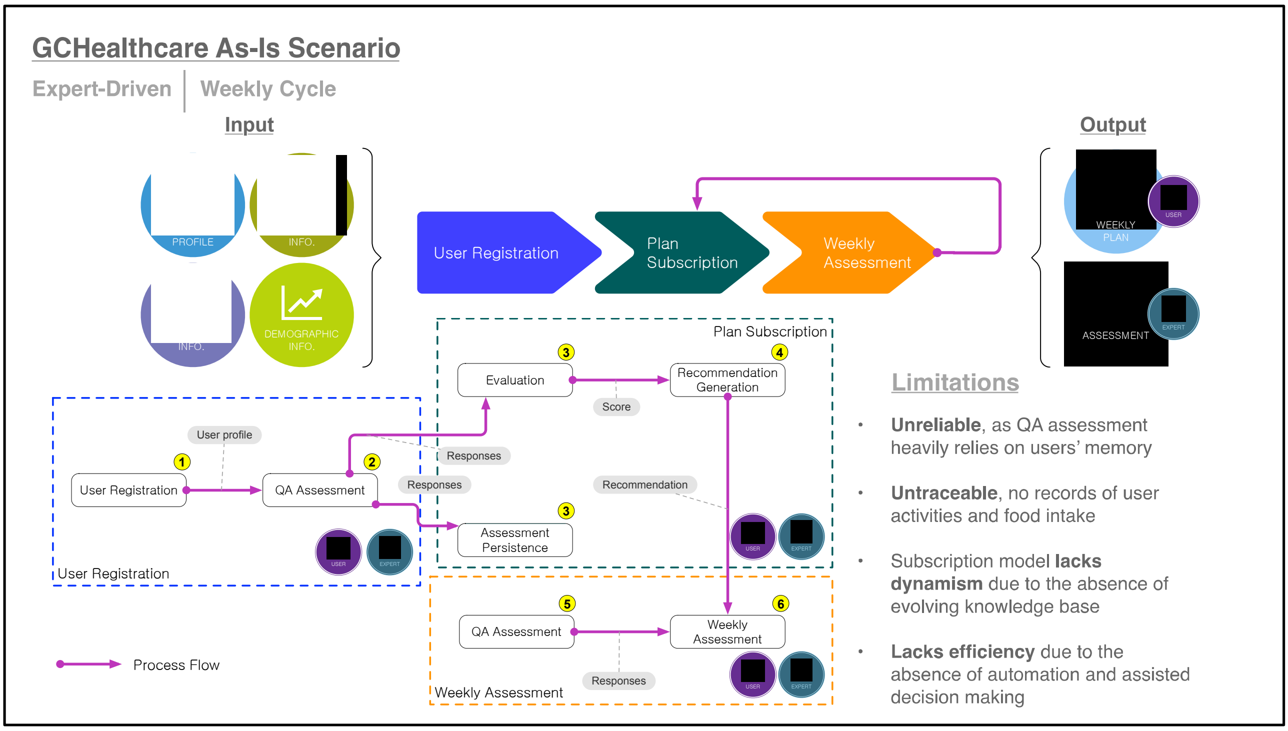
Warm and Cold Service Scenario

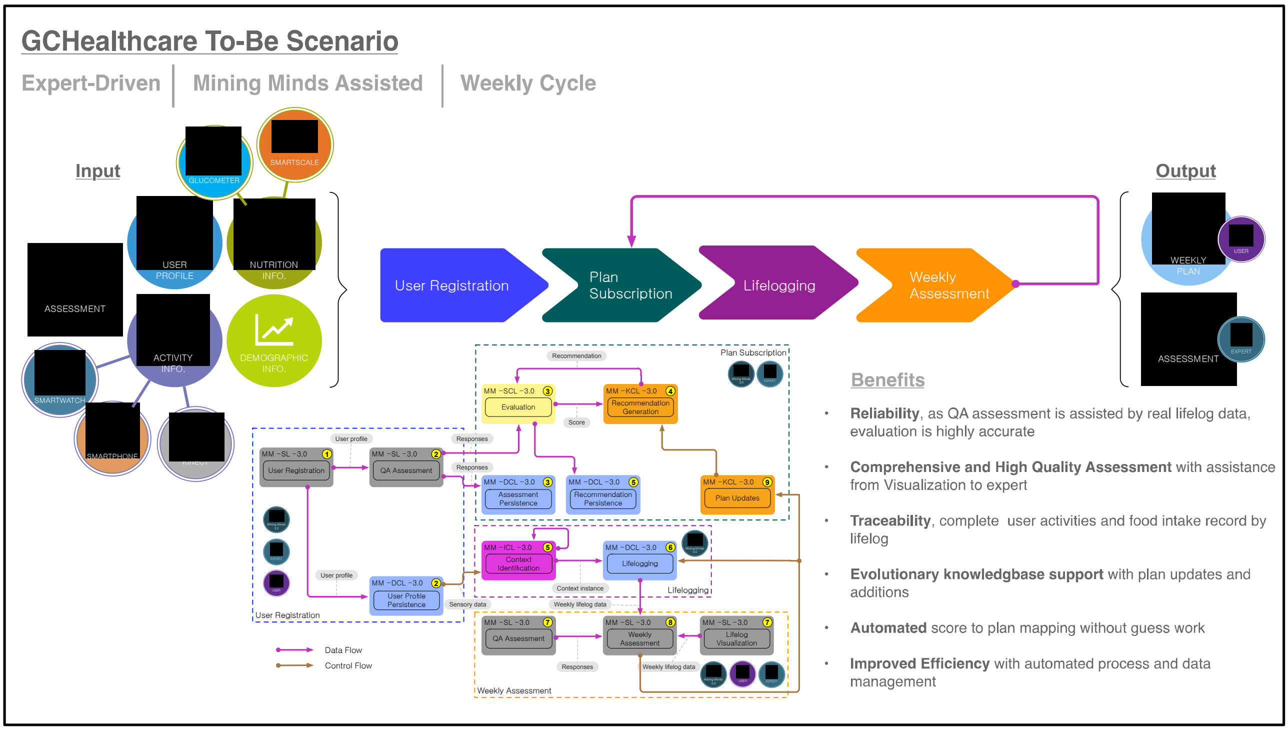
Mining Minds Version 3.0

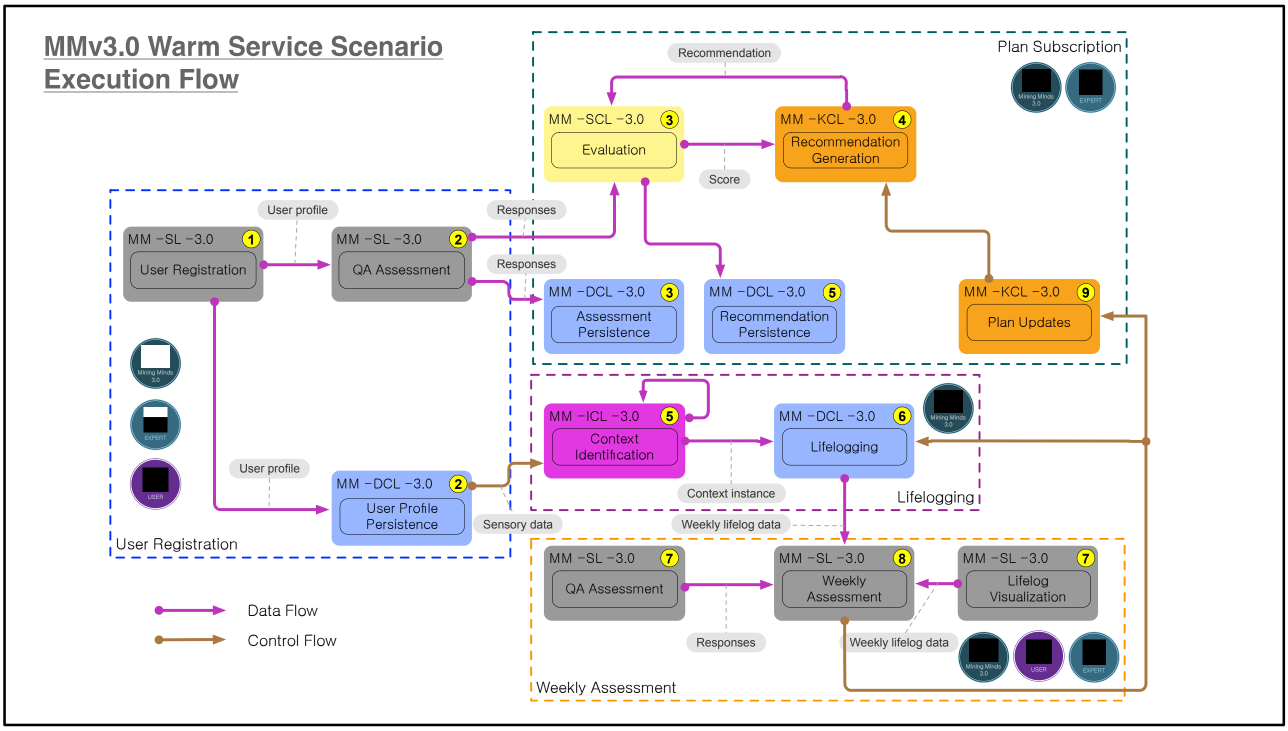
Service Scenario

Mining Minds Version 3.0

Warm Service Scenario


GCHealthcare As-Is Scenario
Expert-Driven
Weekly Cycle
Input
PROFILE
INFO.
INFO.
DEMOGRAPHIC INFO.
User Registration
Plan Subscription
Weekly Assessment
Output
WEEKLY PLAN
ASSESSMENT
USER
EXPERT
Plan Subscription
Evaluation
Recommendation Generation
Score
Responses
Recommendation
Assessment Persistence
User profile
User Registration
QA Assessment
Responses
User Registration
QA Assessment
Weekly Assessment
Responses
Weekly Assessment
Process Flow
Limitations
Unreliable, as QA assessment heavily relies on users' memory
Untraceable, no records of user activities and food intake
Subscription model lacks dynamism due to the absence of evolving knowledge base
Lacks efficiency due to the absence of automation and assisted decision making


GCHealthcare To-Be Scenario
Expert-Driven
Mining Minds Assisted
Weekly Cycle
Input
GLUCOMETER
SMARTSCALE
USER PROFILE
NUTRITION INFO.
ASSESSMENT
ACTIVITY INFO.
DEMOGRAPHIC INFO.
SMARTWATCH
SMARTPHONE
User Registration
Plan Subscription
Lifelogging
Weekly Assessment
Output
WEEKLY PLAN
ASSESSMENT
USER
EXPERT
Plan Subscription
Recommendation
Evaluation
Recommendation Generation
Score
Plan Updates
Assessment Persistence
Recommendation Persistence
User Registration
QA Assessment
User Profile Persistence
Context Identification
Lifelogging
Context instance
Sensory data
Weekly lifelog data
QA Assessment
Weekly Assessment
Lifelog Visualization
Responses
Weekly Assessment
Data Flow
Control Flow
Benefits
Reliability, as QA assessment is assisted by real lifelog data, evaluation is highly accurate
Comprehensive and High Quality Assessment with assistance from Visualization to expert
Traceability, complete user activities and food intake record by lifelog
Evolutionary knowledgbase support with plan updates and additions
Automated score to plan mapping without guess work
Improved Efficiency with automated process and data management



Registration Stage – Register and Data Gathering

<Registration>

- User ID (account code)
- Nickname
- Password
- Linked SNS (e.g. the FACEBOOK ID, S-HEALTH ID)

<Contact>

- Contact Number(Mobile)
- Address
- e-mail

<Client Information>

- Client Name
- Date of Birth
- Gender

<Somatology>

- Height
- Weight
- Somatotype of Body

<Medical History on Lifestyle Diseases>

- Diabetes (if checked, **additional input form presented**)
- Hypertension (additional form) **[Mining Minds ver. 3.5]**
- Dyslipidemia (additional form) **[Mining Minds ver. 3.5]**

User Registration Form

General user registration/authorization modules

User's Profile Check-up

CONTACT NUMBER UID.Phone.Mobile
ADDRESS: UID.Address.Home
E-MAIL: UID.Email

NAME: UID.Name
Date of Birth: UID.Date_of_Birth
Gender: UID.Sex

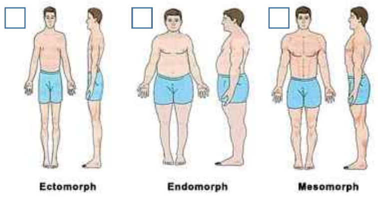
Height: UID.Height
Weight: UID.Weight
Somatotype: UID.Somatotype

Diagnosed Diseased &MedicalStatus.Diabetes.Diagnosed
&MedicalStatus.Hypertension.Diagnosed
&MedicalStauts.Dyslipidemia.Diagnosed

Medical Check-up Data MedicalStatus.Data.FAST
MedicalStatus.Data.OGTT50
MedicalStatus.Data.HbA1c
MedicalStatus.Data.DBP
MedicalStatus.Data.SBP
MedicalStatus.Data.TC
MedicalStatus.Data.HDLC
MedicalStatus.Data.LDLC
MedicalStatus.Data.TG

<Somatotype Assessment>

Question: What is your body type? Choose most similar type in the pictures.
Choices:



Data:

UID	Field Name	Date & Time of Assessment	Toolset	Data	Unit	Certified by
####	UID.Height	yyyyMMddHHmmss	{UserInput}	Integer	cm	USER
####	UID.Weight	yyyyMMddHHmmss	{UserInput, DeviceID}	###.#	kg	USER
####	UID.Somatotype	yyyyMMddHHmmss	{UserChoice, ST_msr01, ST_msr02, …}	{Ectomorph, Endomorph, Mesomorph; NULL}	N/A	USER
####	UID.BMI	yyyyMMddHHmmss	{Rule.Index.BMI}	##.#	kg/m^2	SYSTEM

Rule.Index.BMI
input: UID.Height.Current_date, UID.Weight.Current_date
output: UID.BMI = UID.Weight.Current_date/(UID.Height.Current_date*UID.Height.Current_date)

<Medical History Assessment>

- **[Mining Minds ver. 3.5]**

Question: Have you ever been diagnosed with any of following diseases or conditions?

☐ Diabetes or prediabetes
☐ Hypertension or Prehypertension
☐ Dyslipidemia

If the boxes are checked, present additional questions.

✓ Diabetes or prediabetes
- Input your recent FAST test result : ( ) mg/dL ☐ Don't Know
- OGTT 50g test result : ( ) mg/dL ☐ Don't Know
- HbA1c : ( ) % or ( ) mmol/mol ☐ Don't Know

✓ Hypertension or Prehypertension
- Recent measure of Blood Pressure : Systolic ( )mmHg, Diastolic ( )mmHg

✓ Dyslipidemia
- Recent measure of Total Cholesterol : ( ) mg/dL
- Recent measure of HDL Cholesterol : ( ) mg/dL
- Recent measure of LDL Cholesterol : ( ) mg/dL
- Recent measure of Triglyceride : ( ) mg/dL

Data:

UID	Field Name	Date & Time of Assessment	Toolset	Data	Unit	Certified by
####	MedicalStatus.Diabetes.Diagnosed	yyyyMMddHHmmss	{UserChoice, ExpertInput}	{Yes; NULL}	N/A	USER
####	MedicalStatus.Hypertension.Diagnosed	yyyyMMddHHmmss	{UserChoice, ExpertInput}	{Yes; NULL}	N/A	USER
####	MedicalStauts.Dyslipidemia.Diagnosed	yyyyMMddHHmmss	{UserChoice, ExpertInput}	{Yes; NULL}	N/A	USER
####	MedicalStatus.Data.FAST	yyyyMMddHHmmss	{UserInput, DeviceID}	Integer	mg/dL	USER
####	MedicalStatus.Data.OGTT50	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER
####	MedicalStatus.Data.HbA1c	yyyyMMddHHmmss	{UserInput}	##.# ##.# Integer	% mmol/mol mg/dL	USER
####	MedicalStatus.Data.DBP	yyyyMMddHHmmss	{UserInput, DeviceID}	Integer	mmHg	USER
####	MedicalStatus.Data.SBP	yyyyMMddHHmmss	{UserInput, DeviceID}	Integer	mmHg	USER
####	MedicalStatus.Data.TC	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER
####	MedicalStatus.Data.HDLC	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER
####	MedicalStatus.Data.LDLC	yyyyMMddHHmmss	{UserInput}	###.#	mg/dL	USER
####	MedicalStatus.Data.TG	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER

QA-based Assessment Stage

<Questionnaire: Dietary Habits>

- Eating Regularity
- Food Group Balance
- Whole Grain Uptakes
- Protein Uptakes
- Vegetable Uptakes
- Fruits Uptakes
- Dairy Uptakes
- Red Meat Uptakes
- Sodium Uptakes
- Added Sugar Uptakes

Dietary Habits Questionnaire

Selected dietary habit assessment tools
(ex, GCH's Questionnaire)
1. During the last month, how regularly do you eat?
2. Do you enjoy a wide variety of nutritious foods from the five groups every day?
3. When you eat cooked-rice, do you usually eat cooked-rice with multi-grains?
4. Do you usually take protein foods* in every servings?
5. Do you usually eat vegetables in every servings?
6. Do you usually eat fruits every day?
7. Do you usually dairy foods every day?
8. Do you usually eat red meat or high fat meat more than twice a week?
9. Do you usually take salty side dishes everyday?
10. Do you usually eat snacks or beverages with added sugars every day?

Generate Initial Values for User Profile

<Dietary Habits Data (in User's Profile DB)>
Eating Regularity: DietaryHabits.EP.Regularity (1pt ~ 5pt)
Food Group Balance: DietaryHabits.EP.FGBalance (1pt ~ 5pt)
Whole Grain Uptakes: DietaryHabits.NT.WGUtk (1pt ~ 5pt)
Protein Uptakes: DietaryHabits.NT.PtUtk (1pt ~ 5pt)
Vegetable Uptakes: DietaryHabits.NT.VgUtk (1pt ~ 5pt)
Fruits Uptakes: DietaryHabits.NT.FtUtk (1pt ~ 5pt)
Dairy Uptakes: DietaryHabits.NT.DyUtk (1pt ~ 5pt)
Red Meat Uptakes: DietaryHabits.NT.RMUtk (1pt ~ 5pt)
Sodium Uptakes: DietaryHabits.NT.NaUtk (1pt ~ 5pt)
Added Sugar Uptakes: DietaryHabits.NT.SgUtk (1pt ~ 5pt)

<Lifestyle Questionnaire – Dietary Habits> CALL: GCH-DietPic-QN-DH-v001 questionnaire modules

Q1. Eat Regularity (DietaryHabits.EP.Regularity)

During the last month, how regularly (eat three times a day) do you eat?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q2. Food Group Balance (DietaryHabits.EP.FGBalance)

Do you enjoy a wide variety of nutritious foods from the five groups every day? (the five groups: fruits, vegetables, grains, protein foods, and dairy)

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q3. Whole Grain Uptakes (DietaryHabits.NT.WGUtk)

When you eat cooked-rice, do you usually eat cooked-rice with multi-grains?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q4. Protein Uptakes (DietaryHabits.NT.PtUtk)

Do you usually take protein foods* in every servings? * Lean Meat (except red meat and high fat meat), Poultry, Fish and seafood, Eggs, Nuts and Seeds, Legumes/Beans

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q5. Vegetables Uptakes (DietaryHabits.NT.VgUtk)

Do you usually eat vegetables in every servings?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q6. Fruit Uptakes (DietaryHabits.NT.FtUtk)

Do you usually eat fruits everyday?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q7. Dairy Uptakes (DietaryHabits.NT.DyUtk)

Do you usually dairy foods everyday?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q8. Red Meat Uptakes (DietaryHabits.NT.RMUtk)

Do you usually eat red meat or high fat meat more than twice a week?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q9. Sodium Uptakes (DietaryHabits.NT.NaUtk)

Do you usually take salty side dishes everyday?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

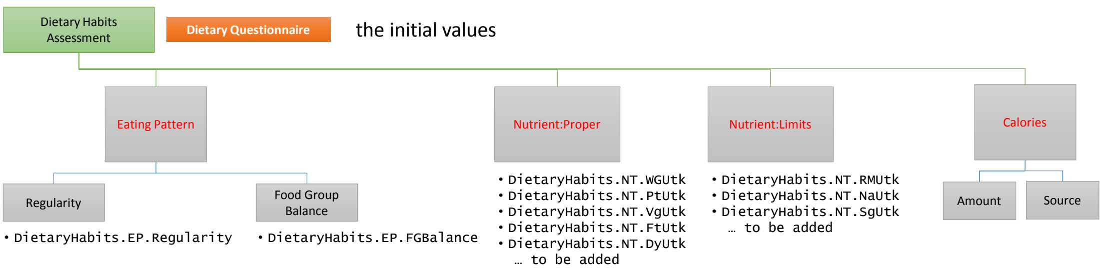
Q10. Added Sugar Uptakes (DietaryHabits.NT.SgUtk)

Do you usually eat snacks or beverages with added sugars every day?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

GCH-DietPic-QN-DH-v001 questionnaire modules

No	Classification			Variable Name	Measure	Score Direction	Question Reference	Answer Choice and Measure Example (Ans to Ref. Question)
1	Dietary Habits Screening	Eating Pattern	Eating Regularity	DQ_EP.Regularity	5-Scale Points	Positive LOW	During the last month, how regularly do you eat? (Eat three times a day)	Almost Always (5) Sometimes (3) Seldom/Never (1)
2	Dietary Habits Screening	Eating Pattern	Food Group Balance	DQ_EP.FGBalance	5-Scale Points	Positive LOW	Do you enjoy a wide variety of nutritious foods from the five groups every day? (the five groups: fruits, vegetables, grains, protein foods, and dairy)	Almost Always (5) Sometimes (3) Seldom/Never (1)
3	Dietary Habits Screening	Nutrient	Whole Grain Uptakes	DQ_NT.WGUtk	5-Scale Points	Positive LOW	When you eat cooked-rice, do you usually eat cooked-rice with multi-grains?	Almost Always (5) Sometimes (3) Seldom/Never (1)
4	Dietary Habits Screening	Nutrient	Protein Uptakes	DQ_NT.PtUtk	5-Scale Points	Positive LOW	Do you usually take protein foods* in every servings? * Lean Meat (except red meat and high fat meat), Poultry, Fish and seafood, Eggs, Nuts and Seeds, Legumes/Beans	Almost Always (5) Sometimes (3) Seldom/Never (1)
5	Dietary Habits Screening	Nutrient	Vegetables Uptakes	DQ_NT.VgUtk	5-Scale Points	Positive LOW	Do you usually eat vegetables in every servings?	Almost Always (5) Sometimes (3) Seldom/Never (1)
6	Dietary Habits Screening	Nutrient	Fruits Uptakes	DQ_NT.FtUtk	5-Scale Points	Positive LOW	Do you usually eat fruits every day?	Almost Always (5) Sometimes (3) Seldom/Never (1)
7	Dietary Habits Screening	Nutrient	Dairy Uptakes	DQ_NT.DyUtk	5-Scale Points	Positive LOW	Do you usually dairy foods every day?	Almost Always (5) Sometimes (3) Seldom/Never (1)
8	Dietary Habits Screening	Nutrient	Red Meat Uptakes	DQ_NT.RMUtk	5-Scale Points	Positive HIGH	Do you usually eat red meat or high fat meat more than twice a week?	Almost Always (1) Sometimes (3) Seldom/Never (5)
9	Dietary Habits Screening	Nutrient	Sodium Uptakes	DQ_NT.NaUtk	5-Scale Points	Positive HIGH	Do you usually take salty side dishes everyday?	Almost Always (1) Sometimes (3) Seldom/Never (5)
10	Dietary Habits Screening	Nutrient	Added Sugar Uptakes	DQ_NT.SgUtk	5-Scale Points	Positive HIGH	Do you usually eat snacks or beverages with added sugars every day?	Almost Always (1) Sometimes (3) Seldom/Never (5)



No	Classification			Variable Name	Measure	Score Direction	Question Reference	Answer Choice and Measure Example (Ans to Ref. Question)
1	Dietary Habits Screening	Eating Pattern	Eating Regularity	DQ_EP.Regularity	5-Scale Points	Positive LOW	During the last month, how regularly do you eat? (Eat three times a day)	Almost Always (5) Sometimes (3) Seldom/Never (1)
2	Dietary Habits Screening	Eating Pattern	Food Group Balance	DQ_EP.FGBalance	5-Scale Points	Positive LOW	Do you enjoy a wide variety of nutritious foods from the five groups every day? (the five groups: fruits, vegetables, grains, protein foods, and dairy)	Almost Always (5) Sometimes (3) Seldom/Never (1)
3	Dietary Habits Screening	Nutrient	Whole Grain Uptakes	DQ_NT.WGUtk	5-Scale Points	Positive LOW	When you eat cooked-rice, do you usually eat cooked-rice with multi-grains?	Almost Always (5) Sometimes (3) Seldom/Never (1)
4	Dietary Habits Screening	Nutrient	Protein Uptakes	DQ_NT.PtUtk	5-Scale Points	Positive LOW	Do you usually take protein foods* in every servings? * Lean Meat (except red meat and high fat meat), Poultry, Fish and seafood, Eggs, Nuts and Seeds, Legumes/Beans	Almost Always (5) Sometimes (3) Seldom/Never (1)
5	Dietary Habits Screening	Nutrient	Vegetables Uptakes	DQ_NT.VgUtk	5-Scale Points	Positive LOW	Do you usually eat vegetables in every servings?	Almost Always (5) Sometimes (3) Seldom/Never (1)
6	Dietary Habits Screening	Nutrient	Fruits Uptakes	DQ_NT.FtUtk	5-Scale Points	Positive LOW	Do you usually eat fruits every day?	Almost Always (5) Sometimes (3) Seldom/Never (1)
7	Dietary Habits Screening	Nutrient	Dairy Uptakes	DQ_NT.DyUtk	5-Scale Points	Positive LOW	Do you usually dairy foods every day?	Almost Always (5) Sometimes (3) Seldom/Never (1)
8	Dietary Habits Screening	Nutrient	Red Meat Uptakes	DQ_NT.RMUtk	5-Scale Points	Positive HIGH	Do you usually eat red meat or high fat meat more than twice a week?	Almost Always (1) Sometimes (3) Seldom/Never (5)
9	Dietary Habits Screening	Nutrient	Sodium Uptakes	DQ_NT.NaUtk	5-Scale Points	Positive HIGH	Do you usually take salty side dishes everyday?	Almost Always (1) Sometimes (3) Seldom/Never (5)
10	Dietary Habits Screening	Nutrient	Added Sugar Uptakes	DQ_NT.SgUtk	5-Scale Points	Positive HIGH	Do you usually eat snacks or beverages with added sugars every day?	Almost Always (1) Sometimes (3) Seldom/Never (5)

QA-based Assessment Stage

<Questionnaire: Physical Activity Habits>

- Performing Exercise Regularly
- Exercise Frequency
- Exercise Duration per unit session
- Exercise Intensity per each session in average
- Performing pre, post exercise or daily stretching
- Experience performing regular exercise
- Existence of restrictions on performing exercise
- Reason of restrictions on performing exercise
- Objective of Exercise
- Willingness on performing exercise
- Daily Walking Time
- Lifetime Movement Pattern - Prefer Active Lifestyle
- Sedentary Lifestyle
- Prefer Walking

Physical Activity and Exercise Habits Questionnaire

Selected physical activity and exercise habit assessment tools
(ex, GCH's Questionnaire)

1. Do you currently perform regular, intended exercise?
2. Do you currently perform exercise more than 3 to 5 times per week regularly?
3. How long do you exercise at each session?
4. How's the intensity level of exercise normally at each session?
5. Do you perform stretching normally before/after exercise or in daily life regularly?
6. Do you have an experience performing regular exercise?
7. Are there any restrictions on starting exercise, even though you are willing to do?
8. If you have any restrictions on starting exercise, why?
9. What is your objective of exercise?
10. Are you willing to start regular exercise?
11. Do you usually walk more than 10 minutes on a daily basis?
12. Do you prefer exercise or working by moving your body?
13. Do you spend more time standing or moving your body rather than sitting?
14. Do you prefer walking short distance rather than ride?

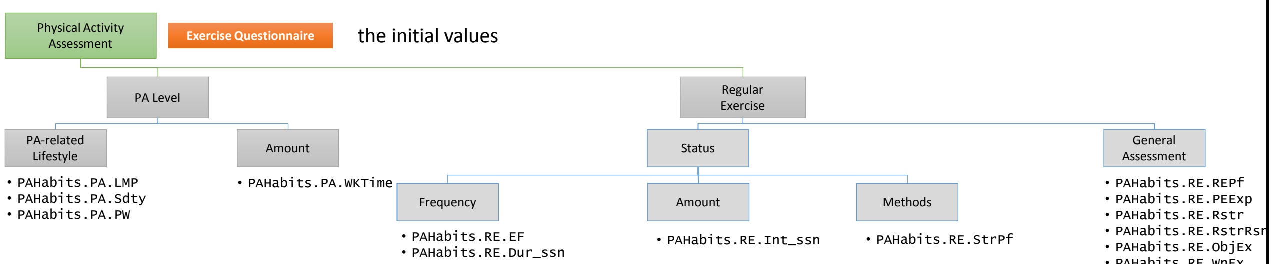
Generate Initial Values for User Profile

<PA & Exercise Habits Data (in User's Profile DB)>

Exercise Regularity: PAHabits.RE.REPf (0 or 1)
Exercise Frequency: PAHabits.RE.EF (0 or 1)
Exercise Duration: PAHabits.RE.Dur_ssn (5 scale pts)
Exercise Intensity: PAHabits.RE.Int_ssn (5 scale pts)
Stretching: PAHabits.RE.StrPf (0 or 1)
Experience on regular exercise: PAHabits.RE.REExp (0 or 1)
Restrictions on Exercise: PAHabits.RE.Rstr (0 or 1)
Source of Restrictions: PAHabits.RE.RstrRsn
Objective of Exercise: PAHabits.RE.ObjEx
Willingness exercise: PAHabits.RE.WnEx (0 or 1)
Daily Walking Time: PAHabits.PA.WkTime (0 or 1)
Prefer Active Lifestyle: PAHabits.PA.LMP (0 or 1)
Sedentary Lifestyle: PAHabits.PA.Sdty (0 or 1)
Prefer Walking: PAHabits.PA.PW (0 or 1)

<Lifestyle Questionnaire – PA & Exercise Habits> CALL: GCH-DietPic-QN-PA-v001 questionnaire modules

No	Classification			Variable Name	Measure	Score Direction	Question Reference	Answer Choice and Measure Example (Ans to Ref. Question)	Questionnaire routine/Comments
1	PA-Exercise Screening	Regular Exercise: General	Performing Exercise Regularly	DQ_RE.REPf	YES/NO	Positive HIGH	Do you currently perform regular, intended exercise?	1=Yes 0=No	If DQ_+K2:K11RE.REPf=0, skip #2~#5 questions (go to #6 question)
2	PA-Exercise Screening	Regular Exercise: Status	Exercise Frequency	DQ_RE.EF	YES/NO	Positive HIGH	Do you currently perform exercise more than 3 to 5 times per week regularly?	1=Yes 0=No	
3	PA-Exercise Screening	Regular Exercise: Status	Exercise Duration per unit session	DQ_RE.Dur_ssn	5-scale pts (conversion)	Positive HIGH	How long do you exericse at each session?	2=More than 1 hour 1=More than 30 minutes 0=Less than 20 minutes	
4	PA-Exercise Screening	Regular Exercise: Status	Exercise Intensity per each session in average	DQ_RE.Int_ssn	5-scale pts (conversion)	Positvie HIGH	How's the intensity level of exercise normaly at each session?	2=Vigorous 1=Moderate 0=Light	
5	PA-Exercise Screening	Regular Exercise: Status	Performing pre, post exercise or daily stretching	DQ_RE.StrPf	YES/NO	Positive HIGH	Do you perform stretching normaly before/after exercise or in daily life regularly?	1=Yes 0=No	End of Questionnaire for the user with DQ_RE.REPf=1
6	PA-Exercise Screening	Regular Exercise: General	Experience performing regular exercise	DQ_RE.REExp	YES/NO	Positive HIGH	Do you have an experience performing regular exercise?	1=Yes 0=No	Begin Questionnaire subroutine for the user with DQ_RE.REPf=0
7	PA-Exercise Screening	Regular Exercise: General	Existence of restrictions on performing exercise	DQ_RE.Rstr	YES/NO	Positive LOW	Are there any restrictions on starting exercise, even though you are willing to do?	1=Yes 0=No	If DQ_RE.Rstr=0, skip #7 question (go to #8 question)
8	PA-Exercise Screening	Regular Exercise: General	Reason of restrictions on performing exericse	DQ_RE.RstrRsn	N/A	NEUTRAL	If you have any restrictions on starting exercise, why?	3=Lack of time or space 2=Feel pains or sick 1=Do not want to exercise alone 0=Don't know how to	
9	PA-Exercise Screening	Regular Exercise: General	Objective of Exercise	DQ_RE.ObjEx	N/A	NEUTRAL	What is your objective of exercise?	2=for Diet 1=for Leisure 0=for Health	
10	PA-Exercise Screening	Regular Exercise: General	Willingness on performing exercise	DQ_RE.WnEx	YES/NO	Positive HIGH	Are you willing to start regular exercise?	1=Yes 0=No	End of Questionnaire for the user with DQ_RE.REPf=0
11	PA-Exercise Screening	PA Level: PA Related Lifestyle	Daily Walking Time	DQ_PA.WkTime	YES/NO	Positive HIGH	Do you usually walk more than 10 minutes on a daily basis?	1=Yes 0=No	
12	PA-Exercise Screening	PA Level: Amount	Lifetime Movement Pattern	DQ_PA.LMP	YES/NO	Positive HIGH	Do you prefer exercise or working by moving your body?	1=Yes 0=No	
.13	PA-Exercise Screening	PA Level: PA Related Lifestyle	Sedentary Lifestyle	DQ_PA.Sdty	YES/NO	Positive LOW	Do you spend more time standing or moving your body rather than sitting?	1=No 0=Yes	
14	PA-Exercise Screening	PA Level: PA Related Lifestyle	Prefer Walking	DQ_PA.PW	YES/NO	Positive HIGH	Do you prefer walking short distance rather than ride?	1=Yes 0=No	



No	Classification			Variable Name	Measure	Score Direction	Question Referenence	Answer Choice and Measure Example (Ans to Ref. Question)	Questionnaire routine/Comments
1	PA-Exercise Screening	Regular Exercise: General	Performing Exercise Regularly	DQ_RE.REPf	YES/NO	Positive HIGH	Do you currently perform regular, intended exercise?	1=Yes 0=No	If DQ_+K2:K11RE.REPf=0, skip #2~#5 questions (go to #6 question)
2	PA-Exercise Screening	Regular Exercise: Status	Exercise Frequency	DQ_RE.EF	YES/NO	Positive HIGH	Do you currently perform exercise more than 3 to 5 times per week regularly?	1=Yes 0=No	
3	PA-Exercise Screening	Regular Exercise: Status	Exercise Duration per unit session	DQ_RE.Dur_ssn	5-scale pts (conversion)	Positive HIGH	How long do you exericse at each session?	2=More than 1 hour 1=More than 30 minutes 0=Less than 20 minutes	
4	PA-Exercise Screening	Regular Exercise: Status	Exercise Intensity per each session in average	DQ_RE.Int_ssn	5-scale pts (conversion)	Positvie HIGH	How's the intensity level of exercise normaly at each session?	2=Vigorous 1=Moderate 0=Light	
5	PA-Exercise Screening	Regular Exercise: Status	Performing pre, post exercise or daily stretching	DQ_RE.StrPf	YES/NO	Positive HIGH	Do you perform stretching normaly before/after exercise or in daily life regularly?	1=Yes 0=No	End of Questionnaire for the user with DQ_RE.REPf=1
6	PA-Exercise Screening	Regular Exercise:	Experience performing regular exercise	DQ_RE.REExp	YES/NO	Positive HIGH	Do you have an experience performing regular exercise?	1=Yes 0=No	Begin Questionnaire subroutine for the user with
7	PA-Exercise Screening	Regular Exercise:	Existence of restrictions on	DQ_RE.Rstr	YES/NO	Positive LOW	Are there any restrictions on starting exercise, even though you are willing to do?	1=Yes 0=No	If DQ_RE.Rstr=0, skip #7 question (go to #8 question)
8	PA-Exercise Screening	Regular Exercise: General	Reason of restrictions on performing exericse	DQ_RE.RstrRsn	N/A	NEUTRAL	If you have any restrictions on starting exercise, why?	3=Lack of time or space 2=Feel pains or sick 1=Do not want to exercise alone	
9	PA-Exercise Screening	Regular Exercise: General	Objective of Exercise	DQ_RE.ObjEx	N/A	NEUTRAL	What is your objective of exercise?	2=for Diet 1=for Leisure 0=for Health	
10	PA-Exercise Screening	Regular Exercise:	Willingness on performing exercise	DQ_RE.WnEx	YES/NO	Positive HIGH	Are you willing to start regular exercise?	1=Yes 0=No	End of Questionnaire for the user with DQ_RE.REPf=0
11	PA-Exercise Screening	PA Level: PA Related	Daily Walking Time	DQ_PA.WkTime	YES/NO	Positive HIGH	Do you usually walk more than 10 minutes on a daily basis?	1=Yes 0=No	
12	PA-Exercise Screening	PA Level: Amount	Lifetime Movement Pattern	DQ_PA.LMP	YES/NO	Positive HIGH	Do you prefer exercise or working by moving your body?	1=Yes 0=No	
.13	PA-Exercise Screening	PA Level: PA Related	Sedentary Lifestyle	DQ_PA.Sdty	YES/NO	Positive LOW	Do you spend more time standing or moving your body rather than sitting?	1=No 0=Yes	
14	PA-Exercise Screening	PA Level: PA Related	Prefer Walking	DQ_PA.PW	YES/NO	Positive HIGH	Do you prefer walking short distance rather than ride?	1=Yes 0=No	

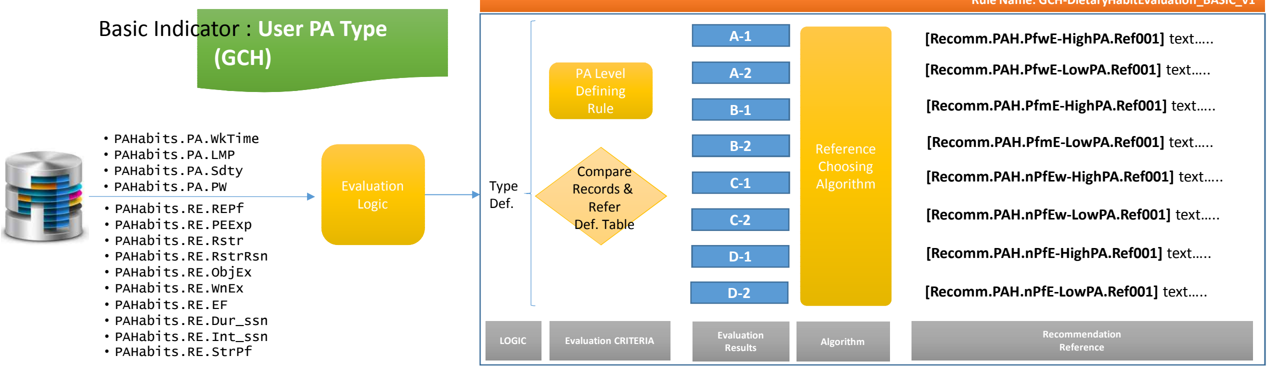
Plan Subscription Stage

<User's Dietary Habit Score>


```
<Rule Example>
Rule: GCH-DietaryHabitsEvaluation_BASIC_v1
included Logic: DietaryHabitScore{} //Sum DietaryHabits Points of 5-scale, with considering scoring direction Positive LOW
Result evaluation: if DietaryHabitScore = < 29, DietaryHabitStatus = BAD; 30 ~ 39, DietaryHabitStatus = Moderate, 40+, DietaryHabitStatus = Proper
Choosing Recommendation Reference:
  if DietaryHabitStatus = Moderate, call "Recom.DH.Moderate.Ref001"; if DietaryHabitStatus = Proper, call "Recomm.DH.Proper.Ref001"; if DietaryHabitStatus = Bad, call "Recomm.DH.Bad.Ref001"
 * Reference Recommendation can varies during service time with analyzing service result database or expert's choice/preferences or company/service aims or specific user's preferences/
   context or user group's general responses. So Reference Choosing Algorithm is needed in future work. This could be done by tagging attributes to the "Reference Recommendations"
```

Plan Subscription Stage

<User's PA Habit Groups>




```
<Rule Example>
Rule: GCH-PAHabitsEvaluation_BASIC_v1
included Logic: UserPAType{} // X(A:D)-Y(1:2)
                Determine Y value (1 or 2):  if PALevel=HIGH, Y=1; PALevel=LOW, Y=0, where PALevel is defined by a PALevel defining rule.
                            in case of initial value (based on GCH questionnaire), if PAHabits.PA.WkTime + PAHabits.PA.LMP + ~PAHabits.PA.Sdty +
PAHabits.PA.PW >= 3, PALevel=HIGH; else LOW
                            during the service time, analysis of step counts will replace PALevel value.
                Determine X value (A, B, C, D):  if ((DQ_RE.REPf==1) AND ((PAHabits.PE.EF && PAHabits.RE.StrPf) == 1) AND PAHabits.RE.Int_ssn !=0), X=A
                                                 if ((DQ_RE.REPf==1) AND ((PAHabits.RE.EF && PAHabits.RE.StrPf) != 1) OR
(PAHabits.RE.Dur_ssn*PAHabits.RE.Int_ssn==0)), X=B
                                                 if ((DQ_RE.REPf!=1) AND (DQ_RE.WnEx = 1), X=C
                                                 if ((DQ_RE.REPf!=1) AND (DQ_RE.WnEx = 0), X=D
```

Registration Stage – Register and Data Gathering

• **[Mining Minds ver. 3.5]**

<Diabetes Risk Assessment>

if MedicalStatus.Diabetes.Diagnosed == 0,
&& MedicalStatus.Data.BGTestResult = NULL
call DiabetesRiskAssessmentTool()

(results)
sum of risk points
5 or less: UID.Diabetes.Risk = Low
6 – 11: UID.Diabetes.Risk = Moderate
12 or more: UID.Diabetes.Risk = High

Diabetes Risk Assessment Tool

Diabetes Risk Assessment Tool
(ex, GCH's Questionnaire)

<Data from User Profile Database>
1. Age
2. Gender
3. Ethnicity (Default)
4. High Blood Pressure
5. Smoking
6. Eat Vegetables or Fruits
7. PA at least 2.5 hours per week

<Additional Questionnaire>
1. Family History
2. High Blood Glucose Level
3. Waist Measurement

Data for Diabetes Assessment

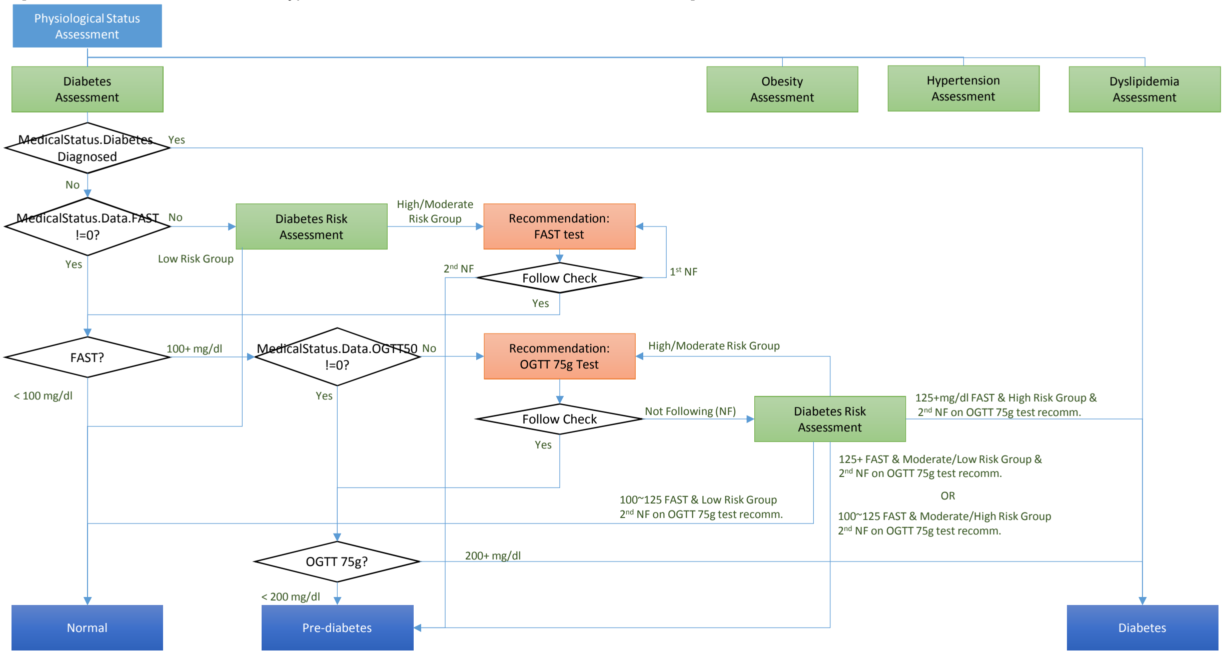
<Data in User's Profile DB)>

Age:	UID.Data_of_Birth
Gender:	UID.Sex
Ethnicity:	Asian(Default)
High BP:	MedicalStatus.Hypertension.Diagnosed or MedicalStatus.Data.DBP/SBP
Smoking:	T.B.D. (*BQ_LH.Smoking*)
Eat Veg or Fruits	DietaryHabits.NT.FtUtk/VgUtk
PA 2.5+ hr/week	*PAHabits.PA.PATimeperWk*
Family History	T.B.D. (*BQ_MI.FH*)
High BG Level	*MedicalStatus.Data.BGTestResult or BQ_MI.HBG*
Waist Measurement	*UID.WaistCircumference*

Registration Stage – User Assessment

• **[Mining Minds ver. 3.5]**

[Classification of Users' Status Type – reference for reference of recommendation]



Diabetes Assessment

Reference: The Australian Type 2 Diabetes

• **[Mining Minds ver. 3.5]**

1. Your Age Group
   - ☐ Under 35 years ;0 points
   - ☐ 35 – 44 years ;2 points
   - ☐ 45 – 54 years ;4 points
   - ☐ 55 – 64 years ;6 points
   - ☐ 65 years or over ;8 points
2. Your Gender
   - ☐ Female ;0 points
   - ☐ Male ;3 points
3. Your ethnicity
   - ☐ Asia, Middle East, North Africa, Southern Europe ;2 points
   - ☐ Other ;0 points
4. Have either of your parents, or any of your brothers or sisters been diagnosed with diabetes (type 1 or type 2)
   - ☐ No ;0 points
   - ☐ Yes ;3 points
5. Have you ever been found to have high blood glucose (sugar) (for example, in a health examination, during an illness, during pregnancy)?
   - ☐ No ;0 points
   - ☐ Yes ;6 points
6. Are you currently taking medication for high blood pressure?
   - ☐ No ;0 points
   - ☐ Yes ;2 points
7. Do you currently smoke cigarettes or any other tobacco products on a daily basis?
   - ☐ No ;0 points
   - ☐ Yes ;2 points
8. How often do you eat vegetables or fruit?
   - ☐ No ;0 points
   - ☐ Yes ;1 points
9. On average, would you say you do at least 2.5 hours of physical activity per week (for example, 30 minutes a day on 5 or more days a week)?
   - ☐ No ;0 points
   - ☐ Yes ;1 points
10. Your waist measurement taken below the ribs (usually at the level of the navel, and while standing)
    - ☐ Waist Measurement (cm) __________

Add up points: ______ points
- ☐ 5 or less: Low Risk
- ☐ 6 – 11: Moderate Risk
- ☐ 12 or more: High Risk

MEN
- ☐ Less than 90cm;0 points
- ☐ 90 – 100cm ;4 points
- ☐ More than 100cm ;7 points

WOMEN
- ☐ Less than 80cm;0 points
- ☐ 80 – 90cm ;4 points
- ☐ More than 90cm ;7 points

Registration Stage – Register and Data Gathering

• **[Mining Minds ver. 3.5]**

<Hypertension Risk Assessment>

if MedicalStatus.Hypertension.Diagnosed == 0,
&& MedicalStatus.Data.DBP/SBP = NULL
call HypertensionRiskAssessmentTool()

(results)
sum of risk points
~ ##: UID.Hypertension.Risk = Low
~ ##: UID.Hypertension.Risk = Moderate
~ ##: UID.Hypertension.Risk = High

Diabetes Risk Assessment Tool

Hypertension Risk Assessment Tool
(ex, GCH's Questionnaire)
<Data from User Profile Database>

<Additional Questionnaire>

Data for Diabetes Assessment

<Data in User's Profile DB)>

Registration Stage – Register and Data Gathering

• **[Mining Minds ver. 3.5]**

<Dyslipidemia Risk Assessment>

if MedicalStatus.Dyslipidemia.Diagnosed == 0,
&& MedicalStatus.Data.TC/HDLC/LDLC/TC = NULL
call DyslipidemiaRiskAssessmentTool()

(results)
sum of risk points
~ ##: UID.Dyslipidemia.Risk = Low
~ ##: UID.Dyslipidemia.Risk = Moderate
~ ##: UID.Dyslipidemia.Risk = High

Diabetes Risk Assessment Tool

Dyslipidemia Risk Assessment Tool
(ex, GCH's Questionnaire)
<Data from User Profile Database>

<Additional Questionnaire>

Data for Diabetes Assessment

<Data in User's Profile DB)>

Registration Stage – Decision of User Type

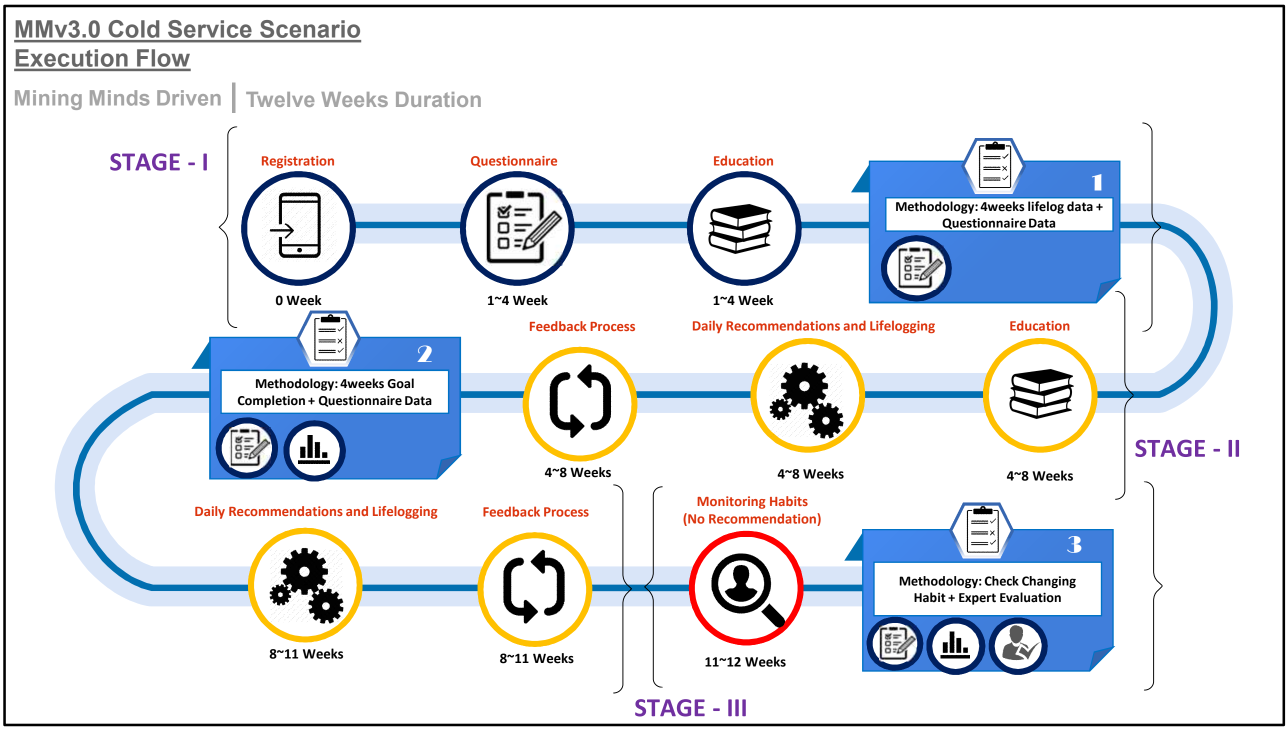
<Classification of User's Status Type>

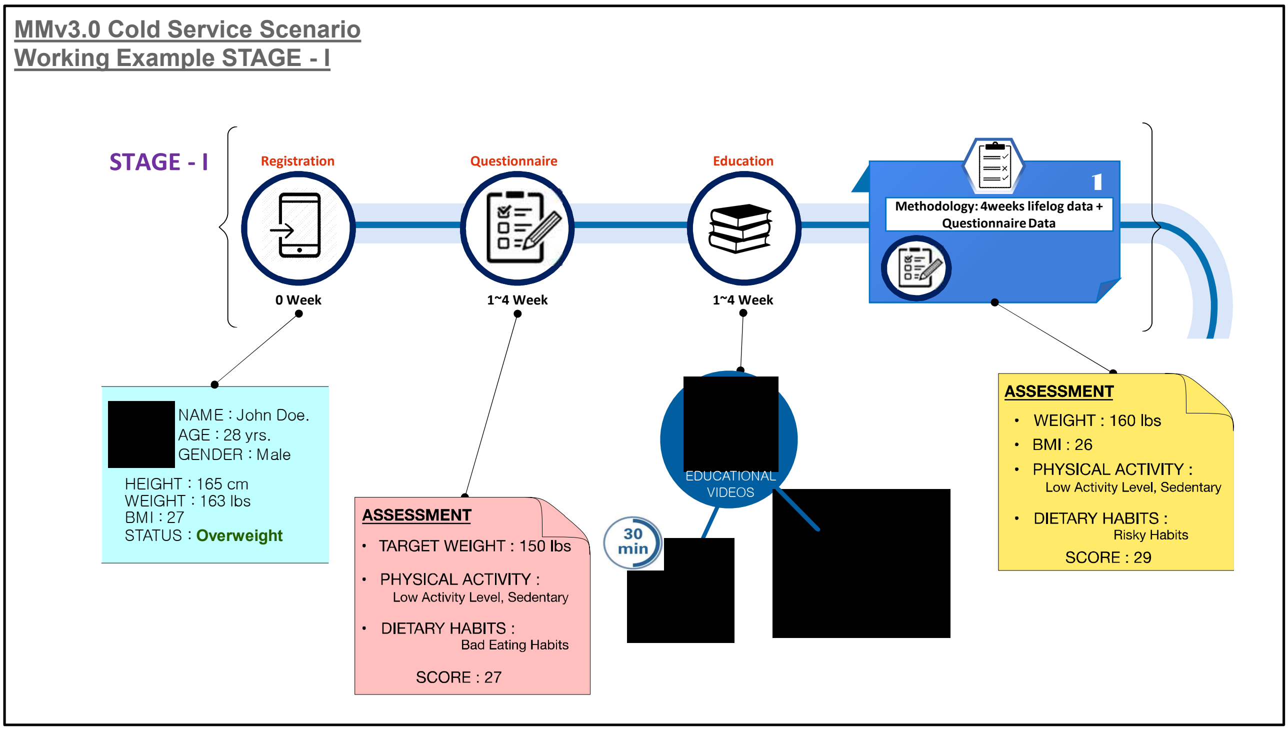
• **[Mining Minds ver. 3.5]**

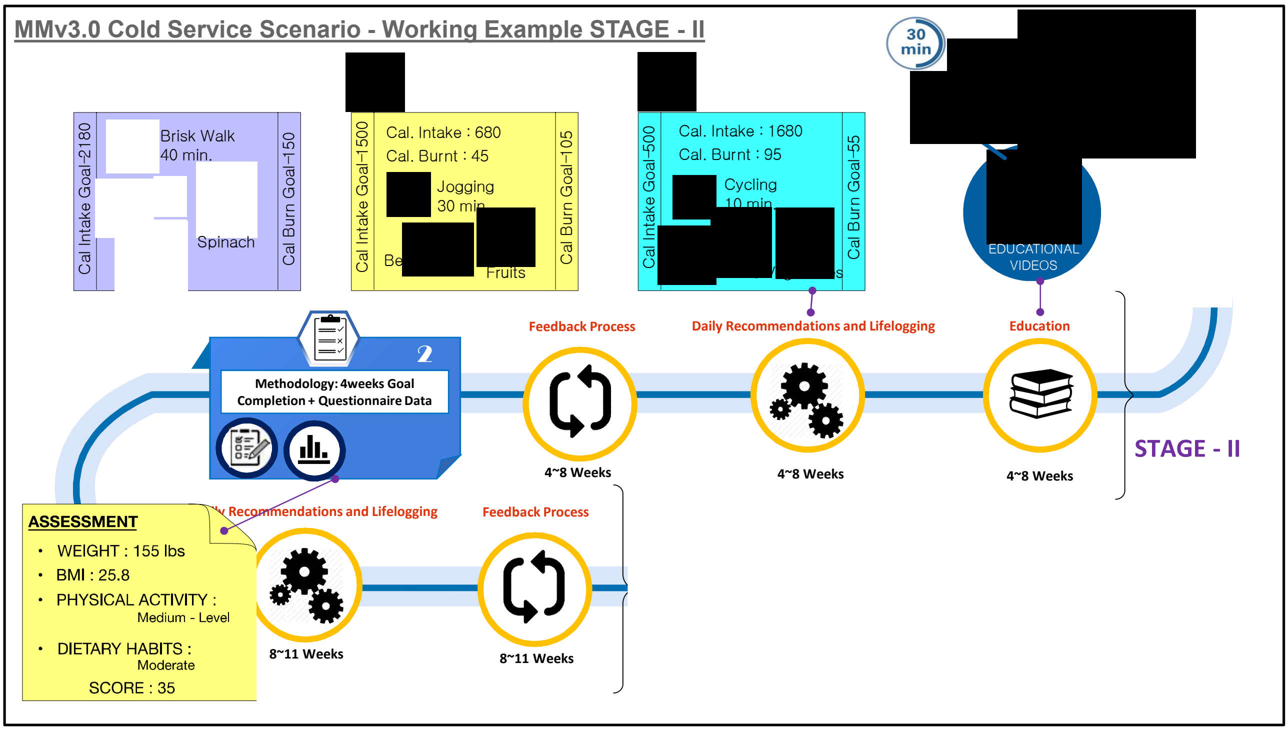
Diabetes	Obesity	Hypertension	Dyslipidemia	Group Code	Recommendation Guidelines(General/Edu.)	Recommendation Guidelines(Nutrition)	Recommendation Guidelines(PA)
Normal	Low Weight	Normal	Group 1 ~ 5	NBG.LW.NBP.DL0 ~ NBG.LW.NBP.DL4	Carbohydrate uptake control Muscular exercise recomm.	Proper Calories, Regular Eating, 3-times Eating, Protein uptakes, Snack Control, Dairy Intakes	Aerobic/Muscular exercise ratio control
		Pre-Hypertension	Group 1 ~ 5	NBG.LW.MHBP.DL0 ~ NBG.LW.MHBP.DL4			
		Hypertension	Group 1 ~ 5	NBG.LW.HBP.DL0 ~ NBG.LW.HBP.DL4			
	Normal Weight	Normal	Group 1 ~ 5	NBG.NW.NBP			
		Pre-Hypertension	Group 1 ~ 5	NBG.NW.MHBP			
		Hypertension	Group 1 ~ 5	NBG.NW.HBP			
	Over weight/ Obesity	Normal	Group 1 ~ 5	NBG.OW.NBP			
		Pre-Hypertension	Group 1 ~ 5	NBG.OW.MHBP			
		Hypertension	Group 1 ~ 5	NBG.OW.HBP			
Pre-Diabetes	Low Weight	Normal	Group 1 ~ 5	MHBG.LW.NBP			
		Pre-Hypertension	Group 1 ~ 5	MHBG.LW.MHBP			
		Hypertension	Group 1 ~ 5	MHBG.LW.HBP			
	Normal Weight	Normal	Group 1 ~ 5	MHBG.NW.NBP			
		Pre-Hypertension	Group 1 ~ 5	MHBG.NW.MHBP			
		Hypertension	Group 1 ~ 5	MHBG.NW.HBP			
	Over weight/ Obesity	Normal	Group 1 ~ 5	MHBG.OW.NBP			
		Pre-Hypertension	Group 1 ~ 5	MHBG.OW.MHBP			
		Hypertension	Group 1 ~ 5	MHBG.OW.HBP			
Diabetes	Low Weight	Normal	Group 1 ~ 5	HBG.LW.NBP			
		Pre-Hypertension	Group 1 ~ 5	HBG.LW.MHBP			
		Hypertension	Group 1 ~ 5	HBG.LW.HBP			
	Normal Weight	Normal	Group 1 ~ 5	HBG.NW.NBP			
		Pre-Hypertension	Group 1 ~ 5	HBG.NW.MHBP			
		Hypertension	Group 1 ~ 5	HBG.NW.HBP			
	Over weight/ Obesity	Normal	Group 1 ~ 5	HBG.OW.NBP			
		Pre-Hypertension	Group 1 ~ 5	HBG.OW.MHBP			
		Hypertension	Group 1 ~ 5	HBG.OW.HBP			

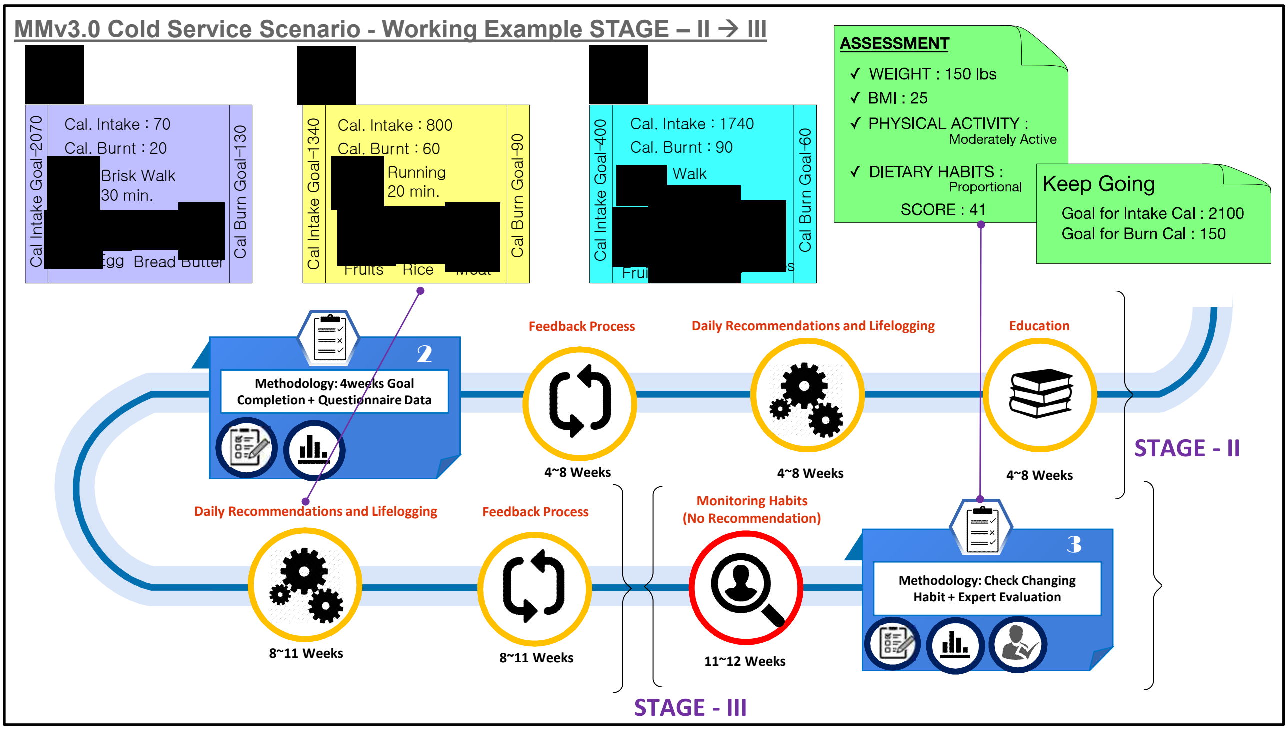
Mining Minds Version 3.0

Cold Service Scenario


MMv3.0 Cold Service Scenario
Execution Flow
Mining Minds Driven | Twelve Weeks Duration
STAGE - I
Registration
0 Week
Questionnaire
1~4 Week
Education
1~4 Week
1
Methodology: 4weeks lifelog data + Questionnaire Data
STAGE - II
Feedback Process
4~8 Weeks
Daily Recommendations and Lifelogging
4~8 Weeks
Education
4~8 Weeks
2
Methodology: 4weeks Goal Completion + Questionnaire Data
Daily Recommendations and Lifelogging
8~11 Weeks
Feedback Process
8~11 Weeks
Monitoring Habits
(No Recommendation)
11~12 Weeks
3
Methodology: Check Changing Habit + Expert Evaluation
STAGE - III


MMv3.0 Cold Service Scenario
Working Example STAGE - I
STAGE - I
Registration
0 Week
Questionnaire
1~4 Week
Education
1~4 Week
1
Methodology: 4weeks lifelog data + Questionnaire Data
NAME : John Doe.
AGE : 28 yrs.
GENDER : Male
HEIGHT : 165 cm
WEIGHT : 163 lbs
BMI : 27
STATUS : Overweight
ASSESSMENT
• TARGET WEIGHT : 150 lbs
• PHYSICAL ACTIVITY :
Low Activity Level, Sedentary
• DIETARY HABITS :
Bad Eating Habits
SCORE : 27
EDUCATIONAL VIDEOS
30 min
ASSESSMENT
• WEIGHT : 160 lbs
• BMI : 26
• PHYSICAL ACTIVITY :
Low Activity Level, Sedentary
• DIETARY HABITS :
Risky Habits
SCORE : 29


MMv3.0 Cold Service Scenario - Working Example STAGE - II
30 min
Cal Intake Goal-2180
Brisk Walk
40 min.
Spinach
Cal Burn Goal-150
Cal Intake Goal-1500
Cal. Intake : 680
Cal. Burnt : 45
[redacted]
Jogging
30 min
Be
[redacted]
Fruits
Cal Burn Goal-105
Cal Intake Goal-500
Cal. Intake : 1680
Cal. Burnt : 95
[redacted]
Cycling
10 min
[redacted]
Cal Burn Goal-55
[redacted]
EDUCATIONAL VIDEOS
Feedback Process
Daily Recommendations and Lifelogging
Education
2
Methodology: 4weeks Goal Completion + Questionnaire Data
4~8 Weeks
4~8 Weeks
4~8 Weeks
STAGE - II
ASSESSMENT
• WEIGHT : 155 lbs
• BMI : 25.8
• PHYSICAL ACTIVITY : Medium - Level
• DIETARY HABITS : Moderate
SCORE : 35
Recommendations and Lifelogging
Feedback Process
8~11 Weeks
8~11 Weeks


MMv3.0 Cold Service Scenario - Working Example STAGE – II → III
[redacted]
Cal Intake Goal-2070
Cal. Intake : 70
Cal. Burnt : 20
Brisk Walk
30 min.
Egg Bread Butter
Cal Burn Goal-130
[redacted]
Cal Intake Goal-1340
Cal. Intake : 800
Cal. Burnt : 60
Running
20 min.
Fruits Rice
Cal Burn Goal-90
[redacted]
Cal Intake Goal-400
Cal. Intake : 1740
Cal. Burnt : 90
Walk
[redacted]
Cal Burn Goal-60
ASSESSMENT
✓ WEIGHT : 150 lbs
✓ BMI : 25
✓ PHYSICAL ACTIVITY : Moderately Active
✓ DIETARY HABITS : Proportional
SCORE : 41
Keep Going
Goal for Intake Cal : 2100
Goal for Burn Cal : 150
2
Methodology: 4weeks Goal Completion + Questionnaire Data
Feedback Process
Daily Recommendations and Lifelogging
Education
4~8 Weeks
4~8 Weeks
4~8 Weeks
STAGE - II
Daily Recommendations and Lifelogging
Feedback Process
Monitoring Habits (No Recommendation)
3
Methodology: Check Changing Habit + Expert Evaluation
8~11 Weeks
8~11 Weeks
11~12 Weeks
STAGE - III

MMv3.0 Cold Service Scenario STAGE - I

<Registration>

- User ID (account code)
- Nickname
- Password
- Linked SNS (e.g. the FACEBOOK ID, S-HEALTH ID)

<Contact>

- Contact Number(Mobile)
- Address
- e-mail

<Client Information>

- Client Name
- Date of Birth
- Gender

<Somatology>

- Height
- Weight
- Somatotype of Body

<Medical History on Lifestyle Diseases>

- Diabetes (if checked, **additional input form presented**)
- Hypertension (additional form) **[Mining Minds ver. 3.5]**
- Dyslipidemia (additional form) **[Mining Minds ver. 3.5]**

User Registration Form

General user registration/authorization modules

User's Profile Check-up

CONTACT NUMBER UID.Phone.Mobile
ADDRESS: UID.Address.Home
E-MAIL: UID.Email

NAME: UID.Name
Date of Birth: UID.Date_of_Birth
Gender: UID.Sex

Height: UID.Height
Weight: UID.Weight
Somatotype: UID.Somatotype

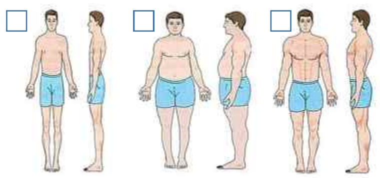
Diagnosed Diseased &MedicalStatus.Diabetes.Diagnosed
&MedicalStatus.Hypertension.Diagnosed
&MedicalStauts.Dyslipidemia.Diagnosed

Medical Check-up Data MedicalStatus.Data.FAST
MedicalStatus.Data.OGTT50
MedicalStatus.Data.HbA1c
MedicalStatus.Data.DBP
MedicalStatus.Data.SBP
MedicalStatus.Data.TC
MedicalStatus.Data.HDLC
MedicalStatus.Data.LDLC
MedicalStatus.Data.TG

MMv3.0 Cold Service Scenario STAGE - I

<Somatotype Assessment>

Question: What is your body type? Choose most similar type in the pictures.
Choices:



Ectomorph Endomorph Mesomorph

Data:

UID	Field Name	Date & Time of Assessment	Toolset	Data	Unit	Certified by
####	UID.Height	yyyyMMddHHmmss	{UserInput}	Integer	cm	USER
####	UID.Weight	yyyyMMddHHmmss	{UserInput, DeviceID}	###.#	kg	USER
####	UID.Somatotype	yyyyMMddHHmmss	{UserChoice, ST_msr01, ST_msr02, …}	{Ectomorph, Endomorph, Mesomorph; NULL}	N/A	USER
####	UID.BMI	yyyyMMddHHmmss	{Rule.Index.BMI}	##.#	kg/m^2	SYSTEM

Rule.Index.BMI
input: UID.Height.Current_date, UID.Weight.Current_date
output: UID.BMI = UID.Weight.Current_date/(UID.Height.Current_date*UID.Height.Current_date)

MMv3.0 Cold Service Scenario STAGE - I

<Medical History Assessment>

- **[Mining Minds ver. 3.5]**

Question: Have you ever been diagnosed with any of following diseases or conditions?

- ☐ Diabetes or prediabetes
- ☐ Hypertension or Prehypertension
- ☐ Dyslipidemia

If the boxes are checked, present additional questions.

✓ Diabetes or prediabetes
- Input your recent FAST test result : (   ) mg/dL ☐ Don't Know
- OGTT 50g test result : (   ) mg/dL ☐ Don't Know
- HbA1c : (   ) % or (   ) mmol/mol ☐ Don't Know

✓ Hypertension or Prehypertension
- Recent measure of Blood Pressure : Systolic (   )mmHg, Diastolic (   )mmHg

✓ Dyslipidemia
- Recent measure of Total Cholesterol : (   ) mg/dL
- Recent measure of HDL Cholesterol : (   ) mg/dL
- Recent measure of LDL Cholesterol : (   ) mg/dL
- Recent measure of Triglyceride : (   ) mg/dL

Data:

UID	Field Name	Date & Time of Assessment	Toolset	Data	Unit	Certified by
####	MedicalStatus.Diabetes.Diagnosed	yyyyMMddHHmmss	{UserChoice, ExpertInput}	{Yes; NULL}	N/A	USER
####	MedicalStatus.Hypertension.Diagnosed	yyyyMMddHHmmss	{UserChoice, ExpertInput}	{Yes; NULL}	N/A	USER
####	MedicalStauts.Dyslipidemia.Diagnosed	yyyyMMddHHmmss	{UserChoice, ExpertInput}	{Yes; NULL}	N/A	USER
####	MedicalStatus.Data.FAST	yyyyMMddHHmmss	{UserInput, DeviceID}	Integer	mg/dL	USER
####	MedicalStatus.Data.OGTT50	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER
####	MedicalStatus.Data.HbA1c	yyyyMMddHHmmss	{UserInput}	##.# ##.# Integer	% mmol/mol mg/dL	USER
####	MedicalStatus.Data.DBP	yyyyMMddHHmmss	{UserInput, DeviceID}	Integer	mmHg	USER
####	MedicalStatus.Data.SBP	yyyyMMddHHmmss	{UserInput, DeviceID}	Integer	mmHg	USER
####	MedicalStatus.Data.TC	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER
####	MedicalStatus.Data.HDLC	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER
####	MedicalStatus.Data.LDLC	yyyyMMddHHmmss	{UserInput}	###.#	mg/dL	USER
####	MedicalStatus.Data.TG	yyyyMMddHHmmss	{UserInput}	Integer	mg/dL	USER

MMv3.0 Cold Service Scenario STAGE - I

QA-based Assessment

<Questionnaire: Dietary Habits>

- Eating Regularity
- Food Group Balance
- Whole Grain Uptakes
- Protein Uptakes
- Vegetable Uptakes
- Fruits Uptakes
- Dairy Uptakes
- Red Meat Uptakes
- Sodium Uptakes
- Added Sugar Uptakes

Dietary Habits Questionnaire

Selected dietary habit assessment tools
(ex, GCH's Questionnaire)

1. During the last month, how regularly do you eat?
2. Do you enjoy a wide variety of nutritious foods from the five groups every day?
3. When you eat cooked-rice, do you usually eat cooked-rice with multi-grains?
4. Do you usually take protein foods* in every servings?
5. Do you usually eat vegetables in every servings?
6. Do you usually eat fruits every day?
7. Do you usually dairy foods every day?
8. Do you usually eat red meat or high fat meat more than twice a week?
9. Do you usually take salty side dishes everyday?
10. Do you usually eat snacks or beverages with added sugars every day?

Generate Initial Values for User Profile

<Dietary Habits Data (in User's Profile DB)>
Eating Regularity: DietaryHabits.EP.Regularity (1pt ~ 5pt)
Food Group Balance: DietaryHabits.EP.FGBalance (1pt ~ 5pt)
Whole Grain Uptakes: DietaryHabits.NT.WGUtk (1pt ~ 5pt)
Protein Uptakes: DietaryHabits.NT.PtUtk (1pt ~ 5pt)
Vegetable Uptakes: DietaryHabits.NT.VgUtk (1pt ~ 5pt)
Fruits Uptakes: DietaryHabits.NT.FtUtk (1pt ~ 5pt)
Dairy Uptakes: DietaryHabits.NT.DyUtk (1pt ~ 5pt)
Red Meat Uptakes: DietaryHabits.NT.RMUtk (1pt ~ 5pt)
Sodium Uptakes: DietaryHabits.NT.NaUtk (1pt ~ 5pt)
Added Sugar Uptakes: DietaryHabits.NT.SgUtk (1pt ~ 5pt)

MMv3.0 Cold Service Scenario STAGE - I

<Lifestyle Questionnaire – Dietary Habits> CALL: GCH-DietPic-QN-DH-v001 questionnaire modules

Q1. Eat Regularity (DietaryHabits.EP.Regularity)

During the last month, how regularly (eat three times a day) do you eat?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q2. Food Group Balance (DietaryHabits.EP.FGBalance)

Do you enjoy a wide variety of nutritious foods from the five groups every day? (the five groups: fruits, vegetables, grains, protein foods, and dairy)

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q3. Whole Grain Uptakes (DietaryHabits.NT.WGUtk)

When you eat cooked-rice, do you usually eat cooked-rice with multi-grains?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q4. Protein Uptakes (DietaryHabits.NT.PtUtk)

Do you usually take protein foods* in every servings? * Lean Meat (except red meat and high fat meat), Poultry, Fish and seafood, Eggs, Nuts and Seeds, Legumes/Beans

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q5. Vegetables Uptakes (DietaryHabits.NT.VgUtk)

Do you usually eat vegetables in every servings?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q6. Fruit Uptakes (DietaryHabits.NT.FtUtk)

Do you usually eat fruits everyday?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q7. Dairy Uptakes (DietaryHabits.NT.DyUtk)

Do you usually dairy foods everyday?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q8. Red Meat Uptakes (DietaryHabits.NT.RMUtk)

Do you usually eat red meat or high fat meat more than twice a week?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q9. Sodium Uptakes (DietaryHabits.NT.NaUtk)

Do you usually take salty side dishes everyday?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

Q10. Added Sugar Uptakes (DietaryHabits.NT.SgUtk)

Do you usually eat snacks or beverages with added sugars every day?

☐ Almost Always ☐ Sometimes ☐ Seldom/Never

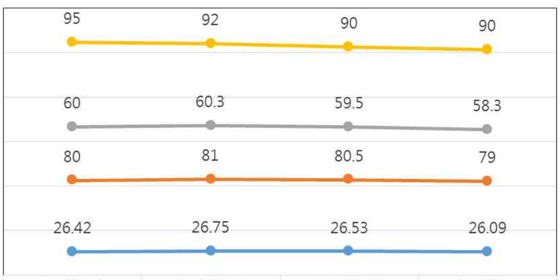
Monthly Report (1/5)

34

Name: John, Gender: Man, Ages: 30

Overall Evaluation - Red (Physical Activities: Green, Nutrition Intakes: Red, Blood Sugar: Red)

1st March 2017
Height : 174cm **Weight: 80kg** **Fat: High (60%)** **Blood Sugar Level: 95mg**
31st March 2017
Height: 174cm **Weight: 79kg** **Fat: High (58.3%)** **Blood Sugar Level: 90mg**

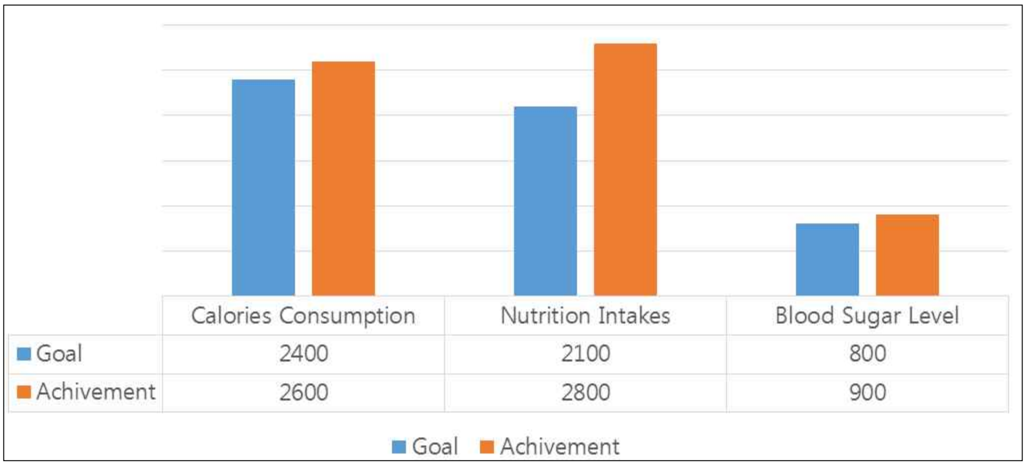


	1st Week	2nd Week	3rd Week	4th Week
BloodSugar	95	92	90	90
Fat	60	60.3	59.5	58.3
Weight	80	81	80.5	79
BMI	26.42	26.75	26.53	26.09

Goal Achievement Rate (Daily Average)

Monthly Report (2/5)

35

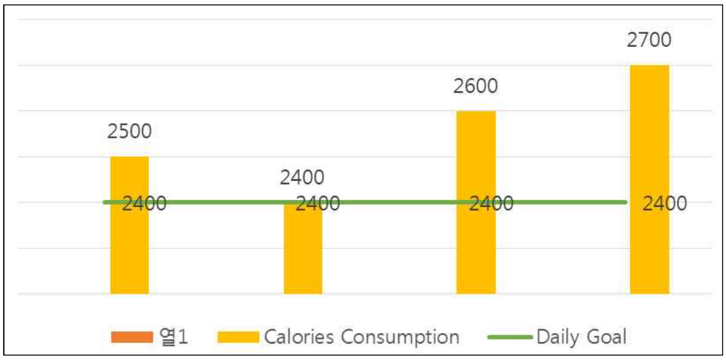


	Calories Consumption	Nutrition Intakes	Blood Sugar Level
Goal	2400	2100	800
Achivement	2600	2800	900

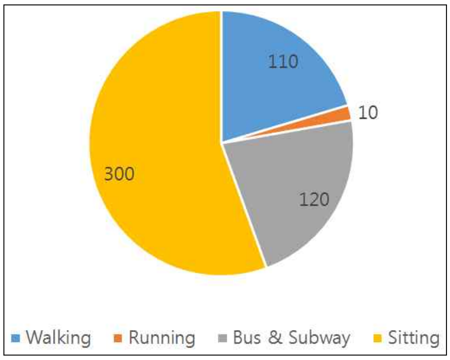
Monthly Report (3/5)

36

Physical Activities Evaluation - Green
Goal: 2400kcal, Achievement: 2600kcal (High)



Daily Average Activities



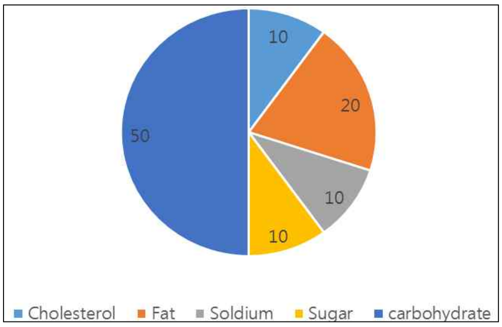
Nutrition Intakes – Red

Monthly Report (4/5)

37

Goal: 2400kcal, Achievement: 2800kcal (Bed)

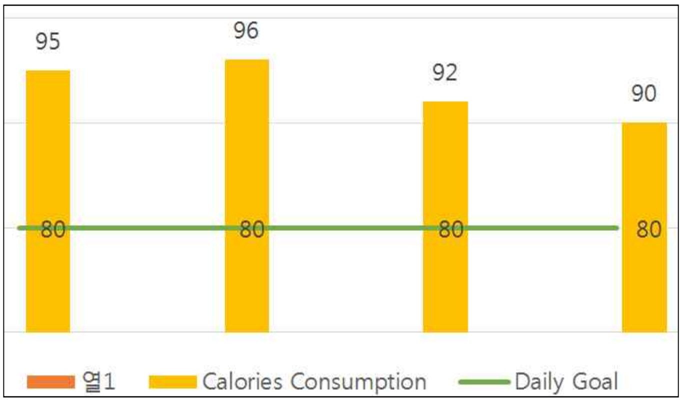
Daily Average Nutrition Rate of Food Intakes



Monthly Report (5/5)

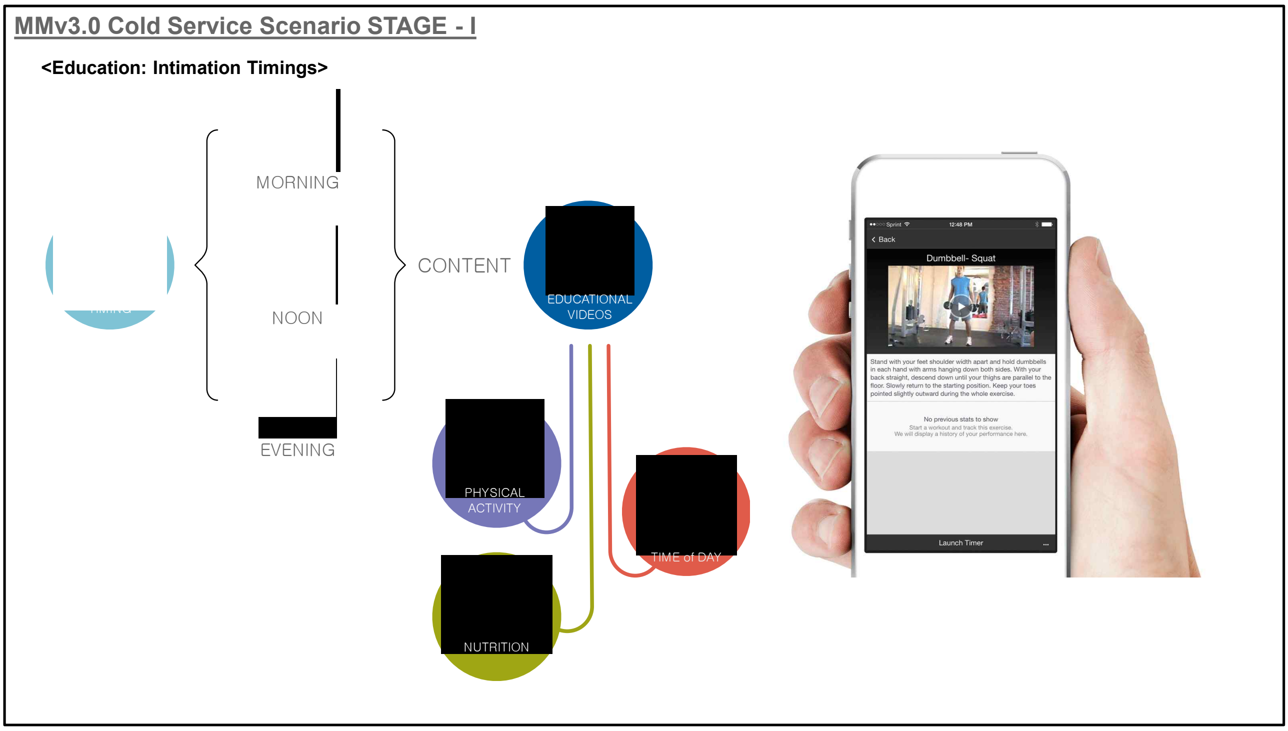
38

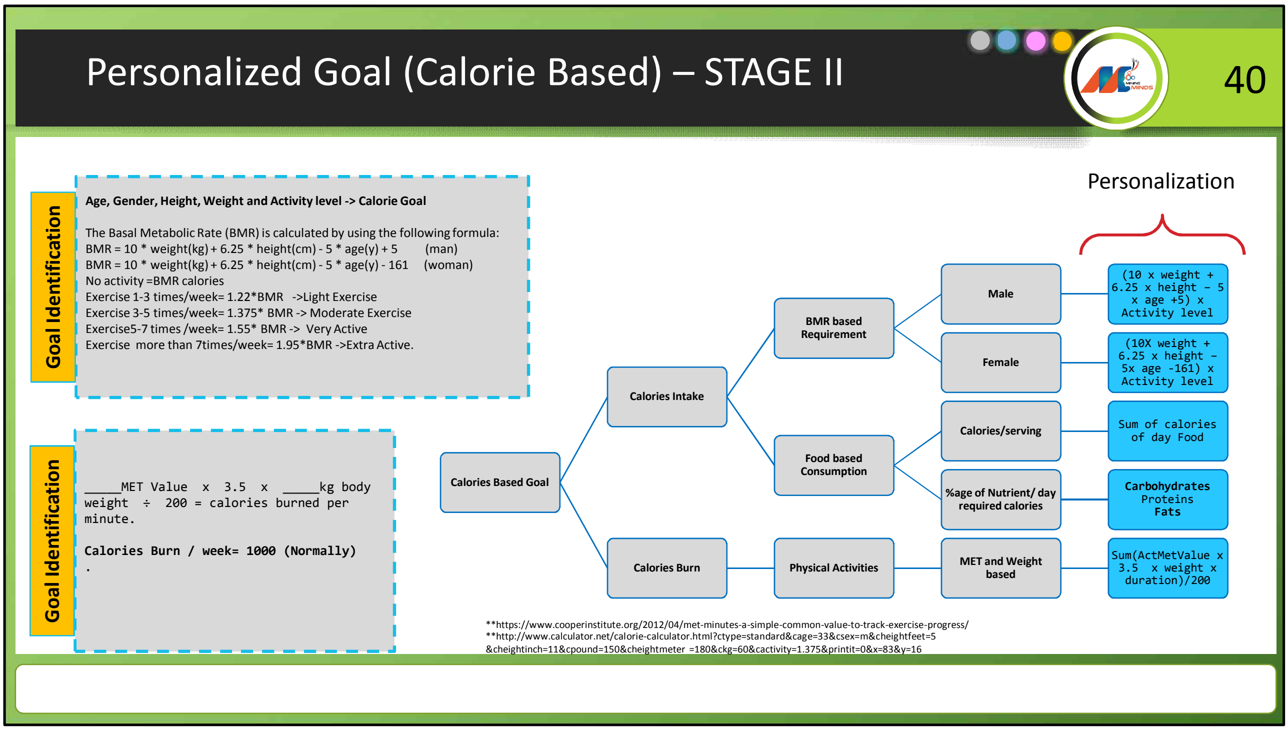
Blood Sugar Level – Red



Expert Guideline

1. Please avoid to take direct sugar
2. Please avoid ice cream and beverage
3. Please control the amount of rice and processed bread
4. Please monitor your blood glucose daily
5. Please avoid sedentary behavior


MMv3.0 Cold Service Scenario STAGE - I
<Education: Intimation Timings>
MORNING
NOON
EVENING
CONTENT
EDUCATIONAL VIDEOS
PHYSICAL ACTIVITY
TIME of DAY
NUTRITION
Dumbbell- Squat
Launch Timer


Personalized Goal (Calorie Based) – STAGE II
40
Goal Identification
Age, Gender, Height, Weight and Activity level -> Calorie Goal
The Basal Metabolic Rate (BMR) is calculated by using the following formula:
BMR = 10 * weight(kg) + 6.25 * height(cm) - 5 * age(y) + 5 (man)
BMR = 10 * weight(kg) + 6.25 * height(cm) - 5 * age(y) - 161 (woman)
No activity =BMR calories
Exercise 1-3 times/week= 1.22*BMR ->Light Exercise
Exercise 3-5 times/week= 1.375* BMR -> Moderate Exercise
Exercise5-7 times /week= 1.55* BMR -> Very Active
Exercise more than 7times/week= 1.95*BMR ->Extra Active.
Goal Identification
____MET Value x 3.5 x ____kg body weight ÷ 200 = calories burned per minute.
Calories Burn / week= 1000 (Normally)
.
Personalization
Calories Based Goal
Calories Intake
Calories Burn
BMR based Requirement
Food based Consumption
Physical Activities
Male
Female
Calories/serving
%age of Nutrient/ day required calories
MET and Weight based
(10 x weight + 6.25 x height − 5 x age +5) x Activity level
(10X weight + 6.25 x height − 5x age -161) x Activity level
Sum of calories of day Food
Carbohydrates Proteins Fats
Sum(ActMetValue x 3.5 x weight x duration)/200
**https://www.cooperinstitute.org/2012/04/met-minutes-a-simple-common-value-to-track-exercise-progress/
**http://www.calculator.net/calorie-calculator.html?ctype=standard&cage=33&csex=m&cheightfeet=5 &cheightinch=11&cpound=150&cheightmeter =180&ckg=60&cactivity=1.375&printit=0&x=83&y=16

Appendix- Nutrition Calories – STAGE II

41

Food Nutrition Data For Mining Minds V3.0

ID	Food Category	FoodItem	Calorie	Fat serv	Protein Serv	Carbohydrate Serv
	GC HealthCARE		Manipulated			
1	Grain	Rice	97.5	0.7275	2.055	19.185
2	Grain	Noodle	82.86	10.554	6.102	36.156
3	Grain	Oats	155.64	2.76	6.756	26.508
4	Grain	Barley	354	0.44	2.26	28.22
5	Grain	Quinoa	220.8	1.152	2.64	12.78
6	Grain	Brown	49.95	1.251	3.2535	34.416
7	Grain	Rye	201	0.978	6.204	45.516
8	Grain	Bread	202.5	2.43	8.04	36.51
9	Meat	Beef	342	33.75	58.5	0
10	Meat	Pork	540	31.5	60.75	0
11	Meat	Chicken	247.5	17.37	66.4875	0
12	SeaFood	Flounder	148.33	3.8631	24.8412	0
13	SeaFood	PacificSaury	79.2	15.795	16.515	0
14	SeaFood	Daegu	189	4.47	30.54	6.36
15	SeaFood	Shrimp	#VALUE!	#VALUE!	#VALUE!	#VALUE!
16	SeaFood	Salmon	320.25	22.75	35	0
17	SeaFood	Early	0	0	0	0
18	SeaFood	Tuna	129.6	7.2	36	0
19	SeaFood	Mackerel	358.75	43.75	33.25	0
20	SeaFood	Head	0	0	0	0
21	SeaFood	FishEggs	0	0	0	0
22	SeaFood	SeaWeed	4.5	0.06	0.17	1
23	Eggs	Egg	171.6	13.2	15.6	1.32
24	Eggs	QuailEggs	190.2	13.2	15.6	0.48
25	MilkAndDairyProducts	Milk	155	20	8	12
26	MilkAndDairyProducts	Yogurt	91.5	0.6	15	5.4
27	MilkAndDairyProducts	LowFatCheese	95.04	11.88	42.24	3.96
28	MilkAndDairyProducts	LowFatMilk	127.5	2.5	8.5	12.5
29	MilkAndDairyProducts	IceCream	231.84	12.32	3.92	26.88

Appendix- Physical Activities – STAGE II

42

Activity Based Calories consumption MMV3.0

Activity	Value	Unit	Weight kg	Duration min	Calorie burn	Weight accepted	Total calorie	By formula
Sitting	very less acive		10	1	0.42	68	2.86	2.856
Jumping	rope		10	1	1	68		6.8
Running	10	mph	10	1	2.66	68	18.09	18.088
Cycling	12-Nov	mph	10	1	1	68	6.8	6.8
ClimbingStairs	walking		10	1	1.33	68	9.04	9.044
Sweeping			10	1	0.66	68	4.49	4.488
Stretching			10	1	0.66	68	4.49	4.488
Dancing			10	1	0.75	68	5.1	5.1
Walking	2	mph	10	1	0.42	68	2.856	2.856
RidingEscalator			10	1		68		0
RidingElevator			10	1		68		0
Standing	less activity		10	1	1	68	6.8	6.8
Eating				15		68	9	#DIV/0!
DescendingStairs				15		68	34	#DIV/0!
LyingDown			10	1	0	68	0	0
Hiking	cross country		10	1	1	68	6.8	6.8
Jogging	general		10	1	1.17	68	7.96	7.956

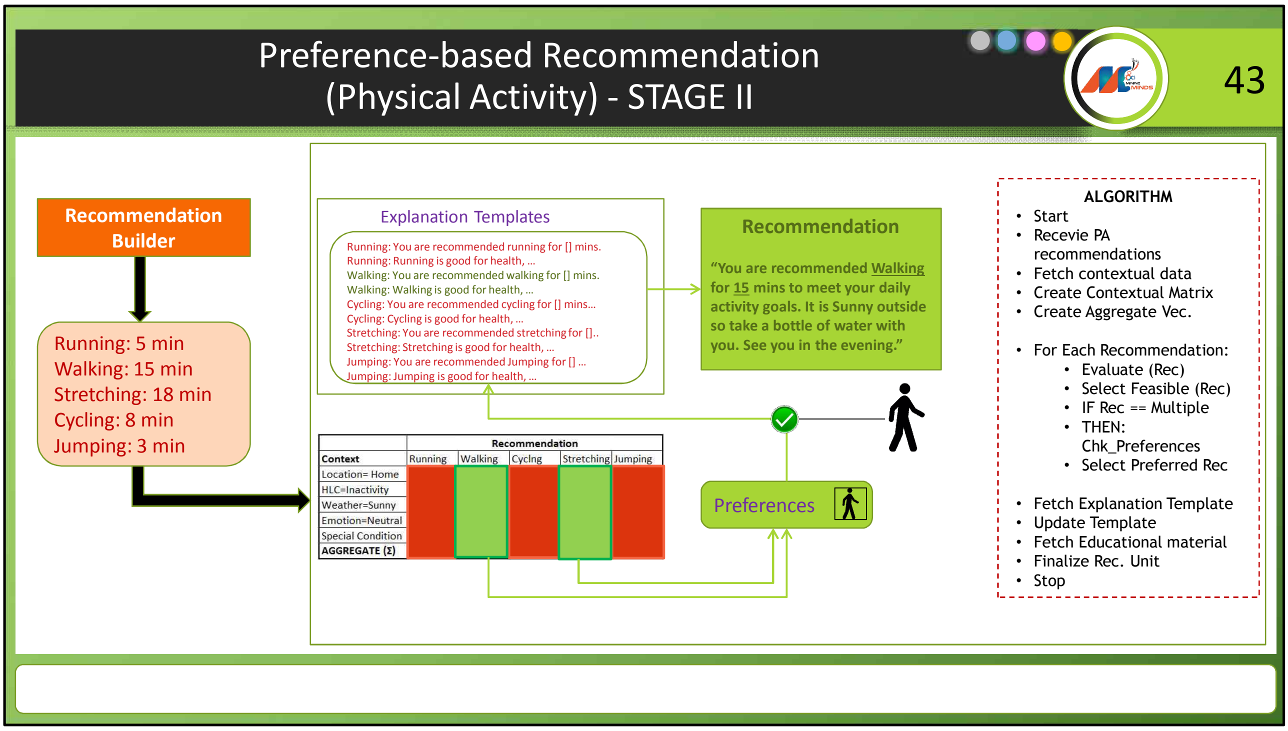
Goal Identification

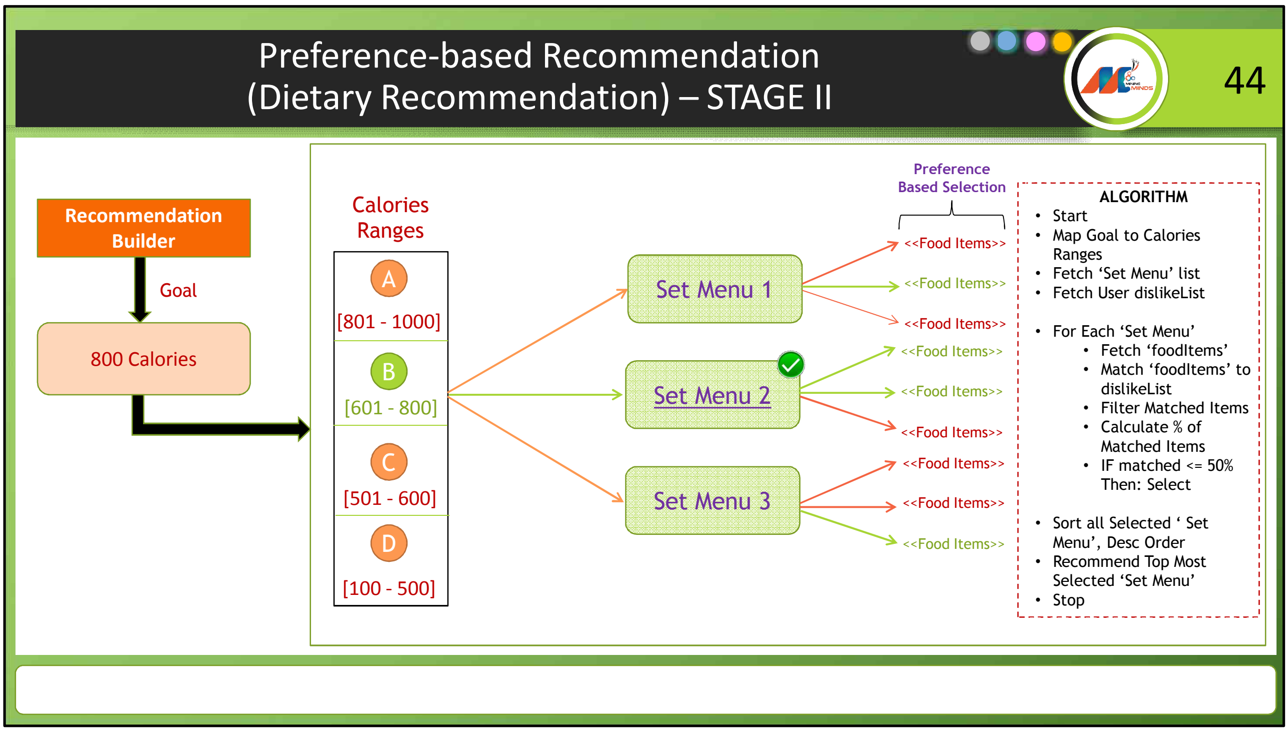
Age, Gender, Height, Weight and Activity level -> Calorie Goal

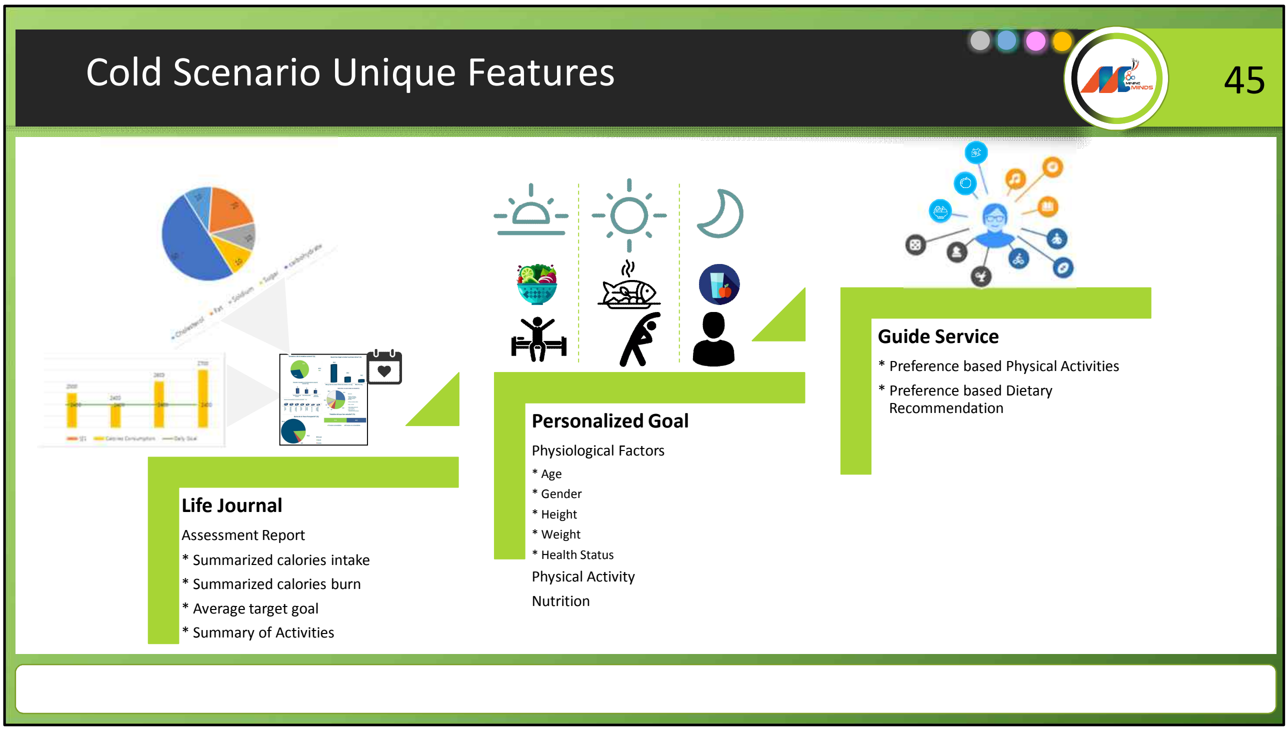
The Basal Metabolic Rate (BMR) is calculated by using the following formula:
BMR = 10 * weight(kg) + 6.25 * height(cm) - 5 * age(y) + 5 (man)
BMR = 10 * weight(kg) + 6.25 * height(cm) - 5 * age(y) - 161 (woman)
No activity =BMR calories
Exercise 1-3 times/week= 1.22*BMR ->Light Exercise
Exercise 3-5 times/week= 1.375* BMR -> Moderate Exercise
Exercise5-7 times /week= 1.55* BMR -> Very Active
Exercise more than 7times/week= 1.95*BMR ->Extra Active.

**http://www.calculator.net/calorie-calculator.html?ctype=standard&cage=33&csex=m&cheightfeet=5&cheightinch=11&cpound=150&cheightmeter =180&ckg=60&cactivity=1.375&printit=0&x=83&y=16
**https://www.bodybuilding.com/fun/calorie-know-how-get-equation-right-to-get-results.htm

Activity	MET Value
Sitting	1.5
Jumping	10
Running	8
Cycling	6
ClimbingStairs	8
Sweeping	3.3
Stretching	2.3
Dancing	7
Walking	3.5
RidingEscalator	
RidingElevator	
Standing	3
Eating	1.5
DescendingStairs	3.5
LyingDown	1.3
Hiking	6.5
Jogging	8
Sleeping	0.9
riding in bus/train	1.3


Preference-based Recommendation
(Physical Activity) - STAGE II
43
Recommendation Builder
Running: 5 min
Walking: 15 min
Stretching: 18 min
Cycling: 8 min
Jumping: 3 min
Explanation Templates
Running: You are recommended running for [] mins.
Running: Running is good for health, ...
Walking: You are recommended walking for [] mins.
Walking: Walking is good for health, ...
Cycling: You are recommended cycling for [] mins...
Cycling: Cycling is good for health, ...
Stretching: You are recommended stretching for []..
Stretching: Stretching is good for health, ...
Jumping: You are recommended Jumping for [] ...
Jumping: Jumping is good for health, ...
Recommendation
"You are recommended Walking for 15 mins to meet your daily activity goals. It is Sunny outside so take a bottle of water with you. See you in the evening."
Context
Recommendation
Running
Walking
Cyclng
Stretching
Jumping
Location= Home
HLC=Inactivity
Weather=Sunny
Emotion=Neutral
Special Condition
AGGREGATE (Σ)
Preferences
ALGORITHM
• Start
• Recevie PA recommendations
• Fetch contextual data
• Create Contextual Matrix
• Create Aggregate Vec.
• For Each Recommendation:
• Evaluate (Rec)
• Select Feasible (Rec)
• IF Rec == Multiple
• THEN: Chk_Preferences
• Select Preferred Rec
• Fetch Explanation Template
• Update Template
• Fetch Educational material
• Finalize Rec. Unit
• Stop


Preference-based Recommendation
(Dietary Recommendation) – STAGE II
44
Recommendation Builder
Goal
800 Calories
Calories Ranges
A
[801 - 1000]
B
[601 - 800]
C
[501 - 600]
D
[100 - 500]
Set Menu 1
Set Menu 2
Set Menu 3
Preference Based Selection
<<Food Items>>
<<Food Items>>
<<Food Items>>
<<Food Items>>
<<Food Items>>
<<Food Items>>
<<Food Items>>
<<Food Items>>
<<Food Items>>
ALGORITHM
• Start
• Map Goal to Calories Ranges
• Fetch 'Set Menu' list
• Fetch User dislikeList
• For Each 'Set Menu'
• Fetch 'foodItems'
• Match 'foodItems' to dislikeList
• Filter Matched Items
• Calculate % of Matched Items
• IF matched <= 50% Then: Select
• Sort all Selected ' Set Menu', Desc Order
• Recommend Top Most Selected 'Set Menu'
• Stop


Cold Scenario Unique Features
45
Life Journal
Assessment Report
* Summarized calories intake
* Summarized calories burn
* Average target goal
* Summary of Activities
Personalized Goal
Physiological Factors
* Age
* Gender
* Height
* Weight
* Health Status
Physical Activity
Nutrition
Guide Service
* Preference based Physical Activities
* Preference based Dietary Recommendation

Section 7

Mining Minds Version 3.0 Architecture Detail Description

Mining Minds

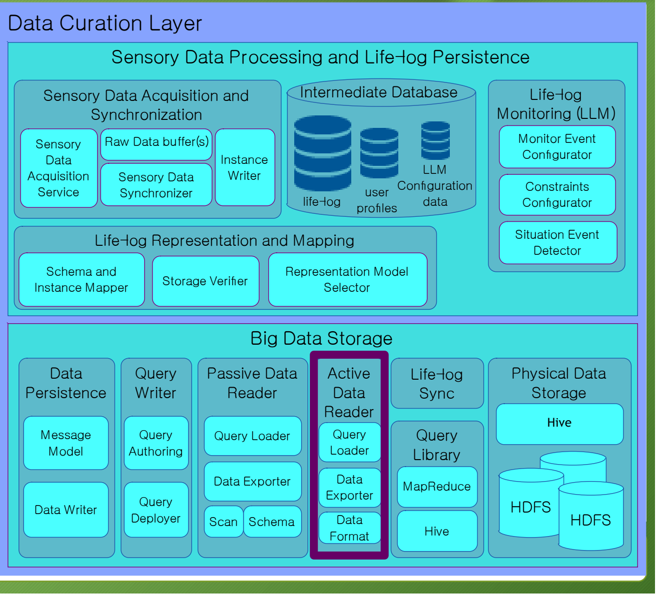
Usman Akhtar

(Version 3.0)

Ubiquitous Computing Lab,
Kyung Hee University, Korea

Introduction

- Active Data Reader:
  - For Analytics and Visualization, SL required real-time data read based on provided parameters.
  - Support for Visualization and Analytics



Motivation

- To store raw sensory, environmental variables in a large-scale non-volatile persistence (Big Data) with CRUD operations.
  - *Real-time data storage*
- For Analytics and Visualization, SL required real-time data read based on provided parameters
  - *Active data read operations*
- Big data storage should handle very large amounts of data and keep scaling to keep up with growth.
  - *Scalability with respect to the data.*

Goals

- Non-volatile storage of data from heterogeneous sources with CRUDS operations.
  - Supports of CRUD operations and REST Service end-point
- Build to handle the REST request and provide data for Visualization and Analytics.
  - High Scalability and Interactive.

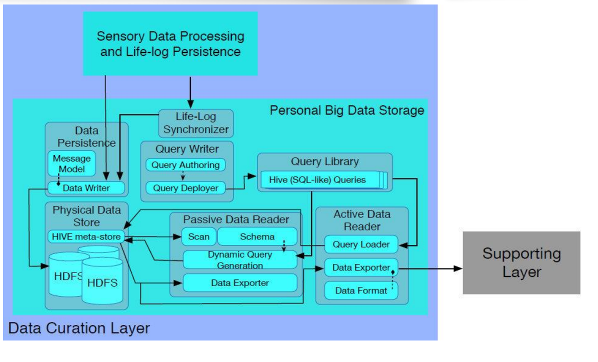
Features

- ✓*Stream-based soft real-time* data read for *Analytics and Visualization*
- ✓*Storage* of Heterogeneous data at *real-time*
- ✓Selection of the particular query depending upon the *query parameters*
- ✓Conversion of the Result-set into data message per defined *data format*
- ✓Support for the *REST Service* through a Specialized Thrift server.
- ✓Support for the *SQL operations* on the top of the Hadoop Cluster.
- ✓Support to evaluate the Apache Hive *Query performance* on raw sensory data, as published in this paper. [1]

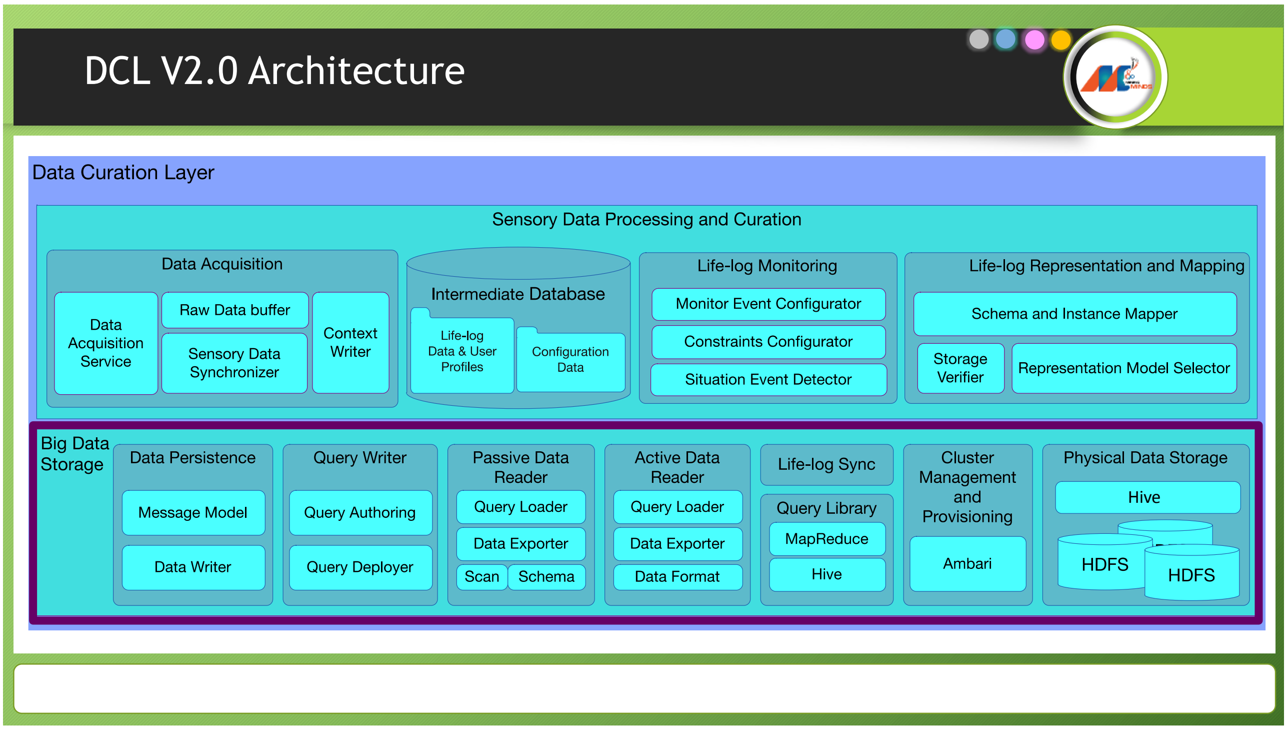
[1] Muhammad Bilal Amin, Oresti Banos, Wajahat Ali Khan, Hafiz Syed Muhammad Bilal, Jinhyuk Gong, Dinh-Mao Bui, Soung Ho Cho, Shujaat Hussain, Taqdir Ali, Usman Akhtar, Tae Choong Chung and Sungyoung Lee, "On Curating Multimodal Sensory Data for Health and Wellness Platforms", Sensors (SCIE, IF: 2.033), vol. 16,no. 7, doi:10.3390/s16070980 , 2016

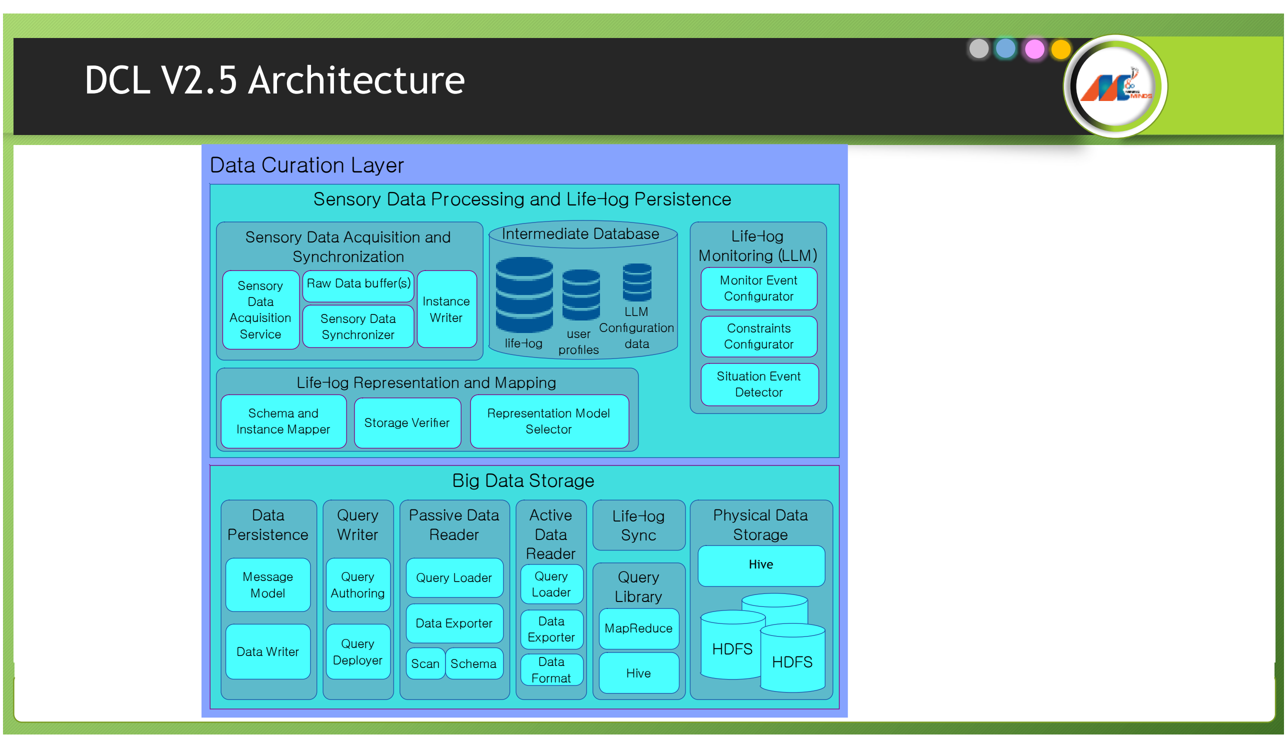
Role of Big Data Storage in Mining Mind

- Role
  - Non-Volatile storage of data from heterogeneous sources with CRUDS operations.
  - Allow active and passive data read operations

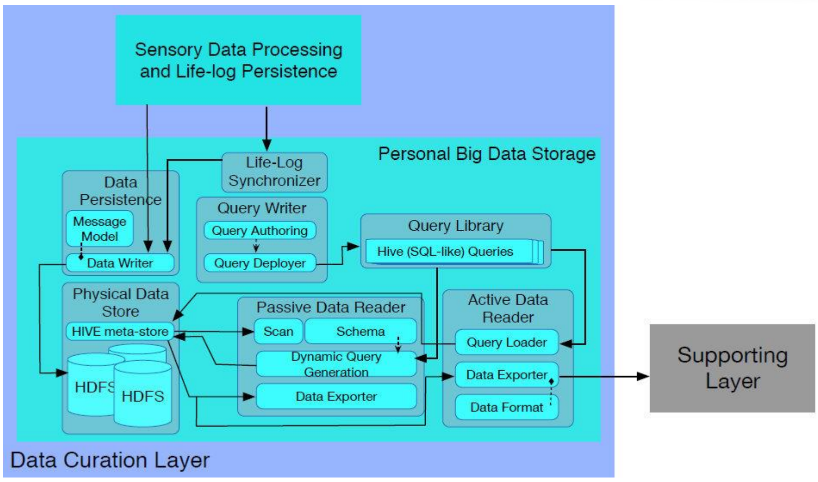


- How is it different from Ver. 1.5 ?
  - Data Raw sensory and environmental data was only written in Big Data Storage
  - Read access for data was not available


DCL V2.0 Architecture
Data Curation Layer
Sensory Data Processing and Curation
Data Acquisition
Data Acquisition Service
Raw Data buffer
Sensory Data Synchronizer
Context Writer
Intermediate Database
Life-log Data & User Profiles
Configuration Data
Life-log Monitoring
Monitor Event Configurator
Constraints Configurator
Situation Event Detector
Life-log Representation and Mapping
Schema and Instance Mapper
Storage Verifier
Representation Model Selector
Big Data Storage
Data Persistence
Message Model
Data Writer
Query Writer
Query Authoring
Query Deployer
Passive Data Reader
Query Loader
Data Exporter
Scan
Schema
Active Data Reader
Query Loader
Data Exporter
Data Format
Life-log Sync
Query Library
MapReduce
Hive
Cluster Management and Provisioning
Ambari
Physical Data Storage
Hive
HDFS
HDFS


DCL V2.5 Architecture
Data Curation Layer
Sensory Data Processing and Life-log Persistence
Sensory Data Acquisition and Synchronization
Sensory Data Acquisition Service
Raw Data buffer(s)
Sensory Data Synchronizer
Instance Writer
Intermediate Database
life-log
user profiles
LLM Configuration data
Life-log Monitoring (LLM)
Monitor Event Configurator
Constraints Configurator
Situation Event Detector
Life-log Representation and Mapping
Schema and Instance Mapper
Storage Verifier
Representation Model Selector
Big Data Storage
Data Persistence
Message Model
Data Writer
Query Writer
Query Authoring
Query Deployer
Passive Data Reader
Query Loader
Data Exporter
Scan
Schema
Active Data Reader
Query Loader
Data Exporter
Data Format
Life-log Sync
Query Library
MapReduce
Hive
Physical Data Storage
Hive
HDFS
HDFS

Big Data Storage - Tools

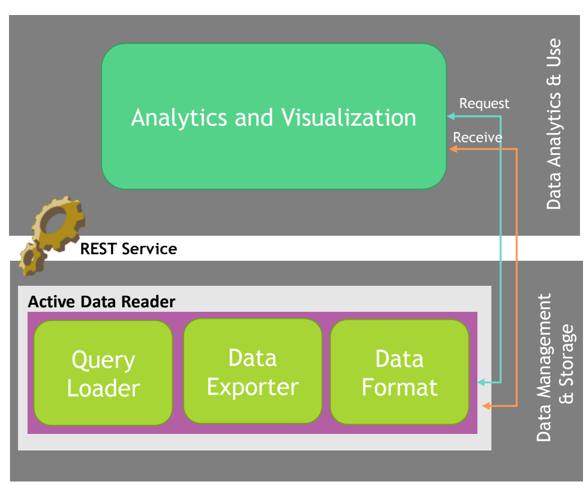


Tools and Technologies

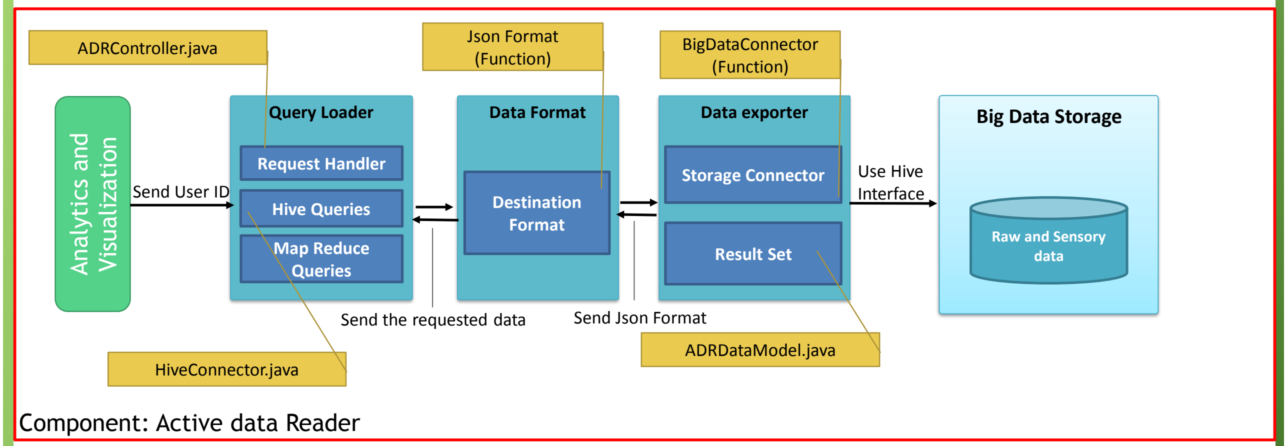
Active Data Reader Implementation

Provide Data for Visualization

1. Built for REST Services (request handle).
2. Support for Visualization and Analytics
3. Highly scalable and interactive

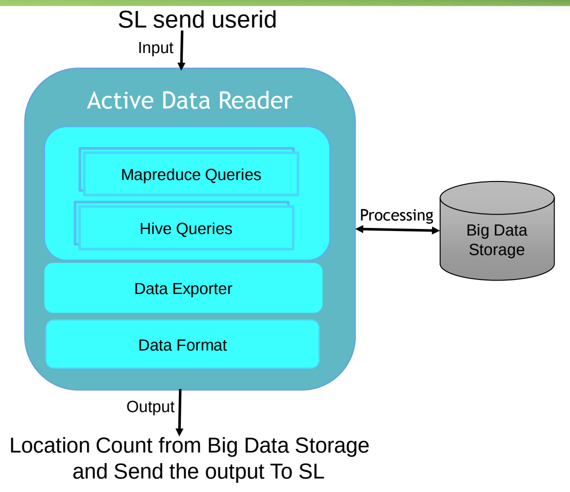


Active Data Reader – Internal Implementation Details



GitHub Source Code Available at: https://github.com/ubiquitous-computing-lab/Mining-Minds/tree/master/data-curation-layer/active-data-reader

MMv3.0 Active Data Reader

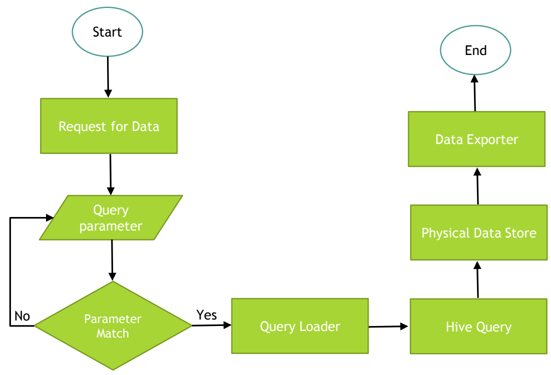


- **Output**
  - ✓ User Location from the Big data (Give the SL on label counts after searching the records the Big Storage)
    - ✓ Using REST SERVICES
- **Input**
  - ✓ User ID to calculate the Location
- **Implementation Module**
  - ✓ Active Data Reader

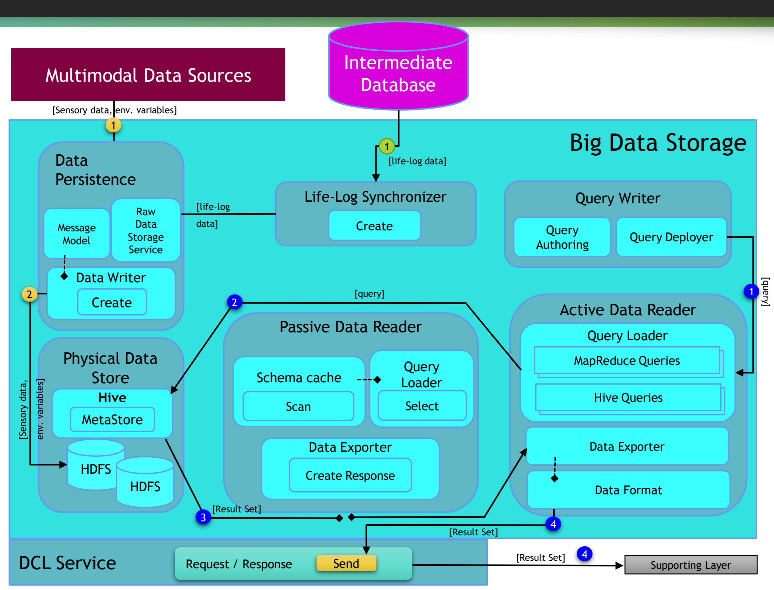
Active Data Reader – Handling Requests

Handling Online Data Request for Data Visualization and Analytics

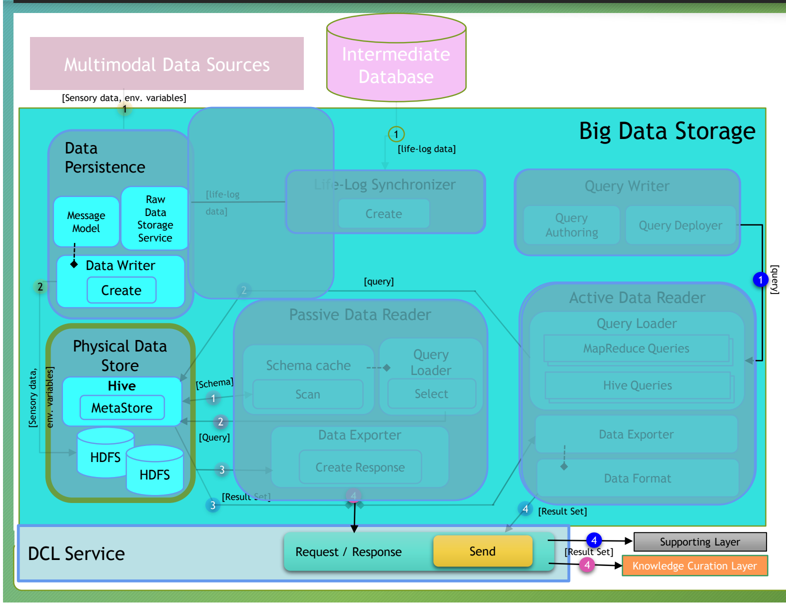
1. Data read request is generated by Visualization and Analytics Components of SL
2. Active data reader selects the particular query depending upon the query parameters
3. Selected Query is sent to physical data store for execution
4. Required data is returned as a result set to Data exporter
5. Result-set is converted into data message per defined data format and send to the SL



Functional Diagram



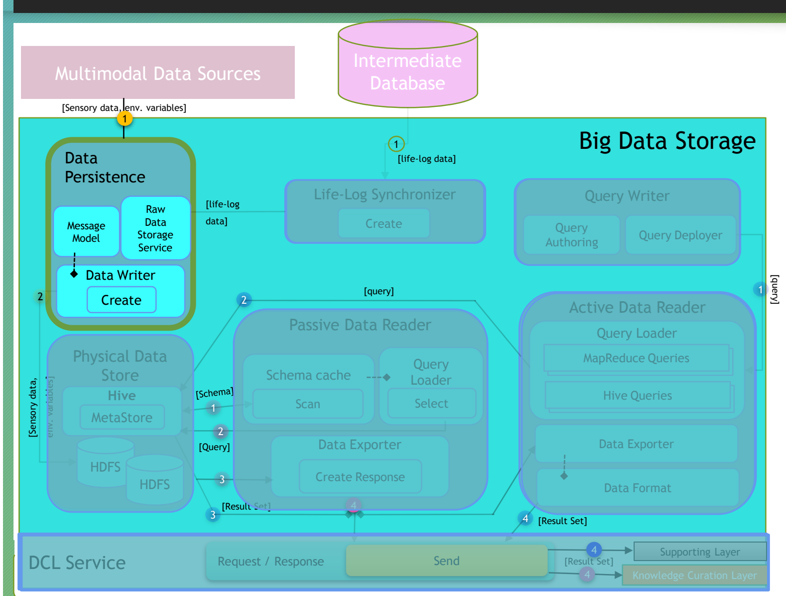
Big Data Storage Components Details



1. Physical Data Storage

- *Role*
  - Non-volatile Persistence to raw-sensory and environmental data
- *Input*
  - Data Read Request
  - Data Write Request
- *Output*
  - Result set in case of Data Read
  - Acknowledgment in case of Data Write

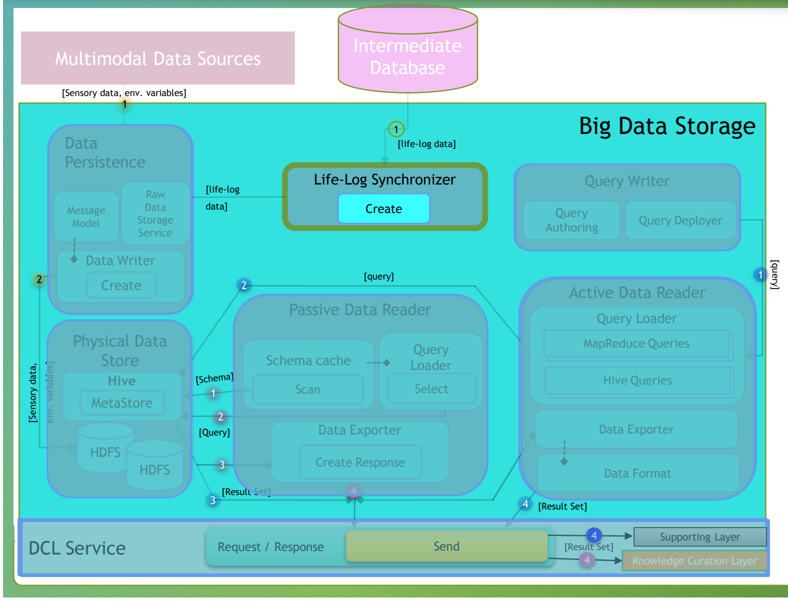
Big Data Storage Components Details



2. Data Persistence

- *Role*
  - De-serialized data according to message model
- *Input*
  - Raw sensory data
- *Output*
  - De-serialized data according to the format

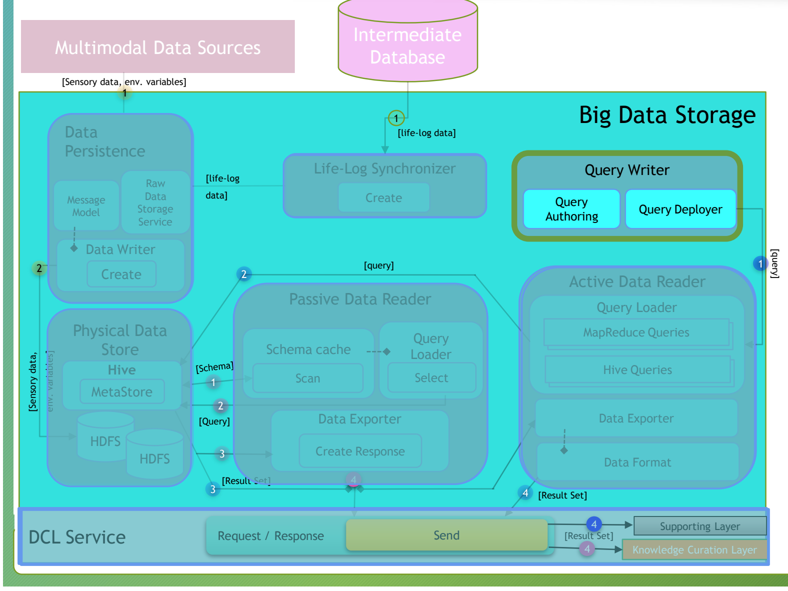
Big Data Storage Components Details



3. <u>Life-log Synchronizer</u>

- *Role*
  - Temporal backups for life-log data in non-volatile persistence
- *Input*
  - Life-log data, timestamps etc.
- *Output*
  - Life-log result set

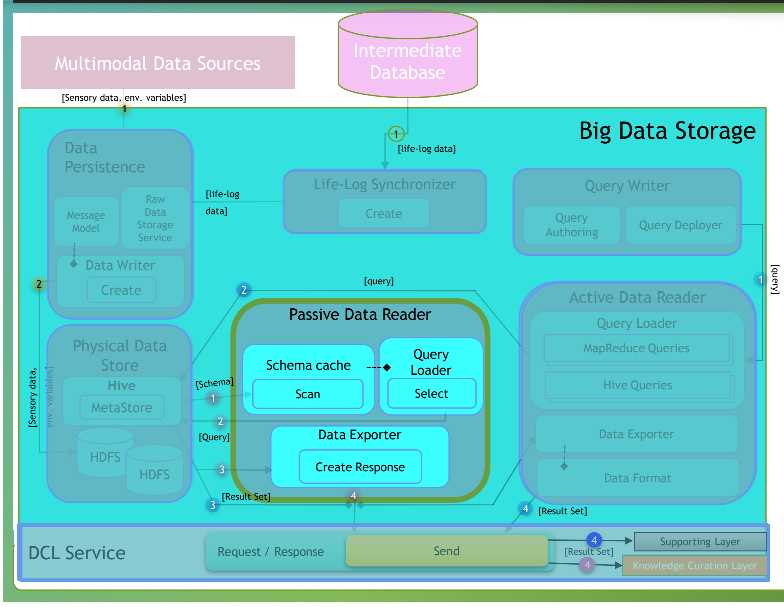
Big Data Storage Components Details



<u>4. Query Writer</u>

- *Role*
  - Build query and execute on data
- *Input*
  - Query library
- *Output*
  - *Query*

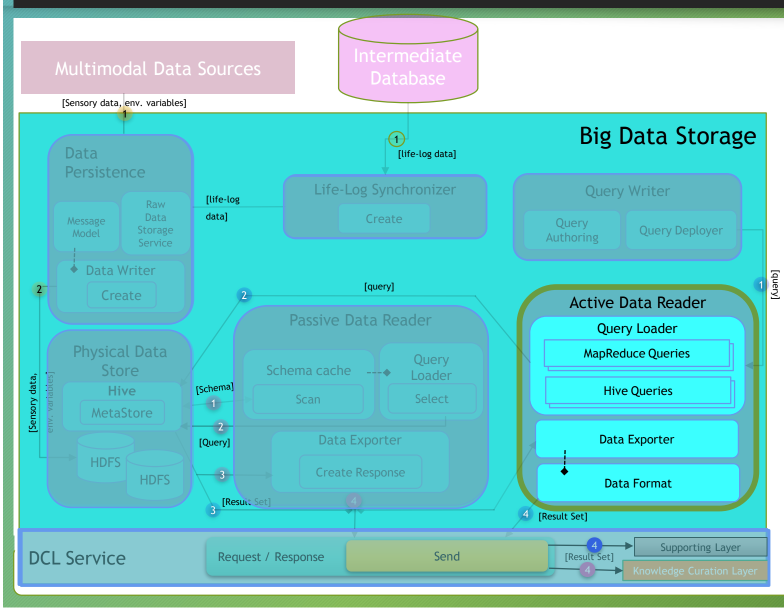
Big Data Storage Components Details



5. Passive Data Reader

- *Role*
  - Received Request for schema for offline data request
- *Input*
  - *Query parameter*
  - *Request for schema*
- *Output*
  - *Result Set*
  - *Schema*

Big Data Storage Components Details



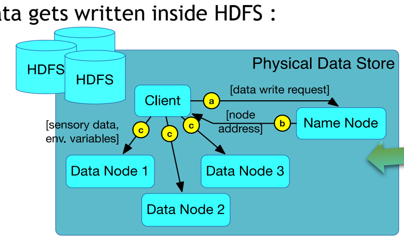
6. Active Data Reader

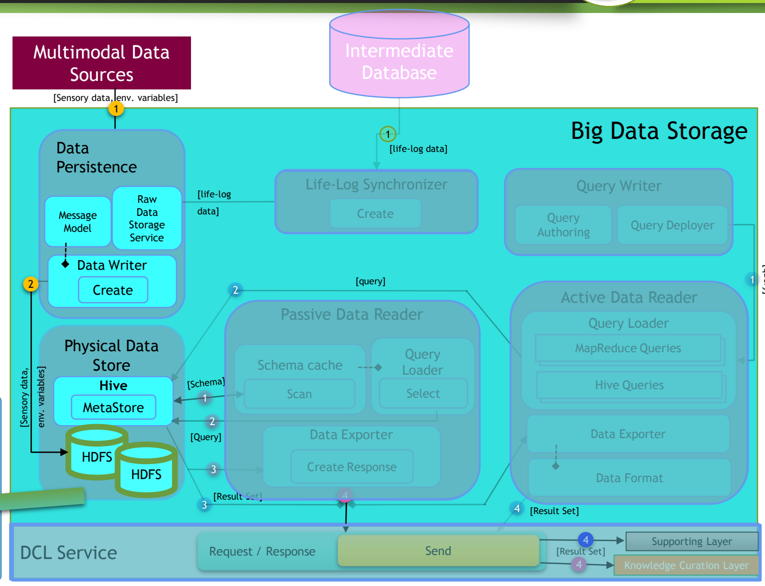
- *Role*
  - Online data request for data visualization and analytics
- *Input*
  - *Query parameter*
  - *Request for schema*
- *Output*
  - *Result Set in message format*

Execution Scenario #1

<u>Writing Raw Sensory Data and environment variables into Non-volatile storage in real-time</u>

1. Raw Sensory data is received by Data Writer of Data Persistence Component
   i. Data is de-serialized according to the Message Model
2. De-serialized message is sent to HDFS for persistence
3. Data gets written inside HDFS :

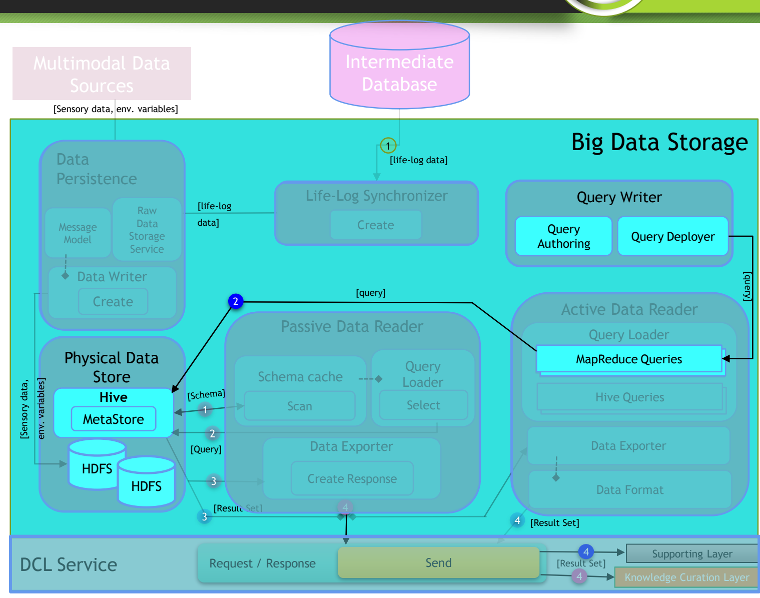




Execution Scenario #2

<u>Offline Query Authoring</u>

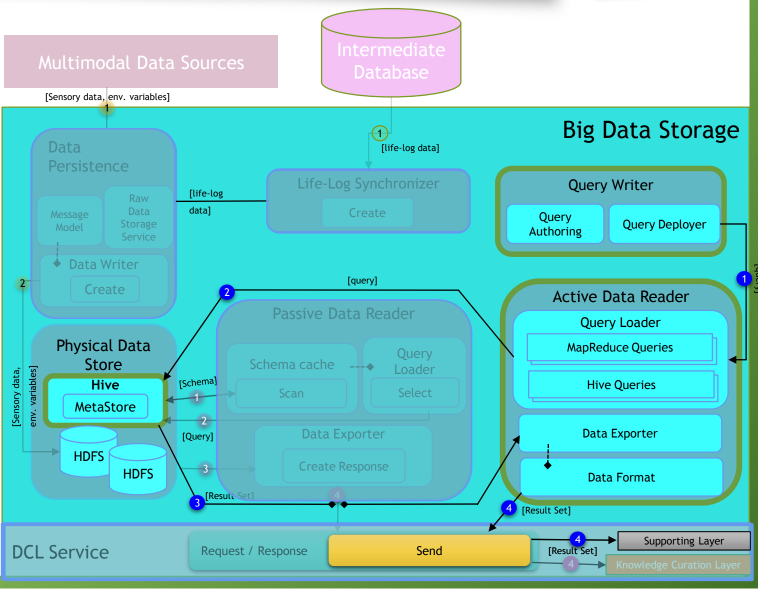
1. Big Data Storage Requirements are converted into MapReduce Queries
2. Queries are written using Eclipse as an IDE
3. Query is tested and deployed as part of Query writer in Big Data Storage



Execution Scenario #3

Online Data Request for Data Visualization and Analytics

1. Data read request is generated by Visualization and Analytics Components of SL
2. Active data reader selects the particular query depending upon the query parameters
3. Selected Query is sent to Hive for execution
4. Required data is returned as a result set to Data exporter
5. Result-set is converted into data message per defined data format

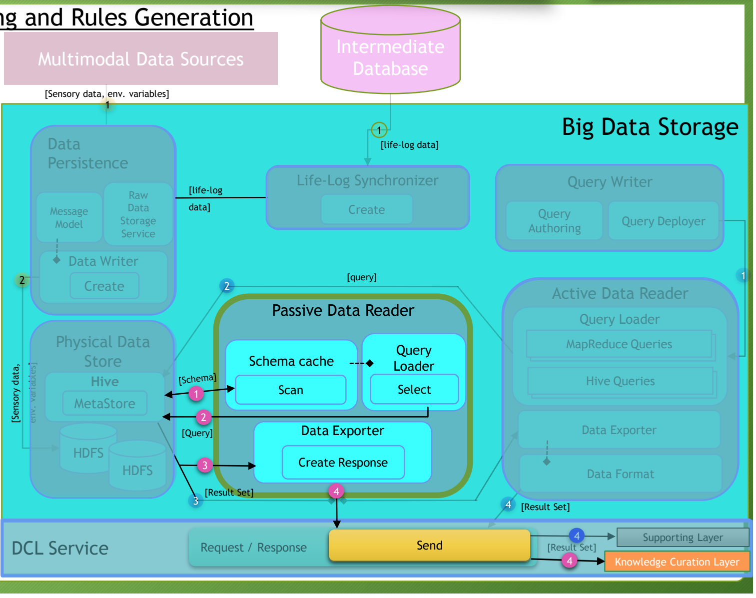


경희대학교 KYUNG HEE UNIVERSITY

Execution Scenario #4

Offline Data Request by KCL for Model Training and Rules Generation

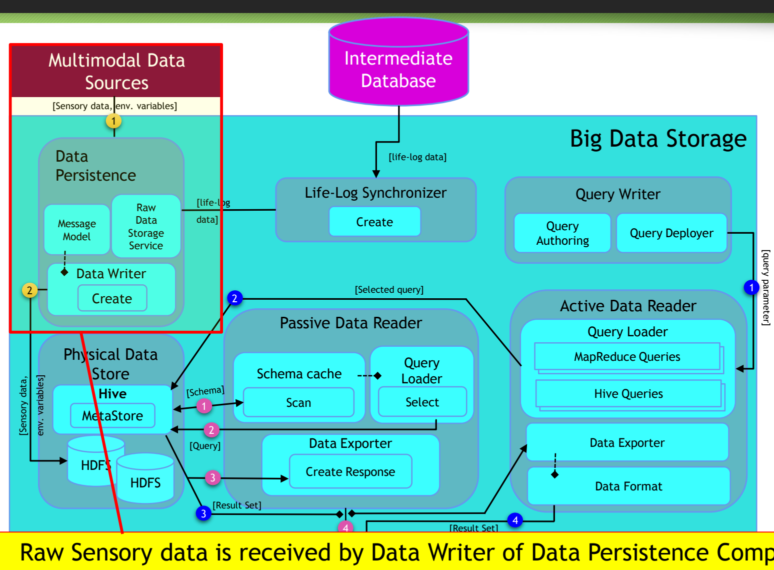
1. Request for Schema of the persisted data is received by the Passive Data Reader
2. Scanned and most updated schema from Non-volatile storage is returned to KCL
3. KCL selects the parameters from the schema to generate a query and submits the request to Passive Data Reader
4. Passive Data Reader selects the query from Query Library
5. Selected Query is sent to Hive for execution
6. Required data is returned as a result set to Data exporter
7. Result-set is converted into data message and returned to KCL



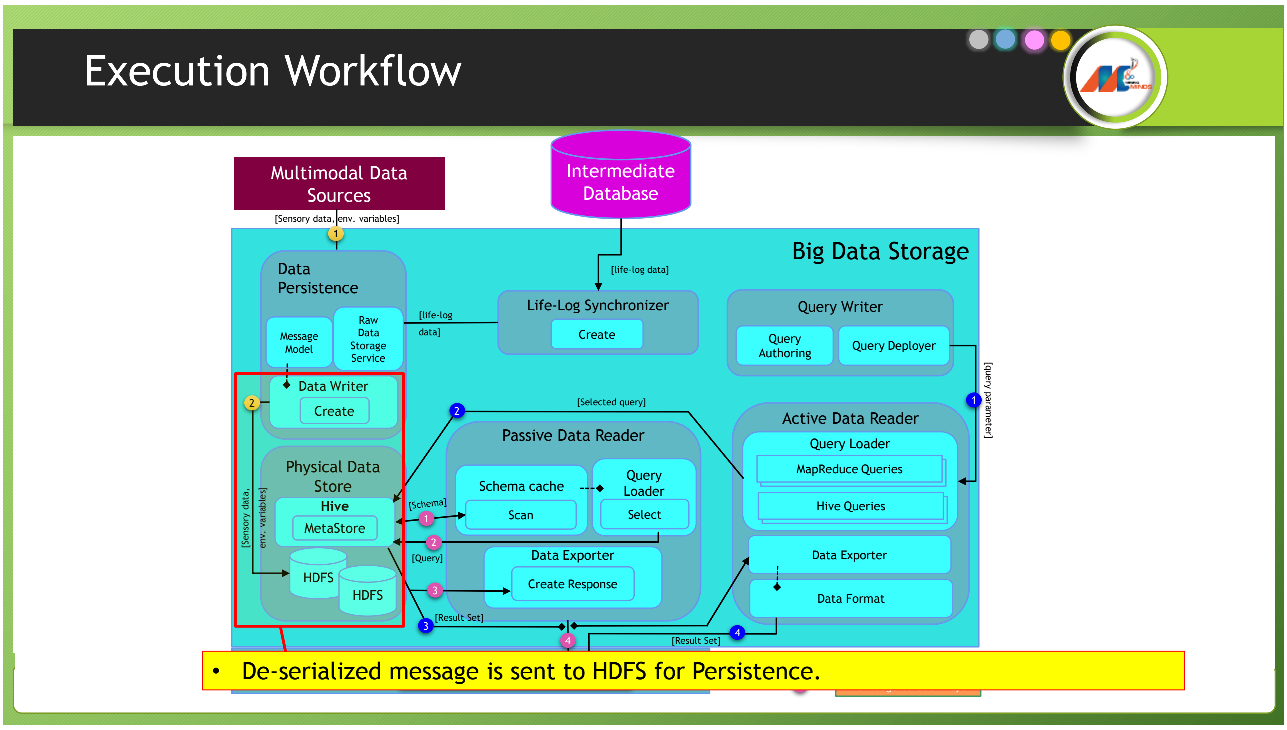
경희대학교 KYUNG HEE UNIVERSITY

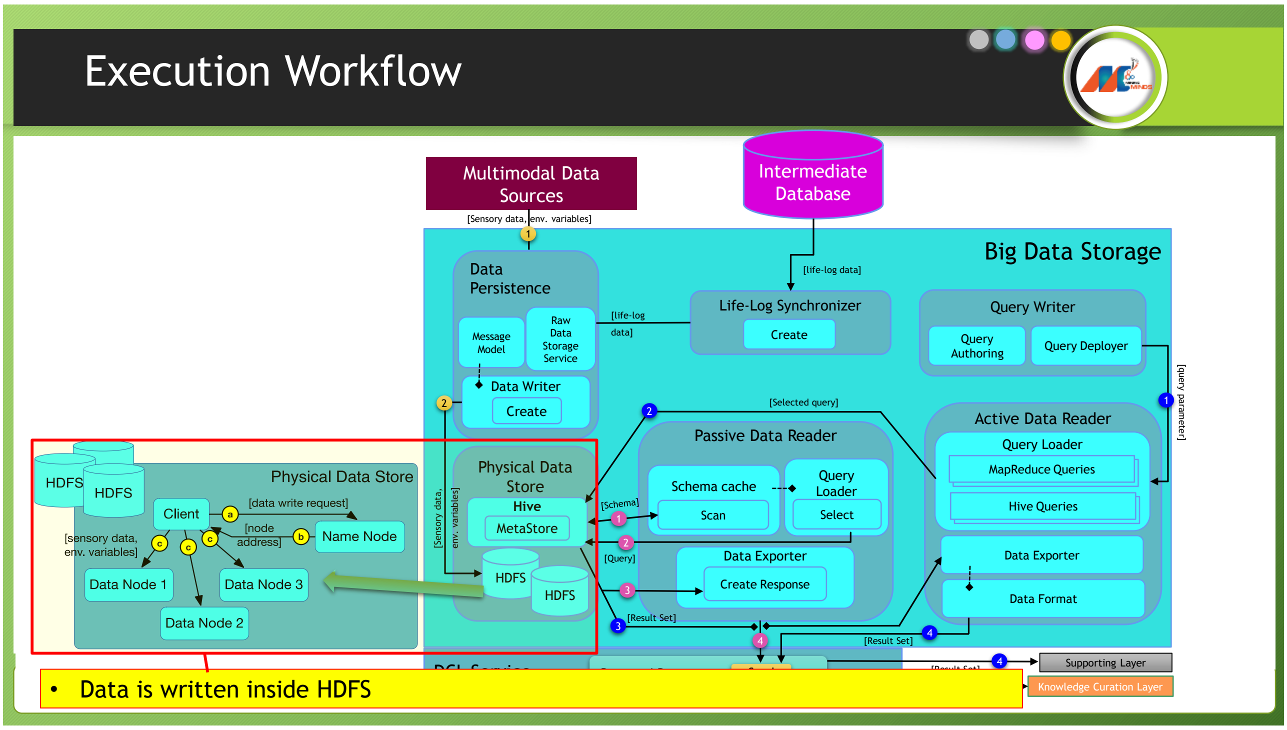
Execution Workflow

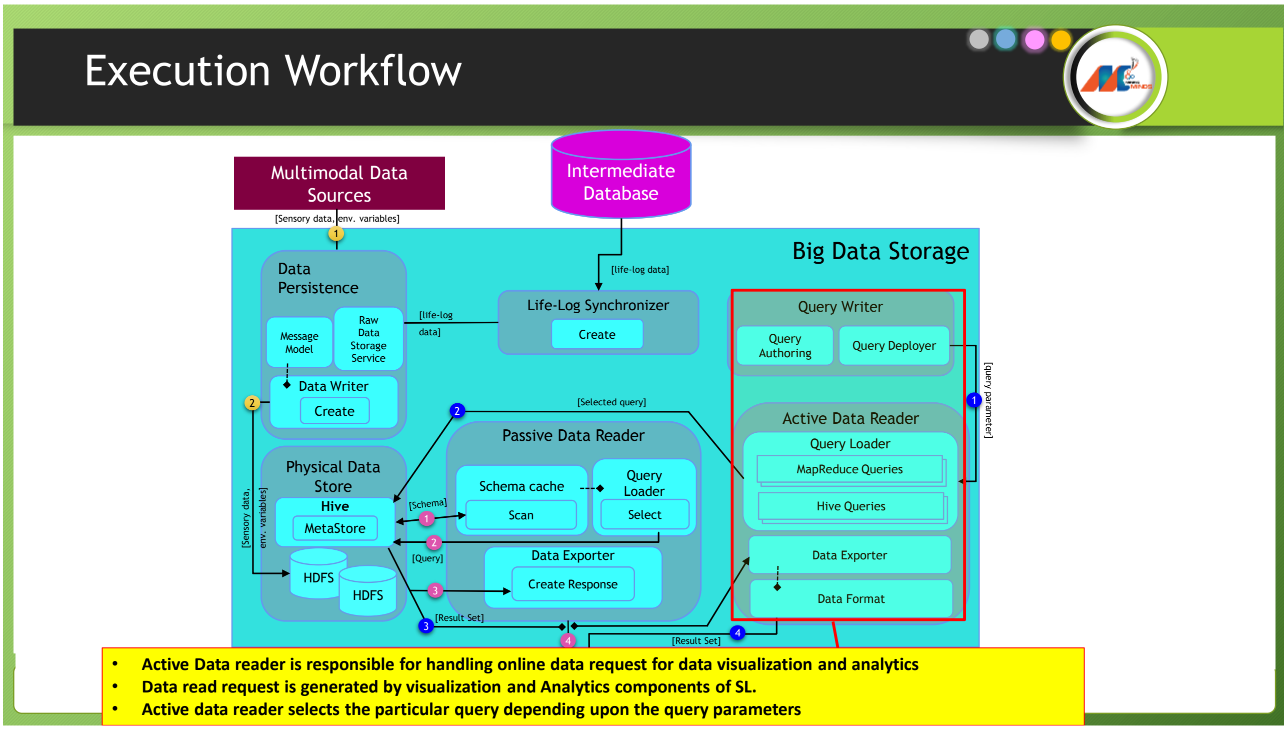
Execution Workflow

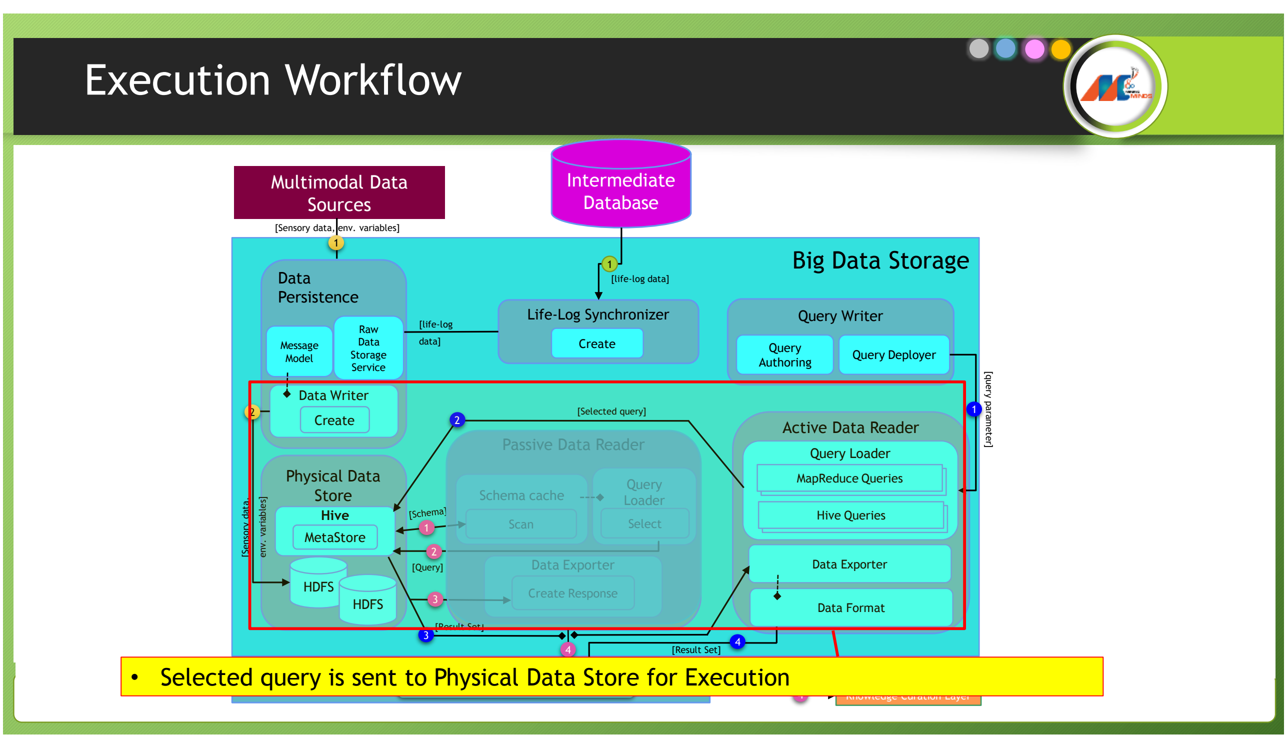


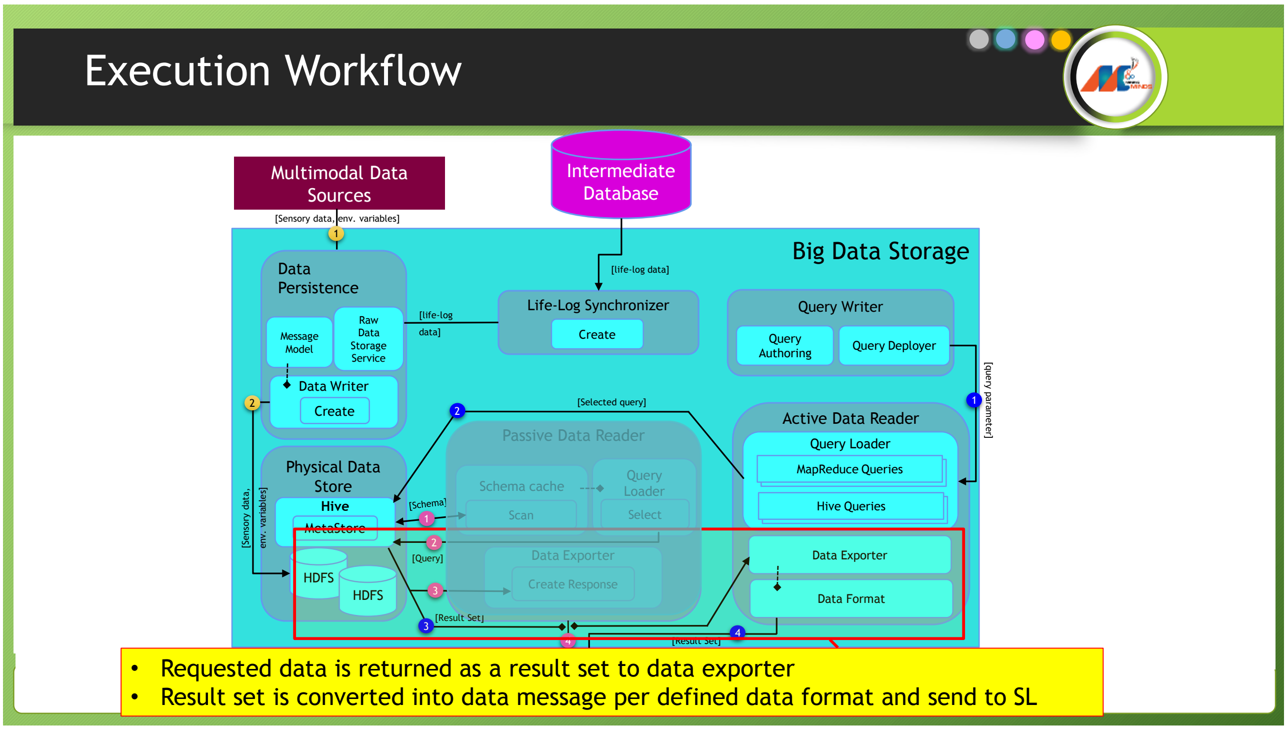
- Raw Sensory data is received by Data Writer of Data Persistence Component.
- Data is de-serialized according to the message model

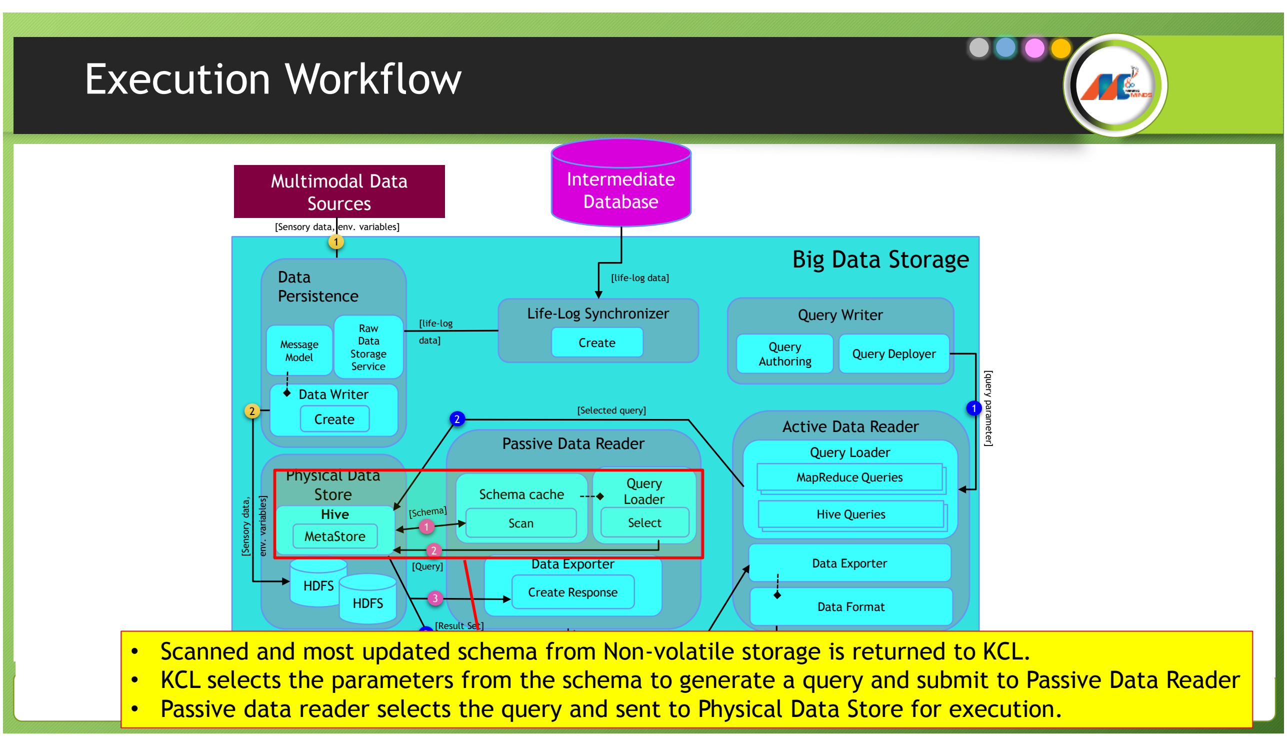

Execution Workflow
Multimodal Data Sources
[Sensory data, env. variables]
Intermediate Database
Big Data Storage
[life-log data]
Data Persistence
Message Model
Raw Data Storage Service
[life-log data]
Life-Log Synchronizer
Create
Query Writer
Query Authoring
Query Deployer
[query parameter]
Data Writer
Create
[Selected query]
Passive Data Reader
Active Data Reader
Query Loader
MapReduce Queries
Hive Queries
Physical Data Store
Hive
MetaStore
[Sensory data, env. variables]
Schema cache
Query Loader
Scan
Select
[Schema]
[Query]
Data Exporter
Create Response
Data Exporter
Data Format
HDFS
HDFS
[Result Set]
[Result Set]
• De-serialized message is sent to HDFS for Persistence.


Execution Workflow
Multimodal Data Sources
[Sensory data, env. variables]
Intermediate Database
Big Data Storage
[life-log data]
Data Persistence
Message Model
Raw Data Storage Service
[life-log data]
Life-Log Synchronizer
Create
Query Writer
Query Authoring
Query Deployer
[query parameter]
Data Writer
Create
[Selected query]
Passive Data Reader
Active Data Reader
Query Loader
MapReduce Queries
Hive Queries
Physical Data Store
HDFS
HDFS
Client
[data write request]
[node address]
Name Node
[sensory data, env. variables]
Data Node 1
Data Node 3
Data Node 2
Physical Data Store
Hive
MetaStore
Schema cache
Query Loader
Scan
Select
[Schema]
[Query]
Data Exporter
Create Response
Data Exporter
Data Format
HDFS
HDFS
[Result Set]
[Result Set]
Supporting Layer
Knowledge Curation Layer
• Data is written inside HDFS

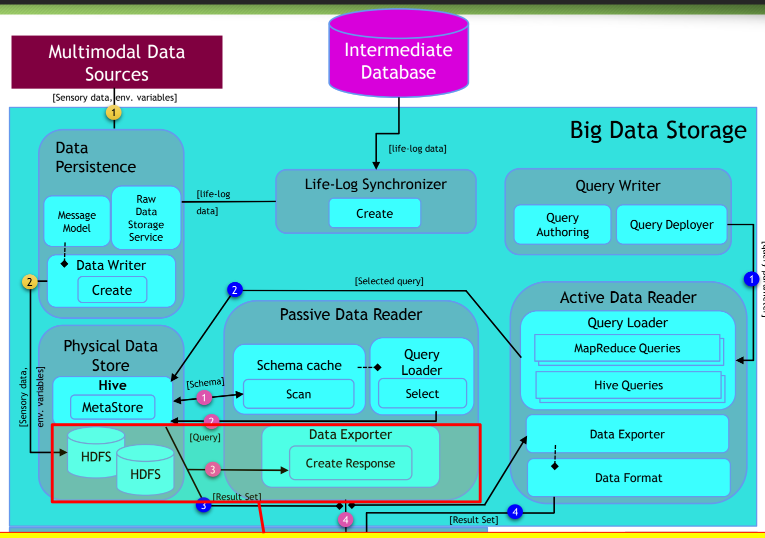

Execution Workflow
Multimodal Data Sources
[Sensory data, env. variables]
Intermediate Database
[life-log data]
Big Data Storage
Data Persistence
Message Model
Raw Data Storage Service
Data Writer
Create
[life-log data]
Life-Log Synchronizer
Create
Query Writer
Query Authoring
Query Deployer
[query parameter]
[Selected query]
Active Data Reader
Query Loader
MapReduce Queries
Hive Queries
Data Exporter
Data Format
Passive Data Reader
Schema cache
Scan
Query Loader
Select
Physical Data Store
Hive
MetaStore
HDFS
HDFS
[Sensory data, env. variables]
[Schema]
[Query]
Data Exporter
Create Response
[Result Set]
[Result Set]
• Active Data reader is responsible for handling online data request for data visualization and analytics
• Data read request is generated by visualization and Analytics components of SL.
• Active data reader selects the particular query depending upon the query parameters


Execution Workflow
Multimodal Data Sources
[Sensory data, env. variables]
Intermediate Database
[life-log data]
Big Data Storage
Data Persistence
Message Model
Raw Data Storage Service
Data Writer
Create
[life-log data]
Life-Log Synchronizer
Create
Query Writer
Query Authoring
Query Deployer
[query parameter]
[Selected query]
Active Data Reader
Query Loader
MapReduce Queries
Hive Queries
Data Exporter
Data Format
Passive Data Reader
Schema cache
Scan
Query Loader
Select
Physical Data Store
Hive
MetaStore
HDFS
HDFS
[Sensory data, env. variables]
[Schema]
[Query]
Data Exporter
Create Response
[Result Set]
• Selected query is sent to Physical Data Store for Execution


Execution Workflow
Multimodal Data Sources
[Sensory data, env. variables]
Intermediate Database
Big Data Storage
[life-log data]
Data Persistence
Message Model
Raw Data Storage Service
Data Writer
Create
[life-log data]
Life-Log Synchronizer
Create
Query Writer
Query Authoring
Query Deployer
[query parameter]
[Selected query]
Passive Data Reader
Schema cache
Query Loader
Scan
Select
Active Data Reader
Query Loader
MapReduce Queries
Hive Queries
Physical Data Store
Hive
MetaStore
HDFS
HDFS
[Sensory data, env. variables]
[Schema]
[Query]
Data Exporter
Create Response
[Result Set]
Data Exporter
Data Format
[Result Set]
Requested data is returned as a result set to data exporter
Result set is converted into data message per defined data format and send to SL

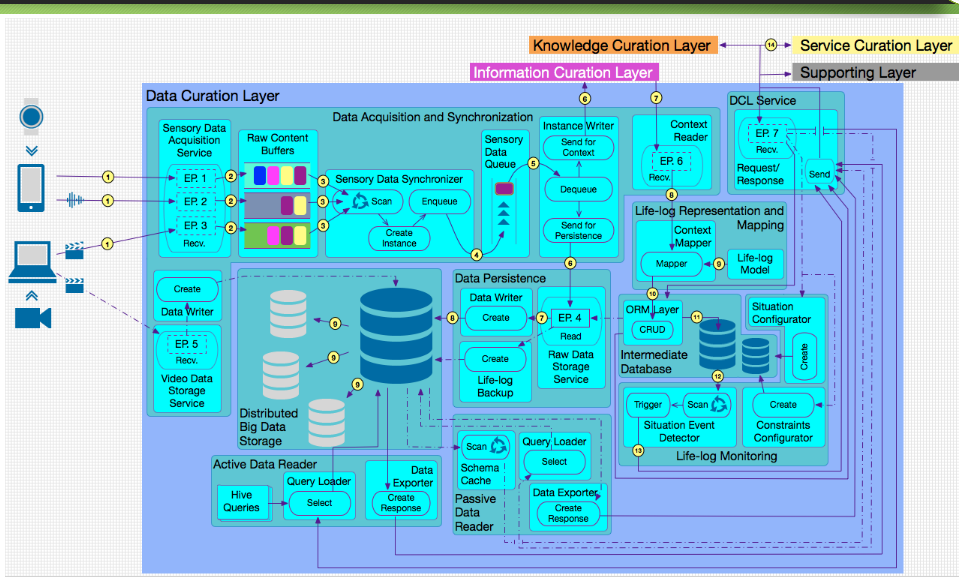

Execution Workflow
Multimodal Data Sources
[Sensory data, env. variables]
Intermediate Database
Big Data Storage
[life-log data]
Data Persistence
Message Model
Raw Data Storage Service
Data Writer
Create
[life-log data]
Life-Log Synchronizer
Create
Query Writer
Query Authoring
Query Deployer
[query parameter]
[Selected query]
Passive Data Reader
Schema cache
Query Loader
Scan
Select
Active Data Reader
Query Loader
MapReduce Queries
Hive Queries
Physical Data Store
Hive
MetaStore
HDFS
HDFS
[Sensory data, env. variables]
[Schema]
[Query]
Data Exporter
Create Response
[Result Set]
Data Exporter
Data Format
Scanned and most updated schema from Non-volatile storage is returned to KCL.
KCL selects the parameters from the schema to generate a query and submit to Passive Data Reader
Passive data reader selects the query and sent to Physical Data Store for execution.

Execution Workflow



- Required data is returned as a result set to data exporter
- The result set is returned to KCL

DCL 2.5 Detailed Architecture



Contribution

- *Storage* of Heterogeneous data at *real-time*
- *Stream-based soft real-time* data read for *Analytics and Visualization*
- Schema-based query selection and execution over Big Data Storage
  - Availability to the most updated schema of persisted Data
- Temporal backups of Life-log data for non-volatile storage
- Able to build the big data ecosystem that facilitate request from the other layers.

경희대학교 KYUNG HEE UNIVERSITY

Mining Minds

Data Synchronization Specification

Dinh-Mao Bui

경희대학교 KYUNG HEE UNIVERSITY KITECH POSTECH UNIVERSITY of TASMANIA 녹십자헬스케어 TAPACROSS U2 system User & Usability system KOCSA

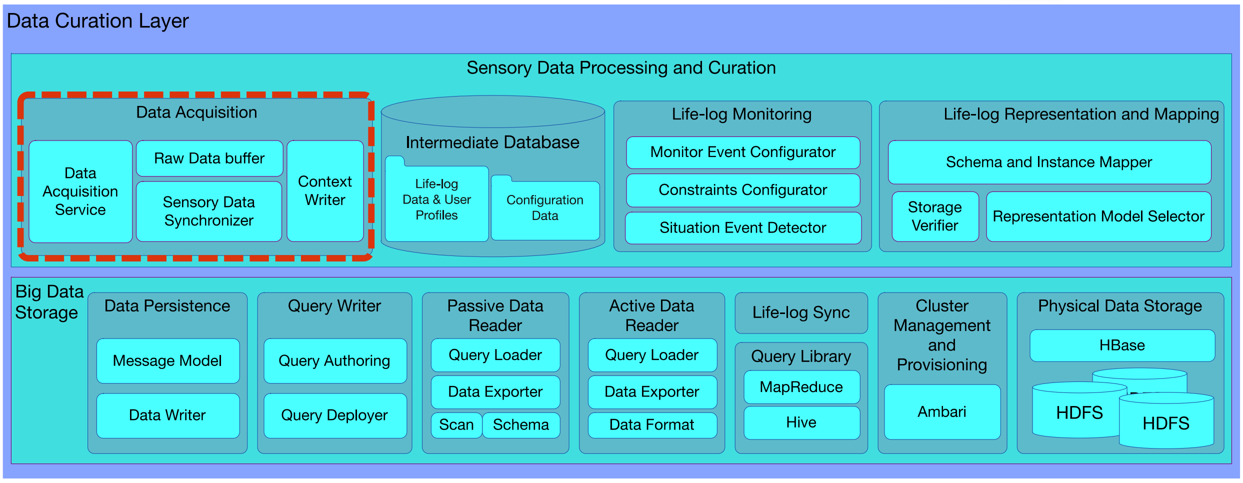
Agenda

- Introduction
- Sync component
- Sync architecture
- Sync features
- Sync strategies
- Contribution

Introduction

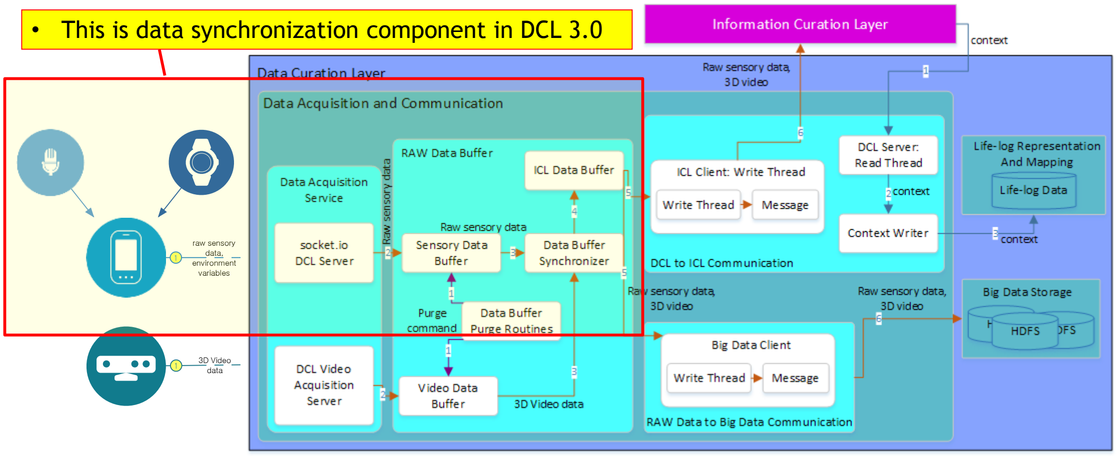
- Sensory Data Acquisition and Synchronization subcomponent obtains the raw sensory data from multi-modal data sources, both in a real-time (active) and off-line (passive) manner.
- This data is synchronized based upon the user identification and the time stamp of the data generation, and subsequently, it is queued for the context determination.
- Implementation of Raw Sensory Data Acquisition and Synchronization (DAS) consists of a REST service that collects raw sensory data from multi-modal data sources.
- The key in this acquisition is the association of accumulated data with their time of origination.
- All data sources subsist independently along with independent clocks; therefore, a logical clock is required for identifying the data origination at the same time from multiple sources.
- Consequently, DAS implements the time frame-based synchronization methods called Complete- and Incomplete-sync.

Synchronization component



Synchronization component

- This is data synchronization component in DCL 3.0

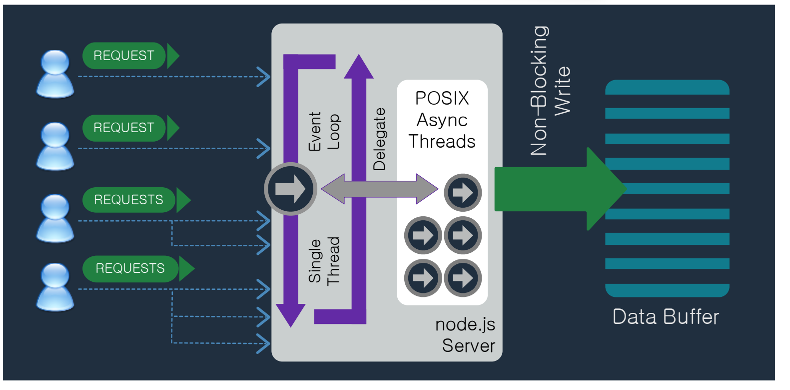


Asynchronous Non-Blocking communication

Synchronization architecture

Node.js

1. Built for HTTP support (device independence)
2. Support for Real-time Communication
3. Non-Blocking IO (Asynchronous)
4. Highly scalable by event-based callback



JavaScript

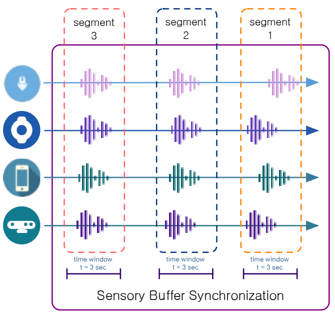
Synchronization features

- Fully implemented on node.js.
- Fully commented.
- Support huge number of sensors (up to 1280 sensors).
- Scalable data structure.
- Can run as a standalone component or be integrated into other software.
- Source code is validated and benchmarked the performance [1].

[1] Amin, Muhammad Bilal, Oresti Banos, Wajahat Ali Khan, Hafiz Syed Muhammad Bilal, Jinhyuk Gong, Dinh-Mao Bui, Soung Ho Cho et al. "On curating multimodal sensory data for health and wellness platforms." *Sensors* 16, no. 7 (2016): 980.

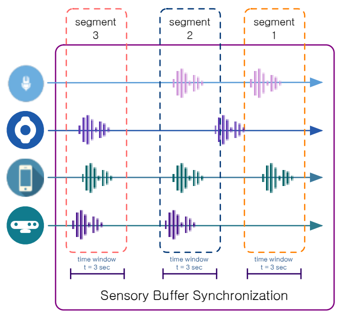
Synchronization strategies

1. Complete Sync



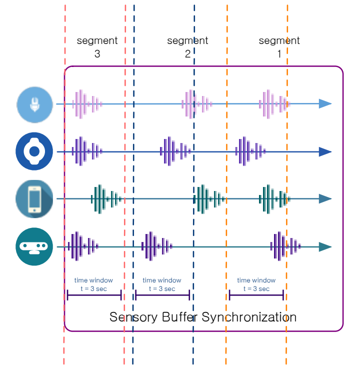
- Executes when all the required sensory data is received in time window
- Support for highly accurate context determination
- Only possible when all the data sources are almost time- and communication-synced

2. Eager Sync



- Executes in regular interval time without the dependence on data sources
- Support for real-time execution with no delays
- Ignores out-windowed packets; resulting in lower accuracy for context determination

3. Rendezvous Sync



- Executes Executes when all the required sensory data is received
- Support for highly accurate context determination
- No guarantee for real-time execution due to delay in sync

Contribution

- Design adaptive data structure.
- Design new synchronization algorithm.
- Integrate synchronization algorithm in to the system (working properly at the moment).
- Conduct the benchmarks and experiments for the anchor paper.
- Code refactoring completed.
- Apply design pattern into source code.
- API design and implementation.

Mining Minds

MMV3.0

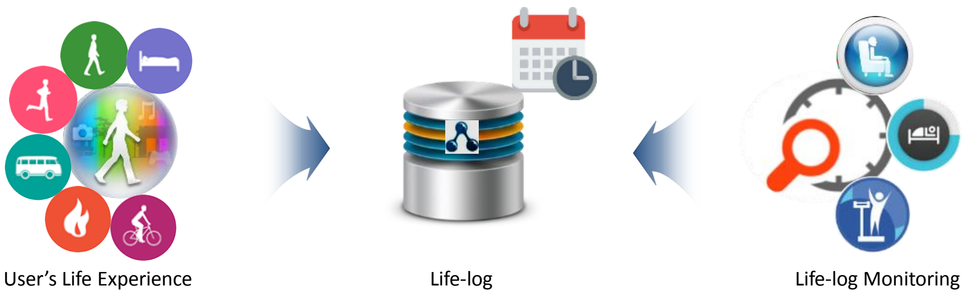
Agenda

- Overview
- Motivation
- Life-log Monitoring Conceptual View
- Lifelog Monitor Architecture
- Offline Process
  - Register Monitoring Condition
- Online Process
  - Monitoring Situation
- Challenges & Contributions

Overview

Lifelog persistently record and archive some informational dimension of *user's* (user lifelog) *life experience* in a particular data category.

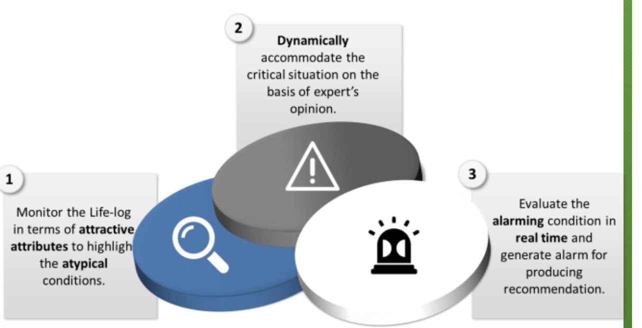
Life-log monitoring is to *trace* the 'threads' of an individual's life in terms of events to observe a *situation* over time.

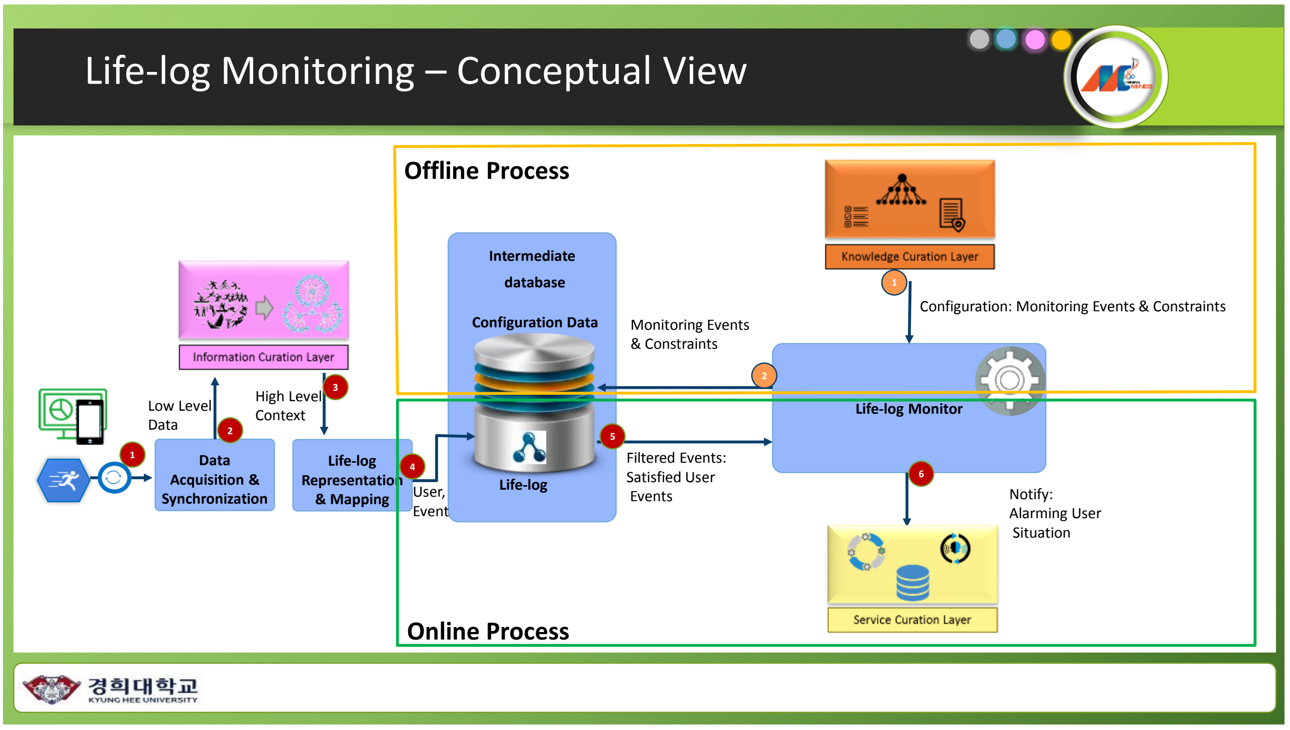


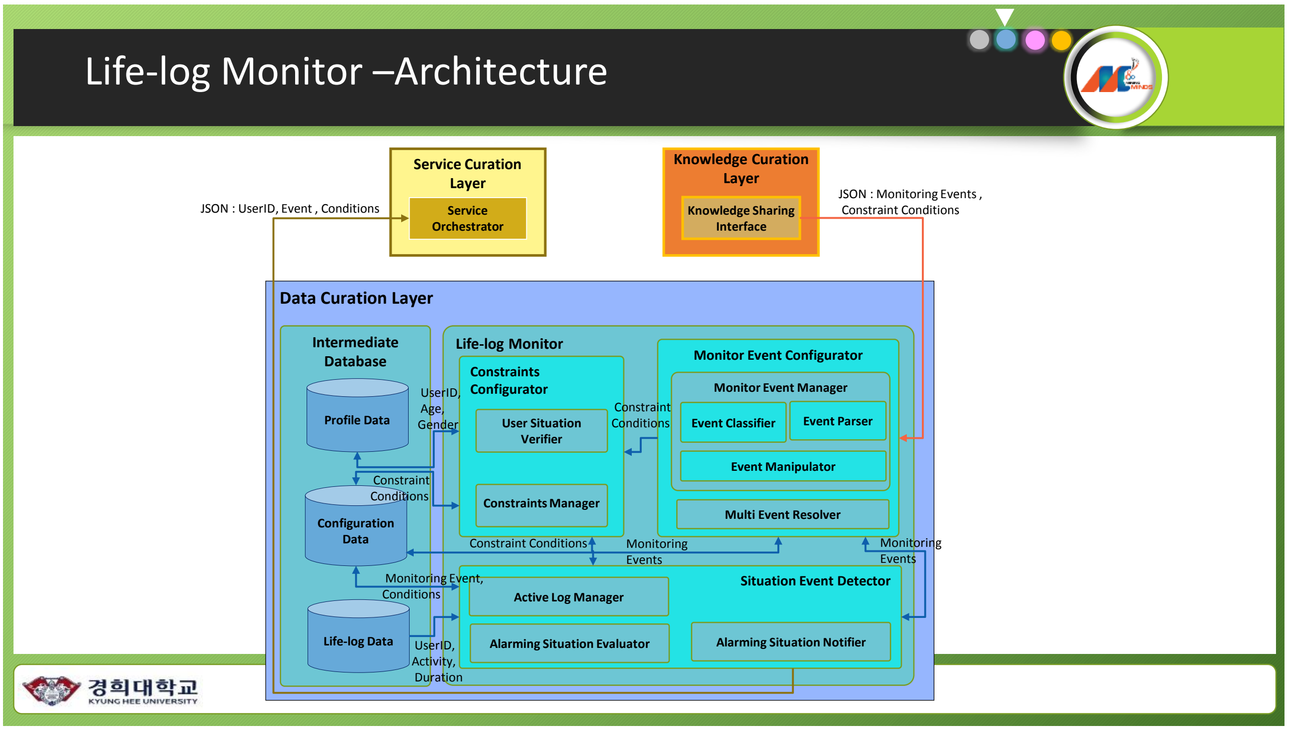
https://en.wikipedia.org/wiki/DARPA_LifeLog

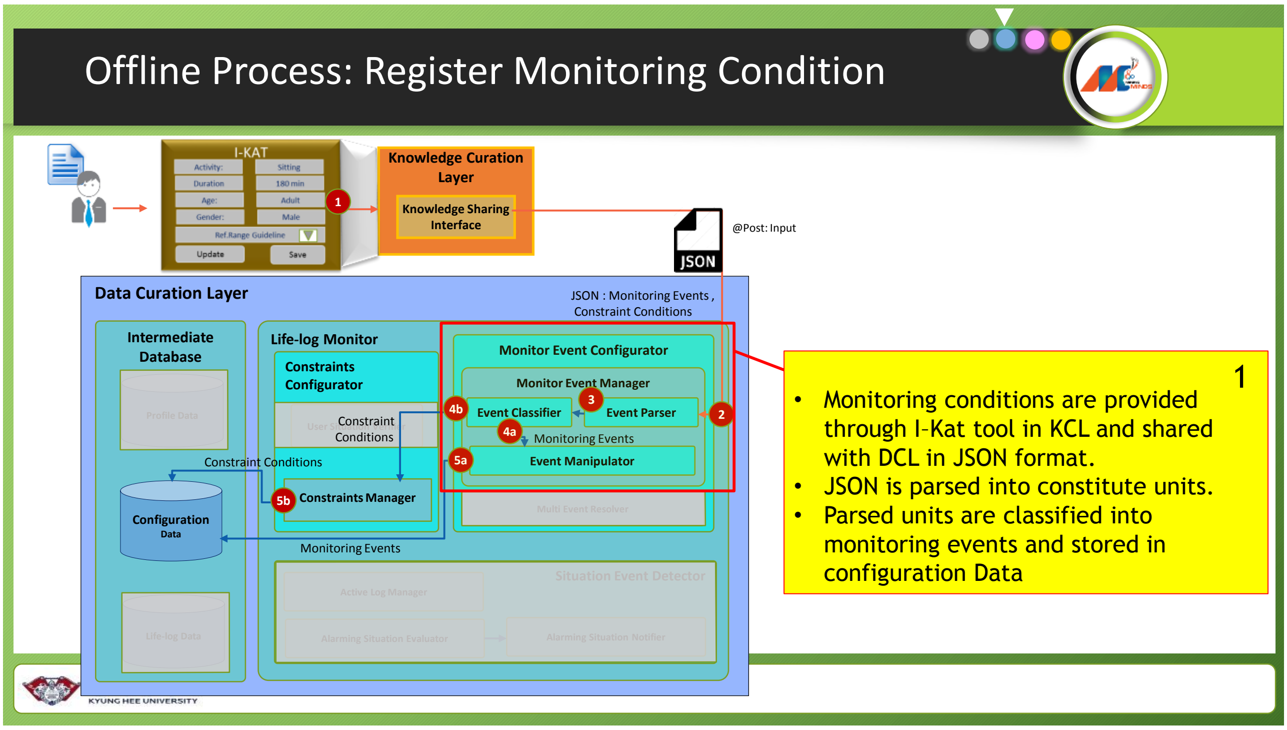
Motivation

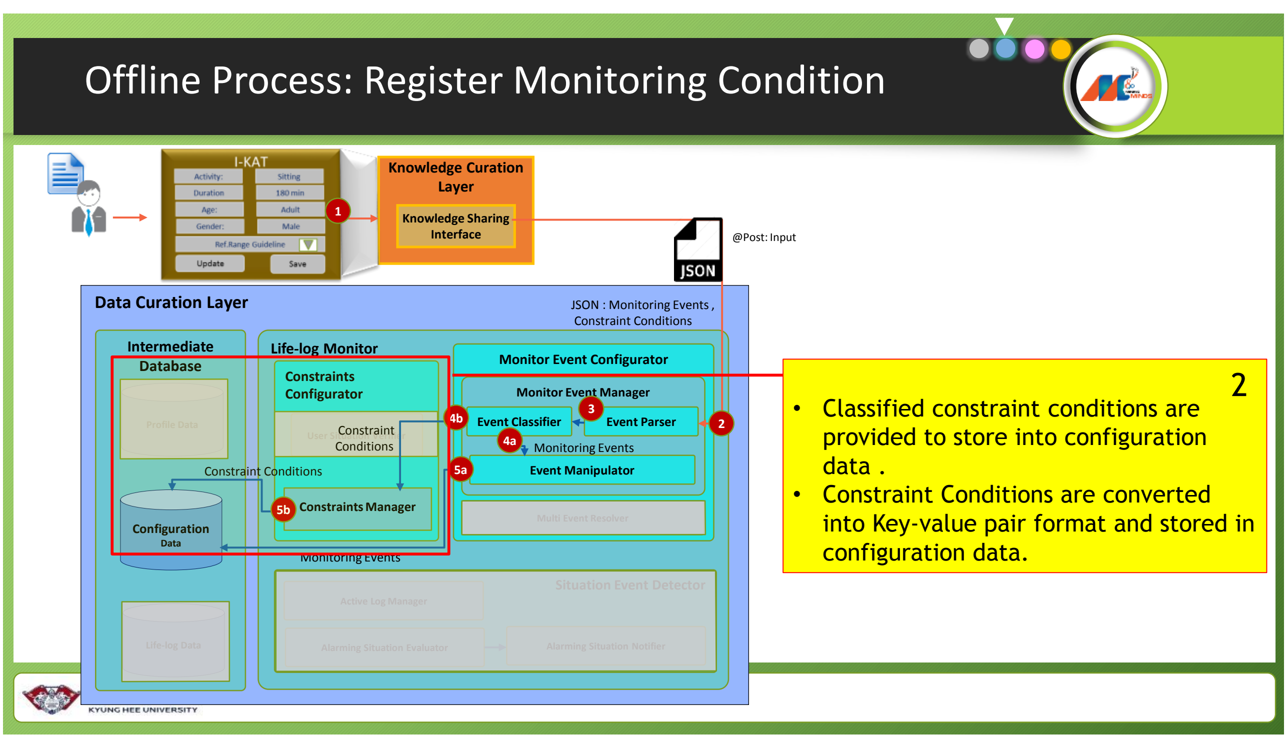
- Recognize the event from stream of life-log
  - Identification of *target* events for efficient monitoring
- Dynamically accommodate critical condition for monitoring
  - Accommodate *critical situations* on the basis of *expert's opinion.*
- Detect the alarming condition from stream of life-log to notify
  - *Evaluate* the alarming situation *with respect to user context and generate intimation.*

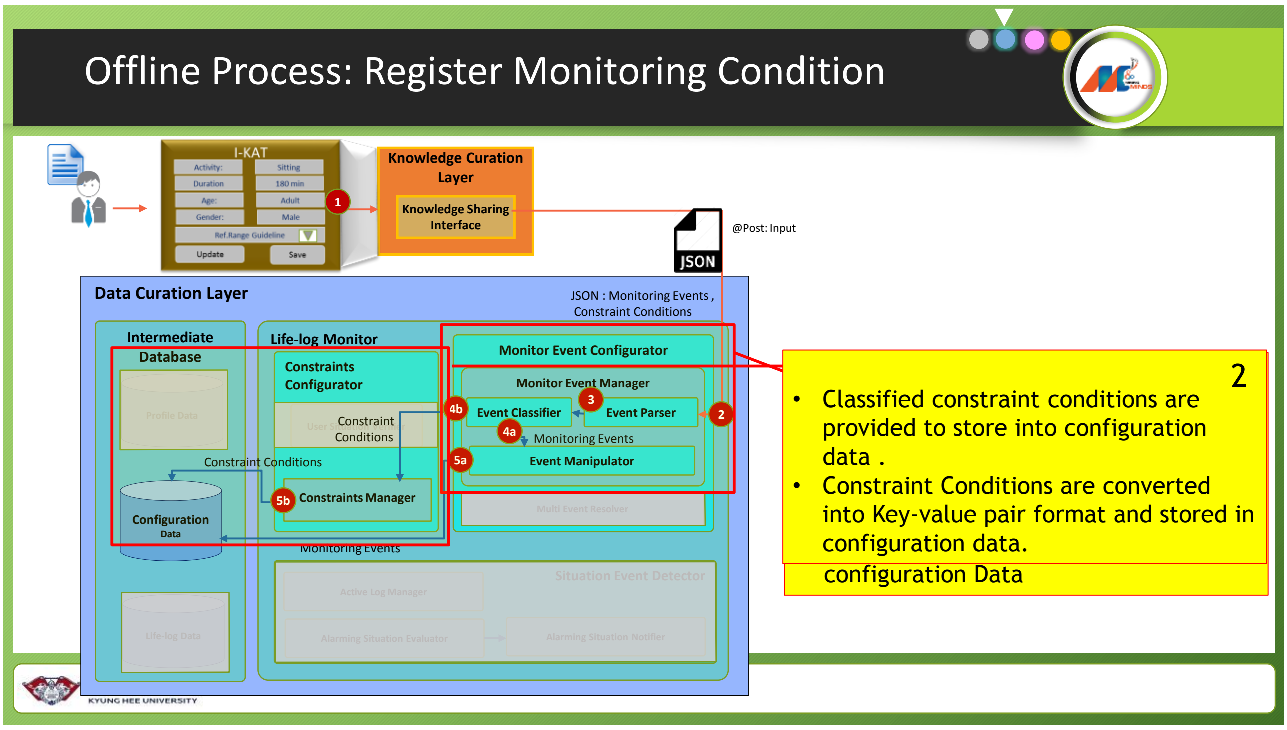


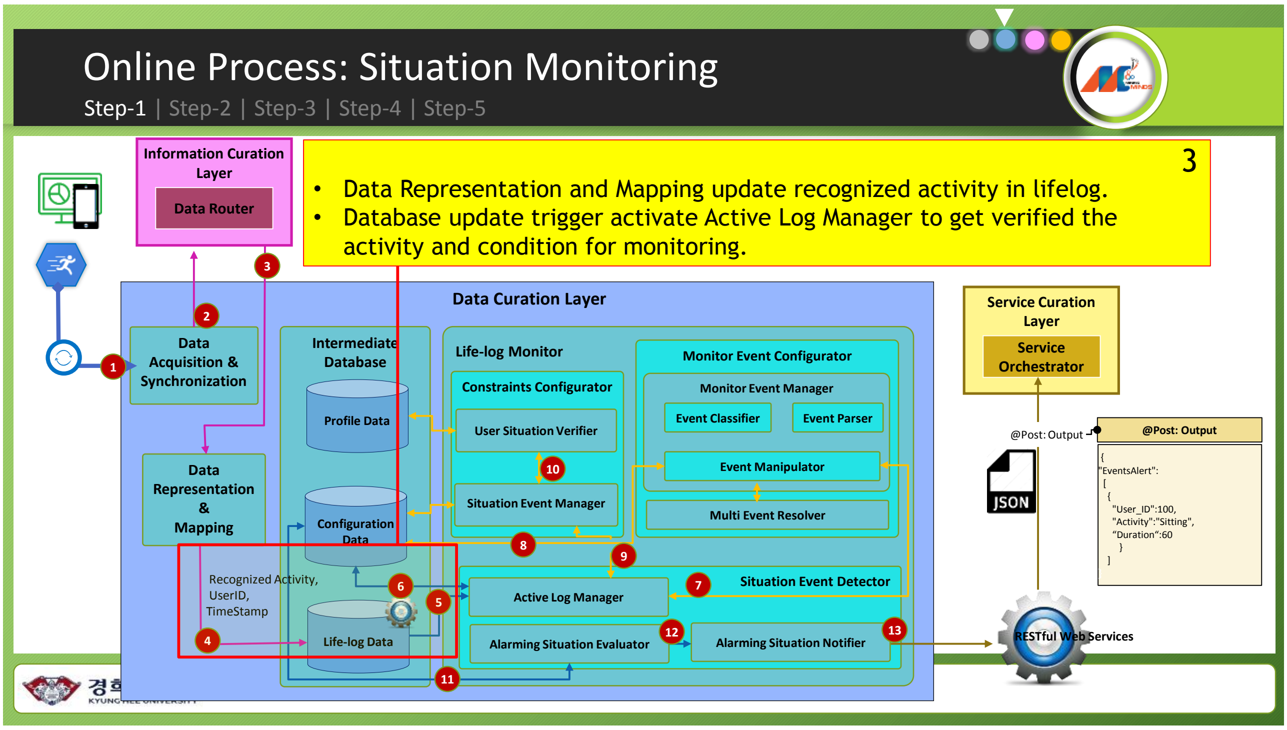

Life-log Monitoring – Conceptual View
Offline Process
Knowledge Curation Layer
Configuration: Monitoring Events & Constraints
Intermediate database
Configuration Data
Monitoring Events & Constraints
Information Curation Layer
Low Level Data
High Level Context
Data Acquisition & Synchronization
Life-log Representation & Mapping
User, Event
Life-log
Life-log Monitor
Filtered Events: Satisfied User Events
Notify: Alarming User Situation
Service Curation Layer
Online Process
경희대학교
KYUNG HEE UNIVERSITY

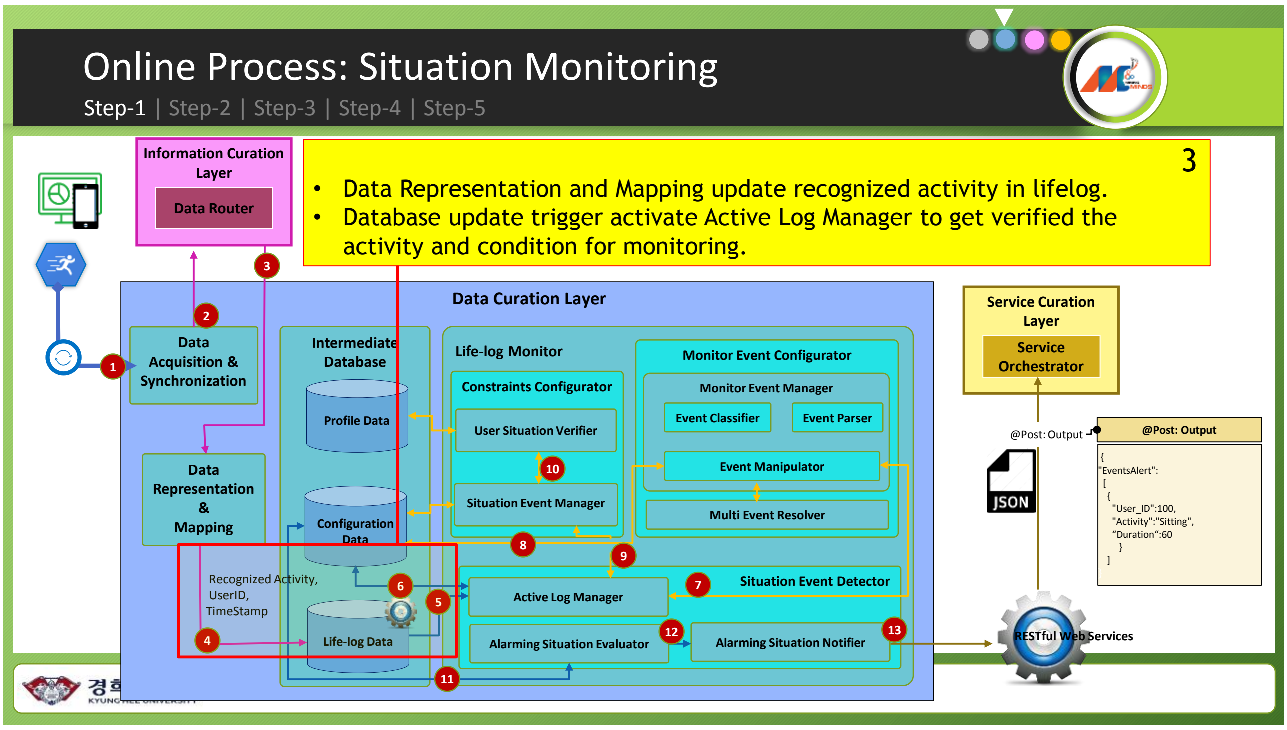

Life-log Monitor –Architecture
Service Curation Layer
Service Orchestrator
JSON : UserID, Event , Conditions
Knowledge Curation Layer
Knowledge Sharing Interface
JSON : Monitoring Events , Constraint Conditions
Data Curation Layer
Intermediate Database
Profile Data
Configuration Data
Life-log Data
UserID, Age, Gender
Constraint Conditions
Monitoring Event, Conditions
UserID, Activity, Duration
Life-log Monitor
Constraints Configurator
User Situation Verifier
Constraints Manager
Constraint Conditions
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Event Manipulator
Multi Event Resolver
Constraint Conditions
Monitoring Events
Monitoring Events
Situation Event Detector
Active Log Manager
Alarming Situation Evaluator
Alarming Situation Notifier
경희대학교
KYUNG HEE UNIVERSITY

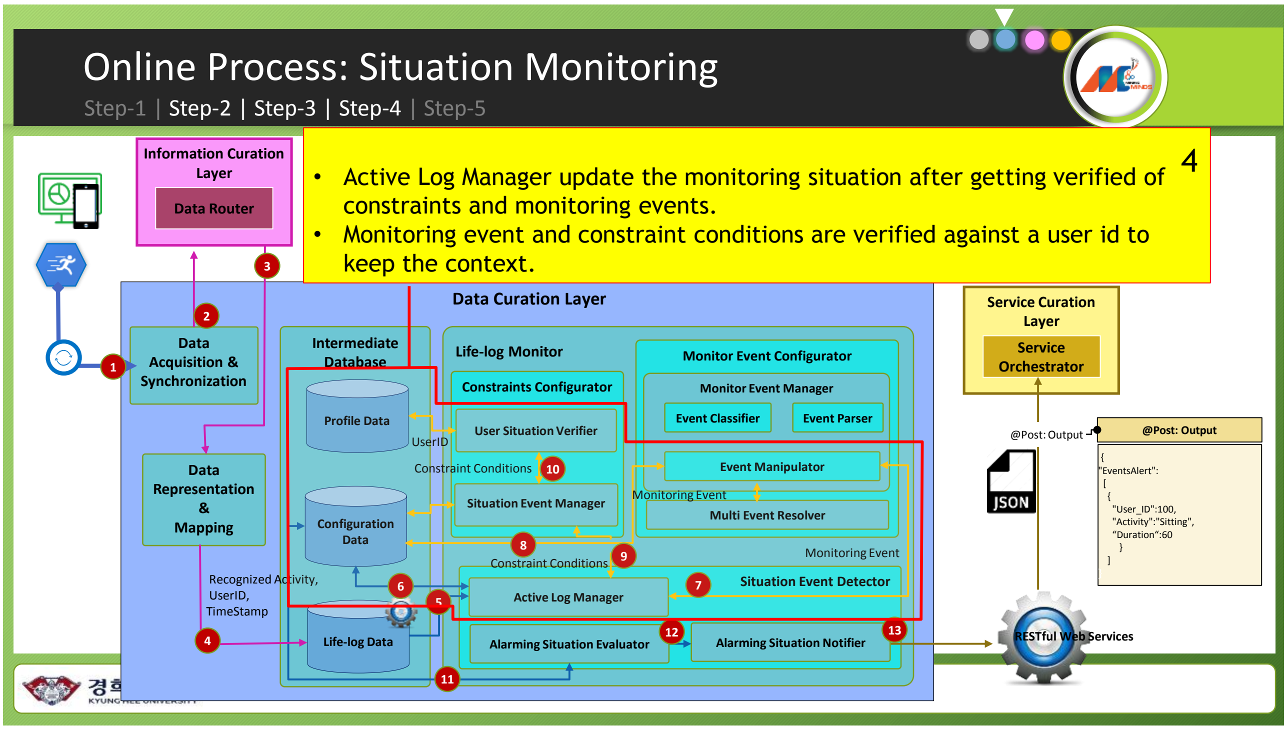

Offline Process: Register Monitoring Condition
I-KAT
Activity:
Sitting
Duration
180 min
Age:
Adult
Gender:
Male
Ref.Range Guideline
Update
Save
Knowledge Curation Layer
Knowledge Sharing Interface
@Post: Input
JSON
Data Curation Layer
JSON : Monitoring Events , Constraint Conditions
Intermediate Database
Life-log Monitor
Constraints Configurator
Constraint Conditions
Constraint Conditions
Configuration Data
Constraints Manager
Monitoring Events
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Monitoring Events
Event Manipulator
1
Monitoring conditions are provided through I-Kat tool in KCL and shared with DCL in JSON format.
JSON is parsed into constitute units.
Parsed units are classified into monitoring events and stored in configuration Data
KYUNG HEE UNIVERSITY

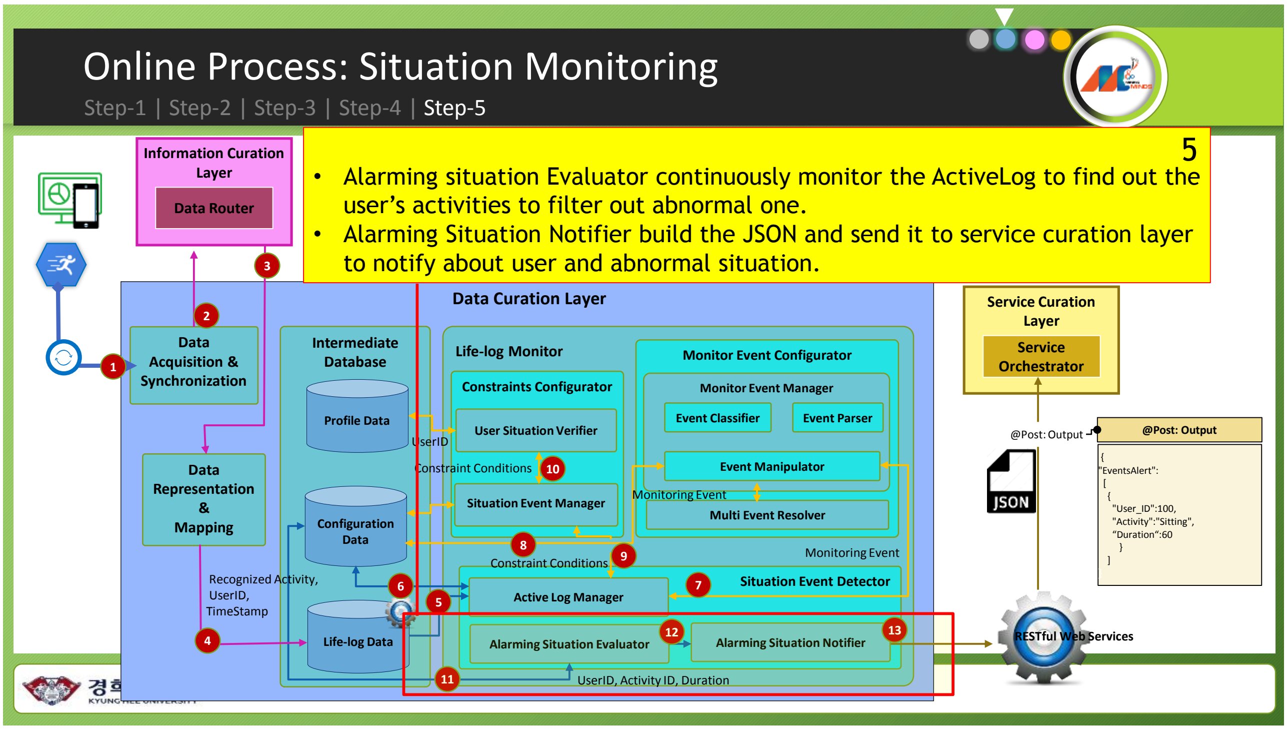

Offline Process: Register Monitoring Condition
I-KAT
Activity:
Sitting
Duration
180 min
Age:
Adult
Gender:
Male
Ref.Range Guideline
Update
Save
Knowledge Curation Layer
Knowledge Sharing Interface
@Post: Input
JSON
Data Curation Layer
JSON : Monitoring Events , Constraint Conditions
Intermediate Database
Life-log Monitor
Constraints Configurator
Constraint Conditions
Constraint Conditions
Configuration Data
Constraints Manager
Monitoring Events
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Monitoring Events
Event Manipulator
2
Classified constraint conditions are provided to store into configuration data .
Constraint Conditions are converted into Key-value pair format and stored in configuration data.
KYUNG HEE UNIVERSITY

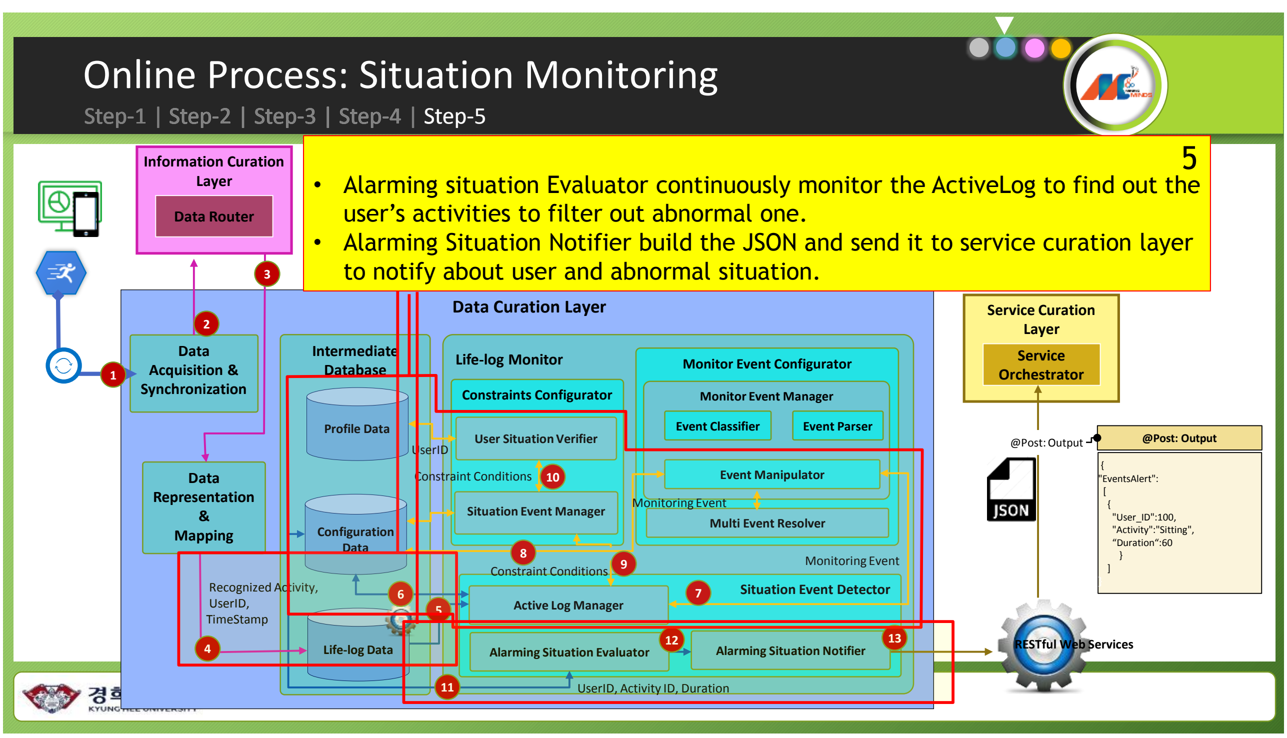

Offline Process: Register Monitoring Condition
I-KAT
Activity: Sitting
Duration 180 min
Age: Adult
Gender: Male
Ref.Range Guideline
Update
Save
Knowledge Curation Layer
Knowledge Sharing Interface
JSON
@Post: Input
Data Curation Layer
JSON : Monitoring Events , Constraint Conditions
Intermediate Database
Profile Data
Configuration Data
Life-log Data
Life-log Monitor
Constraints Configurator
Constraint Conditions
Constraints Manager
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Monitoring Events
Event Manipulator
Multi Event Resolver
Situation Event Detector
Active Log Manager
Alarming Situation Evaluator
Alarming Situation Notifier
2
Classified constraint conditions are provided to store into configuration data .
Constraint Conditions are converted into Key-value pair format and stored in configuration data.
configuration Data
KYUNG HEE UNIVERSITY


Online Process: Situation Monitoring
Step-1 | Step-2 | Step-3 | Step-4 | Step-5
Information Curation Layer
Data Router
3
Data Representation and Mapping update recognized activity in lifelog.
Database update trigger activate Active Log Manager to get verified the activity and condition for monitoring.
Data Curation Layer
Data Acquisition & Synchronization
Data Representation & Mapping
Recognized Activity, UserID, TimeStamp
Intermediate Database
Profile Data
Configuration Data
Life-log Data
Life-log Monitor
Constraints Configurator
User Situation Verifier
Situation Event Manager
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Event Manipulator
Multi Event Resolver
Situation Event Detector
Active Log Manager
Alarming Situation Evaluator
Alarming Situation Notifier
Service Curation Layer
Service Orchestrator
@Post: Output
JSON
{
"EventsAlert":
[
{
"User_ID":100,
"Activity":"Sitting",
"Duration":60
}
]
RESTful Web Services
KYUNG HEE UNIVERSITY


Online Process: Situation Monitoring
Step-1 | Step-2 | Step-3 | Step-4 | Step-5
3
Data Representation and Mapping update recognized activity in lifelog.
Database update trigger activate Active Log Manager to get verified the activity and condition for monitoring.
Information Curation Layer
Data Router
Data Curation Layer
Data Acquisition & Synchronization
Data Representation & Mapping
Intermediate Database
Profile Data
Configuration Data
Life-log Data
Recognized Activity, UserID, TimeStamp
Life-log Monitor
Constraints Configurator
User Situation Verifier
Situation Event Manager
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Event Manipulator
Multi Event Resolver
Active Log Manager
Situation Event Detector
Alarming Situation Evaluator
Alarming Situation Notifier
Service Curation Layer
Service Orchestrator
@Post: Output
JSON
RESTful Web Services
1
2
3
4
5
6
7
8
9
10
11
12
13

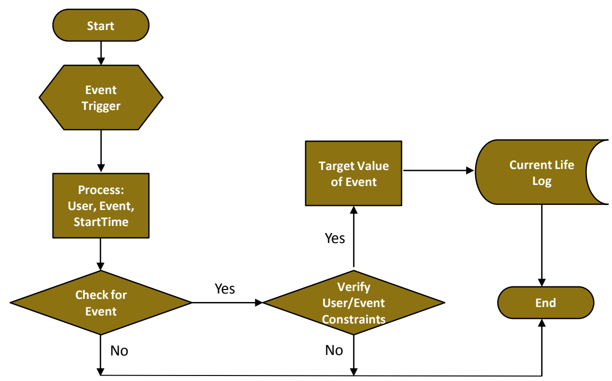

Online Process: Situation Monitoring
Step-1 | Step-2 | Step-3 | Step-4 | Step-5
4
Active Log Manager update the monitoring situation after getting verified of constraints and monitoring events.
Monitoring event and constraint conditions are verified against a user id to keep the context.
Information Curation Layer
Data Router
Data Curation Layer
Data Acquisition & Synchronization
Data Representation & Mapping
Intermediate Database
Profile Data
Configuration Data
Life-log Data
Recognized Activity, UserID, TimeStamp
UserID
Constraint Conditions
Monitoring Event
Life-log Monitor
Constraints Configurator
User Situation Verifier
Situation Event Manager
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Event Manipulator
Multi Event Resolver
Active Log Manager
Situation Event Detector
Alarming Situation Evaluator
Alarming Situation Notifier
Service Curation Layer
Service Orchestrator
@Post: Output
JSON
RESTful Web Services
1
2
3
4
5
6
7
8
9
10
11
12
13


Online Process: Situation Monitoring
Step-1 | Step-2 | Step-3 | Step-4 | Step-5
5
• Alarming situation Evaluator continuously monitor the ActiveLog to find out the user's activities to filter out abnormal one.
• Alarming Situation Notifier build the JSON and send it to service curation layer to notify about user and abnormal situation.
Information Curation Layer
Data Router
Data Curation Layer
Data Acquisition & Synchronization
Data Representation & Mapping
Recognized Activity, UserID, TimeStamp
Intermediate Database
Profile Data
Configuration Data
Life-log Data
Life-log Monitor
Constraints Configurator
User Situation Verifier
UserID
Constraint Conditions
Situation Event Manager
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Event Manipulator
Monitoring Event
Multi Event Resolver
Constraint Conditions
Monitoring Event
Active Log Manager
Situation Event Detector
Alarming Situation Evaluator
Alarming Situation Notifier
UserID, Activity ID, Duration
Service Curation Layer
Service Orchestrator
@Post: Output
@Post: Output
JSON
RESTful Web Services
1
2
3
4
5
6
7
8
9
10
11
12
13


Online Process: Situation Monitoring
Step-1 | Step-2 | Step-3 | Step-4 | Step-5
5
• Alarming situation Evaluator continuously monitor the ActiveLog to find out the user's activities to filter out abnormal one.
• Alarming Situation Notifier build the JSON and send it to service curation layer to notify about user and abnormal situation.
Information Curation Layer
Data Router
Data Curation Layer
Data Acquisition & Synchronization
Data Representation & Mapping
Recognized Activity, UserID, TimeStamp
Intermediate Database
Profile Data
Configuration Data
Life-log Data
Life-log Monitor
Constraints Configurator
User Situation Verifier
UserID
Constraint Conditions
Situation Event Manager
Monitor Event Configurator
Monitor Event Manager
Event Classifier
Event Parser
Event Manipulator
Monitoring Event
Multi Event Resolver
Constraint Conditions
Monitoring Event
Active Log Manager
Situation Event Detector
Alarming Situation Evaluator
Alarming Situation Notifier
UserID, Activity ID, Duration
Service Curation Layer
Service Orchestrator
@Post: Output
@Post: Output
JSON
RESTful Web Services
1
2
3
4
5
6
7
8
9
10
11
12
13

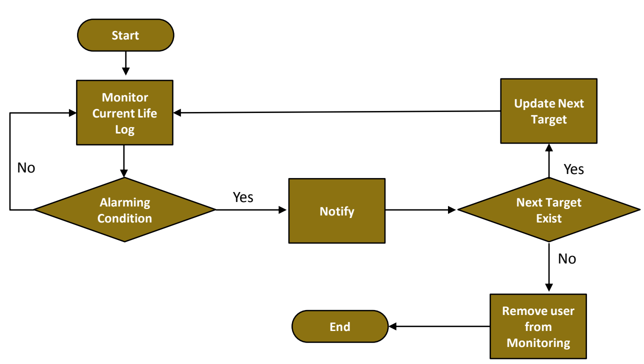
Situation Detection- Algorithm

- **Situation Detection**
  - **Step 1 :Population of Current Life log**
    - Activation: On-Insert Trigger on life-log
    - Retrieval of User, event and Starttime
    - Checking Event for Monitoring
    - Verification of user against event's constraints
    - Retrieval of target value of event
    - Populate current life log with user, event, starttime, and target value.



Situation Detection- Algorithm

- **Situation Detection**
  - **Step 2 : Monitoring of Current Life log**
    - Monitoring: Time Based
    - Filter users whose Alarming condition meet.
    - Notify the user (Just display)
    - Search next available target condition of event
    - Update current lifelog with updated target value of event
    - If no further target value then remove of user from monitoring



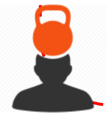
Contributions

- Monitoring of Life-log for occurrence of *alarming situation* in personal life in real time manner.
- Accommodate dynamically *constraints of events* on the recommendation of experts.
- Configuration of Life-log monitor with *contributing factors* / *target variables*.

경희대학교
KYUNG HEE UNIVERSITY

Challenges & Contributions

- Challenges
  - **Real time** monitoring of activities
  - Monitor the **selected** activities of **specific** users.
  - Intimation for **different situations** against **multiple activities** of users at a time.
  - Monitor the **correlated activities** for complex situation i.e nutrition monitoring for sedentary person



- Contributions
  - **Key-value** pair style to manage diverse type of activities with monitoring conditions.
  - **Runtime** identification of monitor-able activities of selected users.
  - Monitoring of **multiple activities** with multiple situation for users at run time.
  - Monitoring of **complex situation**.

경희대학교
KYUNG HEE UNIVERSITY

Mining Minds

경희대학교 KYUNG HEE UNIVERSITY KITECH POSTECH UNIVERSITY of TASMANIA 녹십자헬스케어 TAPACROSS U2 system User & Usability system KOCSA

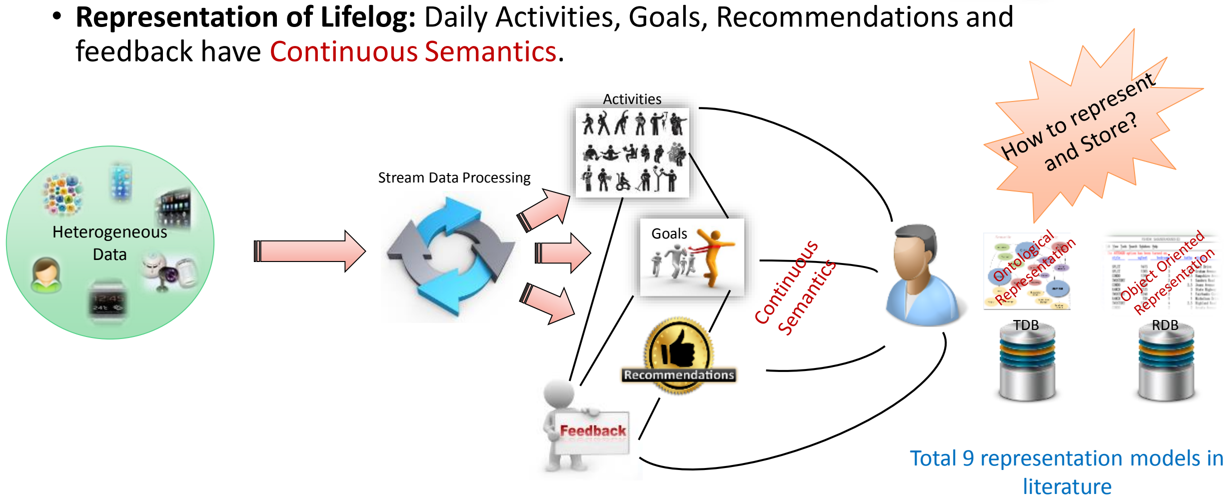
Agenda

- Introduction
- Related Work
- Comparison b/w Object Oriented and Ontological Representation
- Motivation
- Challenges and Solutions
- Proposed Architecture
- Workflow
- Implementation Details
- Use of Object-Relational Mapping (ORM) and RDBMS
- Contributions
- References

경희대학교 KYUNG HEE UNIVERSITY

Introduction

- **Representation of Lifelog:** Daily Activities, Goals, Recommendations and feedback have Continuous Semantics.



Related Work

Graphical Representation [1], [2]

Features

- Diagrammatic Representation at design time
- Focus on issues of formality and generality
- Cover diversity, quality and complex relationships

Limitations

- Needs to translate into usable format
- Limitation of privacy and distribution of data

Logic Based Representation [3]

Features

- Adding data as facts and using expressions or rules
- Sensed data represent in First Order Predicate Logic
- Also represent the meta-propositional properties

Limitations

- Model does not support design and documentation of design process
- The sensed data cannot be used by software tool

Ontological Representation [4], [5], [6], [7], [16], [17]

Features

- Ontology based models for data representation
- Focus on interoperability and uniform format of data
- Store two types of knowledge, Domain and generic knowledge
- Ontological model using case-based reasoning
- Feedback system for verification and falsification

Limitations

- Face challenges of abductive reasoning
- No logical reasoning on different hypothesis
- Facing challenges to exploring temporal and spatial inferences
- No scalability in design model

경희대학교
KYUNG HEE UNIVERSITY

Related Work

Tuple-Based Representation [8], [9]

Features

- Using key-value pair storage system
- Contains number of attributes and corresponding values as a single record
- System gave special emphasis to group activity and data access rights
- Data stores as tuple {ID, Subject, State, Time, Lag, Interval}

Limitations

- Limitation of capturing variety of data
- No Consistency checking
- No reasoning support

Hierarchical Representation [10]

Features

- Represent data in structure like tree
- Enhance the efficiency of searching data
- Stores data of Identity, Location, Status and Time

Limitations

- Static data model for storing specific type of data
- Cannot store the sensed data from environment directly

Domain-Focused Representation [11]

Features

- General data model to represent the facts
- Provides ease of querying, processing and adaptation of incomplete data
- Stored data in 4Ws (Who, What, Where, When)

Limitations

- Less expressiveness
- Less flexibility
- Needs to improve robustness and adaptability

경희대학교 KYUNG HEE UNIVERSITY

Related Work

Spatial Representation [12]

Features

- Stored the spatial data for Geographical Information System
- Five tiers of ontology was proposed for GIS
- handle the occurrence of exceptional situations in consistency constraints

Limitations

- It needs more efforts to gather the location data
- It is hard to keep the location data up to date

Hybrid Representation [13]

Features

- Integrate different existing representation model
- Integration of ontological, tuple based and logic based representation
- Due to ontological model, it has clear advantages of interoperability and heterogeneity

Limitations

- Inappropriate choices that arise from reliance on flawed data that reflect the user's requirements
- Inappropriate privacy
- Preferences exposed to users

Object Oriented Representation [14], [15]

Features

- Data stores as object oriented principles
- It followed object's modelling entities like People, Places and Objects
- Person, place, thing and patient are the main modelled entities

Limitations

- Limitation of interoperability
- Use third party system to overcome this limitation

경희대학교 KYUNG HEE UNIVERSITY

Comparison b/w Object Oriented and Ontological Model

Object Oriented Representation

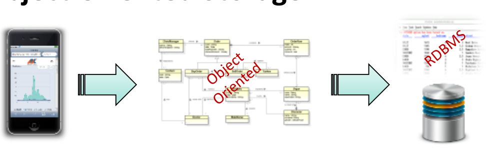
- Pros
  - Data stores as object oriented principals
  - Allows "Real World" to be modeled more closely
  - High extensibility and scalability
  - Support for schema evolution
  - High Performance (Time)
- Cons
  - Lack of universal data model
  - Lack of standards
  - Highly complex to manage
  - Less expressive

Ontological Representation

- Pros
  - Data stores as classes, attributes and individuals with semantics
  - Improve reusability and interoperability
  - More expressive
  - Improvement on searches
  - Permit inferences
  - High extensibility and scalability
- Cons
  - Ontology creation is difficult
  - Low performance (Time)
  - Challenges in abductive reasoning

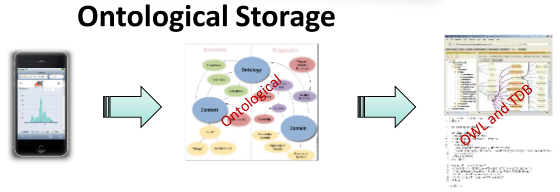
Comparison b/w Object Oriented and Ontological Storage

Object Oriented Storage



- Consists of DB Schema and DB Data
- Close world assumption (CWA)
  - Missing information treated as false
- Unique name assumption (UNA)
  - Each individual has a unique name
- Schema behaves as constraints on structure of data

Ontological Storage

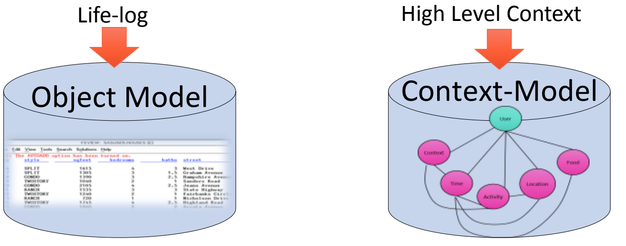


- Consists of axioms (Structure of model) and facts (Concrete Situation)
- Open world assumption (OWA)
  - Missing information treated as unknown
- No-Unique name assumption (No-UNA)
  - Individuals may have more than one name
- Ontology axioms behave like implications (inference rules)

Discussion: Trade off between Object model and ontological model

- It has decided that due to performance issue we will represent and store only the context information (High Level Context) in ontological format while the remaining whole Life-log will be represent and store in the object model format using relational databases.

Solution to performance issue



- There is trade off between performance of reasoning and scalability of heterogeneous source data.
- In ontological format the performance will be low and scalability will be high.
- In object model the performance will be high and scalability will be low.

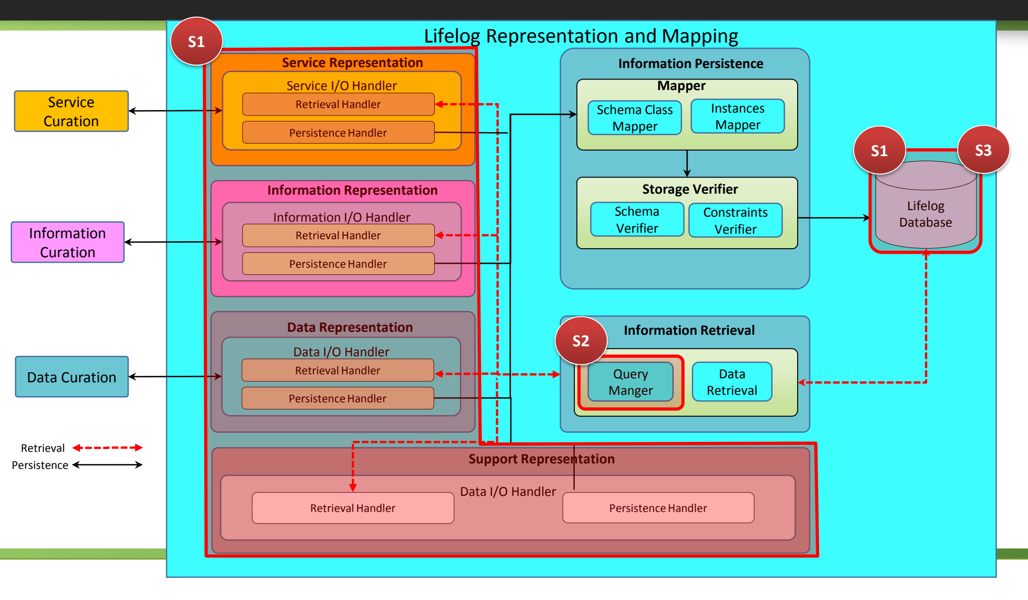
Motivation

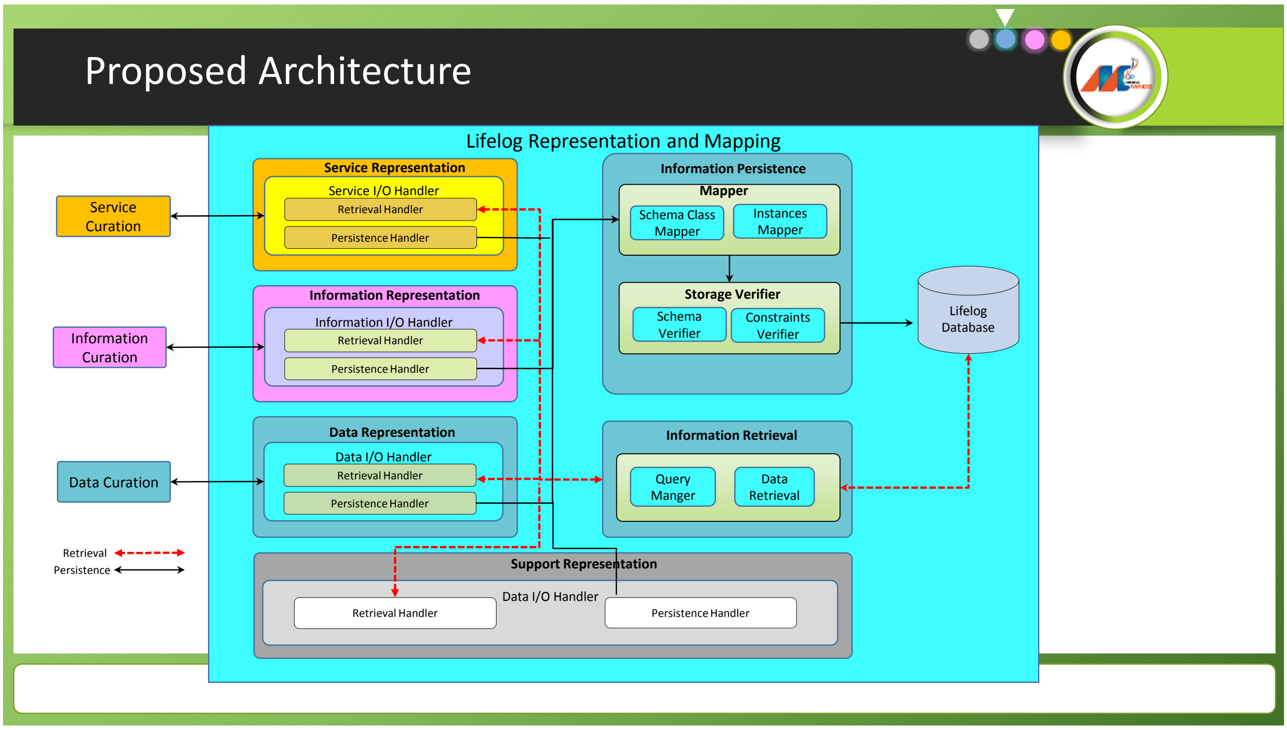
Heterogeneous data Processing Streams	Providing extensible and understandable structured format to variety of data.
Fine-grained Information access	Processable and procedural format to all layers and modules of MM Platform
Extensible Data Model	Integratable new models based on the different input data sources

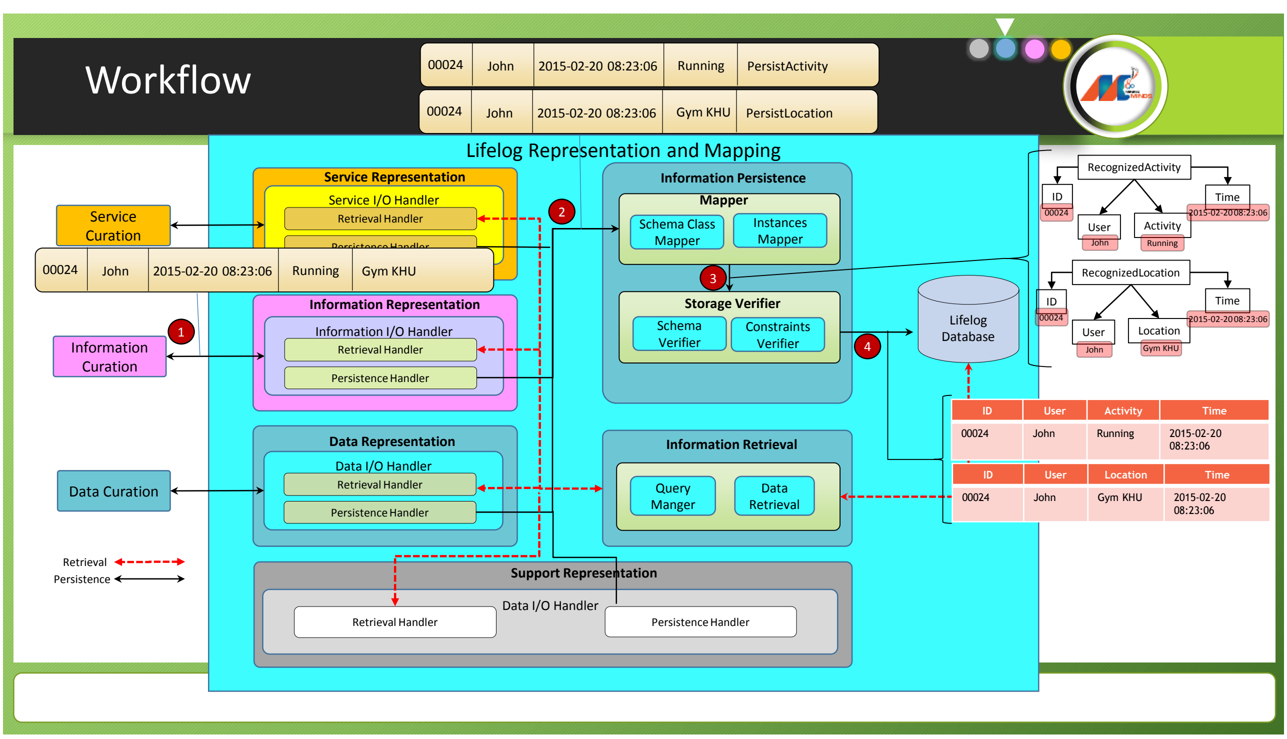
Challenges and Solutions

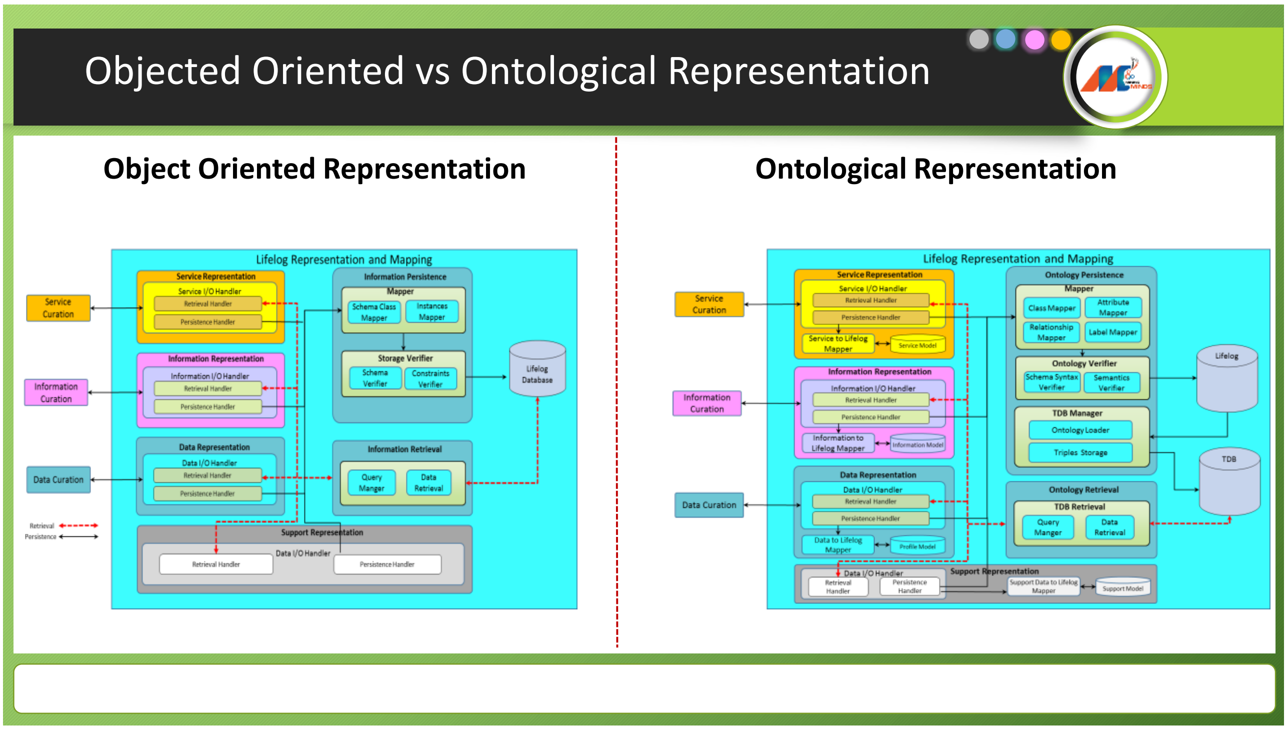
Motivations	Challenges	Solutions	
Heterogeneous data Processing Streams	• Continuous data identification and storage	Representation Models, Lifelog Database	S1
Fine-grained Information access	• Dynamic query generations for different layers	Query Manager	S2
Extensible Data Model	• Design and development of consistent data models for different sources of data	Lifelog Database	S3

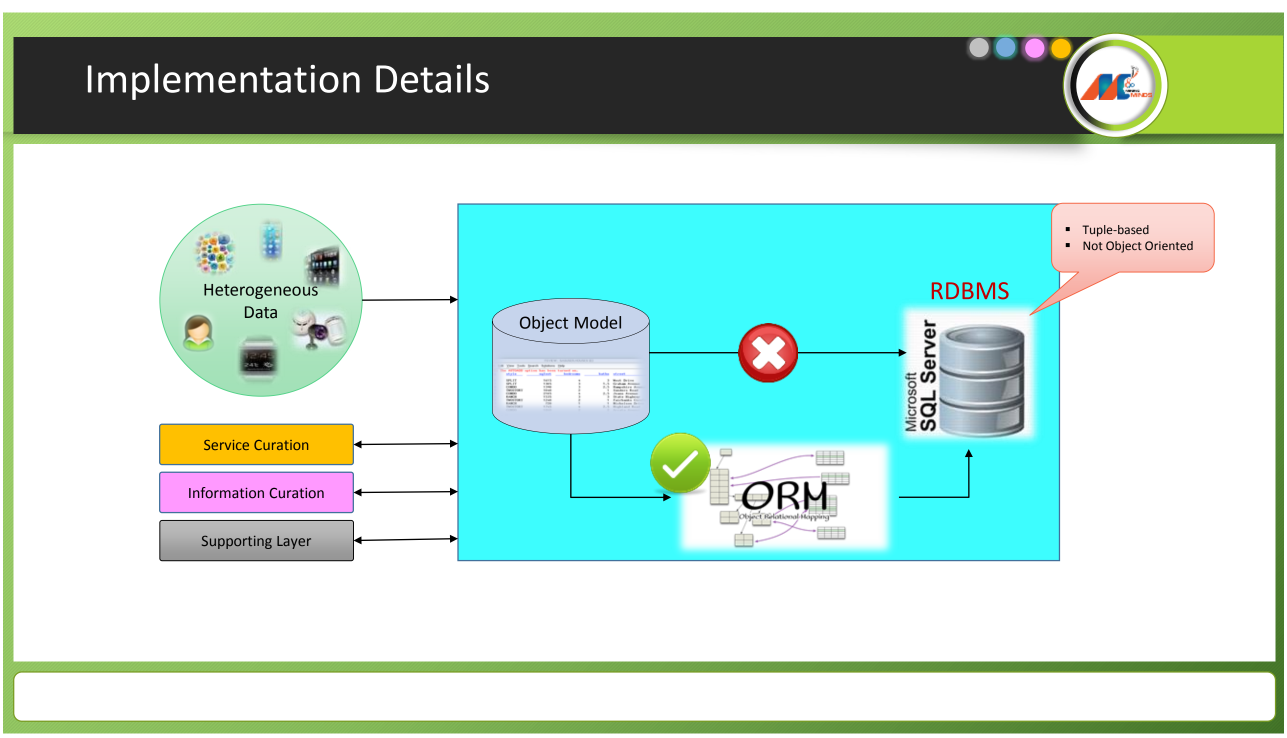
Proposed Architecture and solutions



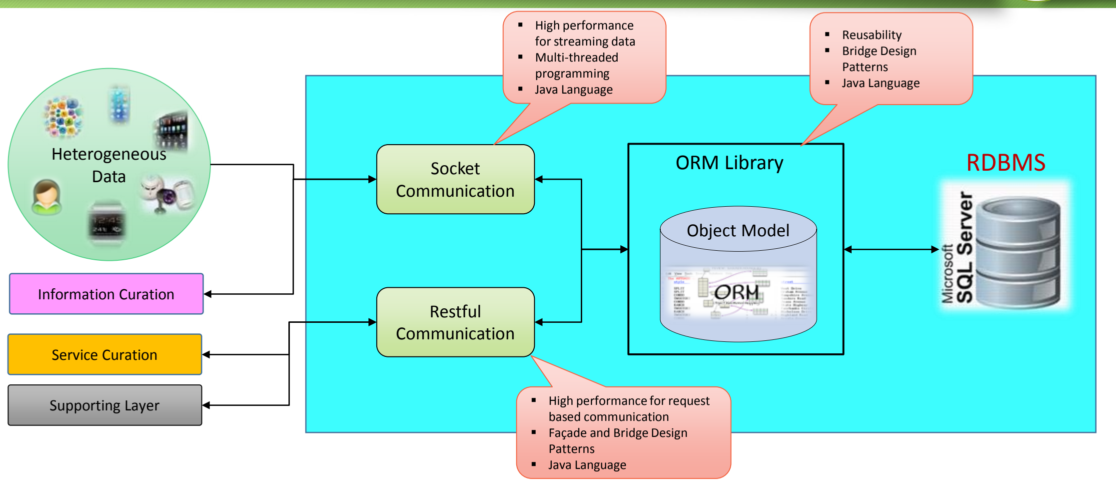

Proposed Architecture
Lifelog Representation and Mapping
Service Representation
Service I/O Handler
Retrieval Handler
Persistence Handler
Information Representation
Information I/O Handler
Retrieval Handler
Persistence Handler
Data Representation
Data I/O Handler
Retrieval Handler
Persistence Handler
Information Persistence
Mapper
Schema Class Mapper
Instances Mapper
Storage Verifier
Schema Verifier
Constraints Verifier
Information Retrieval
Query Manger
Data Retrieval
Lifelog Database
Support Representation
Data I/O Handler
Retrieval Handler
Persistence Handler
Service Curation
Information Curation
Data Curation
Retrieval
Persistence


Workflow
00024 John 2015-02-20 08:23:06 Running PersistActivity
00024 John 2015-02-20 08:23:06 Gym KHU PersistLocation
00024 John 2015-02-20 08:23:06 Running Gym KHU
Lifelog Representation and Mapping
Service Representation
Service I/O Handler
Retrieval Handler
Information Representation
Information I/O Handler
Retrieval Handler
Persistence Handler
Data Representation
Data I/O Handler
Retrieval Handler
Persistence Handler
Information Persistence
Mapper
Schema Class Mapper
Instances Mapper
Storage Verifier
Schema Verifier
Constraints Verifier
Information Retrieval
Query Manger
Data Retrieval
Lifelog Database
Support Representation
Data I/O Handler
Retrieval Handler
Persistence Handler
Service Curation
Information Curation
Data Curation
Retrieval
Persistence
1
2
3
4
RecognizedActivity
ID
User
Activity
Time
RecognizedLocation
ID
User
Location
Time
ID User Activity Time
00024 John Running 2015-02-20 08:23:06
ID User Location Time
00024 John Gym KHU 2015-02-20 08:23:06


Objected Oriented vs Ontological Representation
Object Oriented Representation
Ontological Representation
Lifelog Representation and Mapping
Service Curation
Information Curation
Data Curation
Service Representation
Service I/O Handler
Retrieval Handler
Persistence Handler
Information Representation
Information I/O Handler
Data Representation
Data I/O Handler
Information Persistence
Mapper
Schema Class Mapper
Instances Mapper
Storage Verifier
Schema Verifier
Constraints Verifier
Lifelog Database
Information Retrieval
Query Manger
Data Retrieval
Support Representation
Retrieval
Persistence
Service to Lifelog Mapper
Service Model
Information to Lifelog Mapper
Information Model
Data to Lifelog Mapper
Profile Model
Ontology Persistence
Class Mapper
Attribute Mapper
Relationship Mapper
Label Mapper
Ontology Verifier
Schema Syntax Verifier
Semantics Verifier
TDB Manager
Ontology Loader
Triples Storage
Lifelog
TDB
Ontology Retrieval
TDB Retrieval
Support Data to Lifelog Mapper
Support Model


Implementation Details
Heterogeneous Data
Service Curation
Information Curation
Supporting Layer
Object Model
RDBMS
Tuple-based
Not Object Oriented
Microsoft SQL Server
ORM
Object Relational Mapping

Use of ORM (Object-Relational Mapping)



Object Oriented Design Patterns

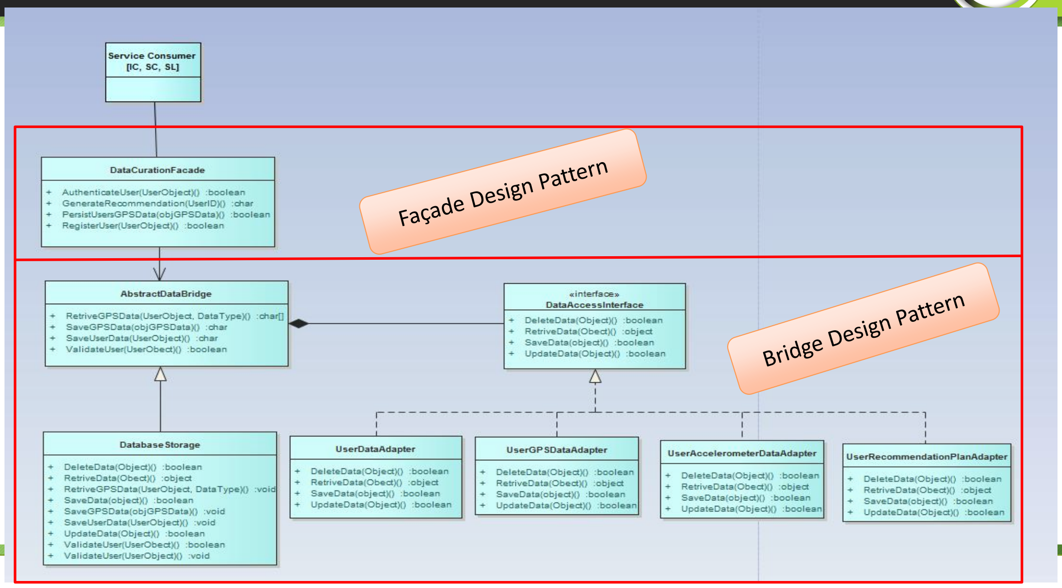
Façade Design Pattern

- Provides a unified interface to multiple inner interfaces
- Software Library easy to use, understand and test
- Library become more readable
- Reduce the dependencies of outer code on inner library classes
- Encapsulate the inner objects and methods of library
- Wrap a complicated sub systems with unified interface

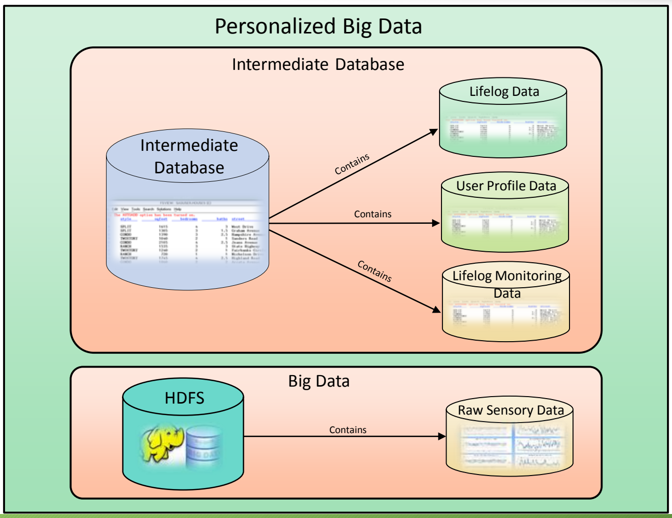
Bridge Design Pattern

- Scalability for data storage
- Decouple the abstraction from its implementations
- Divide the responsibilities among different classes
- Hide details from client

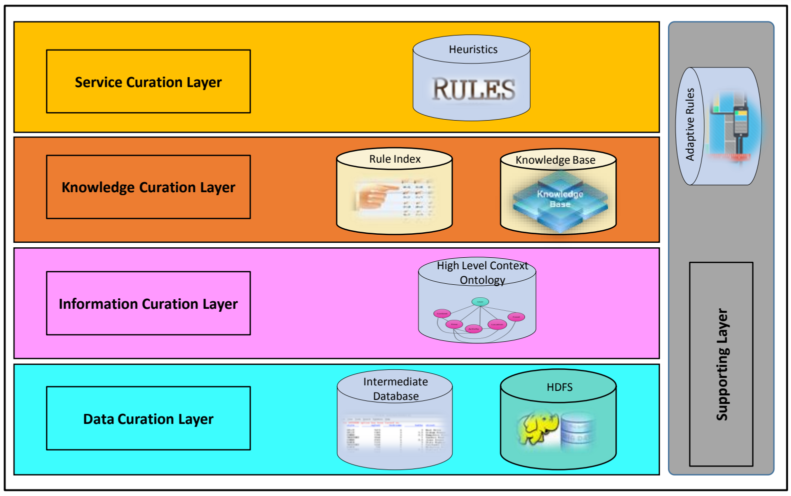
Object Oriented Design Patterns



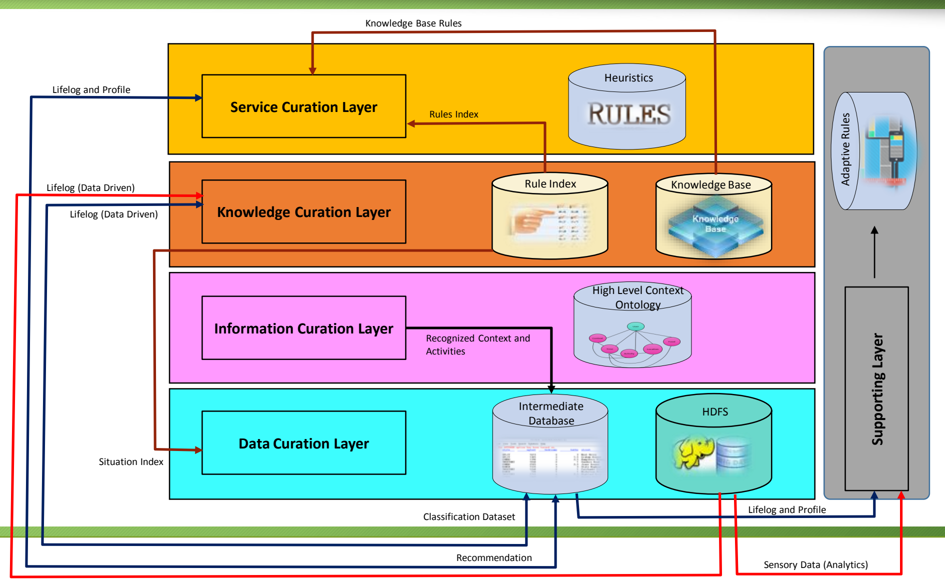
Distributed Database Terminology and definition

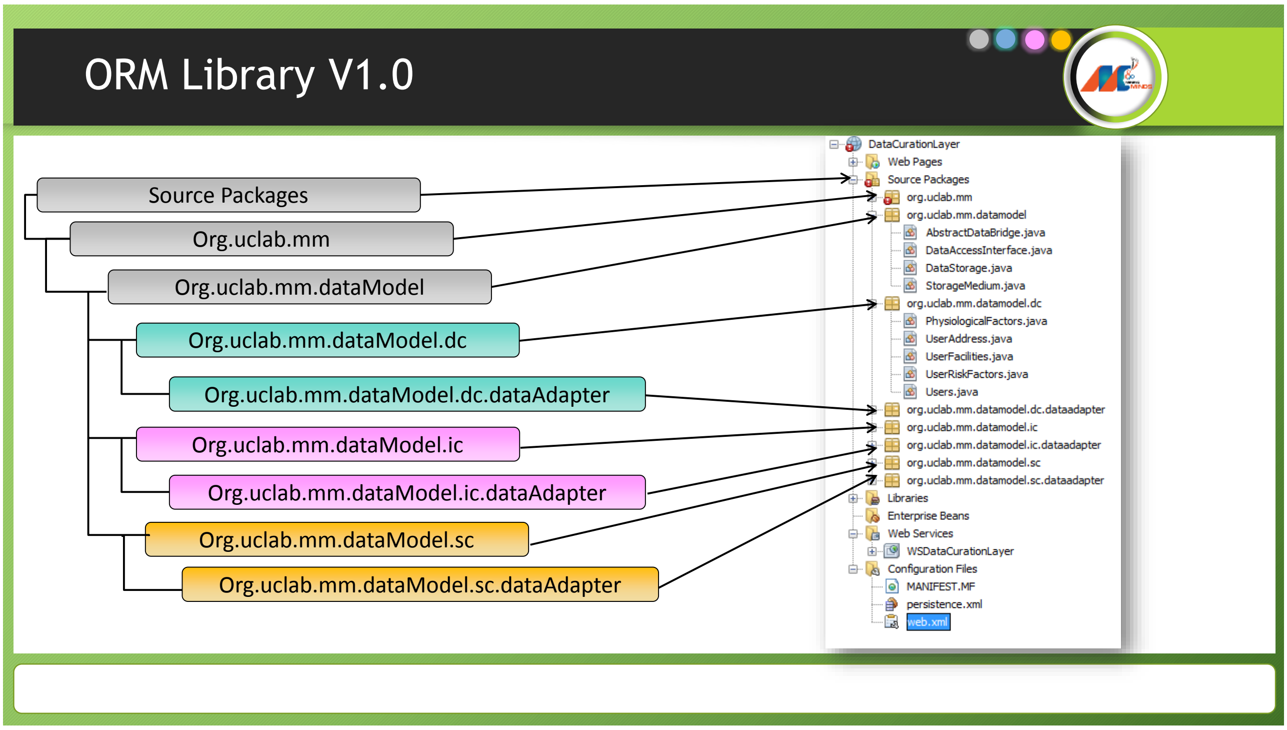


Distributed Database stacks

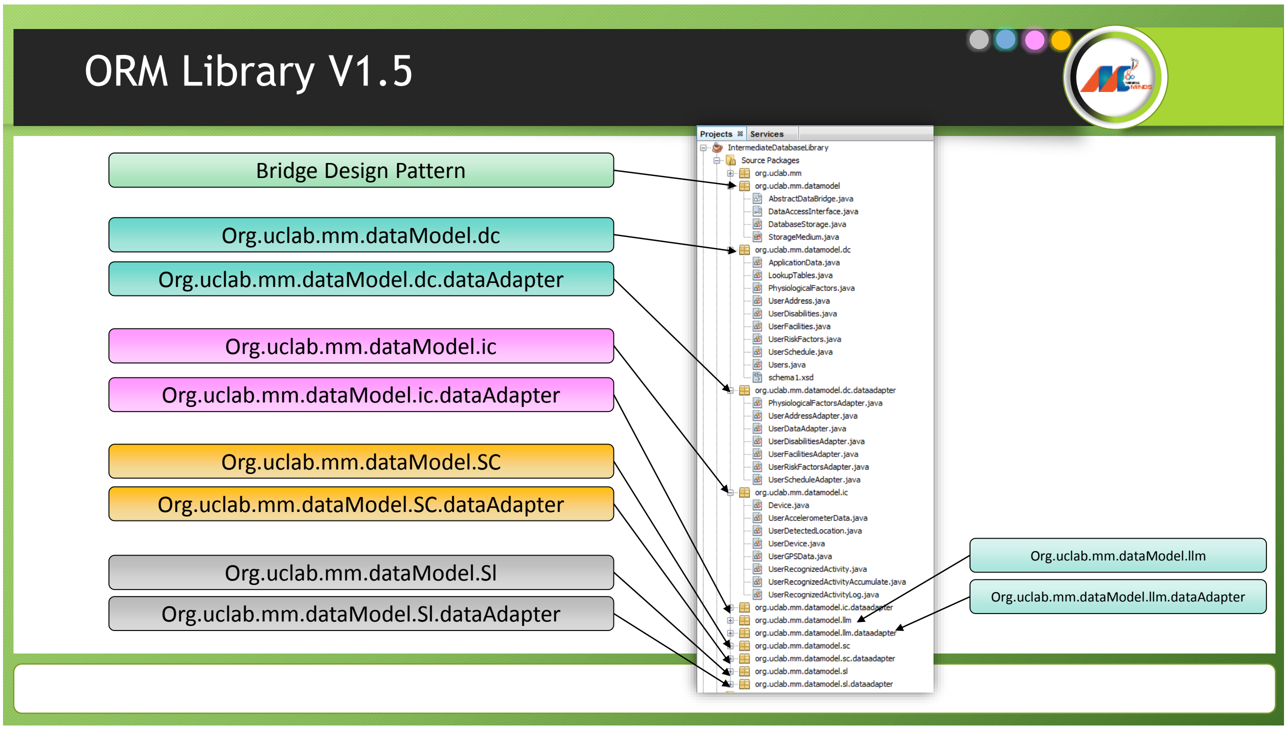


Distributed Database Stacks and Layer Communication



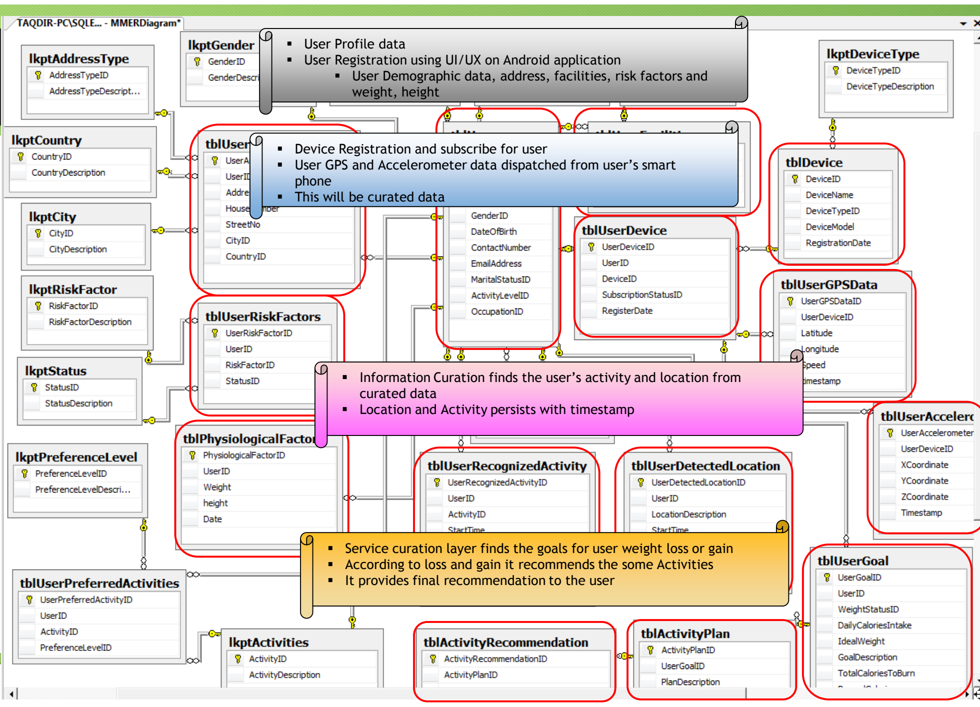


ORM Library V1.0
Source Packages
Org.uclab.mm
Org.uclab.mm.dataModel
Org.uclab.mm.dataModel.dc
Org.uclab.mm.dataModel.dc.dataAdapter
Org.uclab.mm.dataModel.ic
Org.uclab.mm.dataModel.ic.dataAdapter
Org.uclab.mm.dataModel.sc
Org.uclab.mm.dataModel.sc.dataAdapter
DataCurationLayer
Web Pages
Source Packages
org.uclab.mm
org.uclab.mm.datamodel
AbstractDataBridge.java
DataAccessInterface.java
DataStorage.java
StorageMedium.java
org.uclab.mm.datamodel.dc
PhysiologicalFactors.java
UserAddress.java
UserFacilities.java
UserRiskFactors.java
Users.java
org.uclab.mm.datamodel.dc.dataadapter
org.uclab.mm.datamodel.ic
org.uclab.mm.datamodel.ic.dataadapter
org.uclab.mm.datamodel.sc
org.uclab.mm.datamodel.sc.dataadapter
Libraries
Enterprise Beans
Web Services
WSDataCurationLayer
Configuration Files
MANIFEST.MF
persistence.xml
web.xml

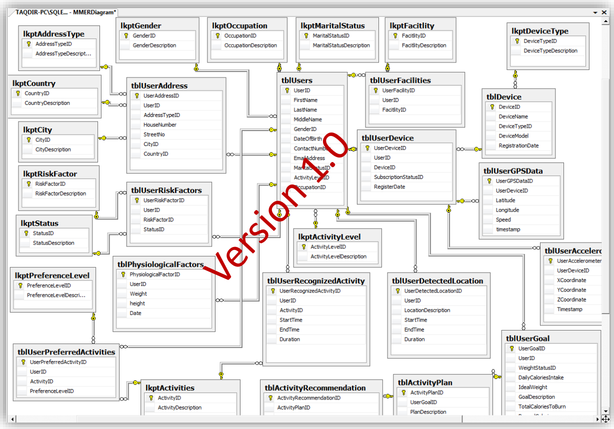


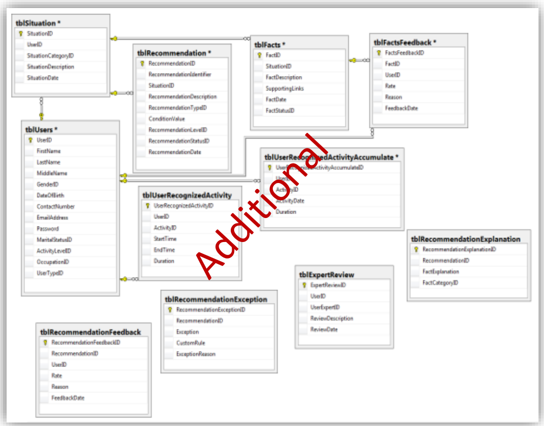
ORM Library V1.5
Bridge Design Pattern
Org.uclab.mm.dataModel.dc
Org.uclab.mm.dataModel.dc.dataAdapter
Org.uclab.mm.dataModel.ic
Org.uclab.mm.dataModel.ic.dataAdapter
Org.uclab.mm.dataModel.SC
Org.uclab.mm.dataModel.SC.dataAdapter
Org.uclab.mm.dataModel.Sl
Org.uclab.mm.dataModel.Sl.dataAdapter
Projects
Services
IntermediateDatabaseLibrary
Source Packages
org.uclab.mm
org.uclab.mm.datamodel
AbstractDataBridge.java
DataAccessInterface.java
DatabaseStorage.java
StorageMedium.java
org.uclab.mm.datamodel.dc
ApplicationData.java
LookupTables.java
PhysiologicalFactors.java
UserAddress.java
UserDisabilities.java
UserFacilities.java
UserRiskFactors.java
UserSchedule.java
Users.java
schema1.xsd
org.uclab.mm.datamodel.dc.dataadapter
PhysiologicalFactorsAdapter.java
UserAddressAdapter.java
UserDataAdapter.java
UserDisabilitiesAdapter.java
UserFacilitiesAdapter.java
UserRiskFactorsAdapter.java
UserScheduleAdapter.java
org.uclab.mm.datamodel.ic
Device.java
UserAccelerometerData.java
UserDetectedLocation.java
UserDevice.java
UserGPSData.java
UserRecognizedActivity.java
UserRecognizedActivityAccumulate.java
UserRecognizedActivityLog.java
org.uclab.mm.datamodel.ic.dataadapter
org.uclab.mm.datamodel.llm
org.uclab.mm.datamodel.llm.dataadapter
org.uclab.mm.datamodel.sc
org.uclab.mm.datamodel.sc.dataadapter
org.uclab.mm.datamodel.sl
org.uclab.mm.datamodel.sl.dataadapter
Org.uclab.mm.dataModel.llm
Org.uclab.mm.dataModel.llm.dataAdapter

Intermediate Database v1.0



Intermediate Database v1.5





Mining Minds Version 2.0

- We will use Object Oriented Representation
- Same communication (Socket and Restful)
- Object-Relational Mapping (ORM)
- Relational Database (MS SQL Server)
- Java Language
- Maintenance of Existing work
  - Changes in DB
  - Changes in ORM
  - Changes in Restful services
  - Changes in SQL Stored Procedures

These changes will be implemented according requirements

References

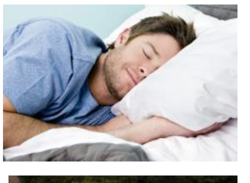
- [1][Henricksen2002] Henricksen, K., Indulska, J., Rakotonirainy, A. Modeling Context Information in Pervasive Computing Systems. In Pervasive Computing; Springer: Berlin, Germany, 2002; pp. 167–180.
- [2][Henricksen2002] Henricksen, K., Indulska, J. A software engineering framework for context-aware pervasive computing. In Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications, Orlando, FL, USA, 14–17 March 2004; pp. 77–86
- [3][Gray2001] Gray, P., Salber, D. Modelling and Using Sensed Context Information in the Design of Interactive Applications. In Engineering for Human-Computer Interaction; Springer: Berlin, Germany, 2001; pp. 317–335
- [4][Chen2004] Chen, H., Perich, F., Finin, T., Joshi, A. SOUPA: Standard ontology for ubiquitous and pervasive applications. In Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services, Boston, MA, USA, 22 August 2004; pp. 258–267
- [5][Petersen2012] Petersen, A.K, Mikalsen, M. Context: Representation and Reasoning. Available Online: http://www.idi.ntnu.no/~anderpe/publications/RIA-05-AKP-MM-paper.pdf (accessed on 27 May 2014).
- [6][Gu2004] Gu, T., Pung, H.K. A middleware for building context-aware mobile services. In Proceedings of the 2004 IEEE 59th Vehicular Technology Conference, Milan, Italy, 17–19 May 2004; pp. 2656–2660
- [7][Website] OWL Web Ontology Language Overview. Available online: http://www.w3.org/TR/owl-features/ (accessed on 27 May 2014)
- [8][Yamabe2005] Yamabe,T., Takagi, A., Nakajima, T. Citron: A context information acquisition framework for personal devices. In Proceedings of the 11th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Hong Kong, China, 17–19 August 2005; pp. 489–495
- [9][Bettini2010] Bettini, C., Brdiczka, O., Henricksen, K., Indulska, J., Nicklas, D., Ranganathan, A., Riboni, D. A survey of context modeling and reasoning techniques. Pervasive Mob. Comput. 2010, 6, 161–180

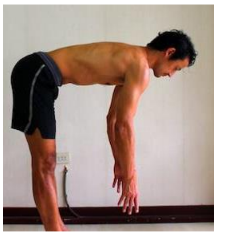
경희대학교
KYUNG HEE UNIVERSITY

Inertial Sensor based Activity Recognition

Overview

- Context awareness is necessary to understand user to provide health related services
  - Health monitoring & management
  - Emergent situation detection/prevention
  - Obesity care
- Activity Recognition is the cornerstone of context awareness





Motivation

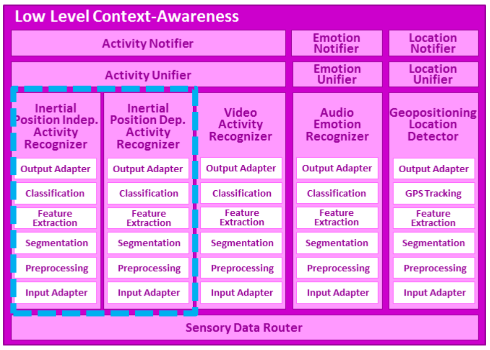
- Why activity recognition is required?
  - To provide health related services to the user, the system must know the user's state
  - User itself is the key factor of the system which user activity represents the daily lifestyle including health status, habits, preference, etc.
  - To infer the user activity, sensors are required to be attached to the user

- Issue of Real-time Inertial sensor based Activity Recognition
  - How accurate the recognizer infers the user activity with regard to the gap between data collection environment and real-world environment

경희대학교
KYUNG HEE UNIVERSITY

Goal and Objectives

- **Goal**: Design and implement a methodology to recognize user activity based on inertial sensors (ACC & Gyro) of wearable devices (Smartphone, Smartwatch, Shimmer)



- **Objectives:**
  - Achieve acceptable accuracy
  - Development of inertial sensor based activity recognition in real-time
  - Collect inertial sensor signal dataset in real-world

경희대학교
KYUNG HEE UNIVERSITY

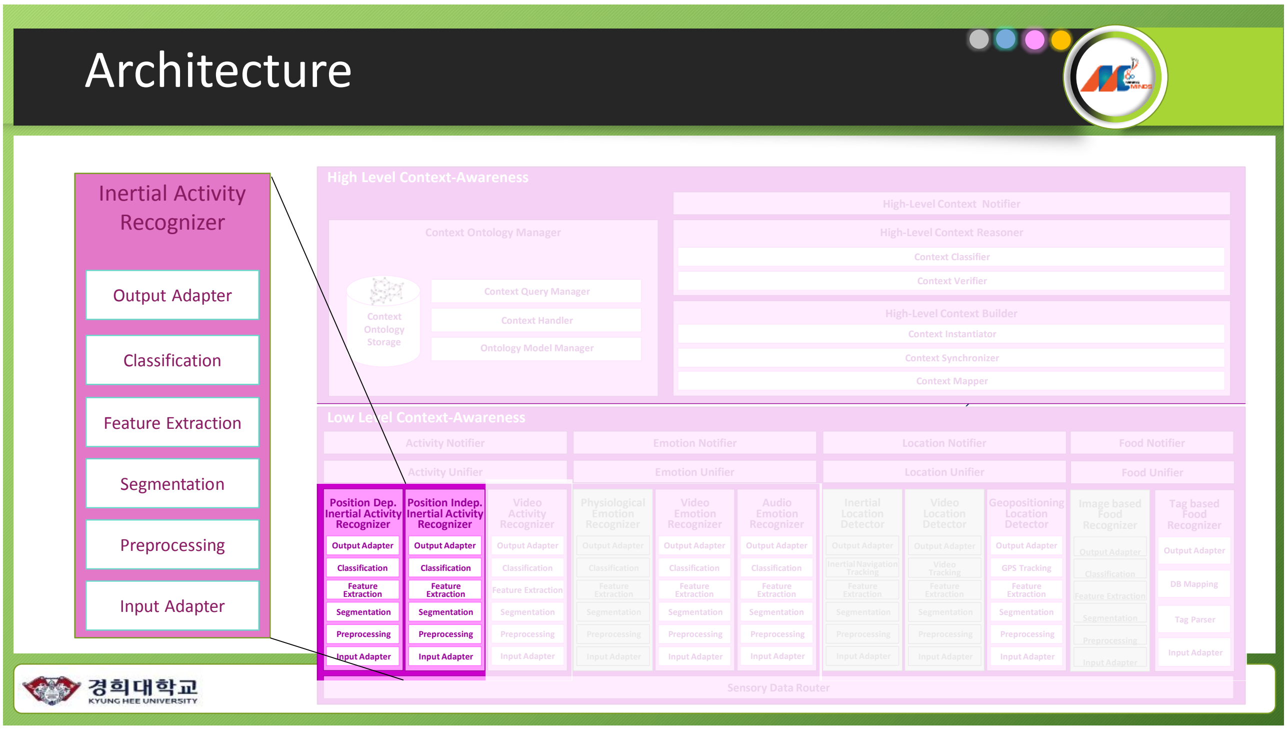
Related Works

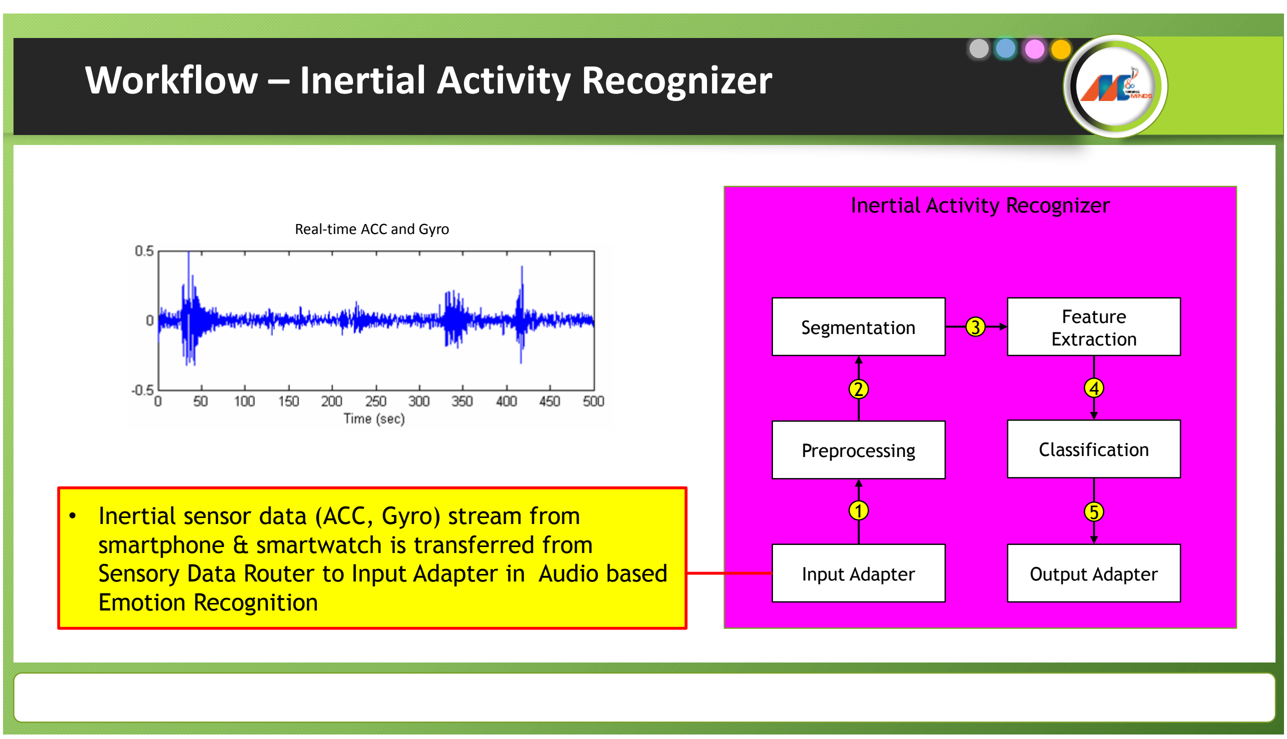
Authors	Published year	Sensor placement	Sensor type	Techniques	Limitation
Chun zhu et al. [1]	2009	Waist, Foot	ACC, Gyro	• Two feed-forward neural networks fusion • Heuristic segmentation	• Few basic activities • Offline evaluation
Jun-Ki Min et al. [3]	2011	Head, two arms and two wrists, fingers	ACC, Gyro, skin temperature, heat flux, galvanic skin	• Dynamic feature selection • Outputs of classifiers are combined and compared	• Device is to bulky to use in real life
Lei Gao et al. [4]	2011	Waist, chest, thigh, side	ACC	• Considered the difference of sensor orientation change using estimate of constant gravity vector • Sensor fault is considered	• Only used ACC
Ming Zeng et al. [2]	2014	Free	ACC, GPS, Speed, Ambient light	• Build separate models for each activity • Feature transformation	• Heavy weighted system
Muhammad Shoaib et al. [5]	2014	Upper arm, wrist, waist, two pockets on pants	ACC, Gyro	• Considered orientation independency • Compared the difference of sensor types and feature sets	• Few basic activities

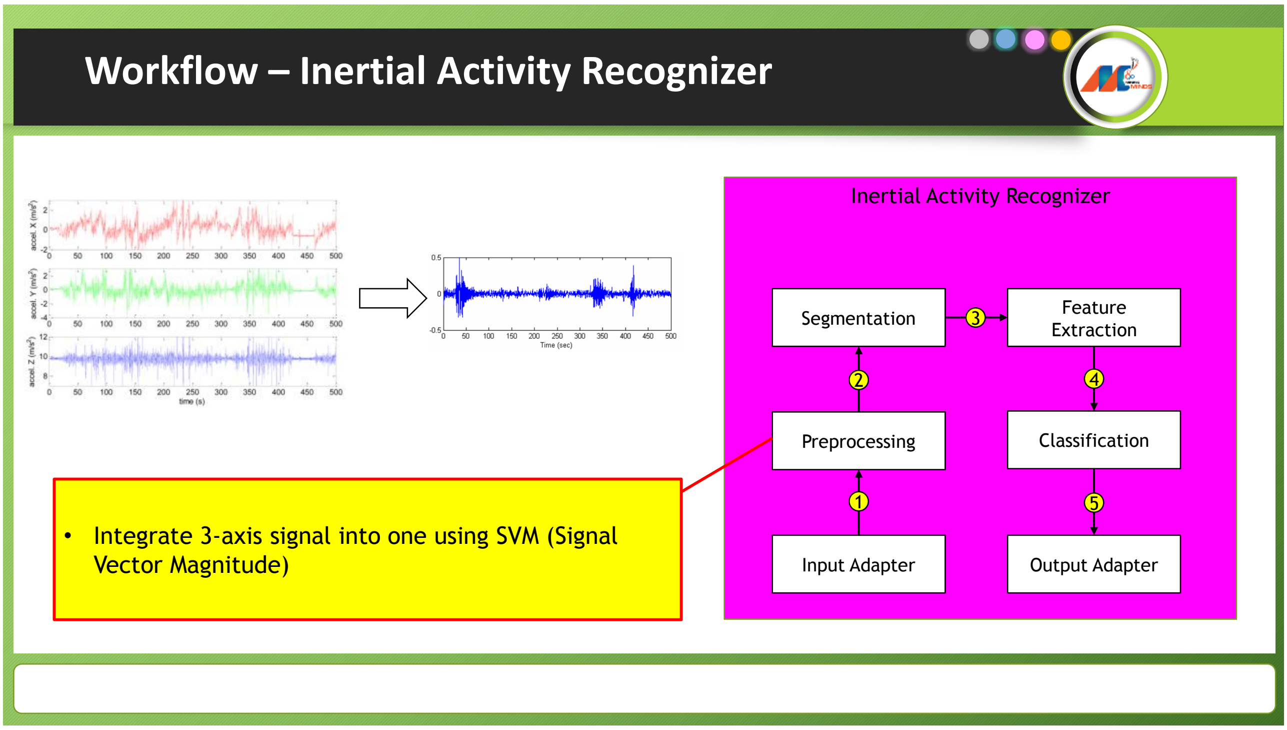
Limitations of existing works

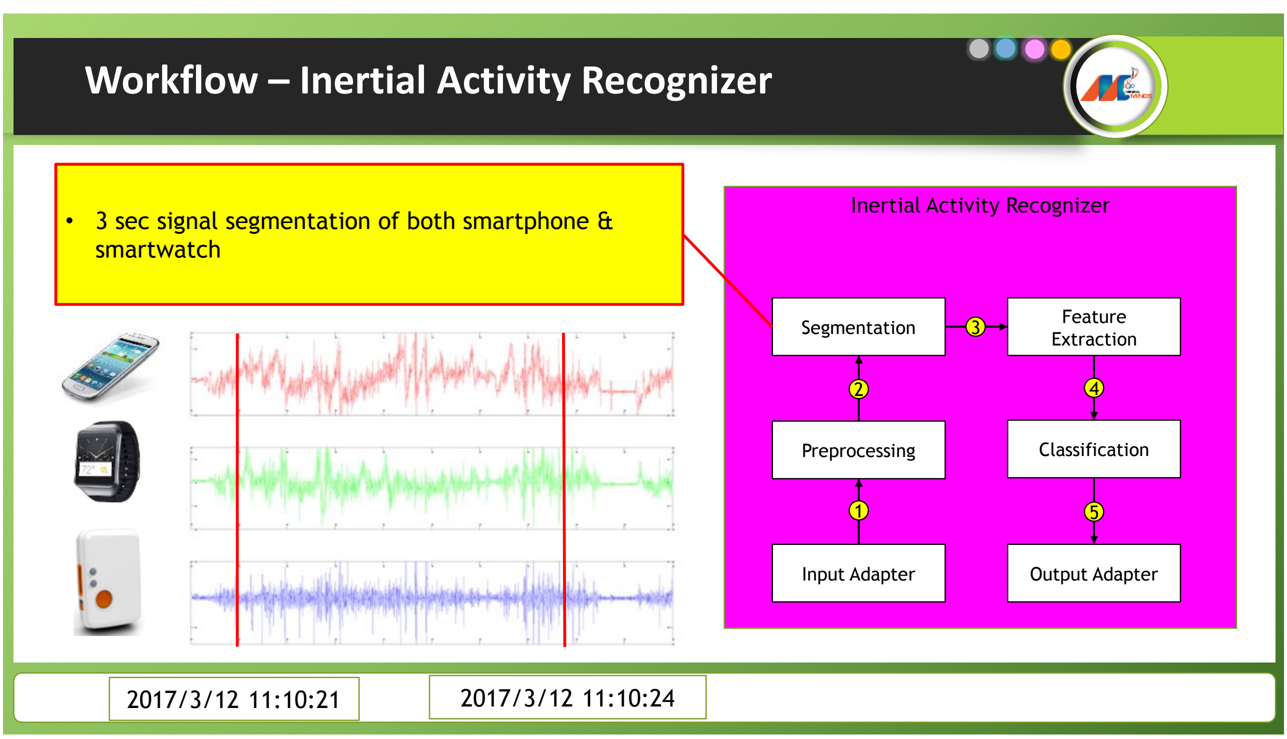
- Many or limited sensor devices
  - Many sensors are conspicuous and uncomfortable
  - Few limited sensors are not appropriate to recognize various activities well
- A gap between offline (Limited) environment and online (Real-world) environment
  - Use multi-model classifier for credibility
  - Weighted decision fusion to conclude final activity

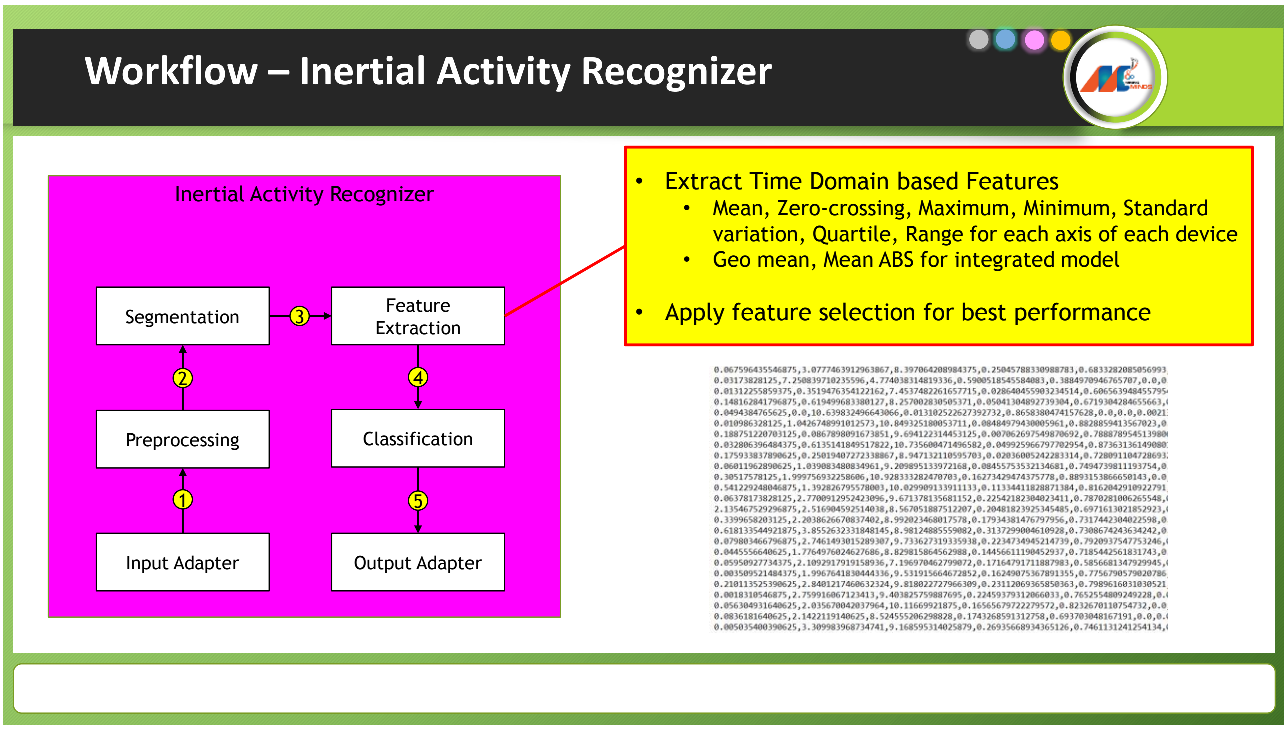
경희대학교 KYUNG HEE UNIVERSITY

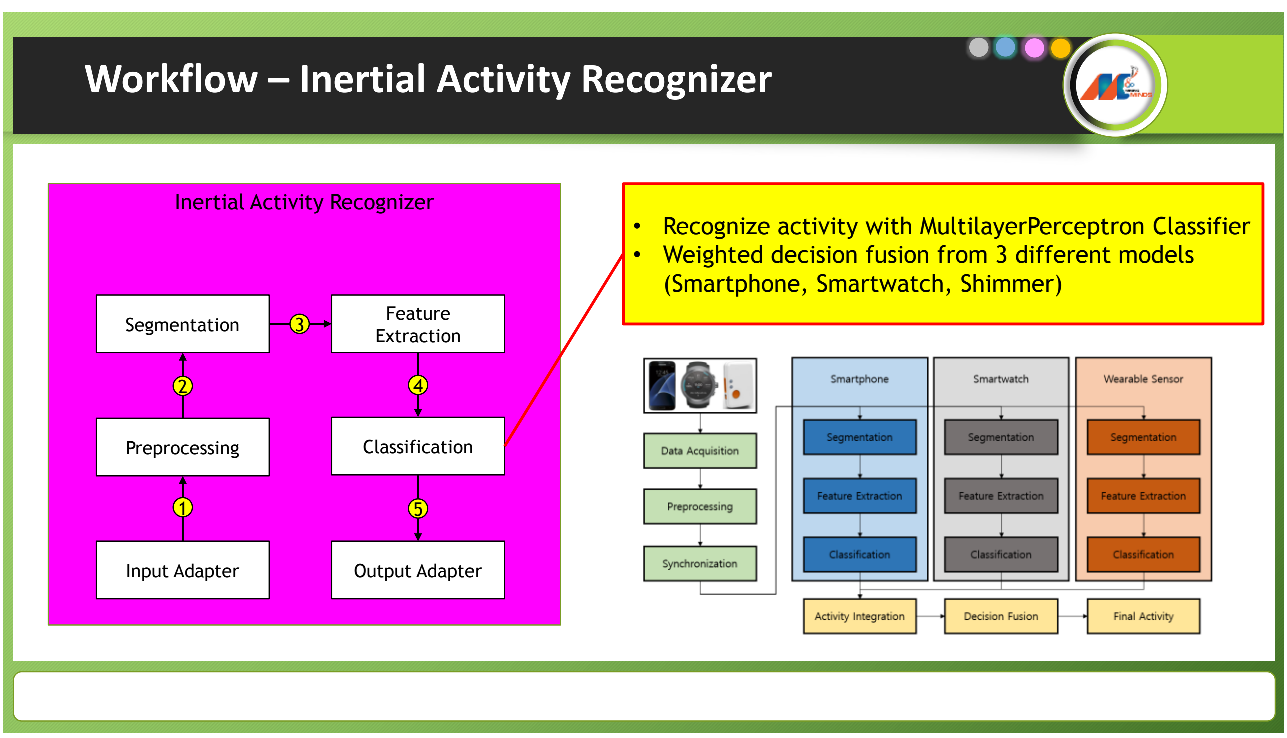

Architecture
Inertial Activity Recognizer
Output Adapter
Classification
Feature Extraction
Segmentation
Preprocessing
Input Adapter
High Level Context-Awareness
High-Level Context Notifier
Context Ontology Manager
High-Level Context Reasoner
Context Classifier
Context Verifier
Context Ontology Storage
Context Query Manager
Context Handler
Ontology Model Manager
High-Level Context Builder
Context Instantiator
Context Synchronizer
Context Mapper
Low Level Context-Awareness
Activity Notifier
Emotion Notifier
Location Notifier
Food Notifier
Activity Unifier
Emotion Unifier
Location Unifier
Food Unifier
Position Dep. Inertial Activity Recognizer
Position Indep. Inertial Activity Recognizer
Video Activity Recognizer
Physiological Emotion Recognizer
Video Emotion Recognizer
Audio Emotion Recognizer
Inertial Location Detector
Video Location Detector
Geopositioning Location Detector
Image based Food Recognizer
Tag based Food Recognizer
GPS Tracking
DB Mapping
Tag Parser
Sensory Data Router
경희대학교
KYUNG HEE UNIVERSITY


Workflow – Inertial Activity Recognizer
Real-time ACC and Gyro
0.5
0
-0.5
0 50 100 150 200 250 300 350 400 450 500
Time (sec)
Inertial Activity Recognizer
Segmentation
Feature Extraction
Preprocessing
Classification
Input Adapter
Output Adapter
1
2
3
4
5
• Inertial sensor data (ACC, Gyro) stream from smartphone & smartwatch is transferred from Sensory Data Router to Input Adapter in Audio based Emotion Recognition


Workflow – Inertial Activity Recognizer
Inertial Activity Recognizer
Segmentation
Feature Extraction
Preprocessing
Classification
Input Adapter
Output Adapter
1
2
3
4
5
• Integrate 3-axis signal into one using SVM (Signal Vector Magnitude)


Workflow – Inertial Activity Recognizer
• 3 sec signal segmentation of both smartphone & smartwatch
Inertial Activity Recognizer
Segmentation
Feature Extraction
Preprocessing
Classification
Input Adapter
Output Adapter
1
2
3
4
5
2017/3/12 11:10:21
2017/3/12 11:10:24


Workflow – Inertial Activity Recognizer
Inertial Activity Recognizer
Segmentation
Feature Extraction
Preprocessing
Classification
Input Adapter
Output Adapter
1
2
3
4
5
• Extract Time Domain based Features
• Mean, Zero-crossing, Maximum, Minimum, Standard variation, Quartile, Range for each axis of each device
• Geo mean, Mean ABS for integrated model
• Apply feature selection for best performance


Workflow – Inertial Activity Recognizer
Inertial Activity Recognizer
Segmentation
Feature Extraction
Preprocessing
Classification
Input Adapter
Output Adapter
1
2
3
4
5
• Recognize activity with MultilayerPerceptron Classifier
• Weighted decision fusion from 3 different models (Smartphone, Smartwatch, Shimmer)
Data Acquisition
Preprocessing
Synchronization
Smartphone
Smartwatch
Wearable Sensor
Segmentation
Feature Extraction
Classification
Activity Integration
Decision Fusion
Final Activity

Contributions

- Proposed a methodology for Inertial sensor based activity recognition
- 3 different roles of sensors (overall, top, bottom)
- Collection of a real-world inertial sensor dataset
- Offline evaluation and validation
- Online validation

경희대학교 KYUNG HEE UNIVERSITY

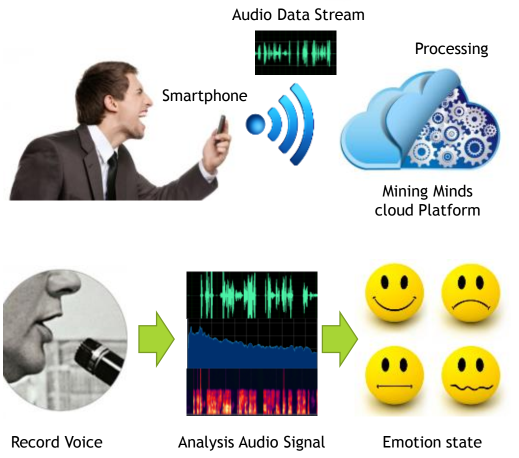
Mining Minds

Information Curation Layer MMV-3.0

경희대학교 KYUNG HEE UNIVERSITY KITECH 한국생산기술연구원 POSTECH 포항공과대학교 UNIVERSITY of TASMANIA 녹십자 헬스케어 TAPACROSS U2 system User & Usability system KOCSA

Overview

- Audio based Emotion Recognizer has an important role for *monitoring mental disease* and *analysis user preference*
- Audio based Emotion Recognizer must adapt the *real-time processing* and achieve the *Reasonable accuracy*
- Audio based Emotion Recognizer is applied user speech in *phone call environments*



Motivation

- Why audio for emotion recognition?
  - Speech is the most commonly used and most natural method of human communication
  - Recently, people frequently communicate with each other by speech on their own machines (Phone, PC etc.)
  - Especially a phone call is powerful data source for recognizing emotion
- Issue of Real-time Speech Emotion Recognition on phone call
  - How to gather emotional audio data sources
  - How to recognize user emotion on real-time using by phone call in conversation environments

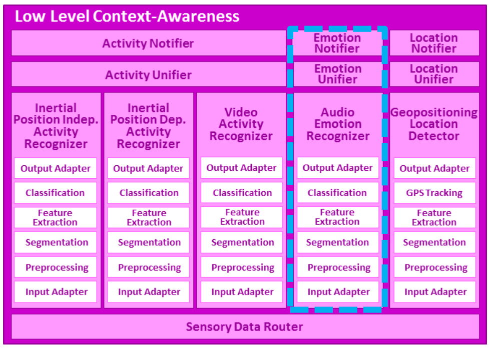
Various emotion expression on a phone call

경희대학교 KYUNG HEE UNIVERSITY

Goal and Objectives

- **Goal**: Design and implement a methodology that is able to recognize emotional states from user speech of phone call

- **Objectives:**
  - Achieve acceptable accuracy
  - Development of smartphone call speech based emotion recognition on the real-time
  - Collect a real-world and diverse call speech dataset



경희대학교
KYUNG HEE UNIVERSITY

Related Works – Datasets

Database	Modalities	Elicitation Method	Emotional Content	Size
AIBO database (Batliner et al., 2004) [1]	Audio	Natural: children interaction with robot	anger, bored, emphatic, helpless, ironic, joyful, motherese, reprimanding, rest, surprise, touchy	110 dialogues, 29200 words
Berlin Database (Burkhardt et al., 2005) [2]	Audio	Acted	anger, boredom, disgust, fear, happiness, sadness, neutral	493 sentences; 5 actors & 5 actresses
ISL meeting corpus (Burger et al., 2002)	Audio	Natural: meeting corpus	negative, positive, neutral	18 meetings; average 5 persons per meeting
Adult Attachment Interview database (Roisman, 2004) [3]	Audio-Visual	Natural: subjects were interviewed to describe the childhood experience	6 basic emotions, contempt, embarrassment, shame, general positive and negative emotions	60 adults: each interview was 30-60 minutes long
Belfast database (Douglas-Cowie et al., 2003) [4]	Audio-Visual	Natural: clips taken from television and realistic interviews with research team	Dimensional labeling/categorical labeling	125 subjects; 209 clips from TV and 30 from interviews
Busso-Narayanan database (Busso et al., 2007) [5]	Audio-Visual	Acted	anger, happiness, sadness, neutral	612 sentences; an actress
IEMOCAP: Interactive emotional dyadic motion capture database (Busso et al, 2008) [6]	Audio-Visual	Acted	happiness, sadness, anger and frustration	10 actor recorded
Haq-Jackson database (Haq & Jackson, 2009) [7]	Audio-Visual	Acted: emotion stimuli were shown on screen	6 basic emotions, neutral	480 sentences; 4 male subjects

경희대학교
KYUNG HEE UNIVERSITY

Related Works – Methodologies

Reference	Data	Features	Classifier	Classes	Accuracy
Lee et al., 2011 [8]	AIBO Dataset	Prosody, MFCC + Statistical Functions	Hierarchical Bayesian Logistic	5	48.2%
Purnima Chandrasekar et, al. ,2014 [9]	EmoDB	MFCC, Pitch, Energy	SVM	7	86.6%
Kun Han et al., 2014 [10]	IEMOCAP: Interactive emotional dyadic motion capture database	Segmented MFCC Features Vector	Deep neural network	4	54.3%
Arianna Mencattini et al., 2014 [11]	EMOVO	520 features (divided 12 different groups); TEO, Energy Sequence, wavelet approximation coefficients etc.	Support Vector Machine (SVM)	7	72%
Wang et al., 2015 [12]	EmoDB, CASIA	MFCC, Fourier Parameters	SVM	6	88.9% (EmoDB), 79% (CASIA)
Poria et al., 2015 [13]	Audio-Visual; eNTERFACE	V: characteristic points, distances; A: MFCC, spectral features	SVM	6	81.2%(V), 78.6%(A), 87.95% (AV)
Amiya Kumar et al., 2015 [14]	600 speech sample by 5 speakers	MFCC, LPCC, MEDC	SVM	7	82.26%

경희대학교
KYUNG HEE UNIVERSITY

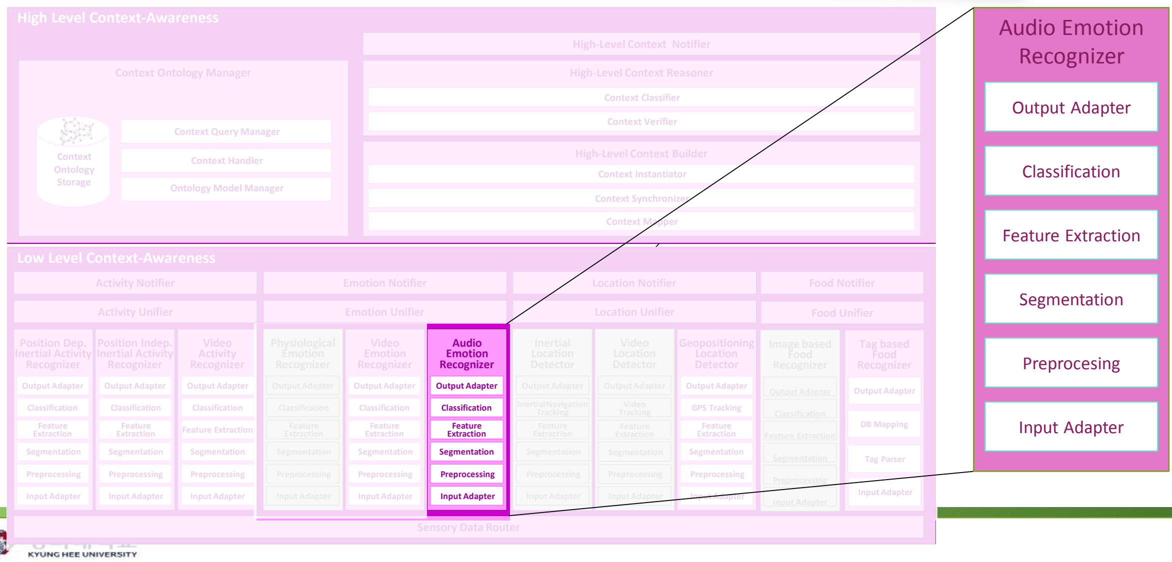
Limitations of existing works

- Focus on a evaluation based on formatted speech database
  - EmoDB, eNTERFACE, Berlin Emotion DB, etc.

- Lack of preprocessing for supporting real-time process
  - silent remover, user speech signal extraction

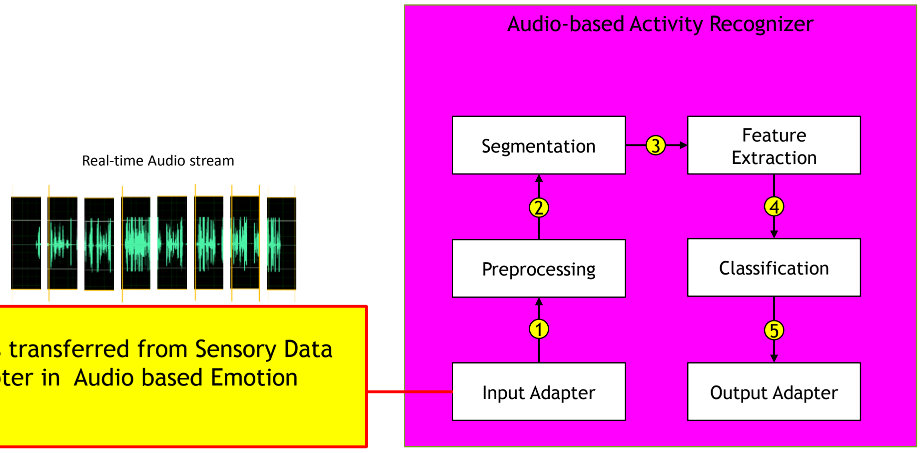
경희대학교
KYUNG HEE UNIVERSITY

Architecture



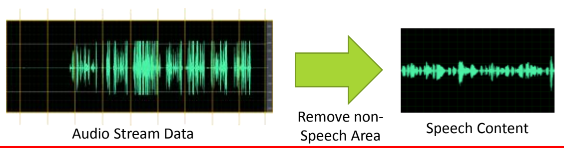
KYUNG HEE UNIVERSITY

Workflow – Emotion Recognizer

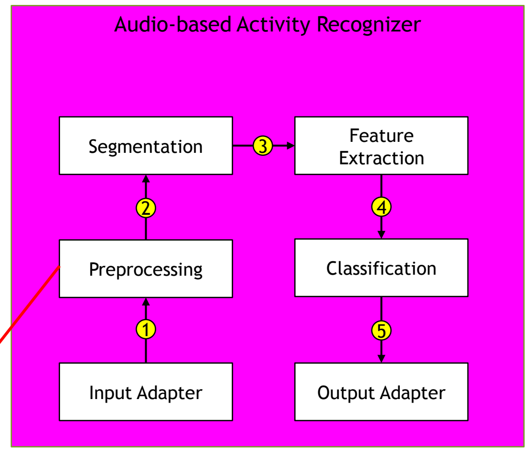


- Audio Stream data is transferred from Sensory Data Router to Input Adapter in Audio based Emotion Recognition

Workflow – Emotion Recognizer

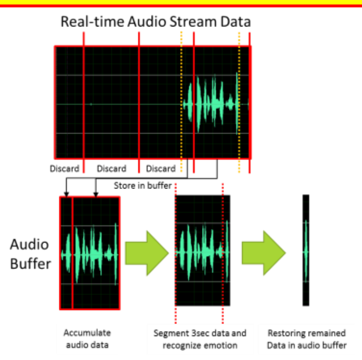


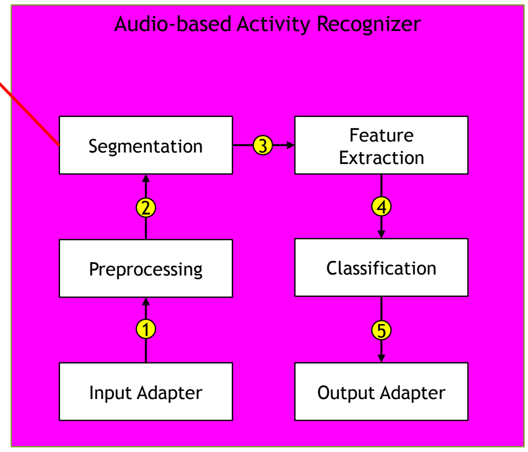
- Removing non-speech area in audio stream data

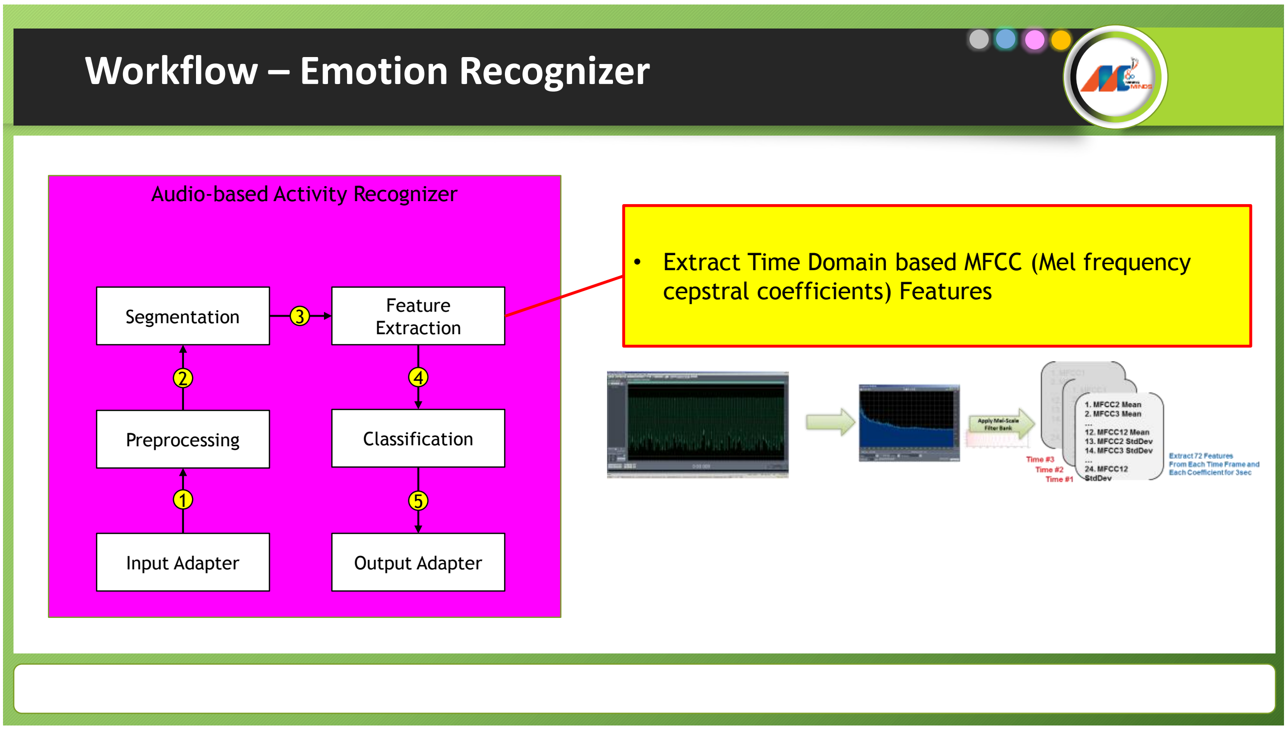


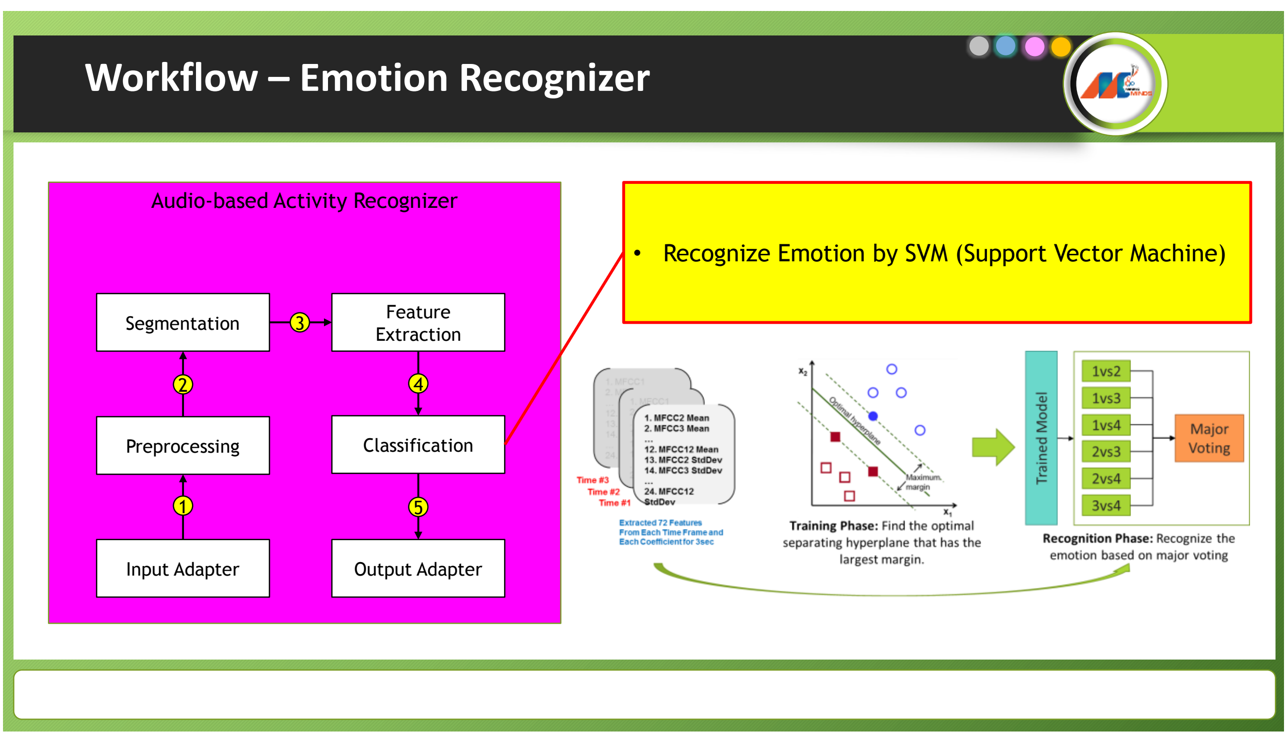
Workflow – Emotion Recognizer

- Window based Segmentation by 3 sec using audio buffer






Workflow – Emotion Recognizer
Audio-based Activity Recognizer
Segmentation
Feature Extraction
Preprocessing
Classification
Input Adapter
Output Adapter
1
2
3
4
5
• Extract Time Domain based MFCC (Mel frequency cepstral coefficients) Features
1. MFCC2 Mean
2. MFCC3 Mean
12. MFCC12 Mean
13. MFCC2 StdDev
14. MFCC3 StdDev
24. MFCC12 StdDev
Time #3
Time #2
Time #1
Extract 72 Features From Each Time Frame and Each Coefficient for 3sec


Workflow – Emotion Recognizer
Audio-based Activity Recognizer
Segmentation
Feature Extraction
Preprocessing
Classification
Input Adapter
Output Adapter
1
2
3
4
5
• Recognize Emotion by SVM (Support Vector Machine)
1. MFCC2 Mean
2. MFCC3 Mean
12. MFCC12 Mean
13. MFCC2 StdDev
14. MFCC3 StdDev
24. MFCC12 StdDev
Time #3
Time #2
Time #1
Extracted 72 Features From Each Time Frame and Each Coefficient for 3sec
Optimal hyperplane
Maximum margin
Training Phase: Find the optimal separating hyperplane that has the largest margin.
Trained Model
1vs2
1vs3
1vs4
2vs3
2vs4
3vs4
Major Voting
Recognition Phase: Recognize the emotion based on major voting

Contributions

- Proposing a methodology for speech based emotion recognition
- Creation of an emotion set based on requirements of services to be delivered
- Collection of a real-world emotional speech dataset
- Offline evaluation and validation of various emotion recognition models
- Online validation

경희대학교 KYUNG HEE UNIVERSITY

Mining Minds

Video-based Activity Recognition

Thien

경희대학교 KYUNG HEE UNIVERSITY KITECH POSTECH UNIVERSITY of TASMANIA 녹십자헬스케어 TAPACROSS U2 system KOCSA

Overview

- Video-based activity recognition has an important role in *surveillance* and *health-care* systems
- Video-based activity recognition must adapt the *real-time processing* and achieve the *high accuracy*

- Capability of indoor *complex activity recognition*
- Combine with inertial sensors for *accuracy improvement*

Motivation

- Natural limitations of color cameras in object detection and pose estimation
  - *Depth camera supports 3D skeleton estimation*
- Complex indoor activities with different posture representations
  - *Combining skeleton-based spatial features with temporal descriptive statistics features in the temporal dimension.*
- Performance balance between accuracy and processing time
  - *Feature selection for feature dimension reduction*

Related works

Authors	Year	Dimension	No. Activity	Activity types	Accuracy	Key points	Limitation
Gu	2010	3D	8	Single action	94%	Hidden Markov Mode	High complexity
Ofli	2013	3D	12	Single action	80%	Annotated joint feature	Low recognition rate
Vantigodi	2013	3D	12	Single action	96%	Temporal joint distance feature	
Kruthiventi	2014	3D	12	Single action	97%	Dynamic time warping	
Wang	2014	3D	12	Single action	95%	Actionlet ensemble model	High complexity and computational cost

- Single action: hand catching, forward punching, two hand waving, ward kicking, high throwing …

Methodology

3D skeleton acquisition → Segmentation → Feature Extraction → Feature Selection → Training & classification

- **Devices**
  - Microsoft Kinect sensor v2
  - Computer
- **Software**
  - Microsoft Visual Basic
  - Java
- **Data**
  - Frame order
  - 3D skeleton data

Kinect v2 highlight specification

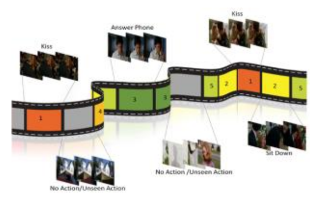
- Up to 6 complete skeleton (compared to 2 with the original sensor)
- 25 joints per person (compared to 20 with the original sensor)
- More anatomically correct and stable and the range of tracking is broader
- Higher depth fidelity and a significantly improved noise floor

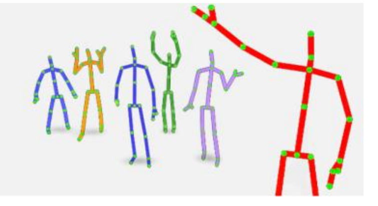
Components inside the Kinect device

Methodology

3D skeleton acquisition → **Segmentation** → Feature Extraction → Feature Selection → Training & classification

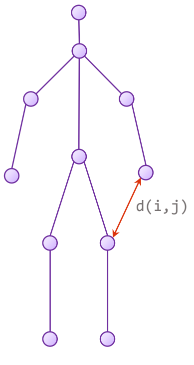
- Segment skeleton data flow (30 body frame per each second) into 3-seconds windows for recognition.
- Each 3-seconds data window
  - Captured at 30 fps
  - Contain 90 complete skeletons
  - Each complete skeleton contains 25 joints
  - Each joint represented by 3 values of *x*, *y*, and *z* in 3D space





Methodology

3D skeleton acquisition → Segmentation → **Feature Extraction** → Feature Selection → Training & classification

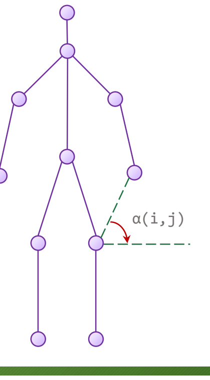


Spatial Joint Distance Feature is measure Euclidean distance between 2 joints of an object in the current frame.

$$d(i,j)=\sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (z_i - z_j)^2}$$

Spatial Joint Angle Feature is measure angle between 2 vector $\overrightarrow{ij}$ and $\overrightarrow{Ox}$ on the plane z=0.

$$\alpha(i,j) = \tan^{-1}\left(\frac{|y_i - y_j|}{|x_i - x_j|}\right)$$



Methodology

3D skeleton acquisition → Segmentation → Feature Extraction → Feature Selection → Training & classification

Descriptive statistics features: along 90 frames corresponding a window of 3-second

- Calculate the mean of the distance and angle feature (N=90)

$$\mu_d = \frac{1}{N}\sum_{i=1}^{N} d_i \qquad \mu_\alpha = \frac{1}{N}\sum_{i=1}^{N} \alpha_i$$

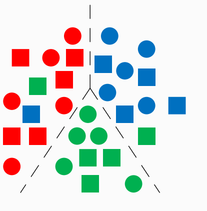
- Calculate the standard deviation of the distance and angle

$$s_d = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} |d_i - \mu_d|^2} \qquad s_\alpha = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N} |\alpha_i - \mu_\alpha|^2}$$

Methodology

3D skeleton acquisition → Segmentation → Feature Extraction → Feature Selection → Training & classification

- Reduce the feature dimension
- Boost the processing speed for real-time recognition
- Ranks the features using an independent evaluation for binary classification
- Absolute value two-sample t-test with pooled variance estimate.
- A number of highly significant features is selected



Square or circle is better ?

Methodology

3D skeleton acquisition → Segmentation → Feature Extraction → Feature Selection → Training & classification

Support Vector Machine

- Has a regularization parameter, which makes the user think about avoiding over-fitting
- Uses the kernel trick, therefore it is built in expert knowledge about the problem via engineering the kernel
- SVM is defined by a convex optimization problem (no local minima) for which there are efficient methods (e.g. SMO)
- An approximation to a bound on the test error rate

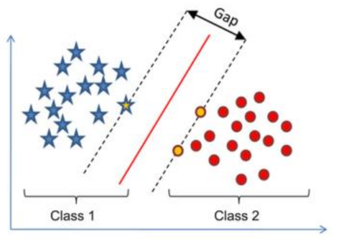


Illustration of a SVM classifier

Tools/Technologies

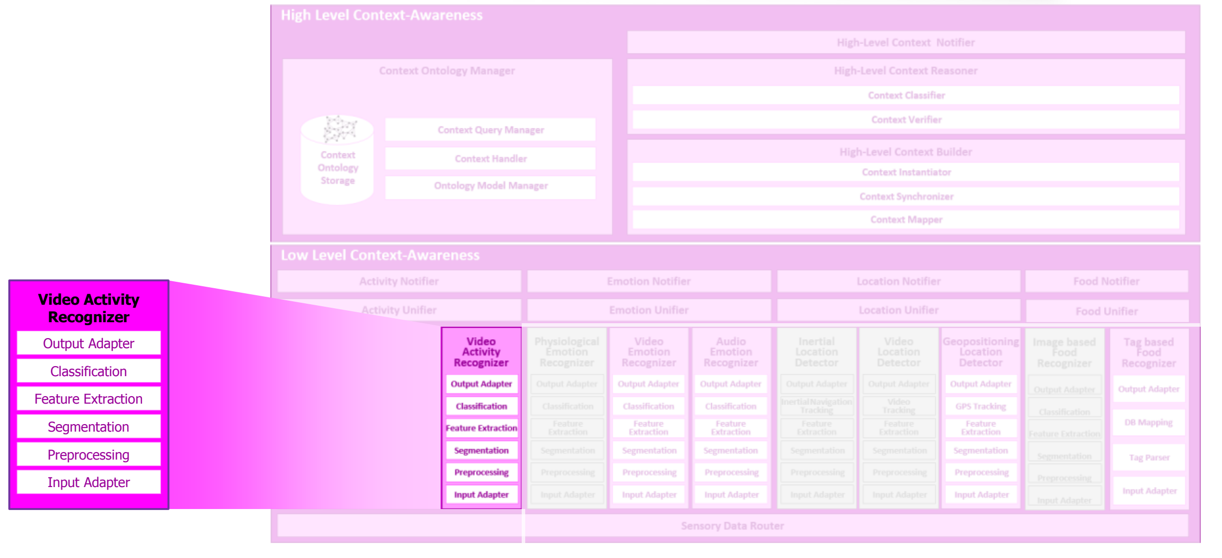
Windows 8

Java
ORACLE

MATLAB & SIMULINK

Execution Workflow

Component Architecture



Execution Workflow

Video-based Activity Recognizer

- Output adapter
- ↑ 5
- Classification
- ↑ 4
- Feature selection
- ↑ 3
- Feature extraction
- ↑ 2
- Segmentation
- ↑ 1
- Input Adapter

Kinect for Windows v2 Sensor

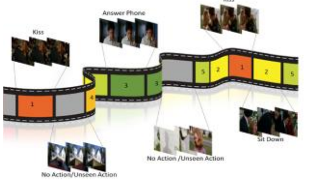
[x1,y1,z1,x2,y2,z2,…,x25,y25,z25
x1,y1,z1,x2,y2,z2,…,x25,y25,z25
…
x1,y1,z1,x2,y2,z2,…,x25,y25,z25]

- Skeleton data maintained in DCL is transferred to ICL
- Data containing 3D coordinates of 25-joint complete skeleton

Execution Workflow

Video-based Activity Recognizer

- Output adapter
- ↑ 5
- Classification
- ↑ 4
- Feature selection
- ↑ 3
- Feature extraction
- ↑ 2
- Segmentation
- ↑ 1
- Input Adapter



x1,y1,z1,x2,y2,z2,…,x25,y25,z25
x1,y1,z1,x2,y2,z2,…,x25,y25,z25
…
x1,y1,z1,x2,y2,z2,…,x25,y25,z25
…

Each data window

- Captured at 30 fps
- Contain 90 complete skeletons
- Each joint represented by 3 values of *x*, *y*, and *z*.

[1,x1,y1,z1,x2,y2,z2,…,x25,y25,z25
2,x1,y1,z1,x2,y2,z2,…,x25,y25,z25
…
90,x1,y1,z1,x2,y2,z2,…,x25,y25,z25] } 3 seconds

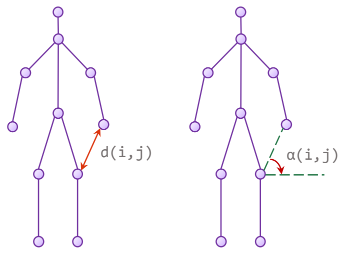
[1,x1,y1,z1,x2,y2,z2,…,x25,y25,z25
2,x1,y1,z1,x2,y2,z2,…,x25,y25,z25
…
90,x1,y1,z1,x2,y2,z2,…,x25,y25,z25] } 3 seconds

- Segment skeleton data flow with 30 fps of frame rate into 3-second windows for recognition.
- The number of frames is reduced from 90 to 30 to boost the processing time

Execution Workflow

Video-based Activity Recognizer

- Output adapter
- ⑤
- Classification
- ④
- Feature selection
- ③
- Feature extraction
- ②
- Segmentation
- ①
- Input Adapter



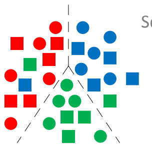

```
[1,x1,y1,z1,x2,y2,z2,…,x25,y25,z25
 2,x1,y1,z1,x2,y2,z2,…,x25,y25,z25
 …
90,x1,y1,z1,x2,y2,z2,…,x25,y25,z25]

[1,d1,d2,…,d300,a1,a2,…,a300
 2,d1,d2,…,d300,a1,a2,…,a300
 …
90,d1,d2,…,d300,a1,a2,…,a300]
```

- Calculate the joint distance and joint angle features based on 3D skeleton data
- Calculate the mean and standard deviation of joint distance and angle

Execution Workflow

Video-based Activity Recognizer

- Output adapter
- ⑤
- Classification
- ④
- Feature selection
- ③
- Feature extraction
- ②
- Segmentation
- ①
- Input Adapter



Square or circle is better ?

Feature ranking using absolute value two-sample t-test with pooled variance estimate.

- Ranks the features using an independent evaluation for binary classification
- Select the most of important features to reduce the processing time of online recognition

```
[1,d1,d2,…,d300,a1,a2,…,a300          [1,d1,d2,…,d70,a1,a2,…,a70
 2,d1,d2,…,d300,a1,a2,…,a300   - - ->   2,d1,d2,…,d70,a1,a2,…,a70
 …                                      …
90,d1,d2,…,d300,a1,a2,…,a300]         90,d1,d2,…,d70,a1,a2,…,a70]

[md1,md2,…,md70,ma1,ma2,…,ma70,stdd1,stdd2,…,stdd70,stda1,stda2,…,stda70]
```

Execution Workflow

Video-based Activity Recognizer

- Output adapter
- ⑤
- **Classification**
- ④
- Feature selection
- ③
- Feature extraction
- ②
- Segmentation
- ①
- Input Adapter

SVM model (Weka Tool)

[md1,md2,…,md70,ma1,ma2,…,ma70,stdd1,stdd2,…,stdd70,stda1,stda2,…,stda70] → **Action class**

- Training the SVM classification model using Weka tool
- The learned model is used for online recognition

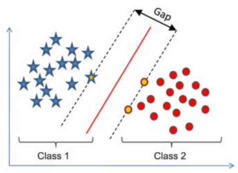


Illustration of a SVM classifier

Why SVM ?

- Avoiding over-fitting
- Uses the kernel trick and built in expert knowledge about the problem via engineering the kernel
- Is defined by a convex optimization problem (no local minima) for which there are efficient methods (e.g. SMO)
- An approximation to a bound on the test error rate

Execution Workflow

Video-based Activity Recognizer

- **Output adapter**
- ⑤
- Classification
- ④
- Feature selection
- ③
- Feature extraction
- ②
- Segmentation
- ①
- Input Adapter

- Result of the classifier is the label of recognized action
- Provide the video-based activity recognizer to the Activity Unifier

Input

```
x1,y1,z1,x2,y2,z2,…,x25,y25,z25
x1,y1,z1,x2,y2,z2,…,x25,y25,z25
…
x1,y1,z1,x2,y2,z2,…,x25,y25,z25
```

→ VAR →

Output

```
[Time,ID,Activity_Inertial,Activity_Kinect]

[12:01:30,Obj_A,standing,standing]
[12:04:30,Obj_A,sweeping,standing]
[12:07:30,Obj_A,sitting,sitting]
[12:11:30,Obj_A,sitting,sitting]
```

Contribution

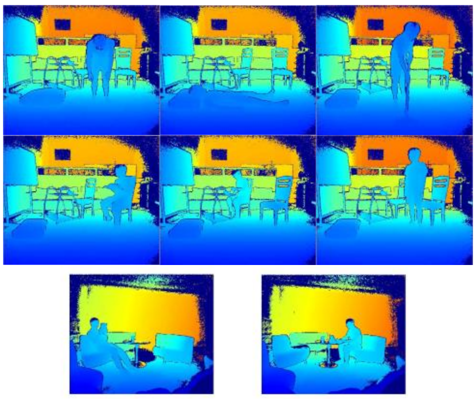
- Collect a new dataset for evaluating the video-based activity recognition using MS Kinect sensor.
- Develop an efficient algorithm for video-based activity recognition using 3D skeleton data.
- Achieve a good trade-off between the accuracy and computational cost for real-time recognition.

MM v3.0 - VAR

- Input : 3D skeleton data
- Output
  - V2.5: stretching, lying, sweeping, sitting, eating, and standing.
  - V3.0:
    - Stretching (up-down, left-right)
    - Lying
    - Sweeping
    - Sitting
    - Eating while sitting
    - Standing
    - Using laptop while sitting
    - Relaxing
- Good tradeoff between accuracy and real-time processing

New Features in VAR 3.0

Evaluation results



Eight activities collected by Kinect (left to right and top to bottom): stretching, lying, sweeping, sitting, eating, standing, using laptop, and relaxing

	Stretching	Lying	Sweeping	Sitting	Eating	Standing	Usinglaptop	Relaxing
Stretching	0.88	0.00	0.04	0.00	0.00	0.08	0.00	0.00
Lying	0.02	0.86	0.02	0.06	0.00	0.04	0.00	0.00
Sweeping	0.00	0.06	0.88	0.00	0.00	0.06	0.00	0.00
Sitting	0.04	0.04	0.00	0.80	0.08	0.04	0.00	0.00
Eating	0.00	0.06	0.04	0.04	0.86	0.00	0.00	0.00
Standing	0.08	0.06	0.04	0.00	0.00	0.82	0.00	0.00
Usinglaptop	0.00	0.00	0.02	0.04	0.00	0.00	0.94	0.00
Relaxing	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.90

Classification result using support vector machine on the testing dataset – Average accuracy 85%

High Level Context Awareness(HLCA)

Component Specification MMV3.0

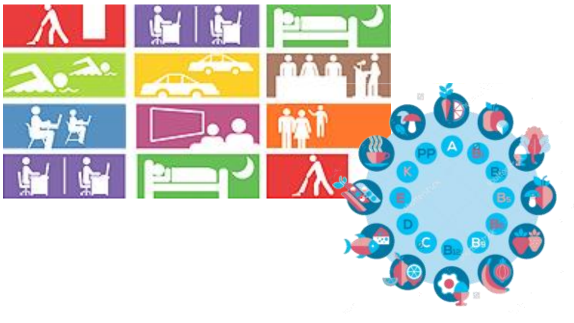
Agenda

- Introduction
- Motivation
- Goal and Objectives
- Related Work
- HLCA Use Cases
- HLCA Architecture
- Context Ontology
- Context Ontology Manager
- High Level Context Builder
- High Level Context Reasoner
- High Level Context Notifier
- Tools and Technologies
- Contributions
- References

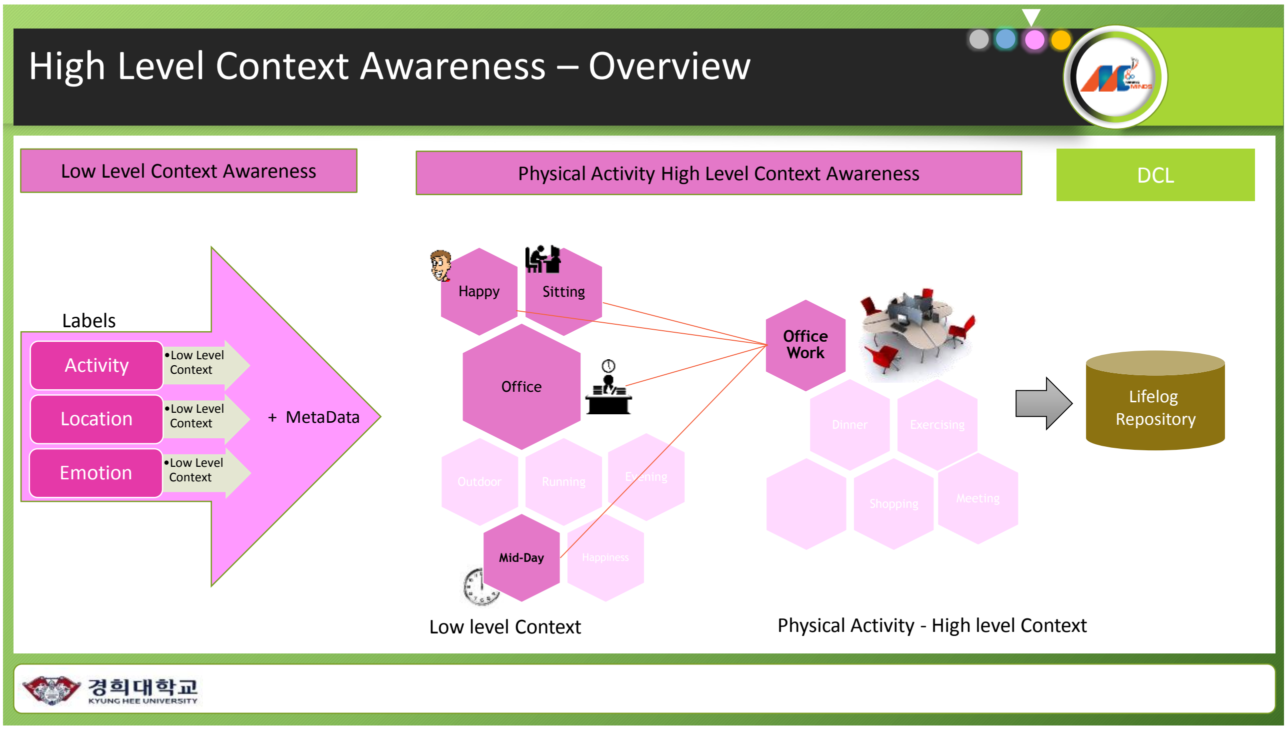
경희대학교 KYUNG HEE UNIVERSITY

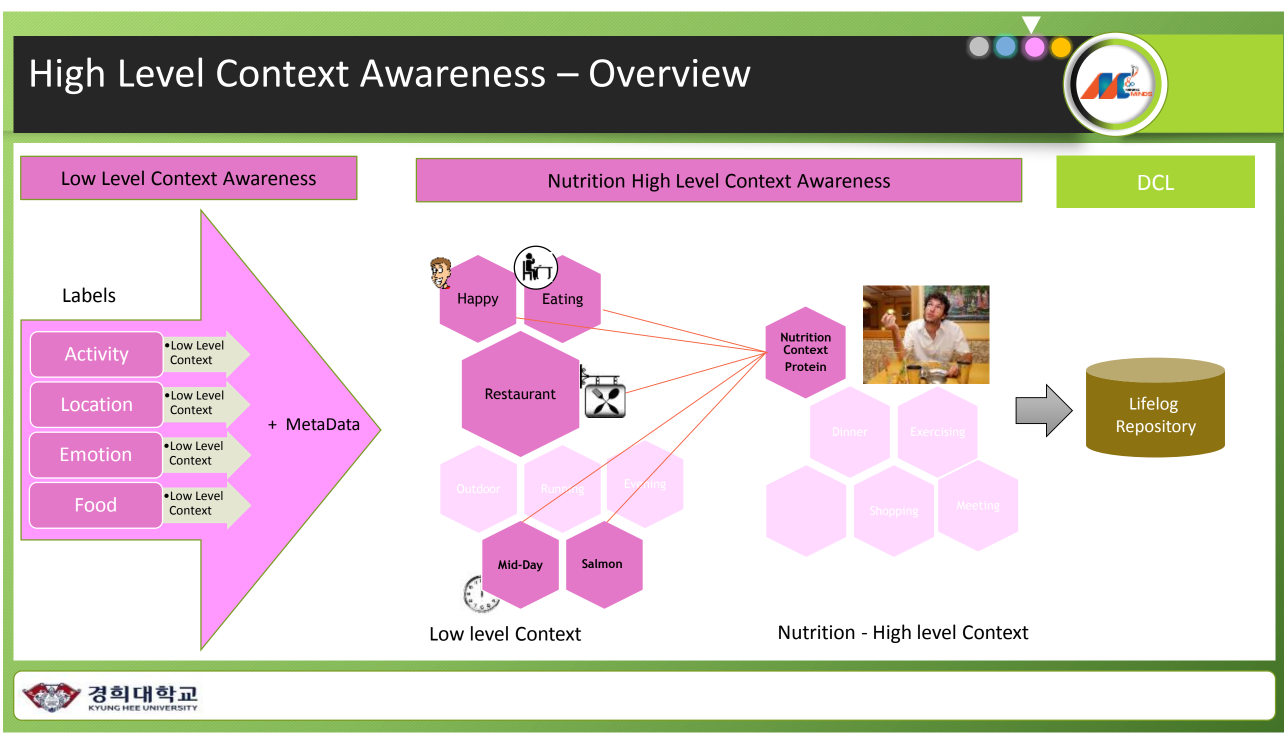
Introduction

- High Level Context is obtained from Processing & Reasoning of Low Level Context based on user's
  - Activity
  - Location
  - Emotion
  - Nutrition



경희대학교 KYUNG HEE UNIVERSITY


High Level Context Awareness – Overview
Low Level Context Awareness
Physical Activity High Level Context Awareness
DCL
Labels
Activity
Location
Emotion
•Low Level Context
•Low Level Context
•Low Level Context
+ MetaData
Happy
Sitting
Office
Outdoor
Running
Evening
Mid-Day
Happiness
Office Work
Dinner
Exercising
Shopping
Meeting
Lifelog Repository
Low level Context
Physical Activity - High level Context
경희대학교
KYUNG HEE UNIVERSITY


High Level Context Awareness – Overview
Low Level Context Awareness
Nutrition High Level Context Awareness
DCL
Labels
Activity
Location
Emotion
Food
•Low Level Context
•Low Level Context
•Low Level Context
•Low Level Context
+ MetaData
Happy
Eating
Restaurant
Outdoor
Running
Evening
Mid-Day
Salmon
Nutrition Context Protein
Dinner
Exercising
Shopping
Meeting
Lifelog Repository
Low level Context
Nutrition - High level Context
경희대학교
KYUNG HEE UNIVERSITY

Motivation

- Abstract description of user's context
- Extraction of High level context from low level context for better *understandability* of *user's context*.
- Identification of High Level Context for *decision making* by upper layers:
  - Personalized recommendations
  - Behavior modeling
  - Personalized predictions

경희대학교 KYUNG HEE UNIVERSITY

Goal and Objectives

- **Goal**: Design and implementation of methodology for high level context recognition.
- **Objectives:**
  - Achieve acceptable accuracy for identifying HLCA
  - Proposal and implementation of context synchronization technique
  - Deployment of Triple storage for ontology persistence
  - Modeling of High-level and Low-level context
  - Reasoning in order to derive High-level context from Low-level context
  - Development of a simulation tool to generate low-level context instances

경희대학교 KYUNG HEE UNIVERSITY

Related Work

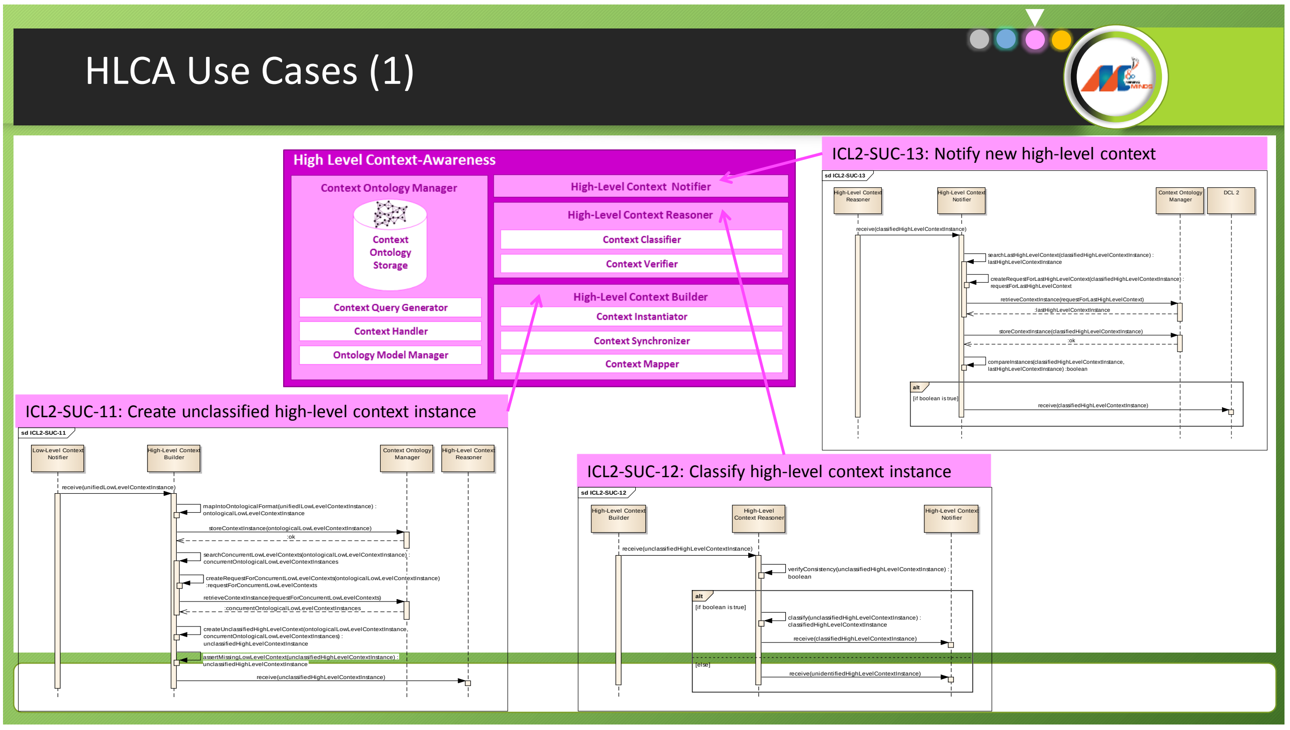
Authors	Domain	Methodology	Features	Limitations
Perera 2014	IoT	Context Aware Computing	• Survey w.r.t IoT • Comprehensive Analysis and Evaluation of Context Aware Techniques	• No Implementation • No Practical Implementation with Results.
Bellavista 2013	Ubiquitous Systems	Unified Architectural model	• Context Data Distribution • Classification of Context • Runtime Adaptation Support	• Context Aggregation and Filtering • Adaptive Context
Khattak 2014	Context Aware Systems	Context Fusion	• Context Fusioning Methods • Survey of Context Representation Schemes	• No Implementation • Evaluation and Proof of Concepts Missing
Moen 2015	Mobile Computing	Activity Recognition Algorithm	• Future Research Methodologies	• Activity Recognition without considering Emotions.
Perera 2013	IoT	Component Level Architecture	• Sensor Selection • Context Aware Architecture	• Semantic and Quantitative Reasoning Missing
Gerhard 2012	Context Aware Systems	Context Aware Framework	• Context Acquisition • Context Representation • Context Utilization	• No Implementation • Evaluation and Proof of Concepts Missing

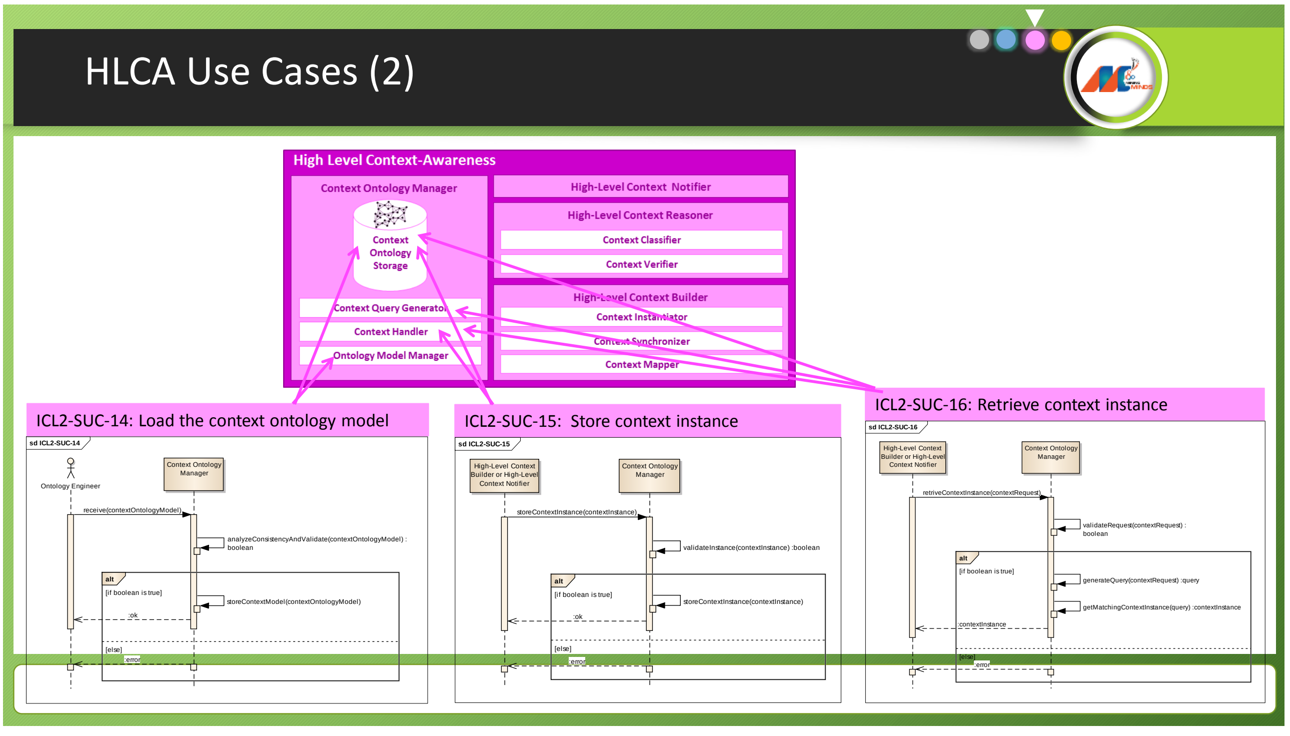
경희대학교 KYUNG HEE UNIVERSITY

Limitations of existing Work

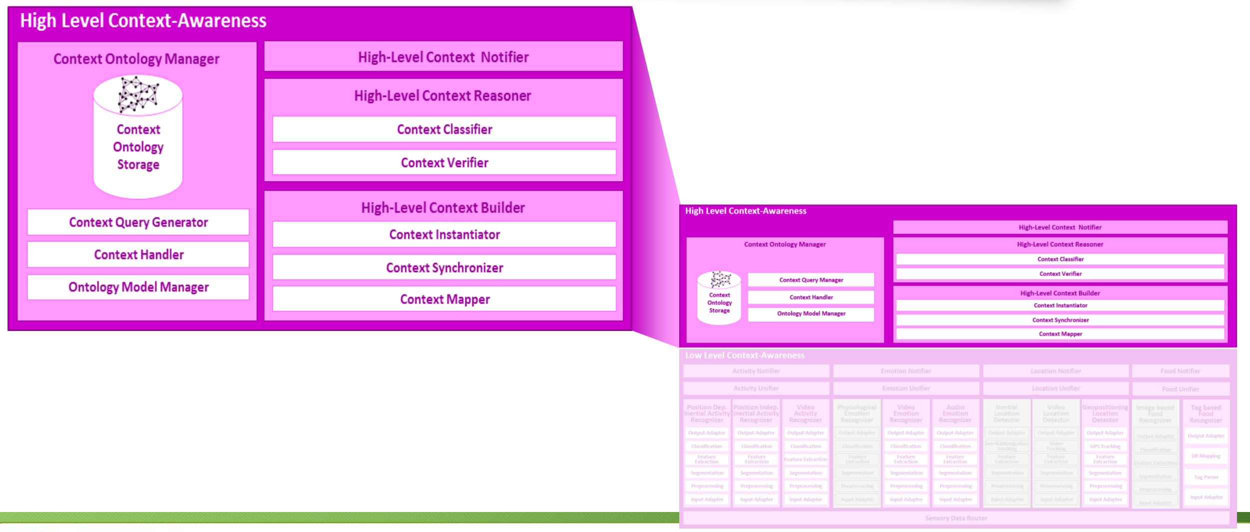
- Lack of Implementation
- Context Aggregation and Abstraction
- Activity Recognition without Emotion Detection in High Level Context Modelling
- Evaluation and Proof of Concept Missing
- Semantic Reasoning Missing

경희대학교 KYUNG HEE UNIVERSITY


HLCA Use Cases (1)
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Context Instantiator
Context Synchronizer
Context Mapper
ICL2-SUC-13: Notify new high-level context
ICL2-SUC-11: Create unclassified high-level context instance
ICL2-SUC-12: Classify high-level context instance

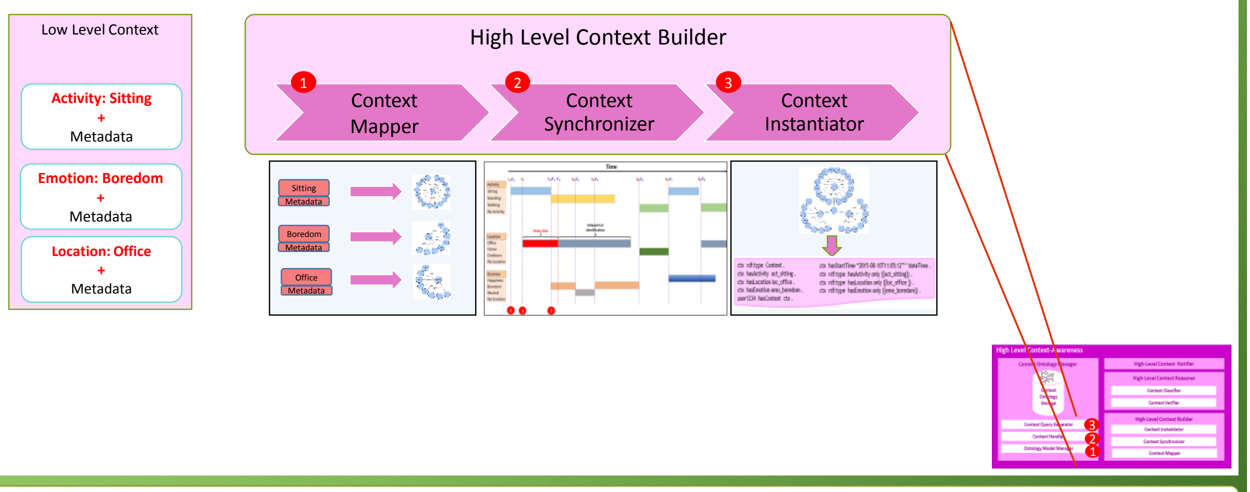

HLCA Use Cases (2)
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Context Instantiator
Context Synchronizer
Context Mapper
ICL2-SUC-14: Load the context ontology model
ICL2-SUC-15: Store context instance
ICL2-SUC-16: Retrieve context instance

HLCA Architecture



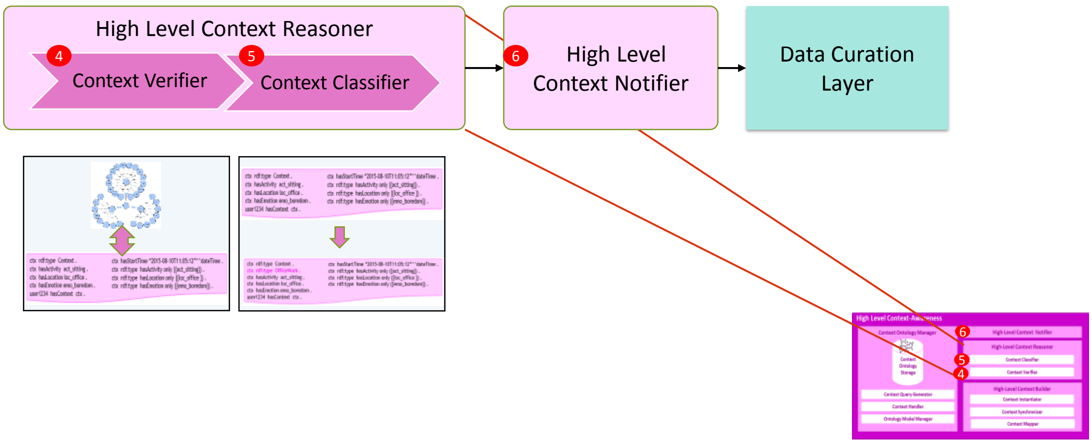
경희대학교
KYUNG HEE UNIVERSITY

Execution Workflow – Conceptual View



경희대학교
KYUNG HEE UNIVERSITY

Execution Workflow – Conceptual View



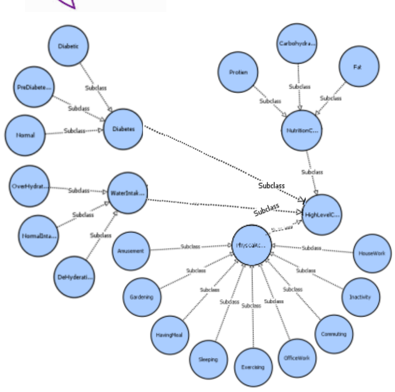
Context Ontology Model

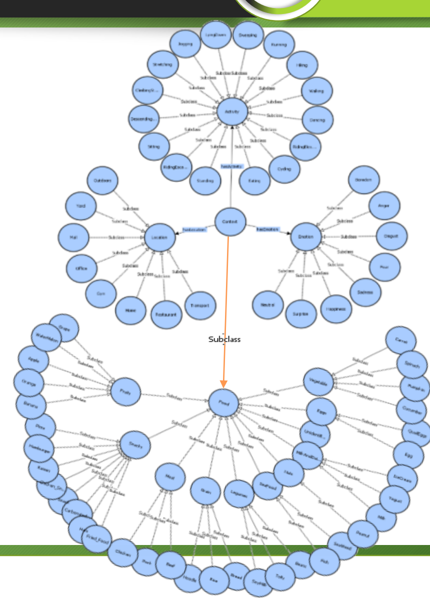
Context Ontology Metrics:

- Context Ontology Metrics for Protégé:
- 9 Physical Activity High-Level Contexts
- 3 Nutrition HLC (Carbohydrates, Proteins, Fats)
- 16 Activities (Low-Level Context)
- 8 Locations (Low-Level Context)
- 8 Emotions (Low-Level Context)
- 10 Food Categories (Low-Level Context)
- 57 Food Items (Low-Level Context)

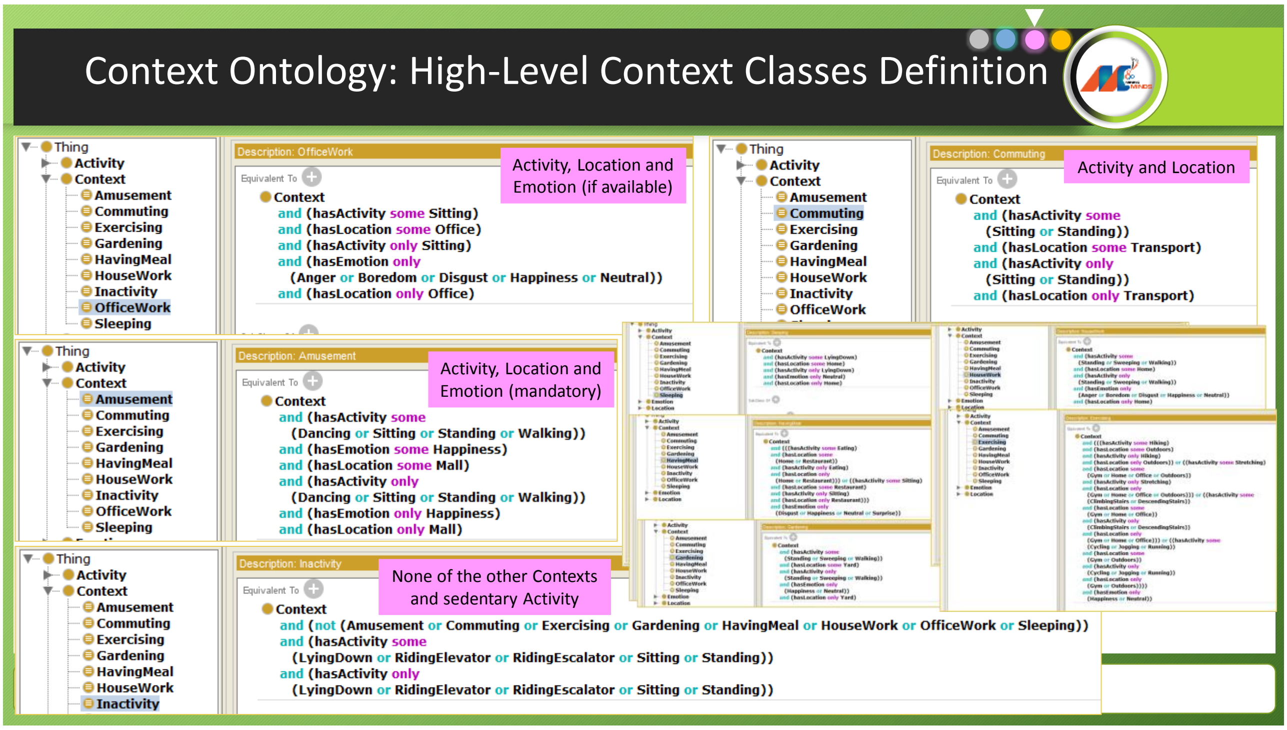
Ontology modeling tool:

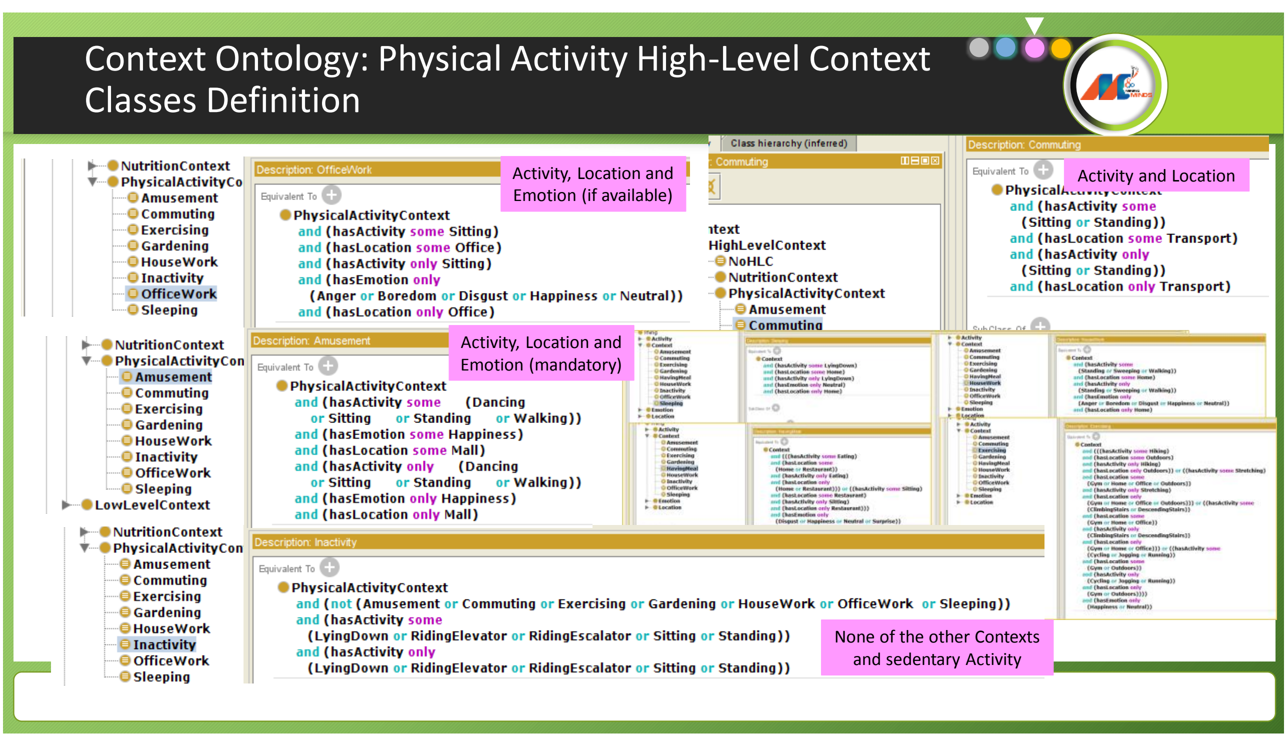
protégé

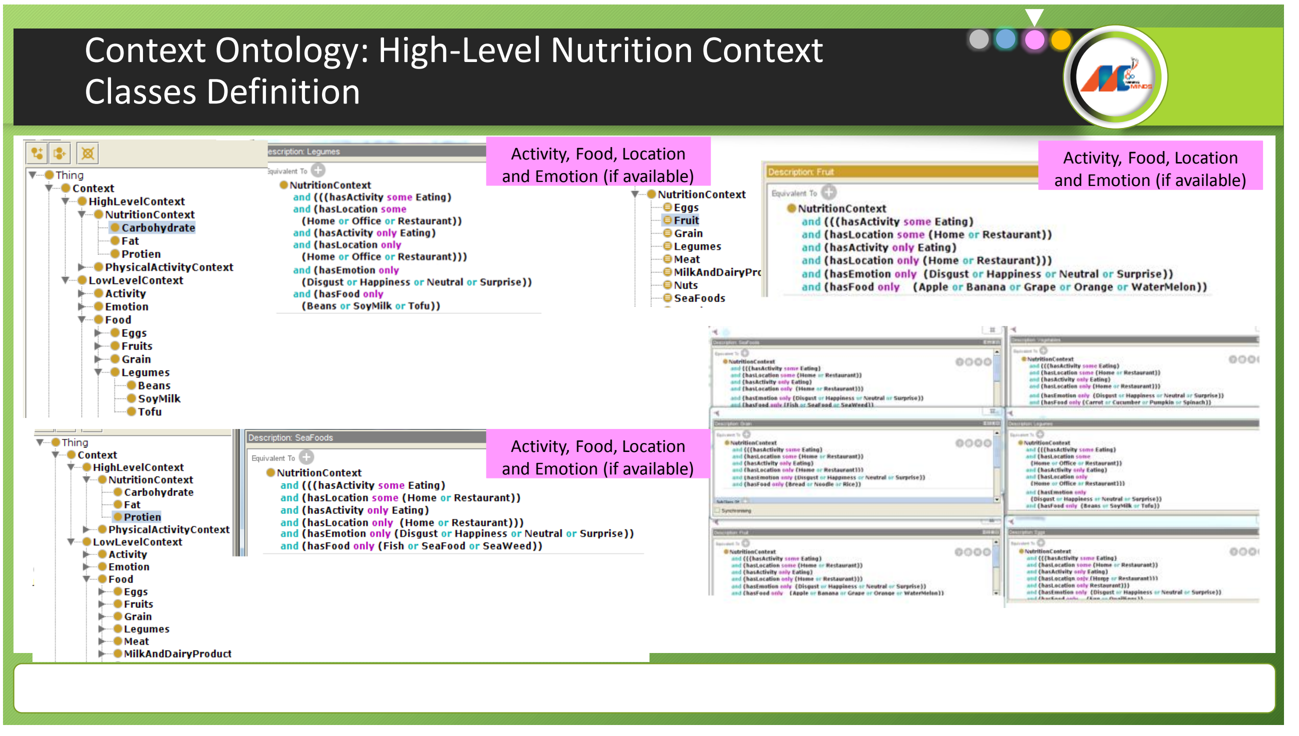


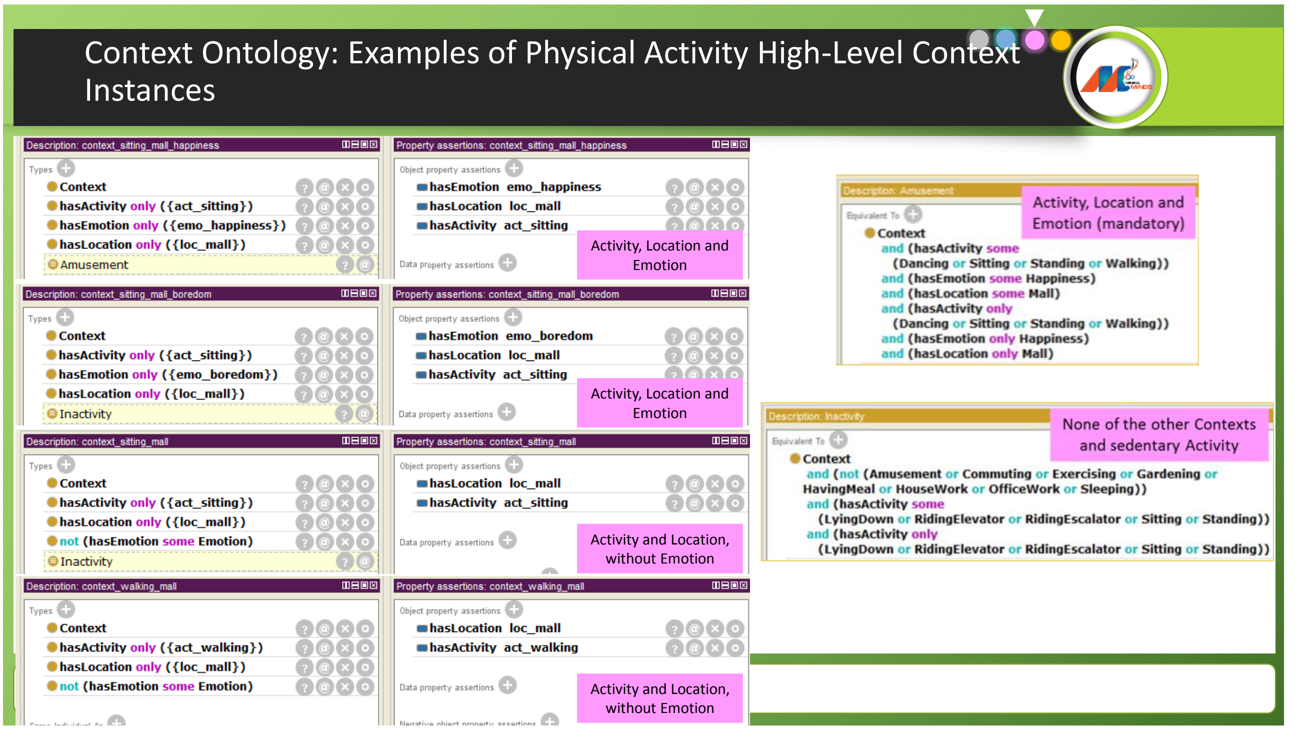


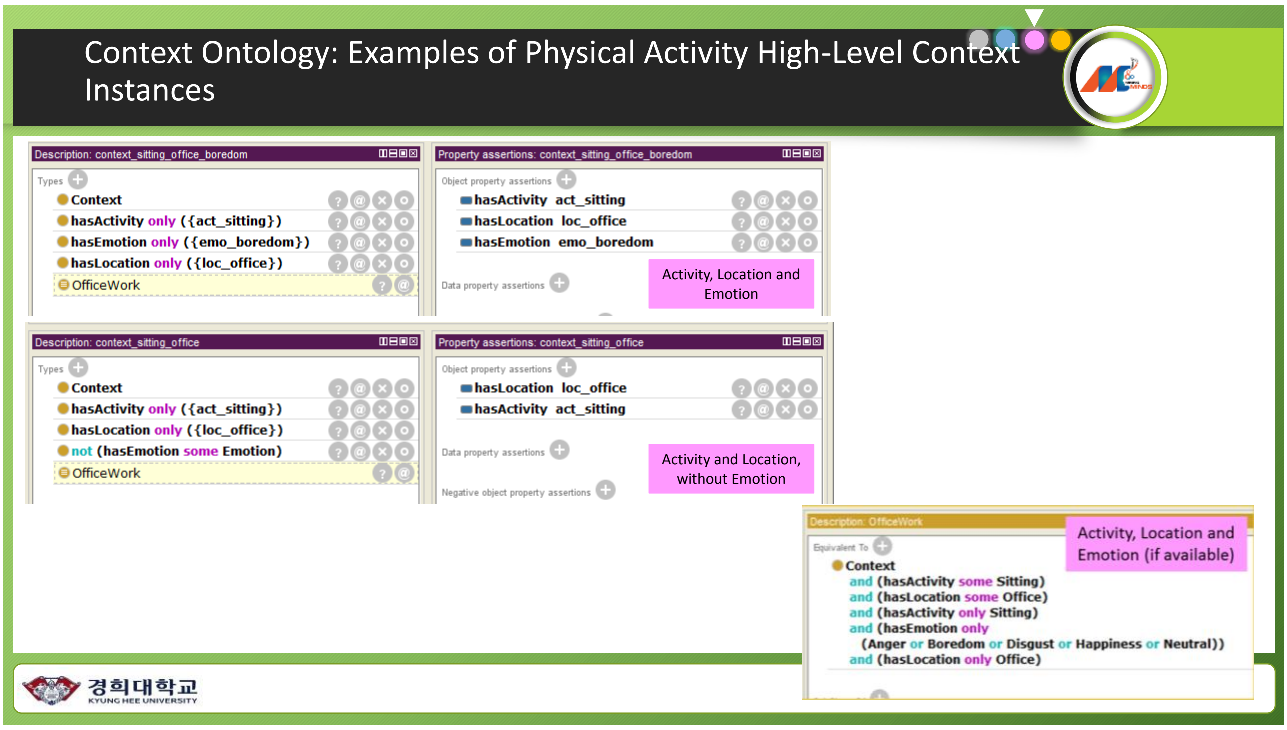
Publication: Villalonga, C., Banos, O., Ali Khan, W., Ali, T., Razzaq, M. A., Lee, S., Pomares, H., Rojas, I. High-Level Context Inference for Human Behavior Identification. *International Work-conference on Ambient Assisted Living an Active Ageing (IWAAL 2015)*, Patagonia, Chile, December 1-4, (2015)


Context Ontology: High-Level Context Classes Definition
Thing
Activity
Context
Amusement
Commuting
Exercising
Gardening
HavingMeal
HouseWork
Inactivity
OfficeWork
Sleeping
Description: OfficeWork
Equivalent To
Activity, Location and Emotion (if available)
Context
and (hasActivity some Sitting)
and (hasLocation some Office)
and (hasActivity only Sitting)
and (hasEmotion only
(Anger or Boredom or Disgust or Happiness or Neutral))
and (hasLocation only Office)
Description: Commuting
Activity and Location
Context
and (hasActivity some
(Sitting or Standing))
and (hasLocation some Transport)
and (hasActivity only
(Sitting or Standing))
and (hasLocation only Transport)
Description: Amusement
Activity, Location and Emotion (mandatory)
Context
and (hasActivity some
(Dancing or Sitting or Standing or Walking))
and (hasEmotion some Happiness)
and (hasLocation some Mall)
and (hasActivity only
(Dancing or Sitting or Standing or Walking))
and (hasEmotion only Happiness)
and (hasLocation only Mall)
Description: Inactivity
None of the other Contexts and sedentary Activity
Context
and (not (Amusement or Commuting or Exercising or Gardening or HavingMeal or HouseWork or OfficeWork or Sleeping))
and (hasActivity some
(LyingDown or RidingElevator or RidingEscalator or Sitting or Standing))
and (hasActivity only
(LyingDown or RidingElevator or RidingEscalator or Sitting or Standing))

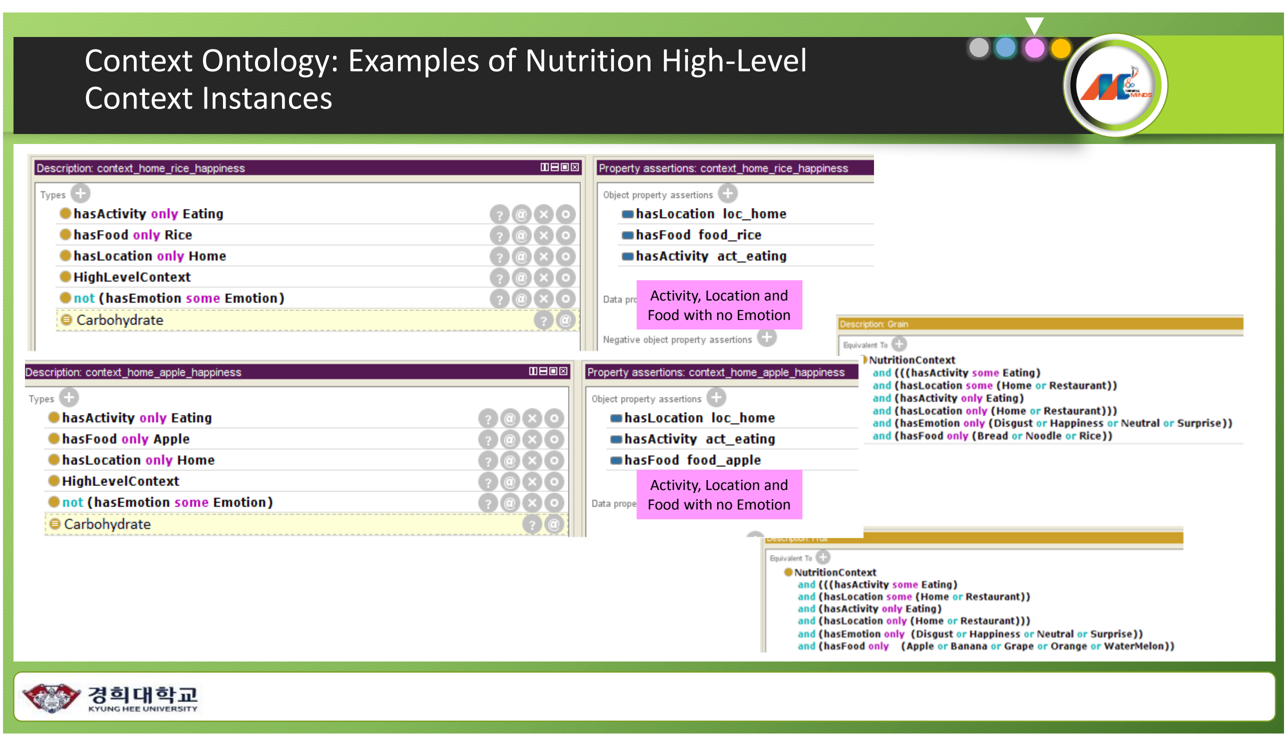

Context Ontology: Physical Activity High-Level Context Classes Definition
NutritionContext
PhysicalActivityContext
Amusement
Commuting
Exercising
Gardening
HouseWork
Inactivity
OfficeWork
Sleeping
LowLevelContext
Description: OfficeWork
Activity, Location and Emotion (if available)
PhysicalActivityContext
and (hasActivity some Sitting)
and (hasLocation some Office)
and (hasActivity only Sitting)
and (hasEmotion only
(Anger or Boredom or Disgust or Happiness or Neutral))
and (hasLocation only Office)
Class hierarchy (inferred)
HighLevelContext
NoHLC
NutritionContext
PhysicalActivityContext
Description: Commuting
Activity and Location
and (hasActivity some
(Sitting or Standing))
and (hasLocation some Transport)
and (hasActivity only
(Sitting or Standing))
and (hasLocation only Transport)
Description: Amusement
Activity, Location and Emotion (mandatory)
PhysicalActivityContext
and (hasActivity some (Dancing
or Sitting or Standing or Walking))
and (hasEmotion some Happiness)
and (hasLocation some Mall)
and (hasActivity only (Dancing
or Sitting or Standing or Walking))
and (hasEmotion only Happiness)
and (hasLocation only Mall)
Description: Inactivity
PhysicalActivityContext
and (not (Amusement or Commuting or Exercising or Gardening or HouseWork or OfficeWork or Sleeping))
and (hasActivity some
(LyingDown or RidingElevator or RidingEscalator or Sitting or Standing))
and (hasActivity only
(LyingDown or RidingElevator or RidingEscalator or Sitting or Standing))
None of the other Contexts and sedentary Activity


Context Ontology: High-Level Nutrition Context Classes Definition
Activity, Food, Location and Emotion (if available)
Activity, Food, Location and Emotion (if available)
Activity, Food, Location and Emotion (if available)
Thing
Context
HighLevelContext
NutritionContext
Carbohydrate
Fat
Protien
PhysicalActivityContext
LowLevelContext
Activity
Emotion
Food
Eggs
Fruits
Grain
Legumes
Beans
SoyMilk
Tofu
Meat
MilkAndDairyProduct
NutritionContext
and (((hasActivity some Eating)
and (hasLocation some
(Home or Office or Restaurant))
and (hasActivity only Eating)
and (hasLocation only
(Home or Office or Restaurant)))
and (hasEmotion only
(Disgust or Happiness or Neutral or Surprise))
and (hasFood only
(Beans or SoyMilk or Tofu))
NutritionContext
Eggs
Fruit
Grain
Legumes
Meat
Nuts
SeaFoods
NutritionContext
and (((hasActivity some Eating)
and (hasLocation some (Home or Restaurant))
and (hasActivity only Eating)
and (hasLocation only (Home or Restaurant)))
and (hasEmotion only (Disgust or Happiness or Neutral or Surprise))
and (hasFood only (Apple or Banana or Grape or Orange or WaterMelon))
NutritionContext
and (((hasActivity some Eating)
and (hasLocation some (Home or Restaurant))
and (hasActivity only Eating)
and (hasLocation only (Home or Restaurant)))
and (hasEmotion only (Disgust or Happiness or Neutral or Surprise))
and (hasFood only (Fish or SeaFood or SeaWeed))

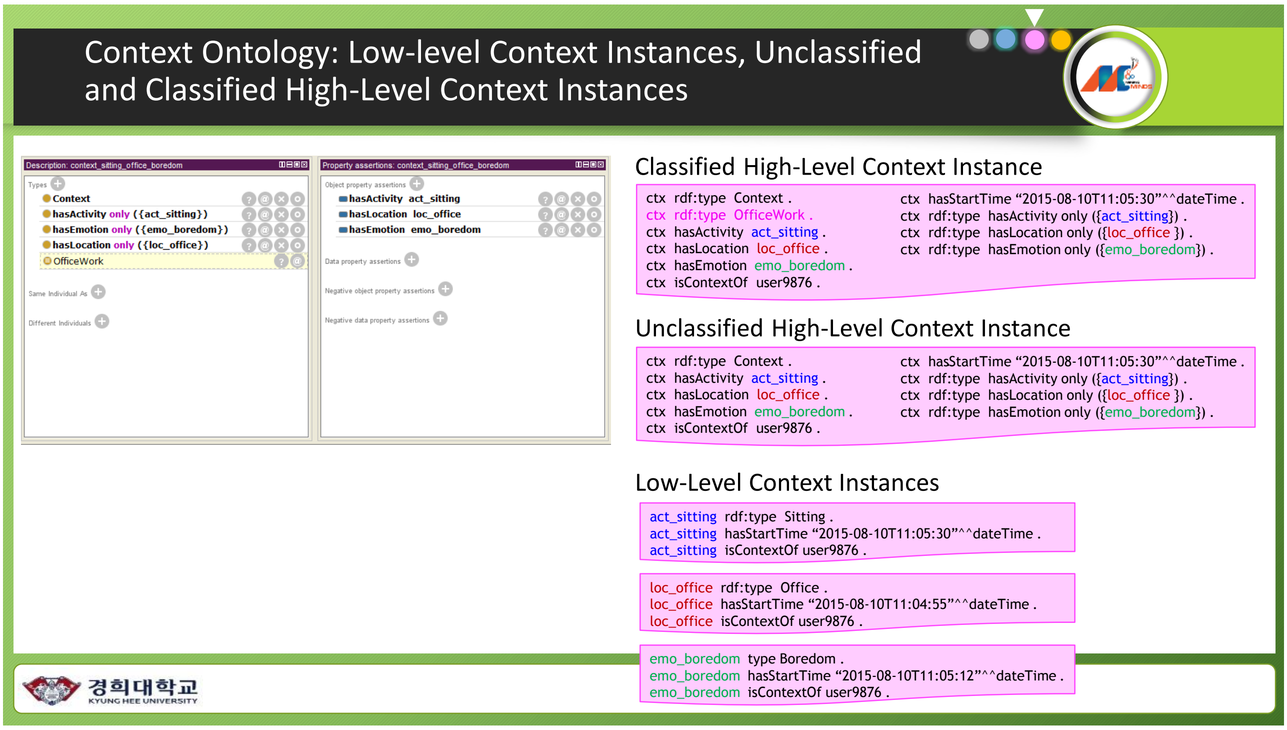

Context Ontology: Examples of Physical Activity High-Level Context Instances
Description: context_sitting_mall_happiness
Context
hasActivity only ({act_sitting})
hasEmotion only ({emo_happiness})
hasLocation only ({loc_mall})
Amusement
Property assertions: context_sitting_mall_happiness
hasEmotion emo_happiness
hasLocation loc_mall
hasActivity act_sitting
Activity, Location and Emotion
Description: context_sitting_mall_boredom
Context
hasActivity only ({act_sitting})
hasEmotion only ({emo_boredom})
hasLocation only ({loc_mall})
Inactivity
Property assertions: context_sitting_mall_boredom
hasEmotion emo_boredom
hasLocation loc_mall
hasActivity act_sitting
Activity, Location and Emotion
Description: context_sitting_mall
Context
hasActivity only ({act_sitting})
hasLocation only ({loc_mall})
not (hasEmotion some Emotion)
Inactivity
Property assertions: context_sitting_mall
hasLocation loc_mall
hasActivity act_sitting
Activity and Location, without Emotion
Description: context_walking_mall
Context
hasActivity only ({act_walking})
hasLocation only ({loc_mall})
not (hasEmotion some Emotion)
Property assertions: context_walking_mall
hasLocation loc_mall
hasActivity act_walking
Activity and Location, without Emotion
Description: Amusement
Activity, Location and Emotion (mandatory)
Context
and (hasActivity some
(Dancing or Sitting or Standing or Walking))
and (hasEmotion some Happiness)
and (hasLocation some Mall)
and (hasActivity only
(Dancing or Sitting or Standing or Walking))
and (hasEmotion only Happiness)
and (hasLocation only Mall)
Description: Inactivity
None of the other Contexts and sedentary Activity
Context
and (not (Amusement or Commuting or Exercising or Gardening or
HavingMeal or HouseWork or OfficeWork or Sleeping))
and (hasActivity some
(LyingDown or RidingElevator or RidingEscalator or Sitting or Standing))
and (hasActivity only
(LyingDown or RidingElevator or RidingEscalator or Sitting or Standing))

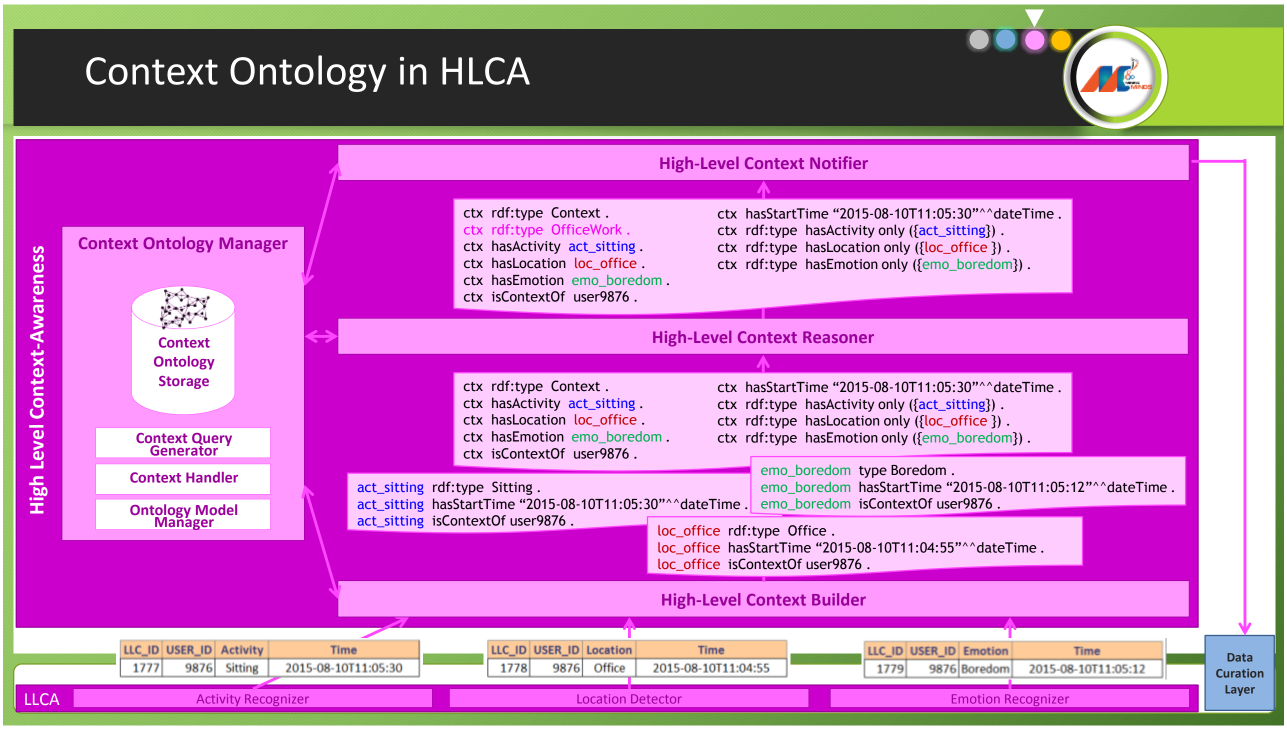


Context Ontology: Examples of Physical Activity High-Level Context Instances
Description: context_sitting_office_boredom
Types
Context
hasActivity only ({act_sitting})
hasEmotion only ({emo_boredom})
hasLocation only ({loc_office})
OfficeWork
Property assertions: context_sitting_office_boredom
Object property assertions
hasActivity act_sitting
hasLocation loc_office
hasEmotion emo_boredom
Data property assertions
Activity, Location and Emotion
Description: context_sitting_office
Types
Context
hasActivity only ({act_sitting})
hasLocation only ({loc_office})
not (hasEmotion some Emotion)
OfficeWork
Property assertions: context_sitting_office
Object property assertions
hasLocation loc_office
hasActivity act_sitting
Data property assertions
Negative object property assertions
Activity and Location, without Emotion
Description: OfficeWork
Equivalent To
Context
and (hasActivity some Sitting)
and (hasLocation some Office)
and (hasActivity only Sitting)
and (hasEmotion only
(Anger or Boredom or Disgust or Happiness or Neutral))
and (hasLocation only Office)
Activity, Location and Emotion (if available)
경희대학교
KYUNG HEE UNIVERSITY

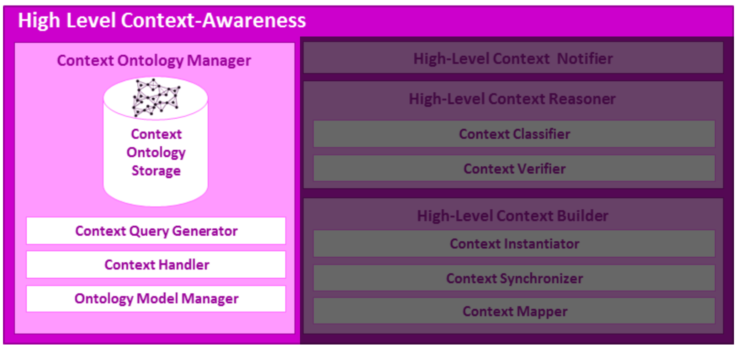


Context Ontology: Examples of Nutrition High-Level Context Instances
Description: context_home_rice_happiness
Types
hasActivity only Eating
hasFood only Rice
hasLocation only Home
HighLevelContext
not (hasEmotion some Emotion)
Carbohydrate
Property assertions: context_home_rice_happiness
Object property assertions
hasLocation loc_home
hasFood food_rice
hasActivity act_eating
Negative object property assertions
Activity, Location and Food with no Emotion
Description: context_home_apple_happiness
Types
hasActivity only Eating
hasFood only Apple
hasLocation only Home
HighLevelContext
not (hasEmotion some Emotion)
Carbohydrate
Property assertions: context_home_apple_happiness
Object property assertions
hasLocation loc_home
hasActivity act_eating
hasFood food_apple
Activity, Location and Food with no Emotion
Description: Grain
Equivalent To
NutritionContext
and (((hasActivity some Eating)
and (hasLocation some (Home or Restaurant))
and (hasActivity only Eating)
and (hasLocation only (Home or Restaurant)))
and (hasEmotion only (Disgust or Happiness or Neutral or Surprise))
and (hasFood only (Bread or Noodle or Rice))
Equivalent To
NutritionContext
and (((hasActivity some Eating)
and (hasLocation some (Home or Restaurant))
and (hasActivity only Eating)
and (hasLocation only (Home or Restaurant)))
and (hasEmotion only (Disgust or Happiness or Neutral or Surprise))
and (hasFood only (Apple or Banana or Grape or Orange or WaterMelon))
경희대학교
KYUNG HEE UNIVERSITY


Context Ontology: Low-level Context Instances, Unclassified and Classified High-Level Context Instances
Description: context_sitting_office_boredom
Types
Context
hasActivity only ({act_sitting})
hasEmotion only ({emo_boredom})
hasLocation only ({loc_office})
OfficeWork
Same Individual As
Different Individuals
Property assertions: context_sitting_office_boredom
Object property assertions
hasActivity act_sitting
hasLocation loc_office
hasEmotion emo_boredom
Data property assertions
Negative object property assertions
Negative data property assertions
Classified High-Level Context Instance
ctx rdf:type Context .
ctx rdf:type OfficeWork .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime “2015-08-10T11:05:30”^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
Unclassified High-Level Context Instance
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime “2015-08-10T11:05:30”^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
Low-Level Context Instances
act_sitting rdf:type Sitting .
act_sitting hasStartTime “2015-08-10T11:05:30”^^dateTime .
act_sitting isContextOf user9876 .
loc_office rdf:type Office .
loc_office hasStartTime “2015-08-10T11:04:55”^^dateTime .
loc_office isContextOf user9876 .
emo_boredom type Boredom .
emo_boredom hasStartTime “2015-08-10T11:05:12”^^dateTime .
emo_boredom isContextOf user9876 .
경희대학교
KYUNG HEE UNIVERSITY


Context Ontology in HLCA
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Notifier
ctx rdf:type Context .
ctx rdf:type OfficeWork .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime “2015-08-10T11:05:30”^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
High-Level Context Reasoner
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime “2015-08-10T11:05:30”^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
emo_boredom type Boredom .
emo_boredom hasStartTime “2015-08-10T11:05:12”^^dateTime .
emo_boredom isContextOf user9876 .
act_sitting rdf:type Sitting .
act_sitting hasStartTime “2015-08-10T11:05:30”^^dateTime .
act_sitting isContextOf user9876 .
loc_office rdf:type Office .
loc_office hasStartTime “2015-08-10T11:04:55”^^dateTime .
loc_office isContextOf user9876 .
High-Level Context Builder
LLC_ID USER_ID Activity Time
1777 9876 Sitting 2015-08-10T11:05:30
LLC_ID USER_ID Location Time
1778 9876 Office 2015-08-10T11:04:55
LLC_ID USER_ID Emotion Time
1779 9876 Boredom 2015-08-10T11:05:12
LLCA
Activity Recognizer
Location Detector
Emotion Recognizer
Data Curation Layer

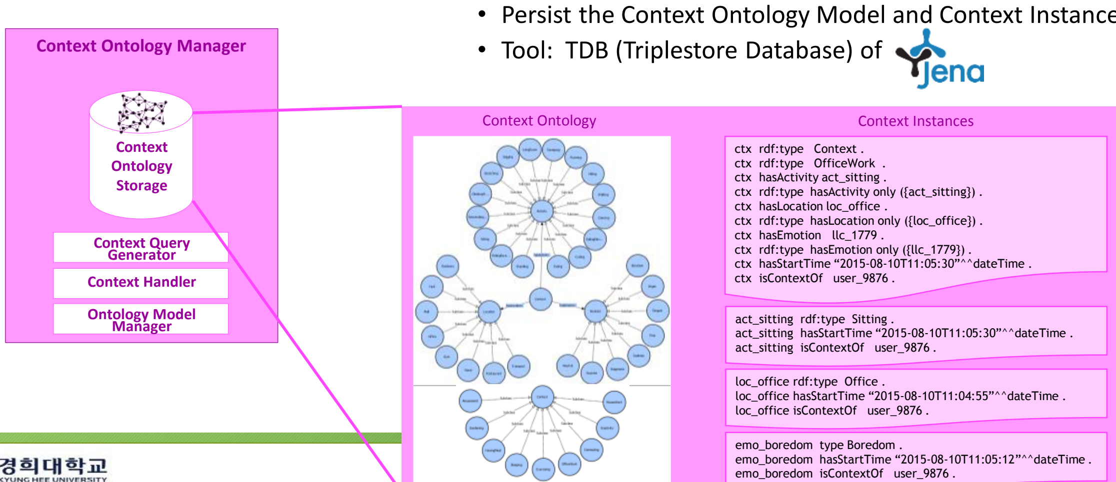
Context Ontology Manager



- Objectives:
  - Provide persistent storage for the Context Ontology and Context Instances
  - Management of the interactions with the persisted Context Information
- Subcomponents:
  - Context Ontology Storage
  - Ontology Model Manager
  - Context Handler
  - Context Query Generator

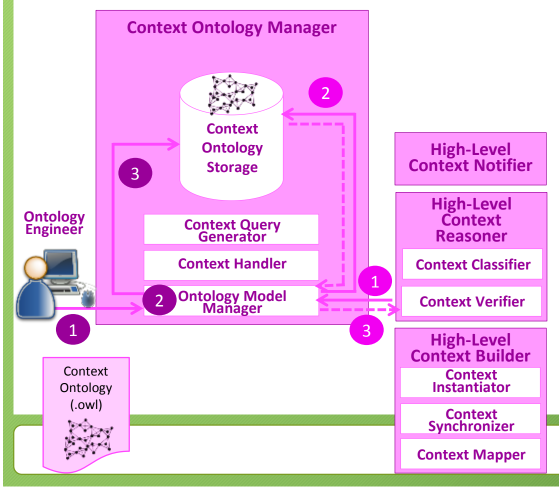
Context Ontology Manager

- **Context Ontology Storage**
  - Persist the Context Ontology Model and Context Instances
  - Tool: TDB (Triplestore Database) of Jena



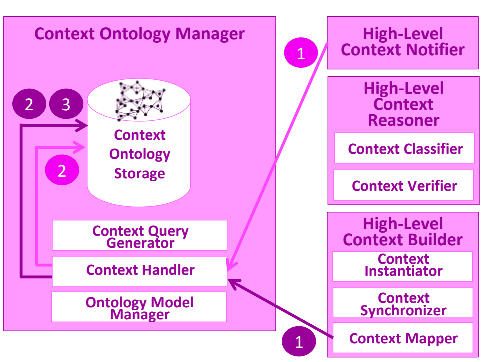
경희대학교
KYUNG HEE UNIVERSITY

Context Ontology Manager



- **Ontology Model Manager**
  - Load the Context Ontology Model and retrieve it
  - Workflow 1 – Storage of the Context Ontology Model:
    1. Receive Context Ontology Model (in OWL format) generated by the ontology engineer
    2. Validate the Context Ontology Model
    3. Store the Context Ontology Model into the triplestore
  - Workflow 2 – Retrieval of the Context Ontology Model:
    1. Receive a request for the Context Ontology from the High-Level Context Reasoner
    2. Retrieve the stored Context Ontology Model
    3. Return the Context Ontology Model to the High-Level Context Reasoner
  - Tool: Jena (including RDF API and Ontology API)

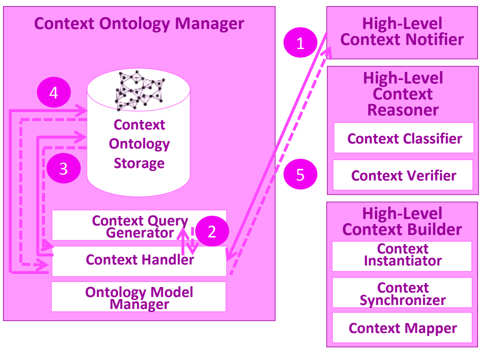
Context Ontology Manager



- **Context Handler**
  - Store and update the Context Instances
  - Workflow 1 – Store HLC:
    1. Receive High Level Context Instance from the High-Level Context Notifier
    2. Store the High Level Context Instance into the triplestore
  - Workflow 2 – Store and update LLC:
    1. Receive Low Level Context Instance from the Context Mapper
    2. Store the Low Level Context Instance into the triplestore
    3. Update the end time of the previous Low Level Context Instance
  - Tool: Jena (including Ontology API)

경희대학교 KYUNG HEE UNIVERSITY

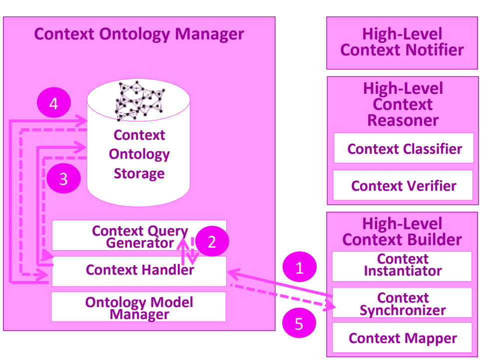
Context Ontology Manager



- **Context Handler**
  - Retrieve the previous High Level Context Instance
  - Workflow:
    1. Receive a request from the High-Level Context Notifier for the previous High Level Context Instance
    2. Invoke the Context Query Generator in order to create the SPARQL query
    3. Execute the SPARQL query on the data stored on the Context Ontology Storage and obtain the identifiers of the matching High Level Context Instances
    4. Retrieve the High Level Context instances which match the SPARQL Query
    5. Return the matching High Level Context Instance
  - Tool: Jena (including Ontology API)

경희대학교 KYUNG HEE UNIVERSITY

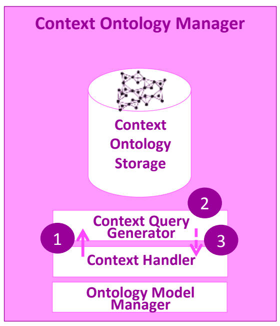
Context Ontology Manager



- **Context Handler**
  - Retrieve concurrent Low Level Context Instances
  - Workflow:
    1. Receive a request from the Context Synchronizer for the concurrent Low Level Context Instances
    2. Invoke the Context Query Generator in order to create the SPARQL query
    3. Execute the SPARQL query on the data stored on the Context Ontology Storage and obtain the identifiers of the matching Low Level Context Instances
    4. Retrieve the Low Level Context instances which match the SPARQL Query
    5. Return the matching Low Level Context Instances
  - Tool: Jena (including Ontology API)

경희대학교 KYUNG HEE UNIVERSITY

Context Ontology Manager

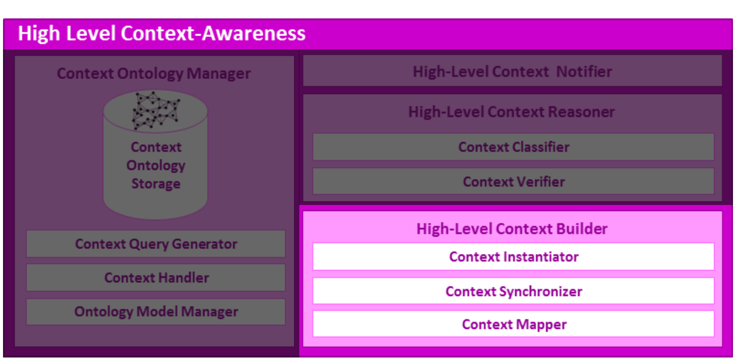


- **Context Query Generator**
  - Automatic generation of SPARQL Queries
  - Workflow:
    1. Receive a query request from the Context Handler
    2. Generate the appropriate SPARQL query

```
SELECT ?hlc
        WHERE {
                ?hlc rdf:type HighLevelContext .
                ?hlc isContextOf user_9876 .
                ?hlc icl2:hasStartTime ?starttime .
                FILTER NOT EXISTS { ?hlc hasEndTime ?endtime}
                FILTER (?starttime  < "2015-08-
10T11:05:30"^^xsd:dateTime)
        }
```

    3. Return the SPARQL query to the Context Handler
  - Tool: Jena (including ARQ)

High-Level Context Builder

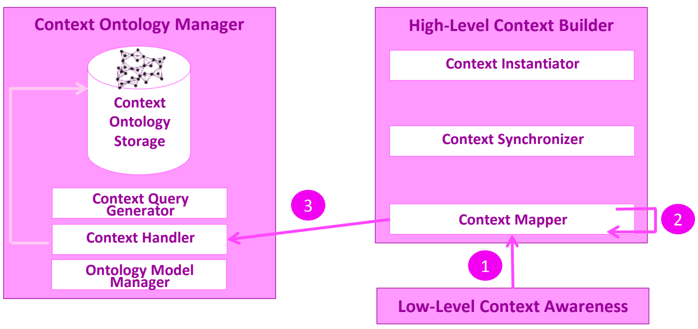


- Objectives:
  - Convert low-level context information into ontological format
  - Align concurrent ontological low-level contexts of the same user
  - Generate unclassified high-level context instances from concurrent low-level contexts
- Subcomponents:
  - Context Mapper
  - Context Synchronizer
  - Context Instantiator

경희대학교 KYUNG HEE UNIVERSITY

High-Level Context Builder

- **Context Mapper:**



1 Receive a new LLC in an unstructured format (label plus metadata).

LLC_ID	USER_ID	Activity	Time
1777	9876	Sitting	2015-08-10T11:05:30

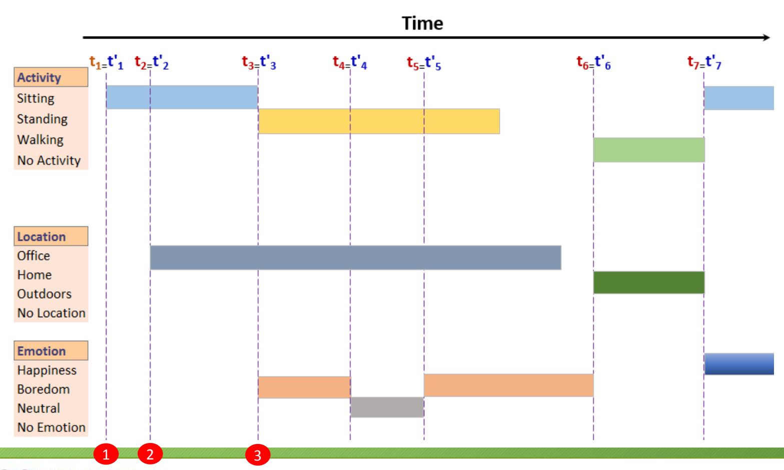
2 Generate the LLC Instance which contains the received information mapped into an ontological format.

```
act_sitting  rdf:type  Sitting .
act_sitting  hasStartTime "2015-08-10T11:05:30"^^dateTime .
act_sitting  isContextOf user9876 .
```

3 Persist the new LLC instance.

High-Level Context Builder

- **Context Synchronization Problem**



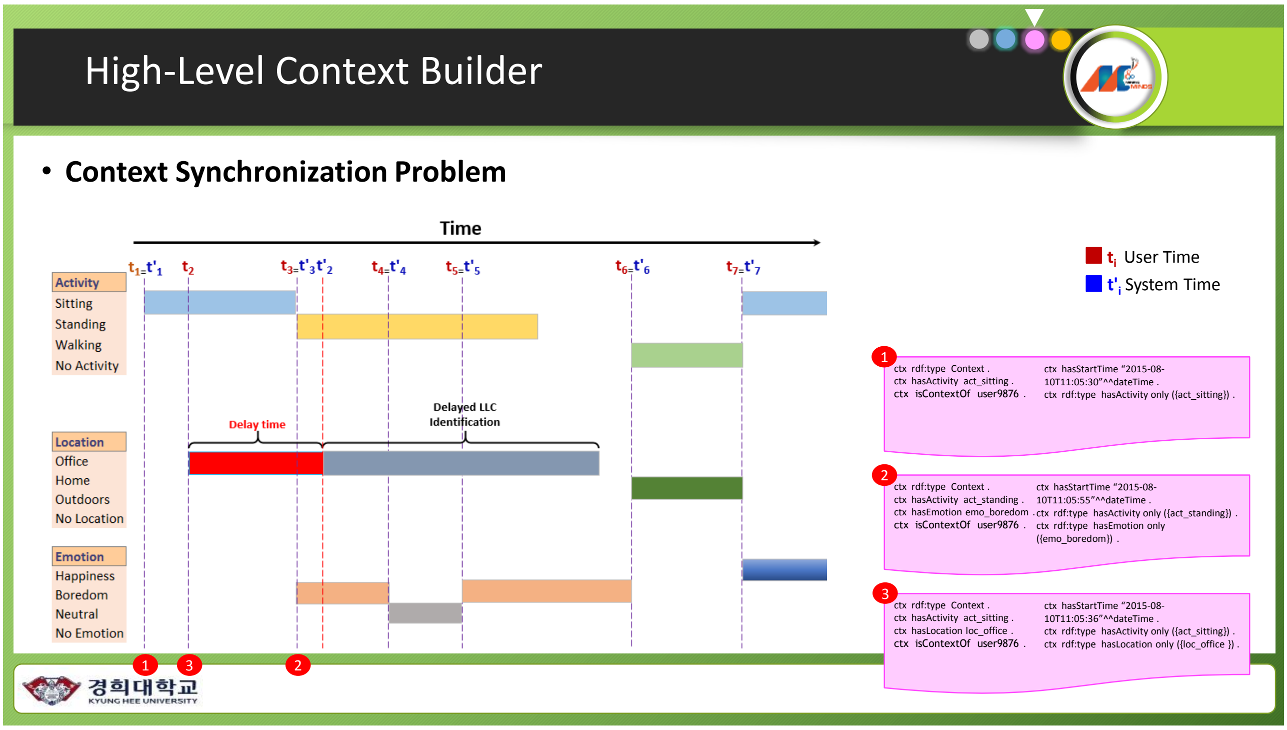
t_i User Time
t'_i System Time

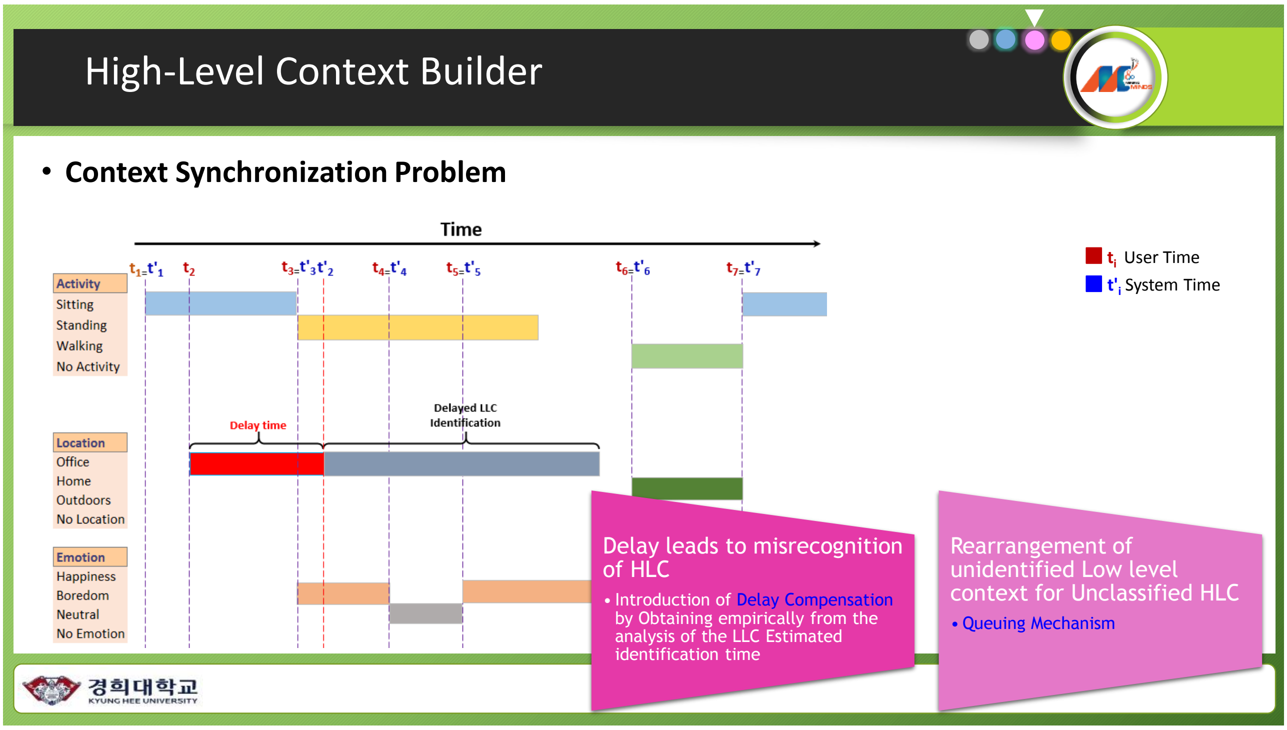
1
```
ctx  rdf:type  Context .
ctx  hasActivity  act_sitting .
ctx  isContextOf  user9876 .
ctx  hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx  rdf:type  hasActivity only ({act_sitting}) .
```

2
```
ctx  rdf:type  Context .
ctx  hasActivity  act_sitting .
ctx  hasLocation loc_office .
ctx  isContextOf  user9876 .
ctx  hasStartTime "2015-08-10T11:05:36"^^dateTime .
ctx  rdf:type  hasActivity only ({act_sitting}) .
ctx  rdf:type  hasLocation only ({loc_office }) .
```

3
```
ctx  rdf:type  Context .
ctx  hasActivity  act_standing .
ctx  hasLocation loc_office .
ctx  hasEmotion emo_boredom .
ctx  isContextOf  user9876 .
ctx  hasStartTime "2015-08-10T11:05:55"^^dateTime .
ctx  rdf:type  hasActivity only ({act_standing}) .
ctx  rdf:type  hasLocation only ({loc_office }) .
ctx  rdf:type  hasEmotion only ({emo_boredom}) .
```

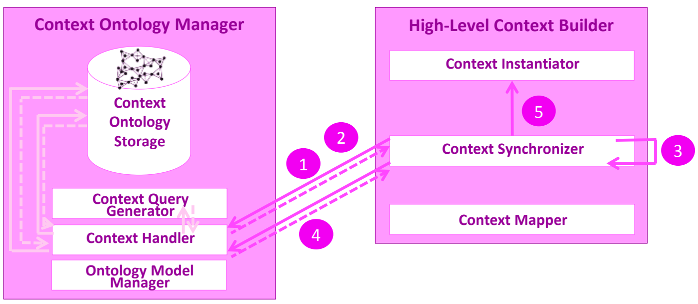
경희대학교 KYUNG HEE UNIVERSITY


High-Level Context Builder
• Context Synchronization Problem
Time
$t_1 = t'_1$ t_2 $t_3 = t'_3$ t'_2 $t_4 = t'_4$ $t_5 = t'_5$ $t_6 = t'_6$ $t_7 = t'_7$
Activity
Sitting
Standing
Walking
No Activity
Location
Office
Home
Outdoors
No Location
Emotion
Happiness
Boredom
Neutral
No Emotion
Delay time
Delayed LLC Identification
1
3
2
t_i User Time
t'_i System Time
1
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
2
ctx rdf:type Context .
ctx hasActivity act_standing .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:55"^^dateTime .
ctx rdf:type hasActivity only ({act_standing}) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
3
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:36"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
경희대학교
KYUNG HEE UNIVERSITY


High-Level Context Builder
• Context Synchronization Problem
Time
$t_1 = t'_1$ t_2 $t_3 = t'_3$ t'_2 $t_4 = t'_4$ $t_5 = t'_5$ $t_6 = t'_6$ $t_7 = t'_7$
Activity
Sitting
Standing
Walking
No Activity
Location
Office
Home
Outdoors
No Location
Emotion
Happiness
Boredom
Neutral
No Emotion
Delay time
Delayed LLC Identification
t_i User Time
t'_i System Time
Delay leads to misrecognition of HLC
• Introduction of Delay Compensation by Obtaining empirically from the analysis of the LLC Estimated identification time
Rearrangement of unidentified Low level context for Unclassified HLC
• Queuing Mechanism
경희대학교
KYUNG HEE UNIVERSITY

High-Level Context Builder

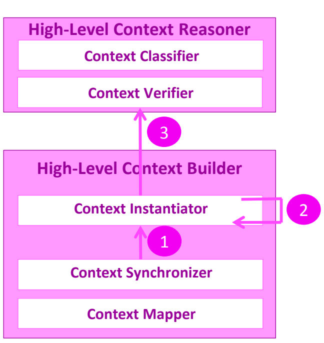
- **Context Synchronizer:**



1. Retrieve the LLC instances which start within a time window.
2. Retrieve the LLC instances which end within the time window.
3. Order chronologically the LLC start and end events
4. For each LLC instance retrieve the concurrent LLC
5. Notify the Context Instantiator

High-Level Context Builder

- **Context Instantiatior:**



1. Receive the trigger LLC instance and its concurrent ones.

Trigger LLC

```
act_sitting  rdf:type  Sitting .
act_sitting  hasStartTime “2015-08-10T11:05:30”^^dateTime .
act_sitting  isContextOf user9876 .
```

Concurrent LLC

```
loc_office  rdf:type  Office .
loc_office  hasStartTime “2015-08-10T11:04:55”^^dateTime .
loc_office  isContextOf user9876 .
```

Concurrent LLC

```
emo_boredom  type Boredom .
emo_boredom  hasStartTime “2015-08-10T11:05:12”^^dateTime .
emo_boredom  isContextOf user9876 .
```

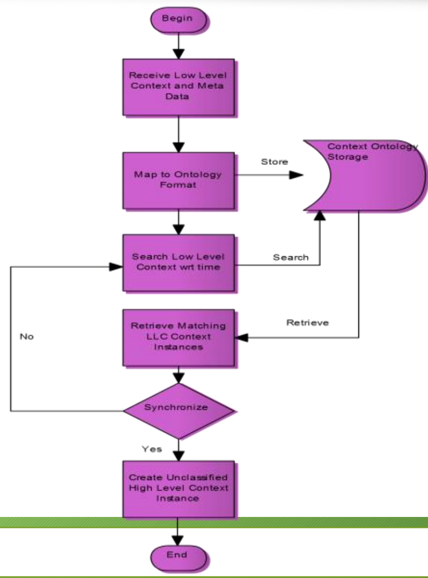
2. Generate an unclassified HLC Instance which contains the LLC instances

Unclassified HLC

```
ctx  rdf:type  Context .
ctx  hasActivity  act_sitting .
ctx  hasLocation  loc_office .
ctx  hasEmotion  emo_boredom .
ctx  isContextOf  user9876 .
ctx  hasStartTime “2015-08-10T11:05:30”^^dateTime .
ctx  rdf:type  hasActivity only ({act_sitting}) .
ctx  rdf:type  hasLocation only ({loc_office }) .
ctx  rdf:type  hasEmotion only ({emo_boredom}) .
```

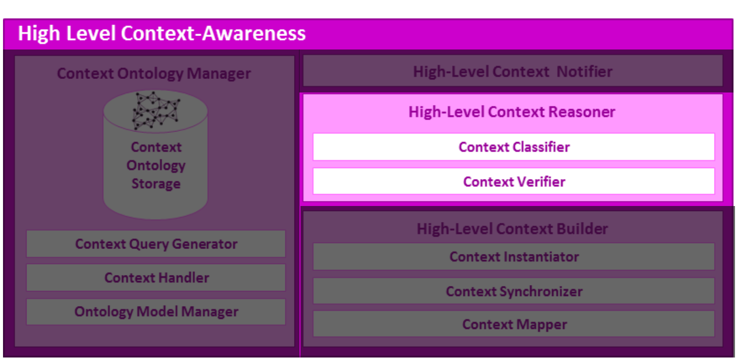

3. Notify the new HLC to the High Level Context Reasoner

High-Level Context Builder Flowchart



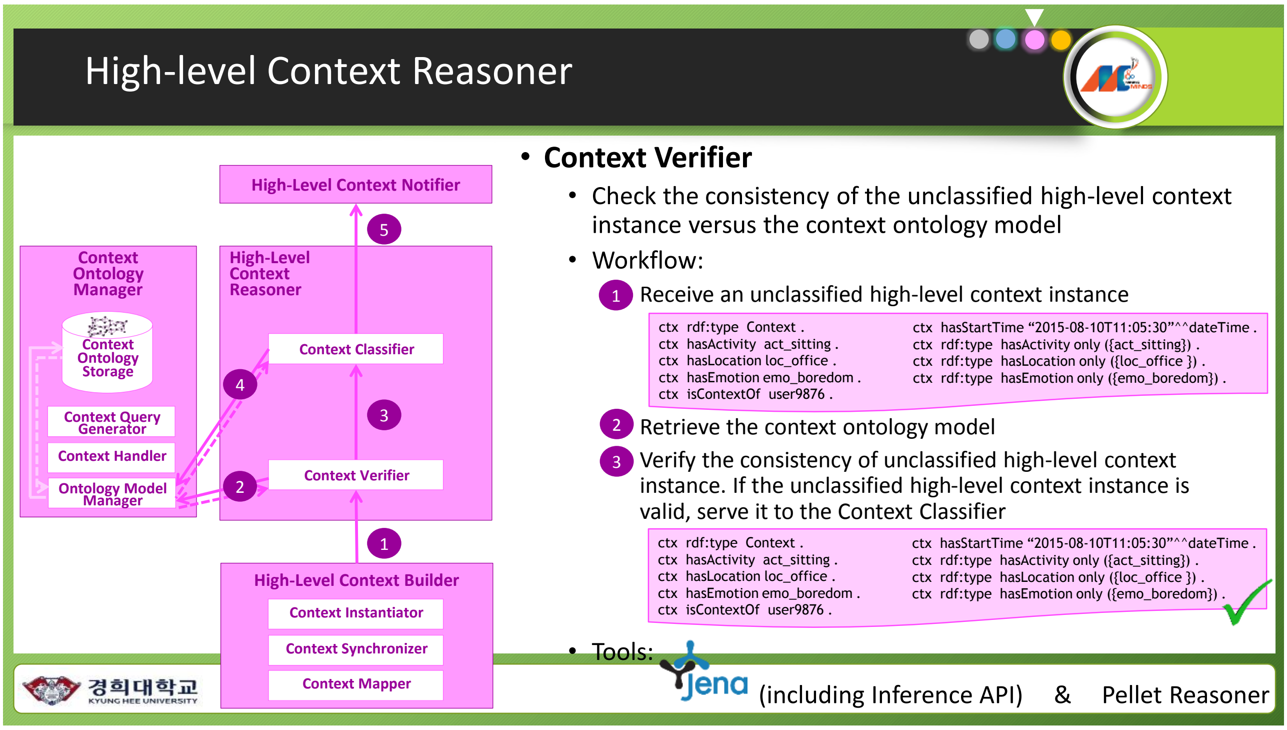
경희대학교 KYUNG HEE UNIVERSITY

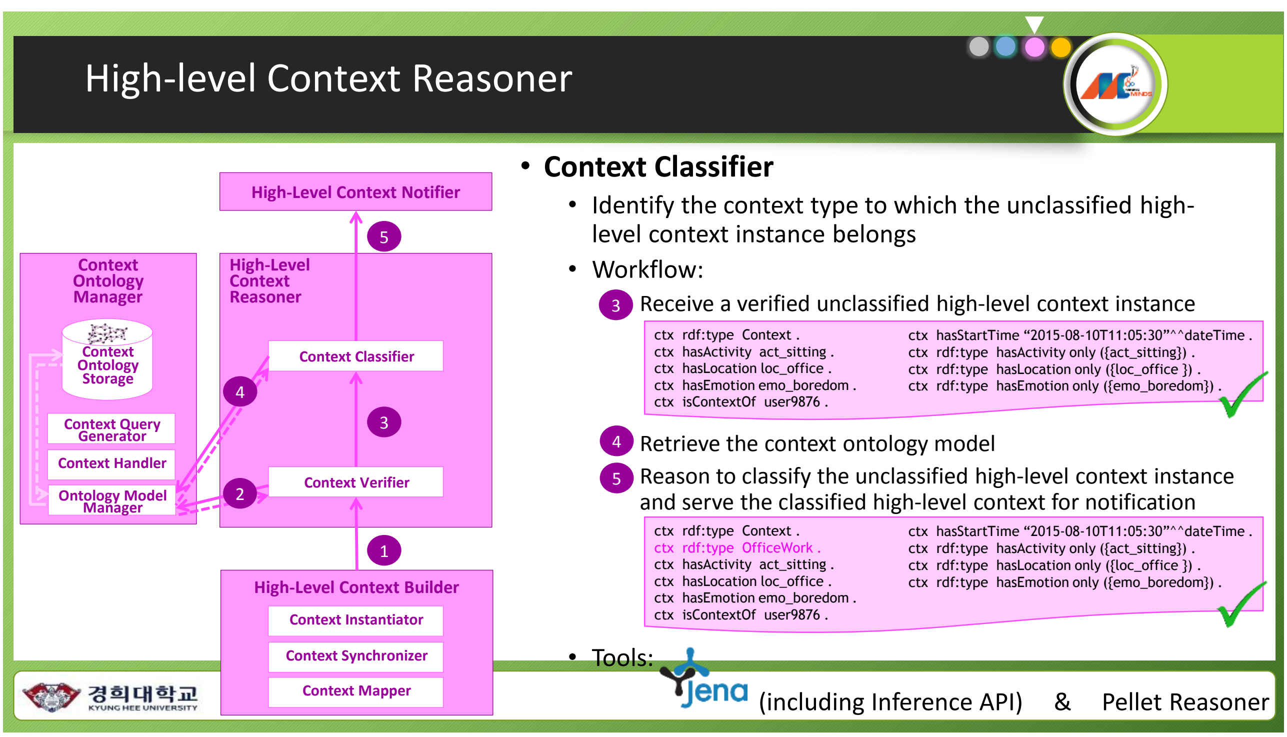
High-level Context Reasoner



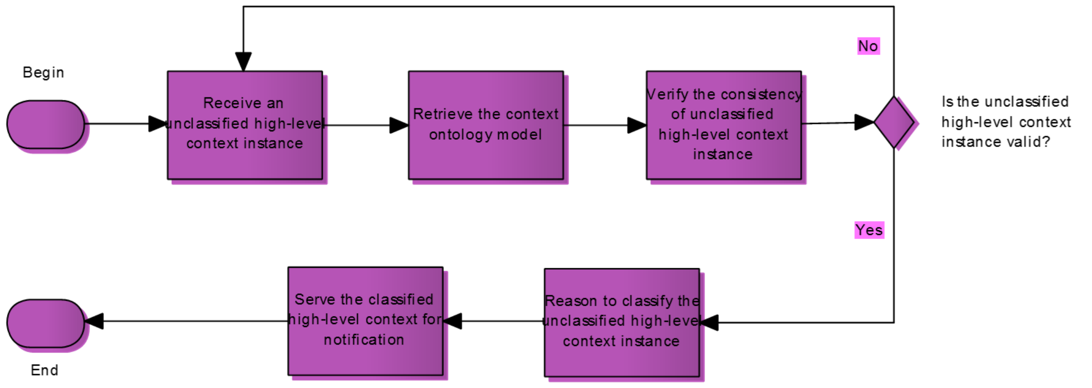
- Objectives:
  - Consistency check (validation and verification) of the unclassified high-level context instance versus the Context Ontology Model
  - Classification or identification of the context type to which the unclassified high-level context instance belongs
- Subcomponents:
  - Context Verifier
  - Context Classifier

경희대학교 KYUNG HEE UNIVERSITY

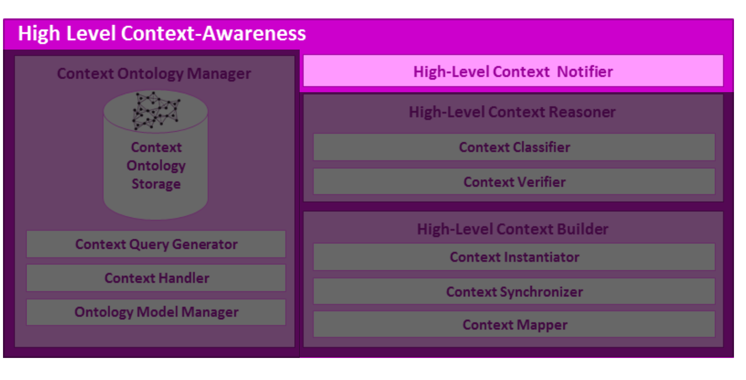

High-level Context Reasoner
Context Verifier
Check the consistency of the unclassified high-level context instance versus the context ontology model
Workflow:
1 Receive an unclassified high-level context instance
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
2 Retrieve the context ontology model
3 Verify the consistency of unclassified high-level context instance. If the unclassified high-level context instance is valid, serve it to the Context Classifier
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
Tools:
Jena (including Inference API) & Pellet Reasoner
High-Level Context Notifier
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Context Instantiator
Context Synchronizer
Context Mapper
경희대학교
KYUNG HEE UNIVERSITY


High-level Context Reasoner
Context Classifier
Identify the context type to which the unclassified high-level context instance belongs
Workflow:
3 Receive a verified unclassified high-level context instance
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
4 Retrieve the context ontology model
5 Reason to classify the unclassified high-level context instance and serve the classified high-level context for notification
ctx rdf:type Context .
ctx rdf:type OfficeWork .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
Tools:
Jena (including Inference API) & Pellet Reasoner
High-Level Context Notifier
Context Ontology Manager
Context Ontology Storage
Context Query Generator
Context Handler
Ontology Model Manager
High-Level Context Reasoner
Context Classifier
Context Verifier
High-Level Context Builder
Context Instantiator
Context Synchronizer
Context Mapper
경희대학교
KYUNG HEE UNIVERSITY

High-level Context Reasoner Flowchart

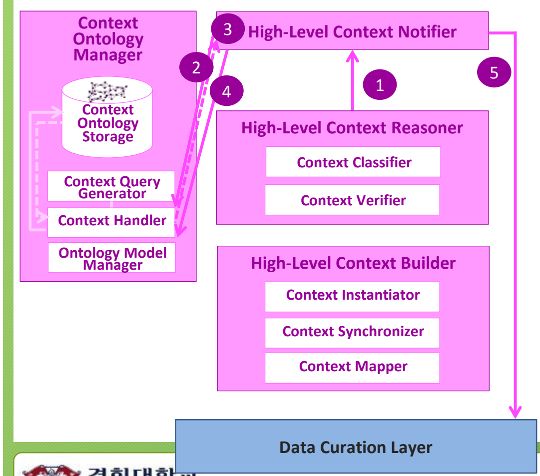


High-Level Context Notifier



- Objectives:
  - Communication of the newly recognized high-level context to Data Curation Layer for storage into the LifeLog
  - Storage of high-level context into the Context Ontology Storage

High-Level Context Notifier



- **Context Notifier**
  - Communicate the newly recognized high-level context to Data Curation Layer for storage into the LifeLog
  - Workflow:
    1. Receive a classified high-level context instance

```
ctx rdf:type Context .
ctx rdf:type OfficeWork .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876.
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
```

    2. Retrieve previous high-level context instance

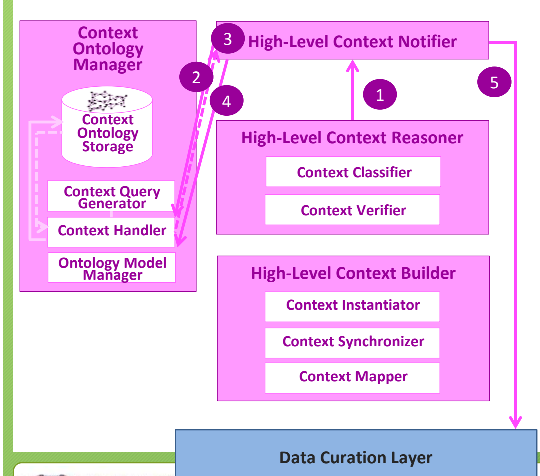
```
ctx0 rdf:type Context .
ctx0 rdf:type Inactivity .
ctx0 hasActivity act_lyingdown .
ctx0 hasLocation loc_office .
ctx0 hasEmotion emo_boredom .
ctx0 isContextOf user9876 .
ctx0 hasStartTime "2015-08-10T11:04:12"^^dateTime .
ctx0 rdf:type hasActivity only ({act_lyingdown}) .
ctx0 rdf:type hasLocation only ({loc_office }) .
ctx0 rdf:type hasEmotion only ({emo_boredom}) .
```

    3. Add end time to previous high-level context instance

```
ctx0 rdf:type Context .
ctx0 rdf:type Inactivity .
ctx0 hasActivity act_lyingdown .
ctx0 hasLocation loc_office .
ctx0 hasEmotion emo_boredom .
ctx0 isContextOf user9876 .
ctx0 hasStartTime "2015-08-10T11:04:12"^^dateTime .
ctx0 hasEndTime "2015-08-10T11:05:30"^^dateTime .
ctx0 rdf:type hasActivity only ({act_lyingdown}) .
ctx0 rdf:type hasLocation only ({loc_office }) .
ctx0 rdf:type hasEmotion only ({emo_boredom}) .
```

경희대학교 KYUNG HEE UNIVERSITY

High-Level Context Notifier



- **Context Notifier**
  - Communicate the newly recognized high-level context to Data Curation Layer for storage into the LifeLog
  - Workflow:
    4. Store the two high-level context instances

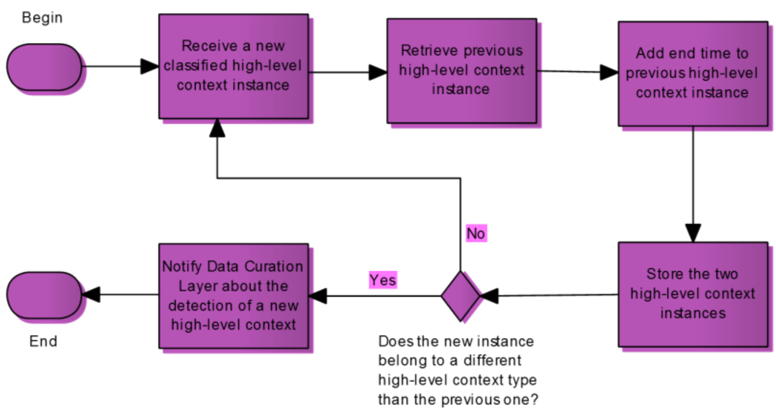
```
ctx rdf:type Context .
ctx rdf:type OfficeWork .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
```

```
ctx0 rdf:type Context .
ctx0 rdf:type Inactivity .
ctx0 hasActivity act_lyingdown .
ctx0 hasLocation loc_office .
ctx0 hasEmotion emo_boredom .
ctx0 isContextOf user9876 .
ctx0 hasStartTime "2015-08-10T11:04:12"^^dateTime .
ctx0 hasEndTime "2015-08-10T11:05:30"^^dateTime .
ctx0 rdf:type hasActivity only ({act_lyingdown}) .
ctx0 rdf:type hasLocation only ({loc_office }) .
ctx0 rdf:type hasEmotion only ({emo_boredom}) .
```

    5. If the new instance belongs to a different high-level context type than the previous one, notify Data Curation Layer about the detection of a new high-level context
- Tool: Jena (including RDF API and Ontology API)

경희대학교 KYUNG HEE UNIVERSITY

High-Level Context Notifier Flowchart



경희대학교 KYUNG HEE UNIVERSITY

Technologies for Data Representation

- Ontologies are **formalized vocabularies** of terms, often covering a specific **domain** and **shared** by a community of users
  - Vocabulary of terms plus explicit characterizations of the assumptions made in interpreting those terms
  - Include some notion of hierarchical classification (is-a)
  - Definition of classes through description of their characteristics
  - Possibility of using inference to help in management and deployment of the knowledge
- **Why ontologies?**
  - The semantics of a language can tell us precisely how to interpret a complex expression
  - Ontologies have formal semantics which provide an **unambiguous** interpretation of the descriptions

경희대학교 KYUNG HEE UNIVERSITY

Technologies for Data Representation

- Standard Ontology Languages
  - **RDF** is a framework for representing information in the Web.
    RDF graphs are sets of subject-predicate-object triples used to express descriptions of resources.
  - **RDF Schema** is a semantic extension of RDF.
    RDFS provides mechanisms for describing groups of related resources and the relationships between these resources.
  - **OWL** is an ontology language for the Semantic Web with formally defined meaning.
    OWL adds an additional layer of semantics on top of RDF

Standard	Features	Limitations or disadvantages
RDF 1.1	Assert statements (rdf:Statement and rdf:subject, rdf:predicate, rdf:object)	Very, very restricted vocabulary No inference
RDF Schema	Define classes (rdfs:Class) and their hierarchy (rdfs:subClassOf) Define properties (rdfs:Property) and their hierarchy (rdfs:subPropertyOf)	Restricted vocabulary No rigid structure, i.e., no constraints
OWL 2	Describe data in terms of set operations (owl:unionOf) Define equivalences (owl:sameAs) Restrict property values (owl:allValuesFrom) Define annotations or meta-meta-data (owl:deprecatedProperty)	More complex ontology

RDF 1.1 (Resource Description Framework) http://www.w3.org/TR/rdf11-concepts/
RDFS (RDF Schema) http://www.w3.org/TR/rdf-schema/
OWL 2 (OWL 2 Web Ontology Language) http://www.w3.org/TR/owl2-overview/

Technologies for Data Representation

- OWL 2 Syntaxes: needed in order to store OWL 2 ontologies and to exchange them among tools and applications

Syntax	Standard Status	Purpose
RDF/XML	Mandatory	Interoperability among all OWL 2 tools
OWL/XML	Optional	Easier to process using XML tools
Functional Syntax	Optional	Easier to see the formal structure of ontologies
Manchester Syntax	Optional	Easier to read/write DL Ontologies
Turtle	Optional	Easier to read/write RDF triples

- OWL 2 Semantics: ways of assigning meaning to OWL 2 ontologies, which are used by reasoners and other tools

Semantics	Name	Advantage	Disadvantage
Direct Semantics	OWL 2 DL	Compatible with the semantics of SROIQ Description Logic (FOL) Decidable	Restrictions on some ontology structures Less expressiveness
RDF-Based Semantics	OWL 2 Full	No restrictions Expressiveness	Undecidable

OWL 2 (OWL 2 Web Ontology Language) http://www.w3.org/TR/owl2-overview/

Technologies for Data Representation

- OWL 2 Profiles: subset of the structural elements in an ontology

Profile	Supported Features	Suitable for …	Benefits
OWL 2 EL	Polynomial time algorithms for standard reasoning	Very large ontologies	Higher performance as a tradeoff for the lower expressive power
OWL 2 QL	Conjunctive queries using standard relational DB technology	Lightweight ontologies with large numbers of individuals	Access the data directly via relational queries (e.g., SQL)
OWL 2 RL	Polynomial time algorithms for reasoning using rule-extended database technologies operating on RDF triples	Lightweight ontologies with large numbers of individuals	Operate directly on data in the form of RDF triples

(!) OWL 2 EL, OWL 2 QL, and OWL 2 RL are even more restrictive than OWL DL

- Standard Ontology Query Language
  - **SPARQL** is a set of specifications that provide languages and protocols to query and manipulate RDF graph content on the Web or in an RDF store

W3C OWL2 W3C SPARQL

OWL 2 (OWL 2 Web Ontology Language) http://www.w3.org/TR/owl2-overview/
SPARQL 1.1 (SPARQL Query Language for RDF) http://www.w3.org/TR/sparql11-overview/

Tools for Ontology Engineering

Tool	Description
protégé	Protégé 4 supports OWL 2 on top of the OWL API. It enables users to load and save OWL ontologies, edit and visualize classes and properties, and check the ontology using an OWL reasoner.
Swoop	SWOOP is a tool for creating, editing, and debugging OWL ontologies. It is an open source project.
NeOn Toolkit	The NeOn Toolkit is an Open Source ontology engineering suite developed in the popular Eclipse environment. Thanks to its modular design and a rich choice of plug-ins, the NeOn Toolkit not only allows editing of ontologies but also provides a variety of leading-edge functionalities, including support for modularization, consistency checking and debugging, alignments and mapping, DB integration, as well as several novel means for visualizing and navigating large ontologies and ontology networks. In addition, it has a unique built-in support for deploying ontology design patterns and for managing ontology development projects, in accordance with the procedures and methods specified in the NeOn Methodology.

protégé

http://www.w3.org/2001/sw/wiki/OWL/Implementations
http://www.w3.org/wiki/Ontology_editors

Ontology Frameworks and APIs

API	Description
OWL API	The OWL API is an API for working with OWL 2 ontologies. It comes with a Java based reference implementation. The OWL API reference implementation includes parsers and renderers for RDF/XML, OWL/XML, Turtle, and Manchester OWL Syntax. It provides a standard interface to various OWL reasoners

Framework	Description
Jena	Jena, a Java RDF API and toolkit (triple store, programming environment, reasoner, rule reasoner, owl reasoner, rdfs reasoner, parser). Directly usable from Java
Sesame	Sesame is a powerful Java framework for processing and handling RDF data. This includes creating, parsing, storing, inferencing and querying over such data. It offers an easy-to-use API that can be connected to all leading RDF storage solutions

경희대학교 KYUNG HEE UNIVERSITY

Tools for Ontological Reasoning

Tool	Native Profiles	Semantics	(Non-) Conformance	Description
CEL	OWL EL	Direct	Lacks support for nominals (ObjectHasValue and ObjectOneOf) and datatypes/values.	CEL is an open-source polynomial-time Classifier for the OWL 2 EL profile. It has demonstrated scalability and proved well suited for several biomedical ontologies.
FaCT++	OWL DL	Direct	Fully conformant except for keys and some datatypes (coming soon).	FaCT++ is an open-source tableaux-based OWL 2 DL reasoner. It is implemented in C++ and shows exceptional performance on expressive ontologies.
HermiT	OWL DL	Direct	Fully conformant	Based on a novel "hypertableau" algorithm, HermiT can determine whether or not the input ontology is consistent, identify subsumption relationships between classes, and much more.
Pellet	OWL DL, EL	Direct	Fully conformant	Pellet is an open source reasoner for OWL 2 DL in Java. It provides standard and cutting-edge reasoning services for OWL ontologies.
RacerPro	OWL DL	Direct		RacerPro is a commercial (but free for research) OWL reasoner and inference server.

Since it supports SWRL Built-in atoms

http://www.w3.org/2001/sw/wiki/OWL/Implementations

Tools and Technologies

- Data Representation
  - Ontological representation (OWL2)
- Ontology Query Language
  - SPARQL
- Ontology Engineering
  - Protégé
- Ontological Reasoner
  - Pellet
- Ontological Framework
  - Jena (Semantic Web Framework)
- Ontology Storage (Triplestore)
  - Jena TDB
- Programming Language
  - Java

경희대학교 KYUNG HEE UNIVERSITY

Uniqueness and Contributions (1/2)

- High-level Context Awareness with nutrition information
  - The High-level context awareness engine recognize more specific context considering 4 kinds information
    - Physical Activities
    - Emotions
    - Nutrition
    - Location

경희대학교 KYUNG HEE UNIVERSITY

Uniqueness and Contributions (2/2)

- Engineering of an *Mining Minds Context Ontology* for context definition
- Design and implementation of methodology for high level context recognition.
- Identification of the issues associated to low-level and high-level context *synchronization*
- Reasoning in order to derive High-level context from Low-level context

경희대학교
KYUNG HEE UNIVERSITY

References

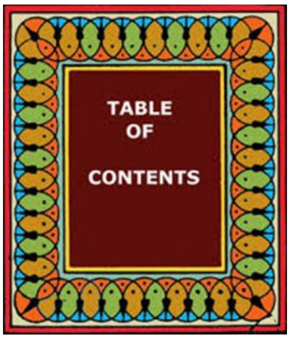
- **[1][G. Chen]** X. Ding, K. Huang, X. Ye, and C. Zhang, "Changing Health Behaviors through Social and Physical Context Awareness," pp. 663–667, 2015.
- **[2][D. Moen]** N. Mckelvey, and K. Curran, "Context Awareness in Mobile Devices," pp. 247–252, 2015.
- **[3][C. Gurrin]** A. F. Smeaton, and A. R. Doherty, "LifeLogging: Personal Big Data," Found. Trends® Inf. Retr., vol. 8, no. 1, pp. 1–125, 2014.
- **[4][Bellavista]** Paolo, et al. "A survey of context data distribution for mobile ubiquitous systems." ACM Computing Surveys (CSUR) 44.4 (2012): 24.
- **[5][H. Lee]** J. Kwon, "Ontology model-based situation and socially-aware health care service in a smart home environment," Int. J. Smart Home, vol. 7, no. 5, pp. 239–250, 2013.
- **[6][C. Perera]** A. Zaslavsky, P. Christen, M. Compton, and D. Georgakopoulos, "Context-aware sensor search, selection and ranking model for internet of things middleware," Proc. - IEEE Int. Conf. Mob. Data Manag., vol. 1, pp. 314–322, 2013.
- **[7][A. M. Khattak]** N. Akbar, M. Aazam, T. Ali, A. M. Khan, S. Jeon, M. Hwang, and S. Lee, "Context Representation and Fusion: Advancements and Opportunities," pp. 9628–9668, 2014.
- **[8][Strang]** Thomas, and Claudia Linnhoff-Popien. "A context modeling survey." Workshop Proceedings. 2004.
- **[9][Dey]** Anind K. "Understanding and using context." Personal and ubiquitous computing 5.1 (2001): 4-7.
- **[10][Abowd]** Gregory D., et al. "Towards a better understanding of context and context-awareness." Handheld and ubiquitous computing. Springer Berlin Heidelberg, 1999.
- **[11][Schilit]** Bill, Norman Adams, and Roy Want. "Context-aware computing applications." Mobile Computing Systems and Applications, 1994. WMCSA 1994. First Workshop on. IEEE, 1994.
- **[12][Wang]** X. H., et al. "Ontology Based Context Modeling and Reasoning using OWL, in 'PERCOMW'04: Proceedings of the Second IEEE Annual Conference on Pervasive Computing and Communications Workshops'." IEEE Computer Society, Washington, DC, USA (2004): 18.

경희대학교
KYUNG HEE UNIVERSITY

Knowledge Curation Layer

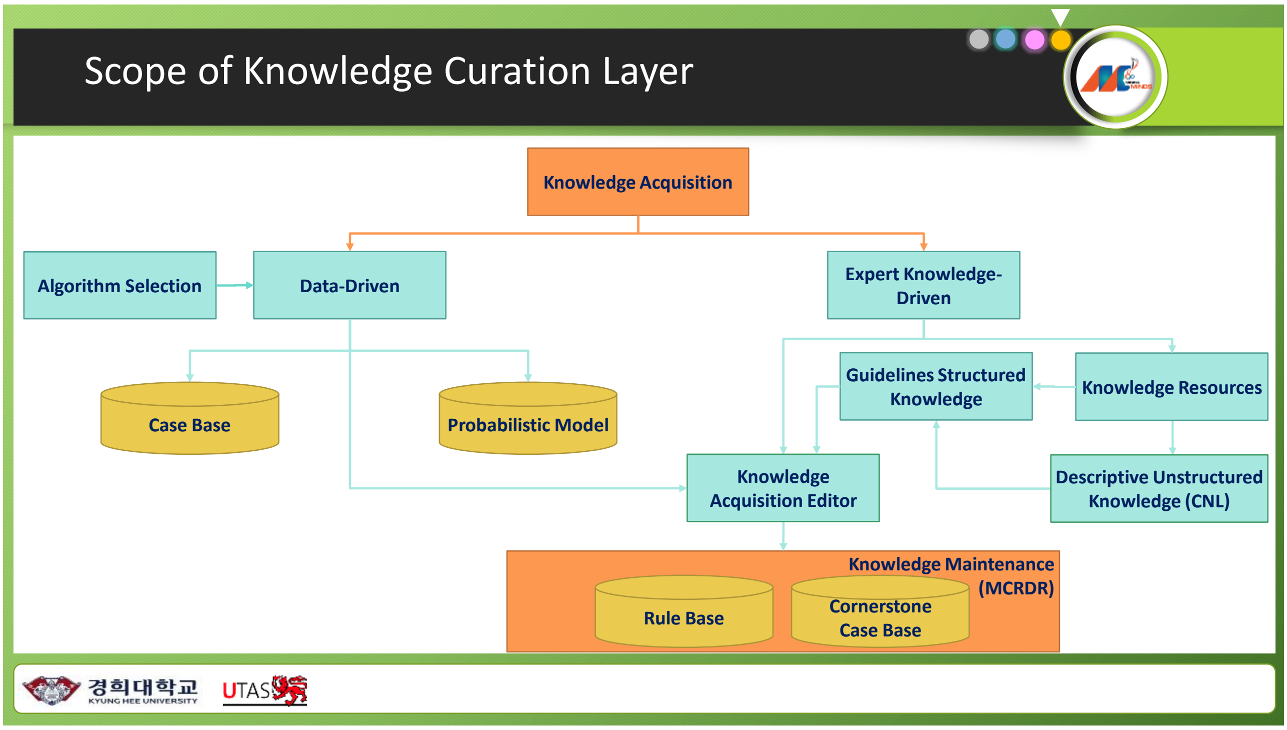
경희대학교 KYUNG HEE UNIVERSITY · KITECH 한국생산기술연구원 · POSTECH 포항공과대학교 · UNIVERSITY of TASMANIA · 녹십자 헬스케어 · TAPACROSS · U2 system User & Usability system · KOCSA (사)한국컨설팅서비스협회

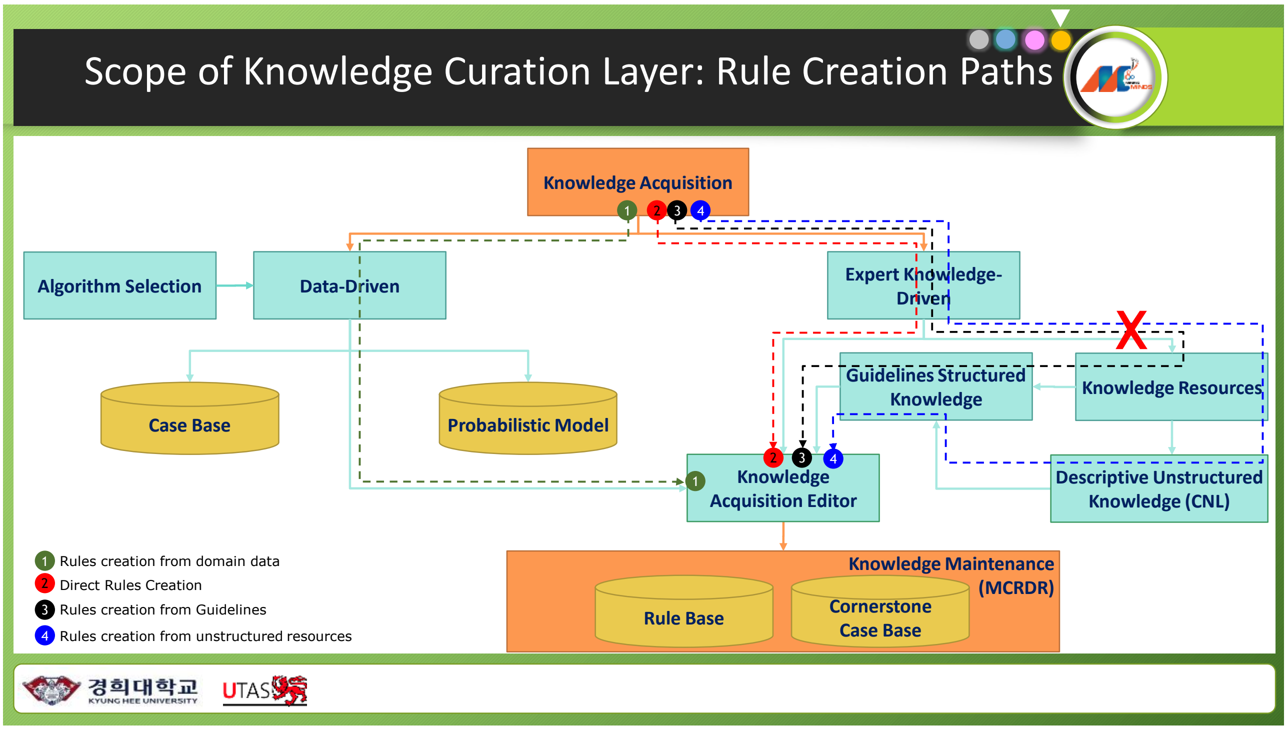
Today Agenda



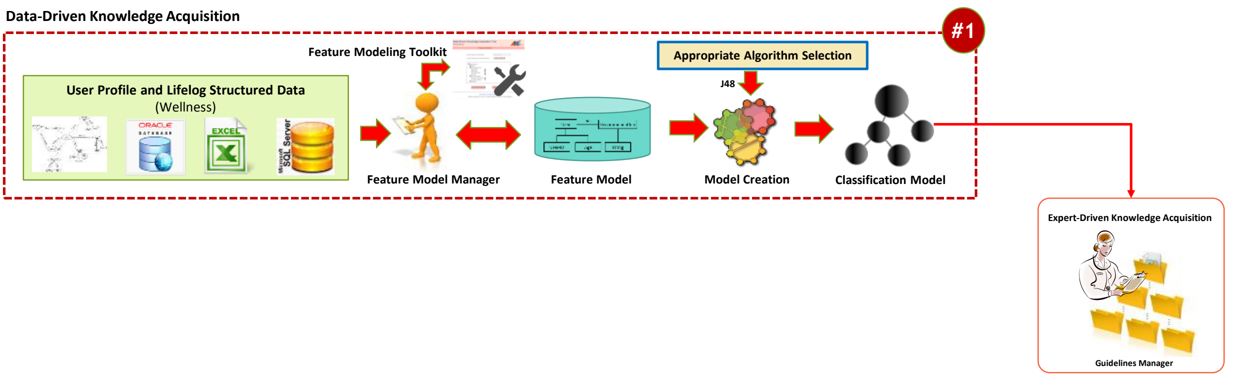
- KCL Overview
  - KCL Taxonomy
  - KCL Intended Acquisition Paths
- KCL Architecture
- Data Driven Knowledge Acquisition
- Expert Driven Knowledge Acquisition

경희대학교 KYUNG HEE UNIVERSITY · UTAS

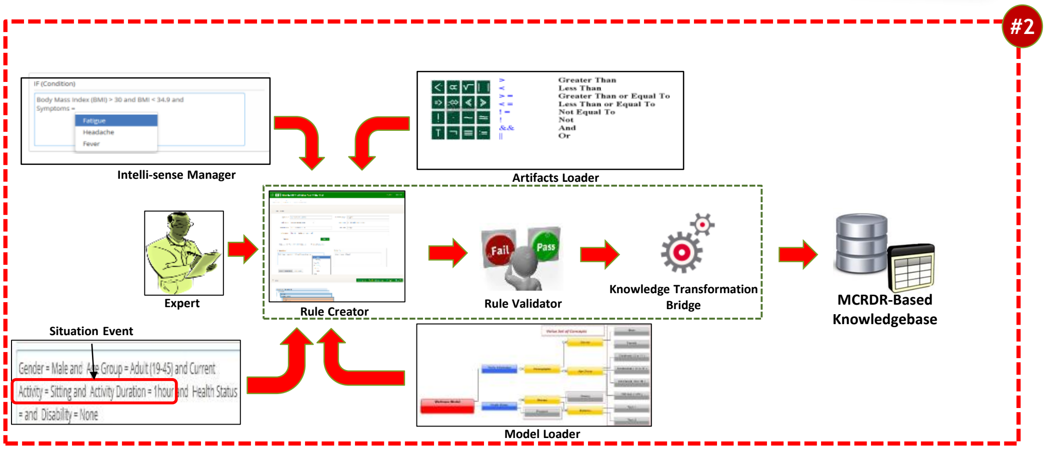

Scope of Knowledge Curation Layer
Knowledge Acquisition
Algorithm Selection
Data-Driven
Expert Knowledge-Driven
Case Base
Probabilistic Model
Guidelines Structured Knowledge
Knowledge Resources
Knowledge Acquisition Editor
Descriptive Unstructured Knowledge (CNL)
Knowledge Maintenance (MCRDR)
Rule Base
Cornerstone Case Base
경희대학교
KYUNG HEE UNIVERSITY
UTAS

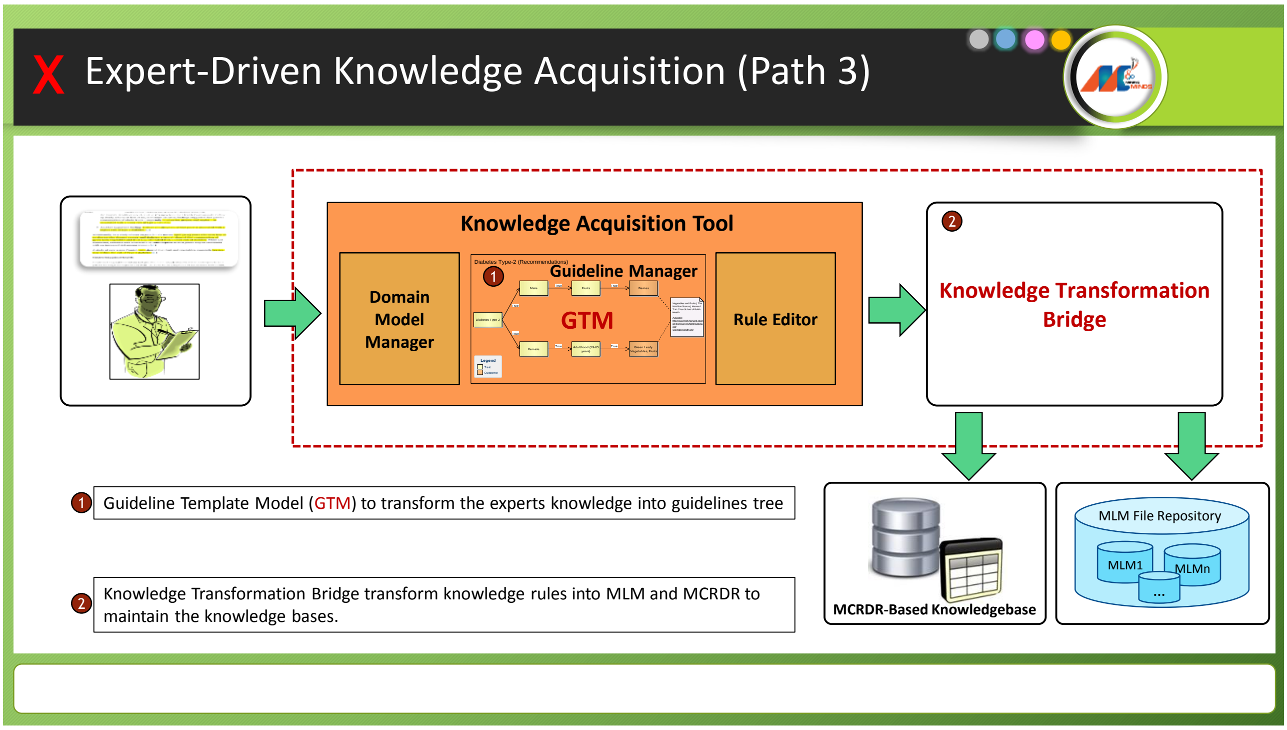

Scope of Knowledge Curation Layer: Rule Creation Paths
Knowledge Acquisition
Algorithm Selection
Data-Driven
Expert Knowledge-Driven
Case Base
Probabilistic Model
Guidelines Structured Knowledge
Knowledge Resources
Knowledge Acquisition Editor
Descriptive Unstructured Knowledge (CNL)
X
1 Rules creation from domain data
2 Direct Rules Creation
3 Rules creation from Guidelines
4 Rules creation from unstructured resources
Knowledge Maintenance (MCRDR)
Rule Base
Cornerstone Case Base
경희대학교
KYUNG HEE UNIVERSITY
UTAS

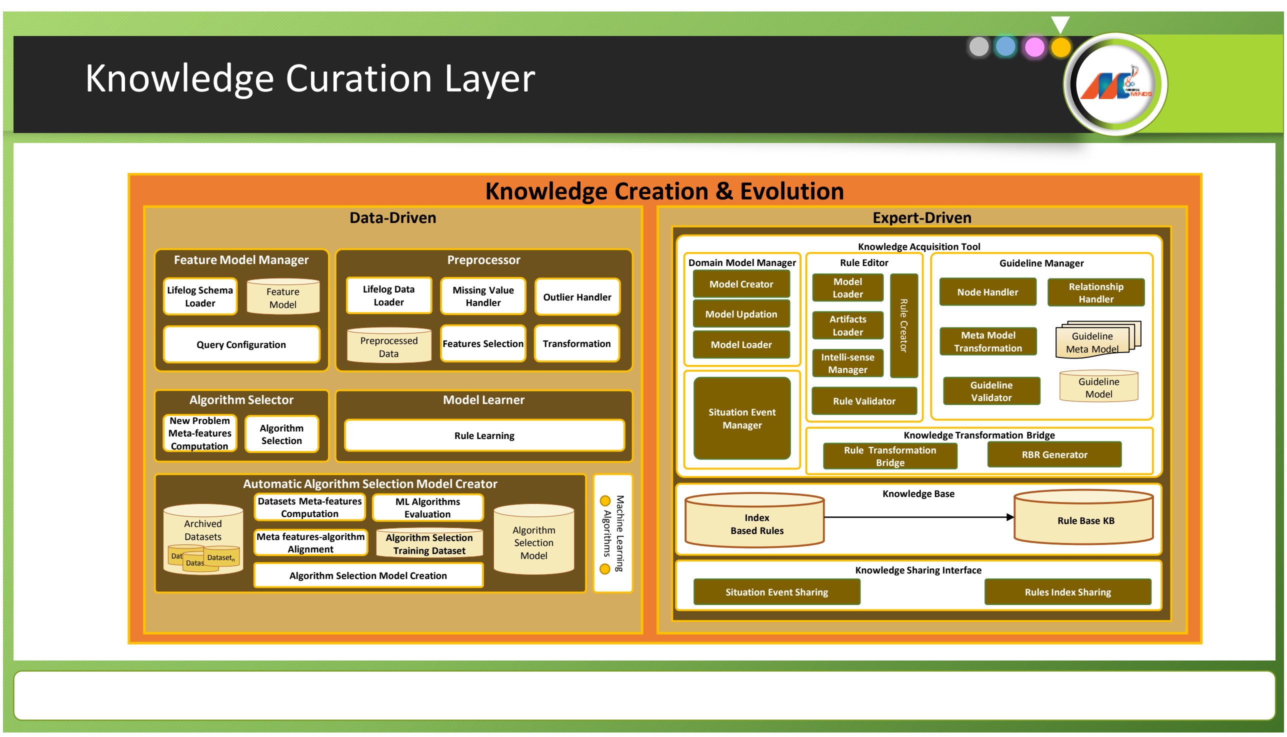
Data-Driven Knowledge Acquisition (Path 1)



Expert-Driven Knowledge Acquisition (Path 2)

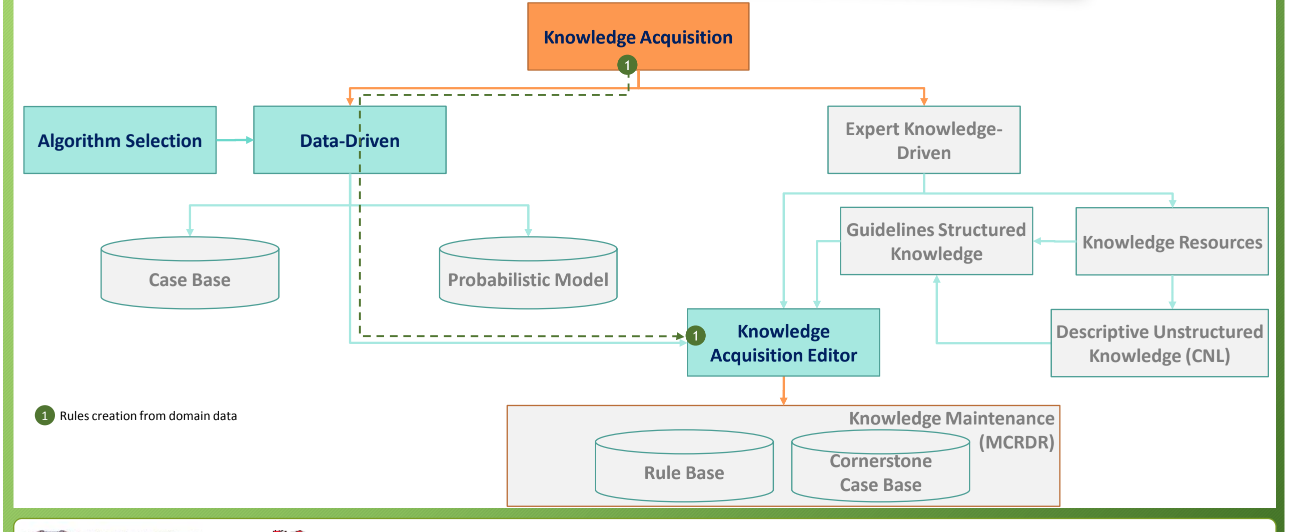



Expert-Driven Knowledge Acquisition (Path 3)
Knowledge Acquisition Tool
Domain Model Manager
Guideline Manager
GTM
Rule Editor
Knowledge Transformation Bridge
Guideline Template Model (GTM) to transform the experts knowledge into guidelines tree
Knowledge Transformation Bridge transform knowledge rules into MLM and MCRDR to maintain the knowledge bases.
MCRDR-Based Knowledgebase
MLM File Repository
MLM1
MLMn
...


Knowledge Curation Layer
Knowledge Creation & Evolution
Data-Driven
Feature Model Manager
Lifelog Schema Loader
Feature Model
Query Configuration
Preprocessor
Lifelog Data Loader
Missing Value Handler
Outlier Handler
Preprocessed Data
Features Selection
Transformation
Algorithm Selector
New Problem Meta-features Computation
Algorithm Selection
Model Learner
Rule Learning
Automatic Algorithm Selection Model Creator
Archived Datasets
Datasets Meta-features Computation
ML Algorithms Evaluation
Meta features-algorithm Alignment
Algorithm Selection Training Dataset
Algorithm Selection Model
Algorithm Selection Model Creation
Machine Learning Algorithms
Expert-Driven
Knowledge Acquisition Tool
Domain Model Manager
Model Creator
Model Updation
Model Loader
Situation Event Manager
Rule Editor
Model Loader
Artifacts Loader
Intelli-sense Manager
Rule Validator
Rule Creator
Guideline Manager
Node Handler
Relationship Handler
Meta Model Transformation
Guideline Meta Model
Guideline Validator
Guideline Model
Knowledge Transformation Bridge
Rule Transformation Bridge
RBR Generator
Knowledge Base
Index Based Rules
Rule Base KB
Knowledge Sharing Interface
Situation Event Sharing
Rules Index Sharing

Data Driven Knowledge Acquisition

Scope of Data Driven Knowledge Acquisition for MM Ver. 2.5



What is Knowledge Acquisition?

- **Knowledge acquisition** is the process of *extracting*, *structuring* and *organizing* knowledge from one **source**
- **Source** can be *Structured* or *Unstructured* Data

Data Driven: Improving Business and Society Through Data

Relatively easy to analyze

Structured Data

Unstructured Data

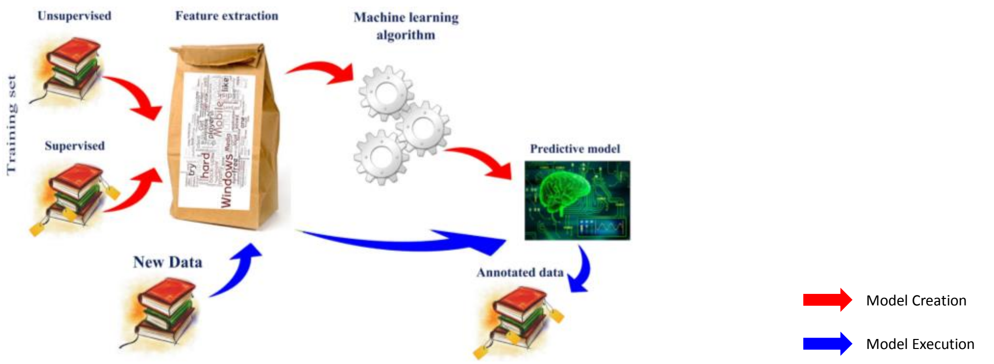
What is Data Driven?

- Data driven is a powerful approach, which
  - process existing data and then create new ones from them.
  - requires less up-front knowledge, but a lot more back-end computation and experimentation

http://www.cad.zju.edu.cn/home/zhx/csmath/lib/exe/fetch.php?media=2013:csmath-01-data-driven.pdf

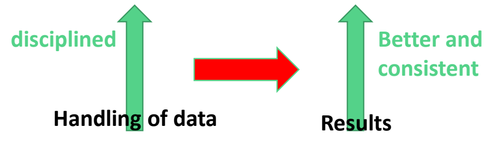
Machine Learning and Data Driven

- Machine learning follows data driven approach
  - Extracts hidden patterns of large data using different computerized algorithms



Machine Learning and Data Preprocessing

- Machine learning algorithms learn from data that should
  - be right
  - be in a useful scale and format
  - includes meaningful features



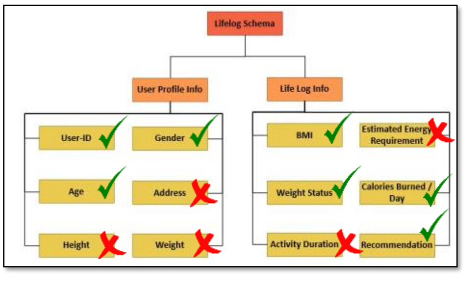
- **Data Preprocessing** describes any type of process performed on raw data to prepare it into an understandable format

Data Preprocessing

- Real world data is usually in raw structured format
  - incomplete: lacking attribute values, containing only aggregate data
    - e.g., occupation=" "
  - noisy: containing errors or outliers
    - e.g., Salary="-10"
  - inconsistent: containing discrepancies in codes or names
    - e.g., Age="42" Birthday="03/07/1997"
    - e.g., Was rating "1,2,3", now rating "A, B, C"
- No quality data, no quality mining results!
  - Quality decisions must be based on quality data
- Data preprocessing lacks the support of Expert-Driven Feature Modeling.

What is Feature Modeling?

- A feature model defines the valid combinations of features in a domain.
- Used widely in domain engineering [1,2]



Related Works

Category	Study	Features	Limitations
Feature Modeling	Sánchez [1], Vranić [2]	• Addresses quality attributes at run time by means of feature models • Context based feature selection • Used in the field of Software Product Lines (SPL)	• Requires extra engineering efforts at development time for designing and mapping the model
Data Preprocessing	Kwiatkowska [3], Dimitriadis [4],	• Prepared consistent and calibrated baseline dataset • Handle noisy, highly variable data	• Knowledge engineer dependency
Algorithm Selection	Smith [5], Ali and Smith [6], Song [7], Wang [8]	• Finds best classification algorithm • Considered multiple datasets • Considered multiple algorithms	• Use a sub-set of meta-features • Use single-metric evaluation criteria • Use black-box learning techniques for model creation
Model Learning	Dimitriadis [4], Bachman [9]	• Extract new knowledge • Create effective set of decision rules • Worked on Time, space, and medical domain	• Used fixed machine learning methods (ZeroR, NaiveBayes, J48, SVM) • Knowledge engineer dependency

Limitations

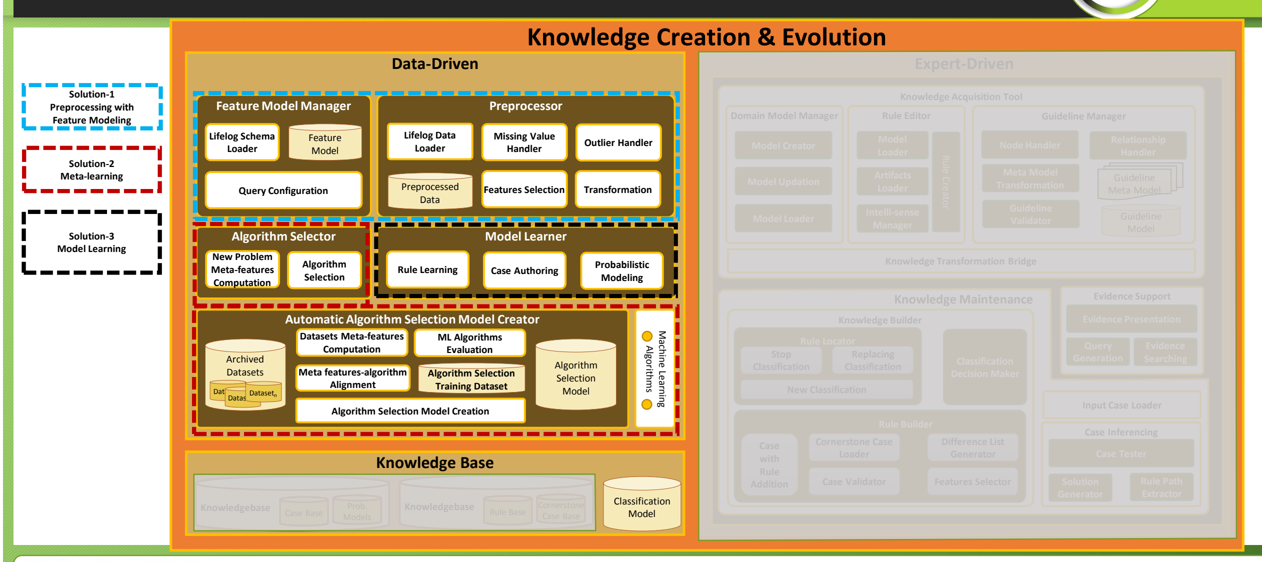
- Expert manually select machine learning algorithm for building classification model
- Use a sub-set of meta-features
- Lack of feature modeling for preprocessing of data

Motivations – Challenges - Solutions

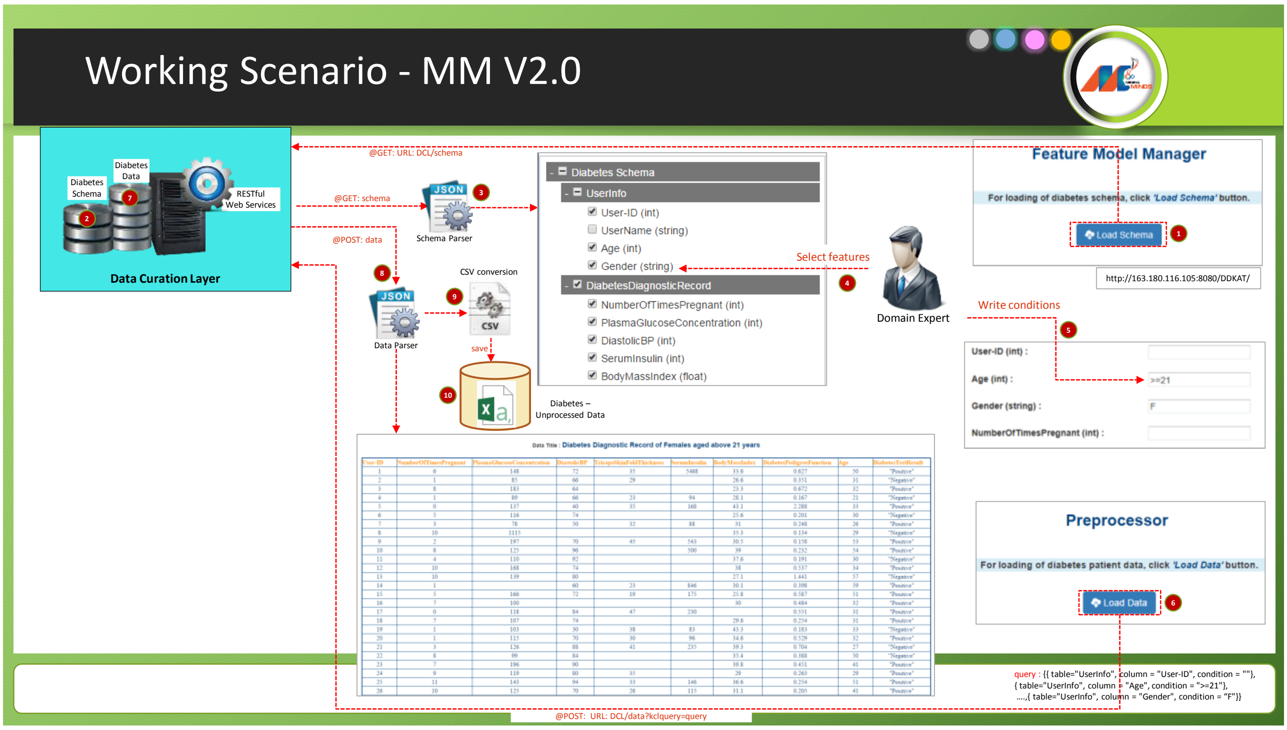
Motivations		Challenges		Solutions
For yielding correct classification model in any domain, selection of quality data requires domain configuration/engineering as well as data preprocessing		Ch-1 How to select, manage, and process quality attributes?		Sol-1 Integration of Data Pre-Processing with Feature Modeling
There exist lot of ML methods that can be utilized to build classification model for large amount of data		Ch-2 How to automatically select an appropriate ML method for learning classification model appropriately?		Sol-2 Meta-learning
A classification model can auto generate recommendation for different services		Ch-3 How to build a classification model?		Sol-3 Model Learning

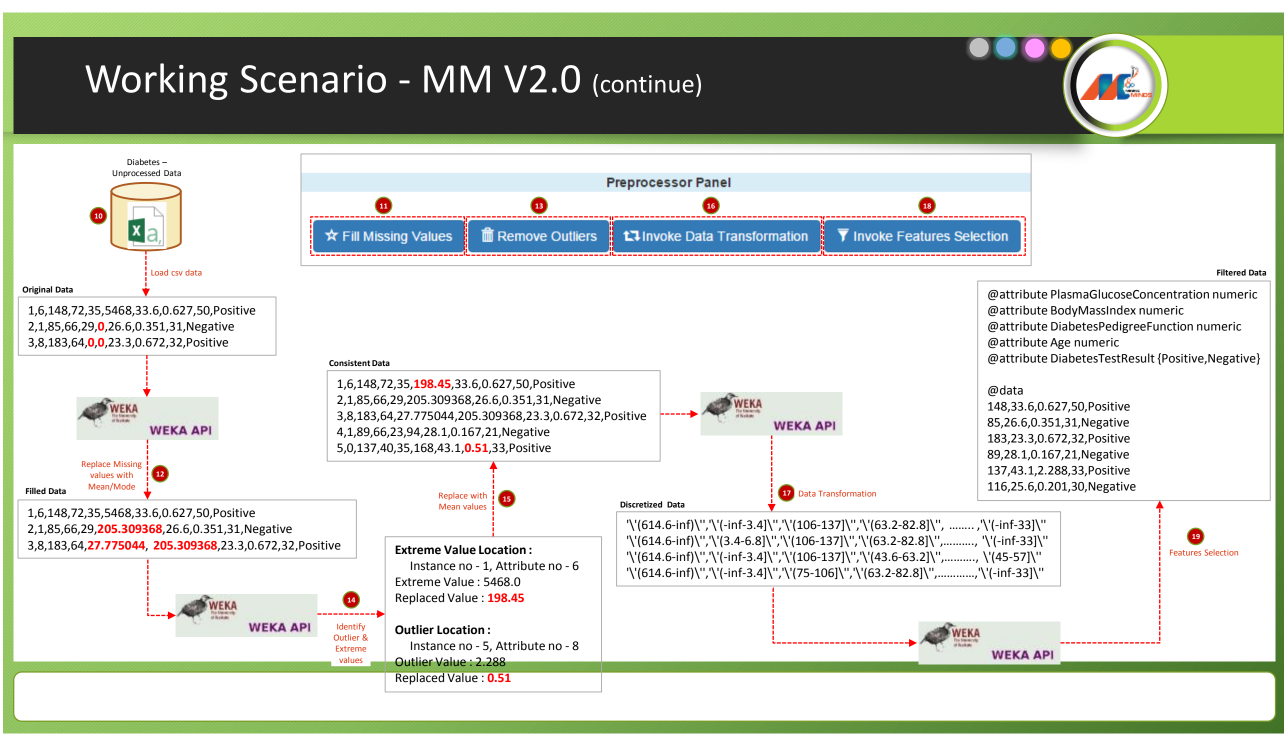
경희대학교 KYUNG HEE UNIVERSITY UTAS

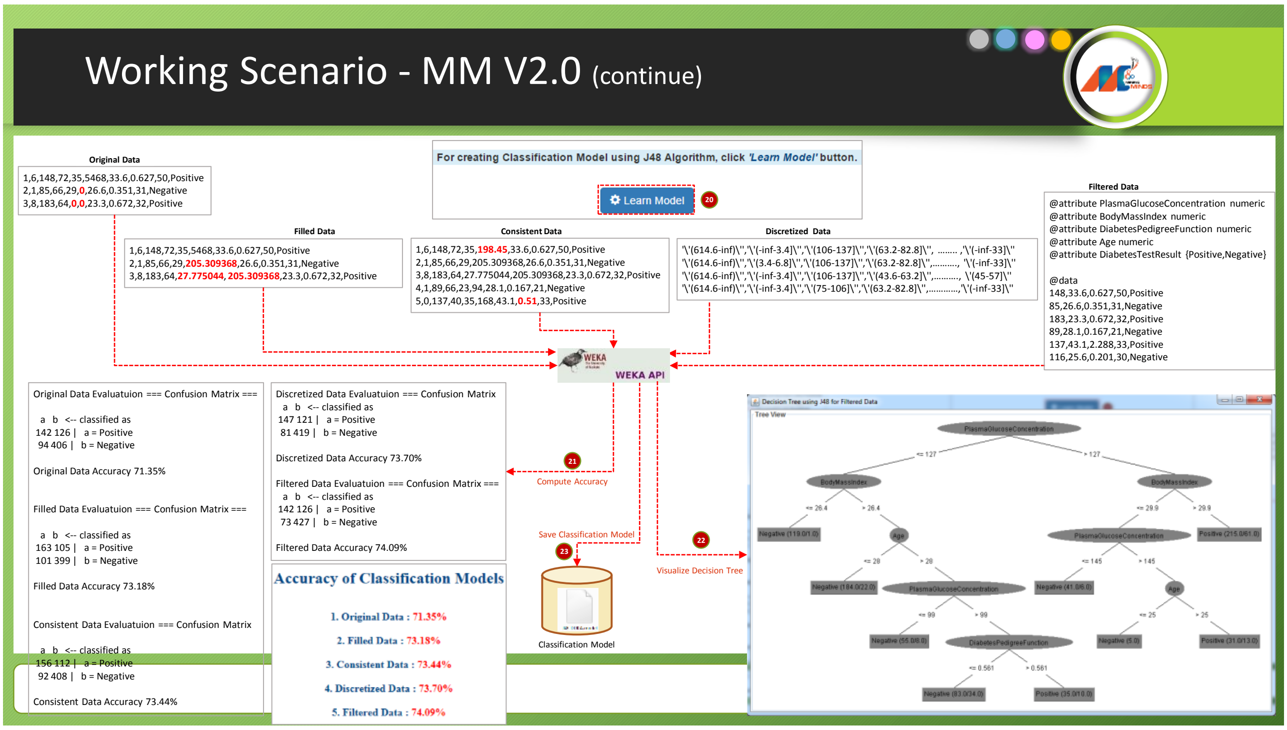
Data Driven Knowledge Acquisition

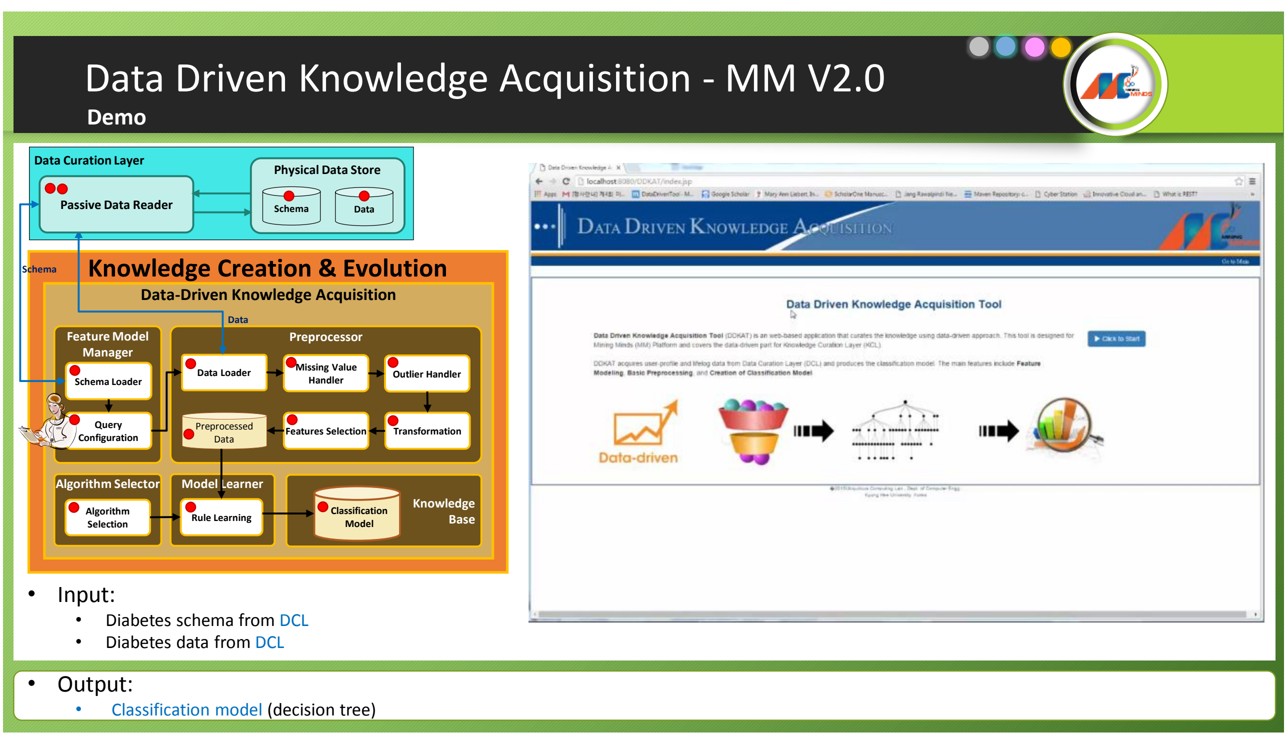


经희대학교 KYUNG HEE UNIVERSITY UTAS


Working Scenario - MM V2.0
Diabetes Data
Diabetes Schema
RESTful Web Services
Data Curation Layer
@GET: URL: DCL/schema
@GET: schema
@POST: data
JSON
Schema Parser
CSV conversion
Data Parser
CSV
save
Diabetes – Unprocessed Data
Diabetes Schema
UserInfo
User-ID (int)
UserName (string)
Age (int)
Gender (string)
DiabetesDiagnosticRecord
NumberOfTimesPregnant (int)
PlasmaGlucoseConcentration (int)
DiastolicBP (int)
SerumInsulin (int)
BodyMassIndex (float)
Select features
Domain Expert
Feature Model Manager
For loading of diabetes schema, click 'Load Schema' button.
Load Schema
http://163.180.116.105:8080/DDKAT/
Write conditions
User-ID (int) :
Age (int) :
>=21
Gender (string) :
F
NumberOfTimesPregnant (int) :
Preprocessor
For loading of diabetes patient data, click 'Load Data' button.
Load Data
Data Title : Diabetes Diagnostic Record of Females aged above 21 years
query : {{ table="UserInfo", column = "User-ID", condition = ""},
{ table="UserInfo", column = "Age", condition = ">=21"},
....{ table="UserInfo", column = "Gender", condition = "F"}}
@POST: URL: DCL/data?kclquery=query


Working Scenario - MM V2.0 (continue)
Diabetes – Unprocessed Data
Load csv data
Original Data
1,6,148,72,35,5468,33.6,0.627,50,Positive
2,1,85,66,29,0,26.6,0.351,31,Negative
3,8,183,64,0,0,23.3,0.672,32,Positive
WEKA API
Replace Missing values with Mean/Mode
Filled Data
1,6,148,72,35,5468,33.6,0.627,50,Positive
2,1,85,66,29,205.309368,26.6,0.351,31,Negative
3,8,183,64,27.775044, 205.309368,23.3,0.672,32,Positive
Identify Outlier & Extreme values
Preprocessor Panel
Fill Missing Values
Remove Outliers
Invoke Data Transformation
Invoke Features Selection
Consistent Data
1,6,148,72,35,198.45,33.6,0.627,50,Positive
2,1,85,66,29,205.309368,26.6,0.351,31,Negative
3,8,183,64,27.775044,205.309368,23.3,0.672,32,Positive
4,1,89,66,23,94,28.1,0.167,21,Negative
5,0,137,40,35,168,43.1,0.51,33,Positive
Replace with Mean values
Extreme Value Location :
Instance no - 1, Attribute no - 6
Extreme Value : 5468.0
Replaced Value : 198.45
Outlier Location :
Instance no - 5, Attribute no - 8
Outlier Value : 2.288
Replaced Value : 0.51
Data Transformation
Discretized Data
'\'(614.6-inf)\'','\'(-inf-3.4]\'','\'(106-137]\'','\'(63.2-82.8]\'', ... ,'\'(-inf-33]\''
'\'(614.6-inf)\'','\'(3.4-6.8]\'','\'(106-137]\'','\'(63.2-82.8]\'',.........., '\'(-inf-33]\''
'\'(614.6-inf)\'','\'(-inf-3.4]\'','\'(106-137]\'','\'(43.6-63.2]\'',.........., '\'(45-57]\''
'\'(614.6-inf)\'','\'(-inf-3.4]\'','\'(75-106]\'','\'(63.2-82.8]\'',..........,'\'(-inf-33]\''
Features Selection
Filtered Data
@attribute PlasmaGlucoseConcentration numeric
@attribute BodyMassIndex numeric
@attribute DiabetesPedigreeFunction numeric
@attribute Age numeric
@attribute DiabetesTestResult {Positive,Negative}
@data
148,33.6,0.627,50,Positive
85,26.6,0.351,31,Negative
183,23.3,0.672,32,Positive
89,28.1,0.167,21,Negative
137,43.1,2.288,33,Positive
116,25.6,0.201,30,Negative


Working Scenario - MM V2.0 (continue)
Original Data
1,6,148,72,35,5468,33.6,0.627,50,Positive
2,1,85,66,29,0,26.6,0.351,31,Negative
3,8,183,64,0,0,23.3,0.672,32,Positive
For creating Classification Model using J48 Algorithm, click 'Learn Model' button.
Learn Model
20
Filtered Data
@attribute PlasmaGlucoseConcentration numeric
@attribute BodyMassIndex numeric
@attribute DiabetesPedigreeFunction numeric
@attribute Age numeric
@attribute DiabetesTestResult {Positive,Negative}
@data
148,33.6,0.627,50,Positive
85,26.6,0.351,31,Negative
183,23.3,0.672,32,Positive
89,28.1,0.167,21,Negative
137,43.1,2.288,33,Positive
116,25.6,0.201,30,Negative
Filled Data
1,6,148,72,35,5468,33.6,0.627,50,Positive
2,1,85,66,29,205.309368,26.6,0.351,31,Negative
3,8,183,64,27.775044,205.309368,23.3,0.672,32,Positive
Consistent Data
1,6,148,72,35,198.45,33.6,0.627,50,Positive
2,1,85,66,29,205.309368,26.6,0.351,31,Negative
3,8,183,64,27.775044,205.309368,23.3,0.672,32,Positive
4,1,89,66,23,94,28.1,0.167,21,Negative
5,0,137,40,35,168,43.1,0.51,33,Positive
Discretized Data
WEKA API
Original Data Evaluatuion === Confusion Matrix ===
a b <-- classified as
142 126 | a = Positive
94 406 | b = Negative
Original Data Accuracy 71.35%
Filled Data Evaluatuion === Confusion Matrix ===
a b <-- classified as
163 105 | a = Positive
101 399 | b = Negative
Filled Data Accuracy 73.18%
Consistent Data Evaluatuion === Confusion Matrix
a b <-- classified as
156 112 | a = Positive
92 408 | b = Negative
Consistent Data Accuracy 73.44%
Discretized Data Evaluatuion === Confusion Matrix
a b <-- classified as
147 121 | a = Positive
81 419 | b = Negative
Discretized Data Accuracy 73.70%
Filtered Data Evaluatuion === Confusion Matrix ===
a b <-- classified as
142 126 | a = Positive
73 427 | b = Negative
Filtered Data Accuracy 74.09%
Accuracy of Classification Models
1. Original Data : 71.35%
2. Filled Data : 73.18%
3. Consistent Data : 73.44%
4. Discretized Data : 73.70%
5. Filtered Data : 74.09%
21
Compute Accuracy
Save Classification Model
23
22
Visualize Decision Tree
Classification Model
Decision Tree using J48 for Filtered Data
Tree View


Data Driven Knowledge Acquisition - MM V2.0
Demo
Data Curation Layer
Passive Data Reader
Physical Data Store
Schema
Data
Knowledge Creation & Evolution
Data-Driven Knowledge Acquisition
Feature Model Manager
Schema Loader
Query Configuration
Preprocessor
Data Loader
Missing Value Handler
Outlier Handler
Preprocessed Data
Features Selection
Transformation
Algorithm Selector
Algorithm Selection
Model Learner
Rule Learning
Classification Model
Knowledge Base
Data Driven Knowledge Acquisition
Data Driven Knowledge Acquisition Tool
Data-driven
Input:
Diabetes schema from DCL
Diabetes data from DCL
Output:
Classification model (decision tree)

Tools and Technologies - MM V2.0

- Language
  - Java
- Frameworks:
  - Spring MVC
  - Bootstrap
- APIs:
  - Weka
- IDEs:
  - Eclipse
- Database
  - UCI - Diabetes Dataset

Technical Contributions - MM V2.0

- Classification model creation
- Feature model creation
- Data preprocessing
- Integration with big data repository

Project Plan: Data-Driven Knowledge Acquisition

Tasks for MM2.5

Feature Model Manager

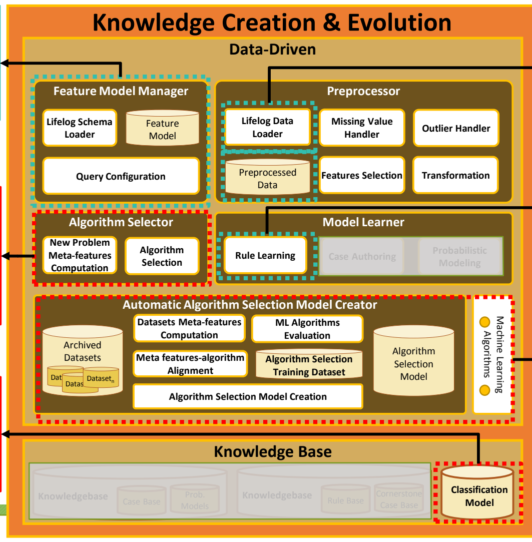
- Persistence of Feature Model
- Features Grouping

Algorithm Selector

- Integrating algorithm selection model into data driven knowledge acquisition

Classification Model

- Rules conformance with Expert-Driven



Preprocessor

- GUI improvement for huge lifelog data display
- Persistence of Processed Data

Model Learner

- Rule learning based on newly selected algorithm

Automatic Algorithm Selection Model Creator

- Developing meta-features extractor for new dataset
- Creating algorithms section model

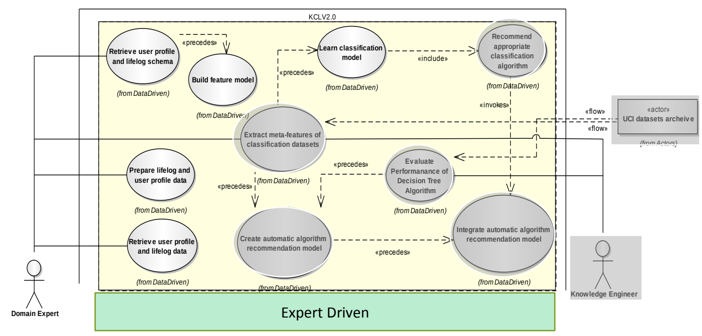
Development Plan

Knowledge Curation Layer Task Schedule for MMV2.5

Modules	Responsible	TASKS	Jan-16 11-16	Jan-16 18-23	Jan-16 25-30	Feb-16 1-6	Feb-16 8-13	Feb-16 15-20	Feb-16 22-27	Feb-16 29-29	Mar-16 1-5	Mar-16 7-12	Mar-16 14-19	Mar-16 21-26	Mar-16 28-31	Apr-16 1-2	Apr-16 4-9	Apr-16 11-16	Apr-16 18-23	Apr-16 25-30	May-16 2-7	May-16 9-14	May-16 16-20
Data Driven: Automatic Algorithm Selection Model Creator	Rahman Ali	Selection of diverse datasets for training																					
		Evaluating different machine learning algorithms on diverse datasets																					
		Implementation of meta-features extrator for computing meta-features of the dataset																					
		Creation of training dataset for model creation																					
		Crteation of algorithm selection model																					
Data Driven: Feature Model Manager	Maqbool Ali	Features grouping research and analysis																					
		Design of Features grouping																					
		Implementation of Features grouping																					
		Implementation of feature model persistence																					
		Integration and Testing																					
Data Driven: Preprocessor		GUI improvement for huge lifelog data display																					
		Implementation of Processed data persistence																					
		Integration and Testing																					
Data Driven: Algorithm Selector		Design of Algorithm Selector																					
		Integration of algorithm selection model with Knowledge Acquisition tool																					
		Integration Testing																					
Data Driven: Model Learner		Design of rule learning																					
		Implementation of Rule learning based on newly selected algorithm																					
		Integration and Testing																					
Data Driven: Classification Model		Design of rules conformance																					
		Implementation of Rules conformance with Expert-Driven																					
		Integration and Testing																					

Data Driven Knowledge Acquisition Ver. 2.0 – Use cases

Use Case #ID	Use Case Name
KCL2-SUC-05	Retrieve user profile and lifelog schema
KCL2-SUC-01	Build feature model
KCL2-SUC-06	Retrieve user profile and lifelog data
KCL2-SUC-02	Prepare lifelog and user profile data
KCL2-SUC-13	Learn classification model



Data Driven Knowledge Acquisition Ver. 2.0 – Use cases

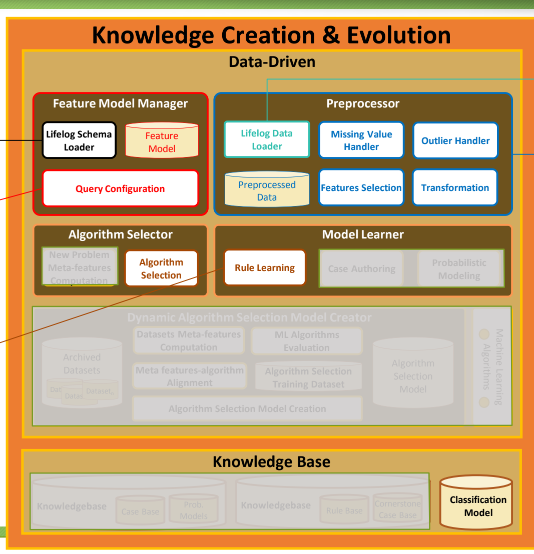
Retrieve user profile and lifelog schema

Build feature model

- Feature Modeling
  - Select the attributes

Learn classification model

- Model Learning
  - Generate the decision tree



Retrieve user profile and lifelog data

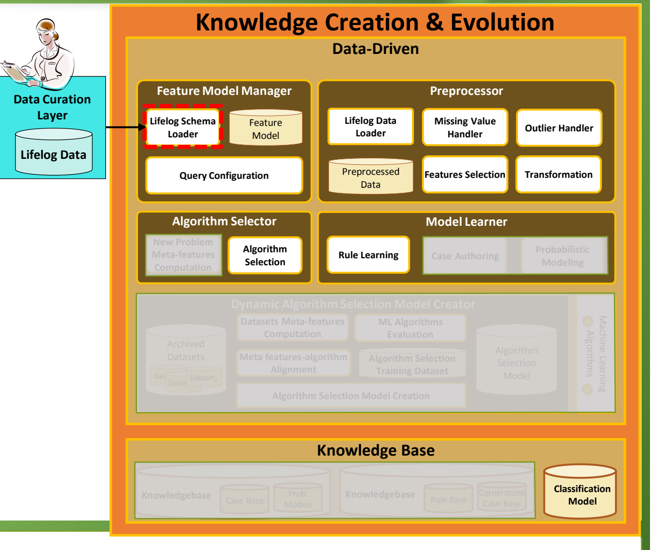
Prepare lifelog and user profile data

- Data Cleaning
  - Fill in missing values, identify or remove outliers
- Data Transformation
  - Generalization: concept hierarchy climbing
- Data Reduction
  - Obtains reduced representation in volume

KCL2-SUC-05: Retrieve user profile and lifelog schema

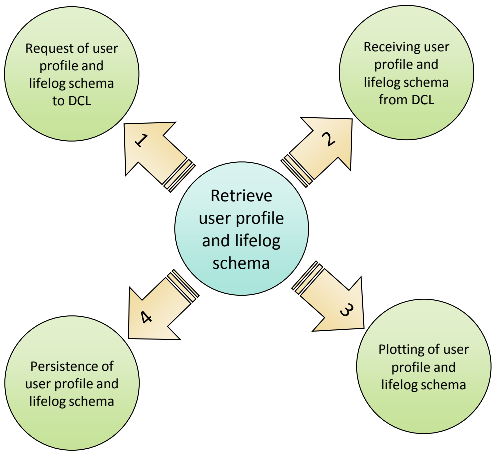
Use Case

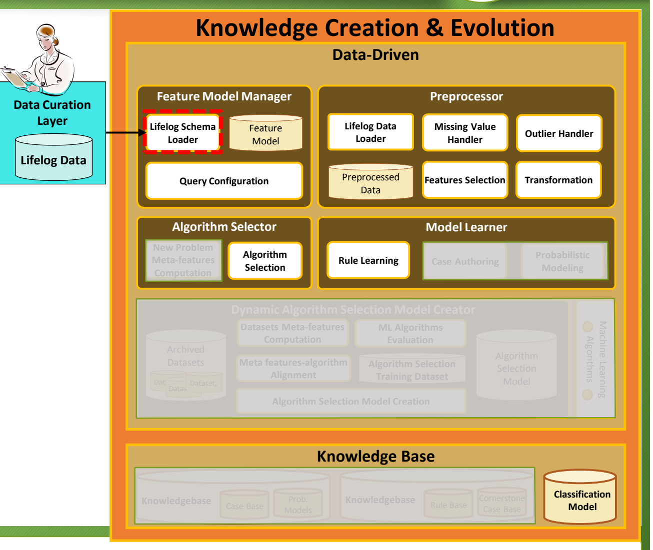
- **Objective**
  - ➢ To help domain expert to view all available features for building feature model
- **Methodology**
  - ➢ ***Communication***: Restful Service between *Feature Model Manager* and *DCL*
  - ➢ ***Lifelog Schema Loader*** *(operation)*
    1. **Input**: Required configuration of domain
    2. **Processing**:
       a. Domain expert selects the domain and sends requests to DCL for user profile and lifelog schema.
       b. DCL shares the user profile and lifelog schema
       c. System receives the user profile and lifelog schema
       d. System saves the received schema into schema storage
    3. **Output**: forwards the received schema to *Query Configuration*

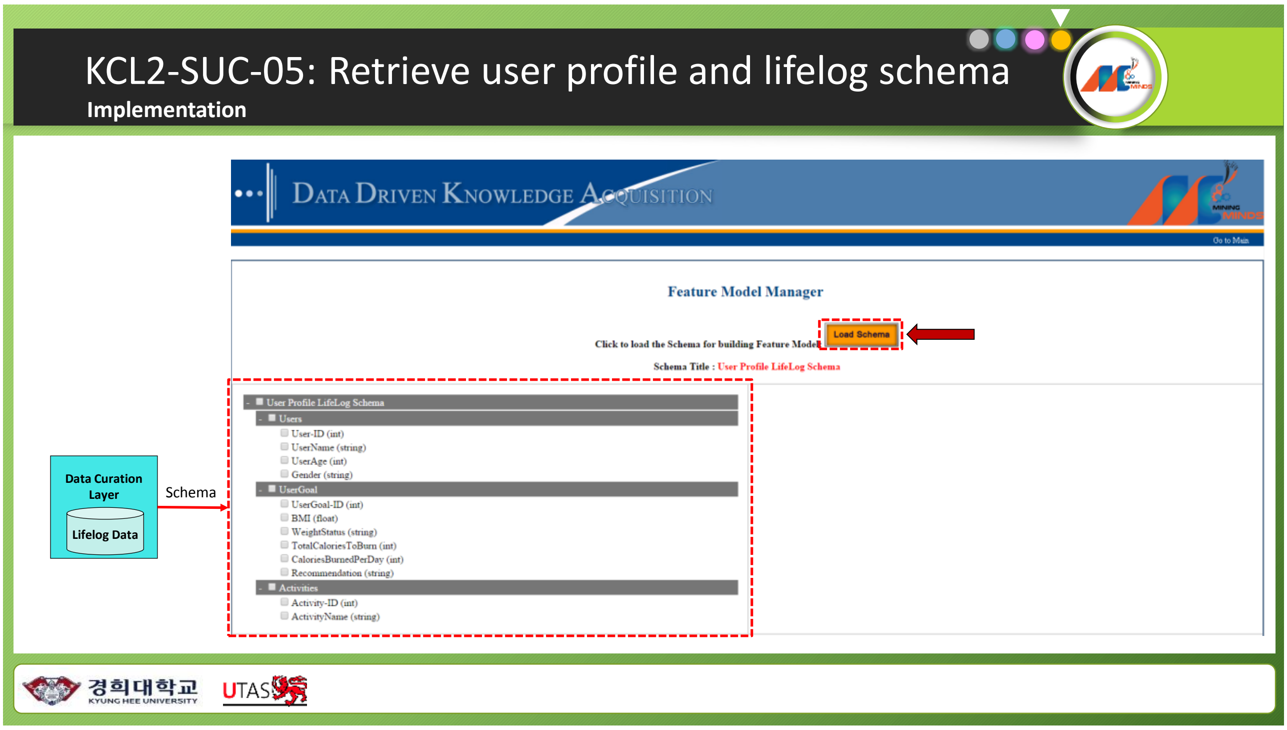


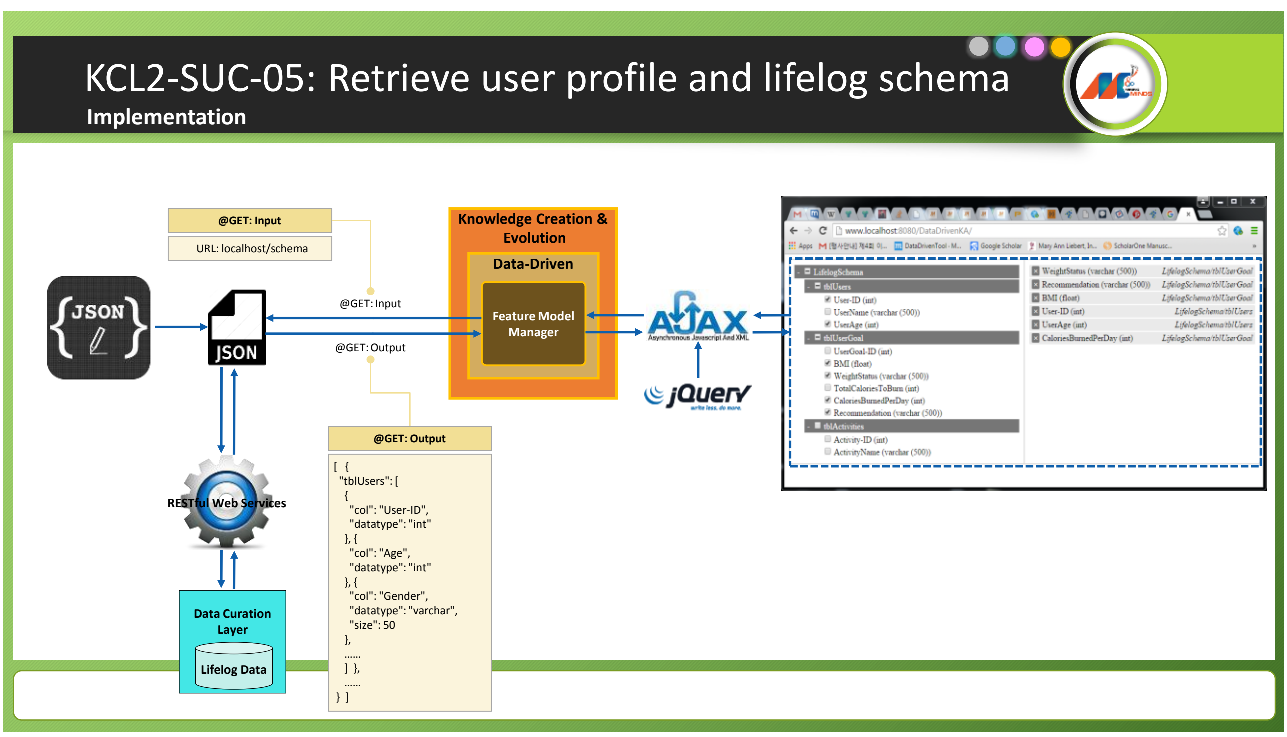
KCL2-SUC-05: Retrieve user profile and lifelog schema

Use Case Functions





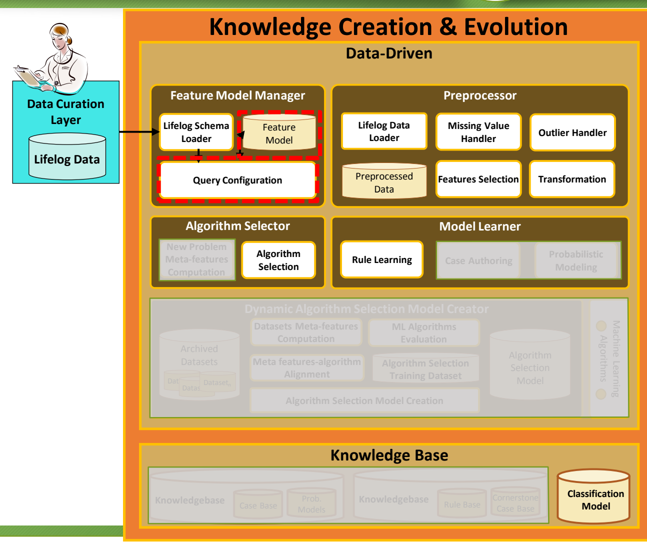

KCL2-SUC-05: Retrieve user profile and lifelog schema
Implementation
DATA DRIVEN KNOWLEDGE ACQUISITION
Go to Main
Feature Model Manager
Click to load the Schema for building Feature Model
Load Schema
Schema Title : User Profile LifeLog Schema
User Profile LifeLog Schema
Users
User-ID (int)
UserName (string)
UserAge (int)
Gender (string)
UserGoal
UserGoal-ID (int)
BMI (float)
WeightStatus (string)
TotalCaloriesToBurn (int)
CaloriesBurnedPerDay (int)
Recommendation (string)
Activities
Activity-ID (int)
ActivityName (string)
Data Curation Layer
Lifelog Data
Schema
경희대학교
KYUNG HEE UNIVERSITY
UTAS


KCL2-SUC-05: Retrieve user profile and lifelog schema
Implementation
@GET: Input
URL: localhost/schema
Knowledge Creation & Evolution
Data-Driven
Feature Model Manager
@GET: Input
@GET: Output
JSON
RESTful Web Services
Data Curation Layer
Lifelog Data
AJAX
Asynchronous Javascript And XML
jQuery
@GET: Output
[{
"tblUsers": [
{
"col": "User-ID",
"datatype": "int"
}, {
"col": "Age",
"datatype": "int"
}, {
"col": "Gender",
"datatype": "varchar",
"size": 50
},
......
] },
......
}]
LifelogSchema
tblUsers
User-ID (int)
UserName (varchar (500))
UserAge (int)
tblUserGoal
UserGoal-ID (int)
BMI (float)
WeightStatus (varchar (500))
TotalCaloriesToBurn (int)
CaloriesBurnedPerDay (int)
Recommendation (varchar (500))
tblActivities
Activity-ID (int)
ActivityName (varchar (500))

KCL2-SUC-01: Build feature model

Use Case

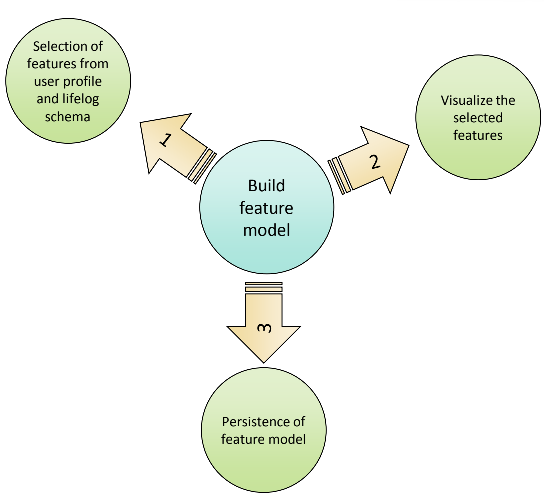
- **Objective**
  - ➢ To define the valid combinations of features in a domain for building feature model
- **Methodology**
  - ➢ ***Query Configuration*** *(operation)*
    1. **Input**: Lifelog schema
    2. **Processing**:
       a. Domain expert retrieves the schema from schema storage.
       b. System loads and plots the schema
       c. Domain expert builds the feature models as follows:
          a. Select the required features for corresponding domain
          b. Verify the consistency of the selected features (such as concept hierarchy)
          c. Save the feature model
       d. System persists the feature model into repository.
    3. **Output**: builds the feature model and forwards that model to *Lifelog Data Loader*

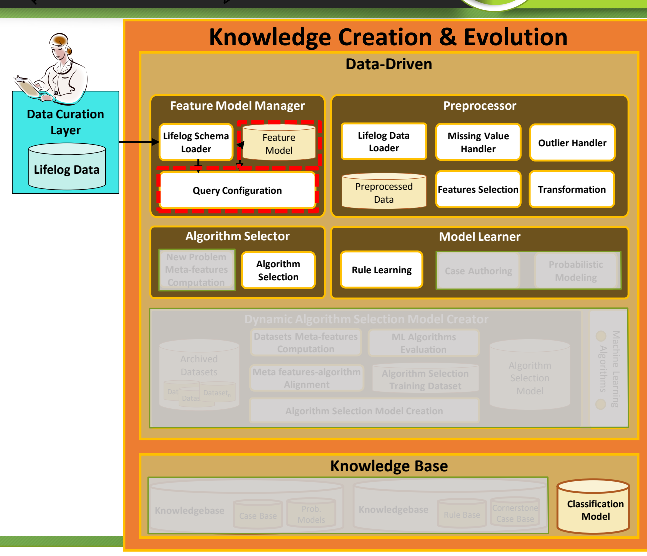


경희대학교 KYUNG HEE UNIVERSITY UTAS

KCL2-SUC-01: Build feature model

Use Case Functions

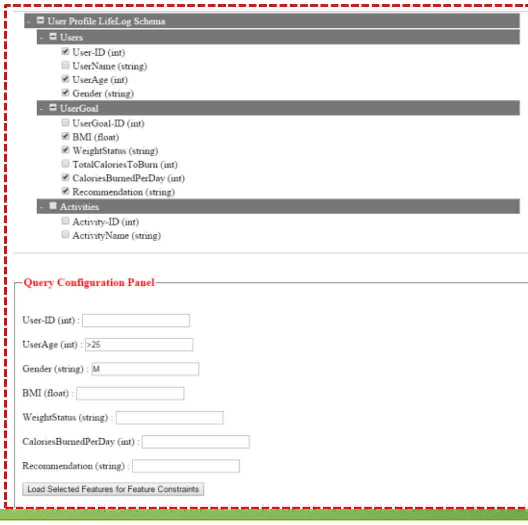




경희대학교 KYUNG HEE UNIVERSITY UTAS

KCL2-SUC-01: Build feature model

Implementation




```
<<Physical Activity Schema>>
<?xml version="1.0" encoding="UTF-8"?>
<xs:element name="tblUsers">
 <xs:complexType>
  <xs:sequence>
       <xs:element name="User-ID" type="xs:int"/>
       <xs:element name="Age" type="xs:int"/>
       <xs:element name="Gender" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
 <xs:key name="tblUserKey" msdata:PrimaryKey="true">
  <xs:selector xpath="./tblUser"/>
  <xs:field xpath="User-ID"/>
 </xs:key>
</xs:element>
<xs:element name="tblUserGoal">
 <xs:complexType>
  <xs:sequence>
       <xs:element name="BMI" type="xs:float"/>
       <xs:element name="WeightStatus" type="xs:string"/>
       <xs:element name="CaloriesBurnedPerDay" type="xs:int"/>
       <xs:element name="Recommendation" type="xs:string"/>
  </xs:sequence>
 </xs:complexType>
 <xs:keyref name="tblUserGoalFKey" refer="tblUserKey">
  <xs:selector xpath="./tblUserGoal" />
  <xs:field xpath="User-ID" />
 </xs:keyref>
</xs:element>
```

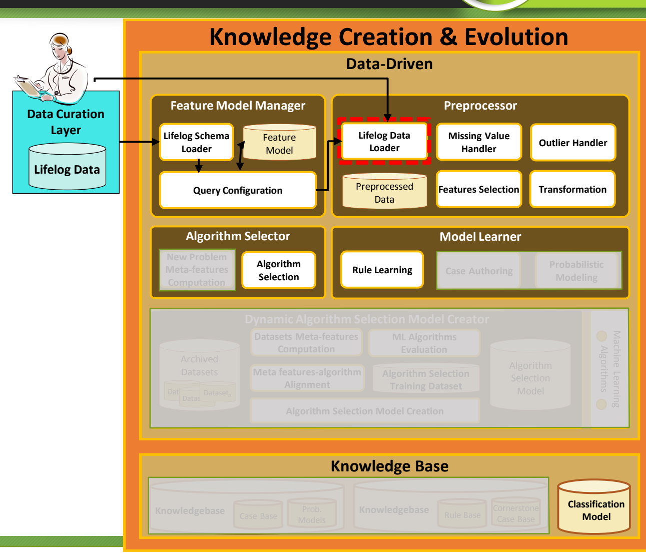
Feature Model

경희대학교 KYUNG HEE UNIVERSITY UTAS

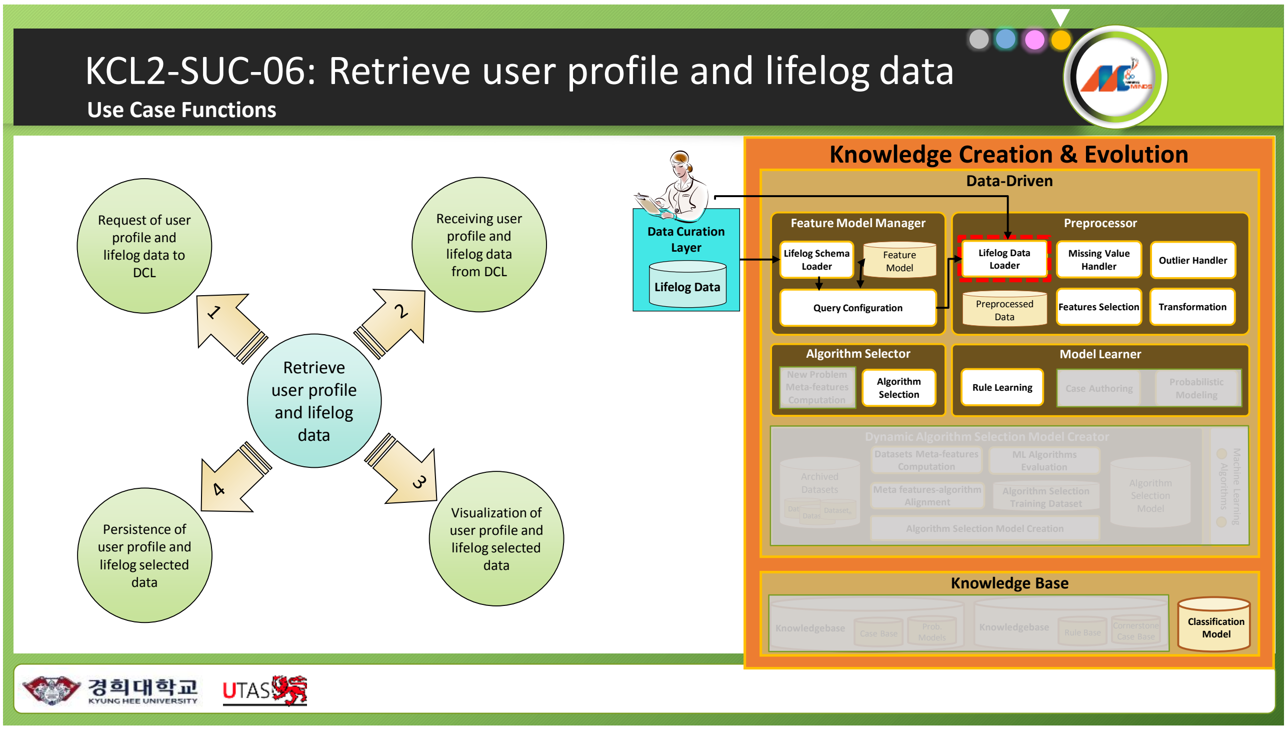
KCL2-SUC-06: Retrieve user profile and lifelog data

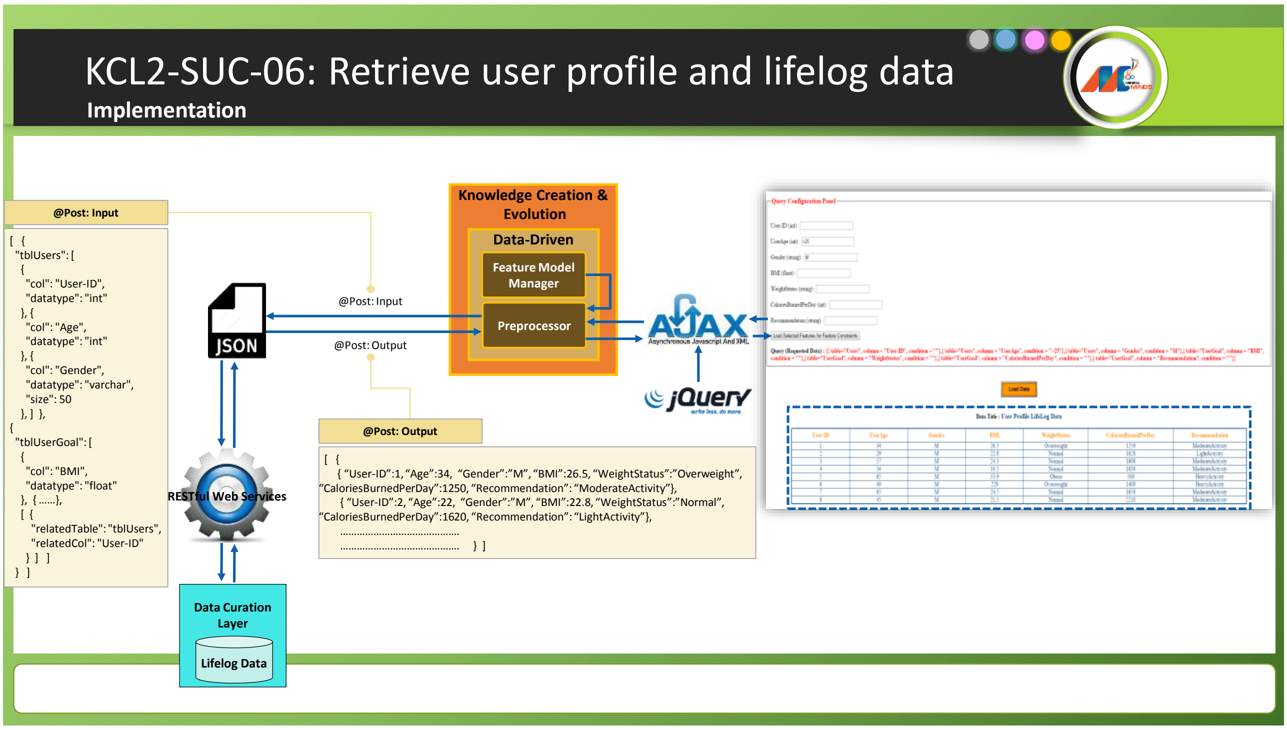
Use Case

- **Objective**
  - To help domain expert to view unprocessed user profile and lifelog data
- **Methodology**
  - ***Communication***: Restful Service between *Preprocessor* and *DCL*
  - ***Lifelog Data Loader*** *(operation)*
    1. **Input**: Feature model and lifelog data
    2. **Processing**:
       a. Domain expert loads the feature model for selected domain
       b. System loads the corresponding feature model
       c. Domain expert sends request to DCL for user profile and lifelog data based on loaded feature model
       d. DCL shares the user profile and lifelog data
       e. System receives the user profile and lifelog data
       f. System saves the received lifelog data
    3. **Output**: forwards the lifelog data to *Missing Value Handler*



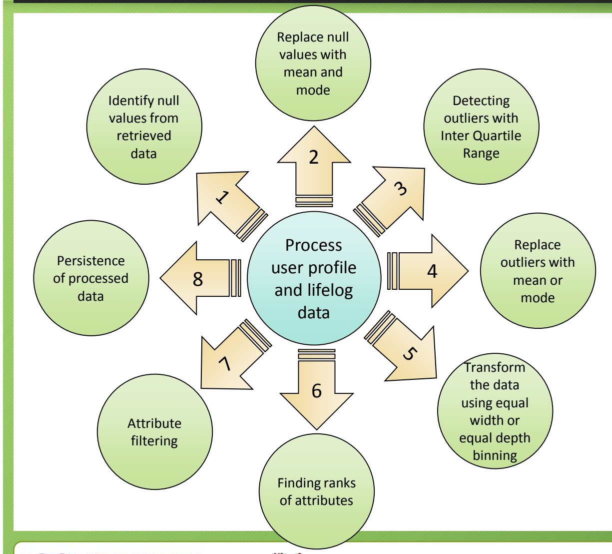
경희대학교 KYUNG HEE UNIVERSITY UTAS

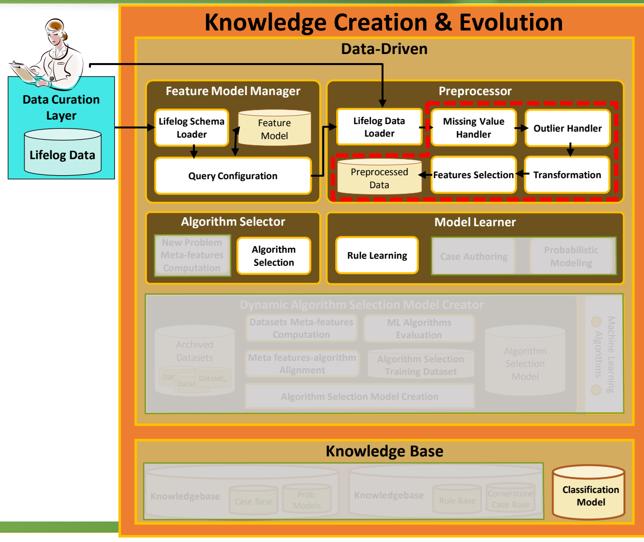

KCL2-SUC-06: Retrieve user profile and lifelog data
Use Case Functions
Request of user profile and lifelog data to DCL
Receiving user profile and lifelog data from DCL
Retrieve user profile and lifelog data
Persistence of user profile and lifelog selected data
Visualization of user profile and lifelog selected data
1
2
3
4
Data Curation Layer
Lifelog Data
Knowledge Creation & Evolution
Data-Driven
Feature Model Manager
Lifelog Schema Loader
Feature Model
Query Configuration
Preprocessor
Lifelog Data Loader
Missing Value Handler
Outlier Handler
Preprocessed Data
Features Selection
Transformation
Algorithm Selector
New Problem Meta-features Computation
Algorithm Selection
Model Learner
Rule Learning
Case Authoring
Probabilistic Modeling
Dynamic Algorithm Selection Model Creator
Knowledge Base
Classification Model
경희대학교
KYUNG HEE UNIVERSITY
UTAS


KCL2-SUC-06: Retrieve user profile and lifelog data
Implementation
@Post: Input
[{
"tblUsers": [
{
"col": "User-ID",
"datatype": "int"
}, {
"col": "Age",
"datatype": "int"
}, {
"col": "Gender",
"datatype": "varchar",
"size": 50
},] },
{
"tblUserGoal": [
{
"col": "BMI",
"datatype": "float"
}, {......},
[{
"relatedTable": "tblUsers",
"relatedCol": "User-ID"
}]]
}]
JSON
@Post: Input
@Post: Output
Knowledge Creation & Evolution
Data-Driven
Feature Model Manager
Preprocessor
AJAX
Asynchronous Javascript And XML
jQuery
write less, do more.
RESTful Web Services
Data Curation Layer
Lifelog Data
@Post: Output
[{
{ "User-ID":1, "Age":34, "Gender":"M", "BMI":26.5, "WeightStatus":"Overweight", "CaloriesBurnedPerDay":1250, "Recommendation": "ModerateActivity"},
{ "User-ID":2, "Age":22, "Gender":"M", "BMI":22.8, "WeightStatus":"Normal", "CaloriesBurnedPerDay":1620, "Recommendation": "LightActivity"},
}]

KCL2-SUC-02: Prepare lifelog and user profile data

Use Case Functions

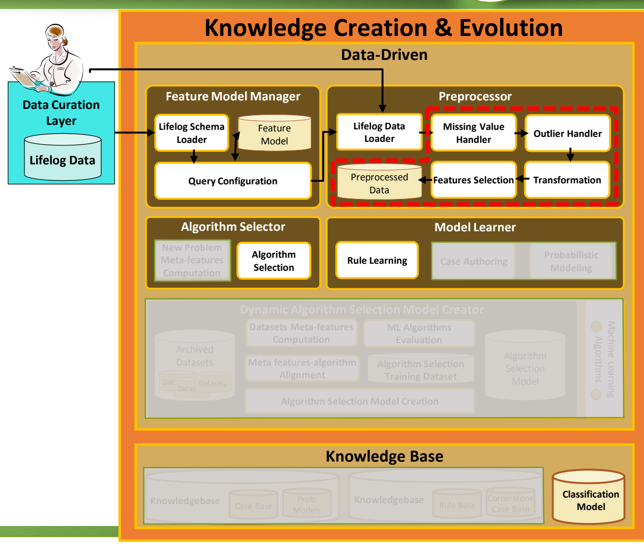




KCL2-SUC-02: Prepare lifelog and user profile data (1/4)

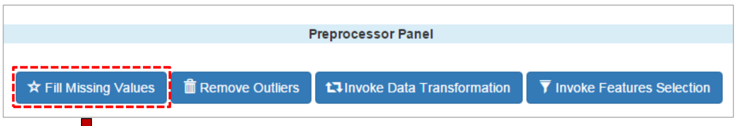
Missing Value Handler | Outlier Handler | Transformation | Features Selection

- **Objective**
  - ➢ To preprocess the data to generate models with high accuracy.
- **Methodology**
  - ➢ ***Missing Value Handler*** *(operation)*
    1. **Input**: unprocessed data
    2. **Processing**:
       a. Domain expert loads the unprocessed data
       b. System displays the retrieved data
       c. For each attribute:
          i. Domain expert identifies the missing values and select appropriate method from following options for missing value replacement.
             - Mean
             - Mode
          ii. System replaces the missing values using selected method.
       d. Domain expert saves the processed data into repository
    3. **Output**: forwards the processed data to *Outlier Handler*



KCL2-SUC-02: Prepare lifelog and user profile data (1/4)

Missing Value Handler | Outlier Handler | Transformation | Features Selection



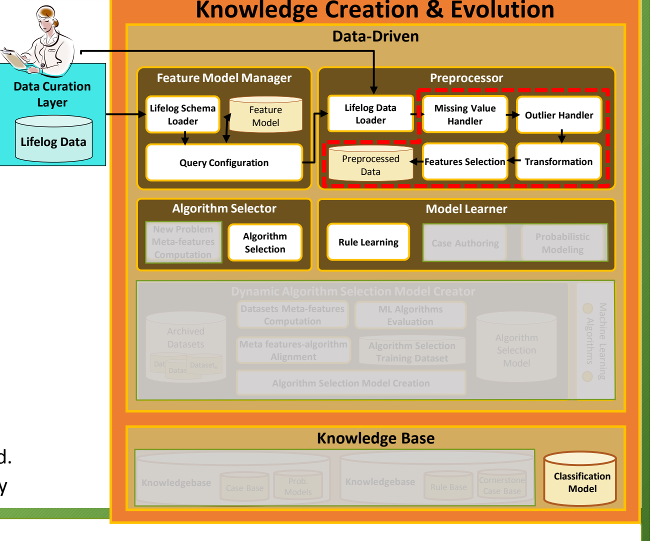
User-ID	Age	Gender	BMI	WeightStatus	CaloriesBurnedPerDay	Recommendation
1	34	M	26.5	Overweight	1250	ModerateActivity
2	22	M	22.8	Normal	1620	LightActivity
3	18	M	24.3	Normal	1600	ModerateActivity
4	34	F		Normal	1630	ModerateActivity
5	65	F	33.9	Obese	500	HeavyActivity
..	..	..	..	..	...	..
19	19	M	229.0	Overweight	1400	HeavyActivity
20	65	M	24.5	Normal	1650	ModerateActivity

- **Mean/Mode substitution**
  - Replace missing value with sample mean or mode, which are two kinds of "averages".
  - The "**mean**" is the "average" you're used to, where you add up all the numbers and then divide by the number of numbers.
  - The "**mode**" is the value that occurs most often. If no number is repeated, then there is no mode for the list.
- Dummy Variable Adjustment
  - Create an indicator for missing value (1=value is missing for observation; 0=value is observed for observation)
  - Impute missing values to a constant (such as the mean)
  - Include missing indicator in regression
- Regression Imputation
  - Replaces missing values with predicted score from a regression equation.

KCL2-SUC-02: Prepare lifelog and user profile data (2/4)

Missing Value Handler | **Outlier Handler** | Transformation | Features Selection

- **Objective**
  - To preprocess the data to generate models with high accuracy.
- **Methodology**
  - ***Outlier Handler*** *(operation)*
    1. **Input**: processed data
    2. **Processing**:
       a. Domain expert loads the processed data
       b. System displays the retrieved data
       c. For each attribute:
          i. Domain expert apply outlier detection method such as interquartile range and scatterplot.
          ii. System display the outliers
          iii. Domain expert select appropriate method from following options for outlier replacement.
             - Mean
             - Mode
          iv. System replace the outlier using selected method.
       d. Domain expert saves the processed data into repository
    3. **Output**: forwards the processed data to *Transformation*



KCL2-SUC-02: Prepare lifelog and user profile data (2/4)

Missing Value Handler | **Outlier Handler** | Transformation | Features Selection

Preprocessor Panel

Fill Missing Values | Remove Outliers | Invoke Data Transformation | Invoke Features Selection

User-ID	Age	Gender	BMI	WeightStatus	CaloriesBurnedPerDay	Recommendation
1	34	M	26.5	Overweight	1250	ModerateActivity
2	22	M	22.8	Normal	1620	LightActivity
3	18	M	24.3	Normal	1600	ModerateActivity
4	34	F		Normal	1630	ModerateActivity
5	65	F	33.9	Obese	500	HeavyActivity
..	..	..	..	..	...	..
19	19	M	229.0	Overweight	1400	HeavyActivity
20	65	M	24.5	Normal	1650	ModerateActivity

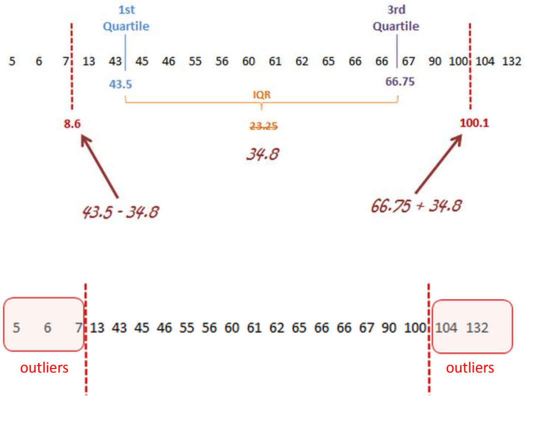
- Outlier
  - An outlier is an observation which deviates so much from the other observations
- Detection Methods
  - **Statistical Methods**
    - **Parametric**
    - **Non-Parametric**
      - **Histogram**
      - **Inter Quartile Ranges (IQR)**
  - Proximity-Based Methods
  - Clustering-Based Methods

경희대학교 KYUNG HEE UNIVERSITY UTAS

KCL2-SUC-02: Prepare lifelog and user profile data (2/4)

Missing Value Handler | **Outlier Handler** | Transformation | Features Selection

- **Inter Quartile Ranges (IQR)**
  - Calculate the interquartile range (IQR) of data
    - IQR = Quartile 3 – Quartile 1
  - Multiply the interquartile range (IQR) by the number 1.5
  - Add 1.5 x (IQR) to the third quartile. Any number greater than this is a suspected outlier.
    - (Quartile 3) + (1.5 × IQR)
  - Subtract 1.5 x (IQR) from the first quartile. Any number less than this is a suspected outlier.
    - (Quartile 1) – (1.5 × IQR)

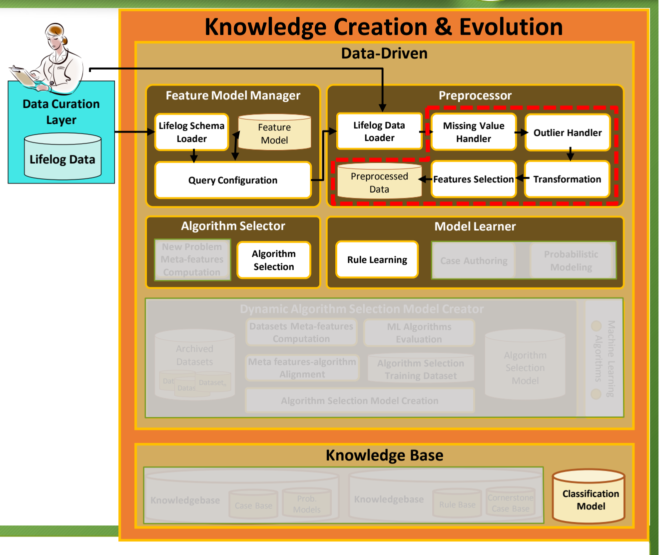


경희대학교 KYUNG HEE UNIVERSITY UTAS

KCL2-SUC-02: Prepare lifelog and user profile data (3/4)

Missing Value Handler | Outlier Handler | **Transformation** | Features Selection

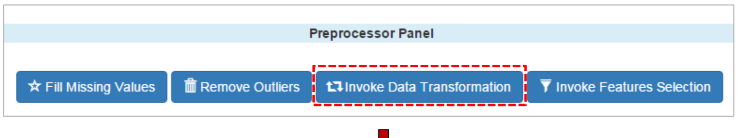
- **Objective**
  - ➢ To preprocess the data to generate models with high accuracy.
- **Methodology**
  - ➢ ***Transformation*** *(operation)*
    1. **Input**: processed data
    2. **Processing**:
       a. Domain expert loads the processed data
       b. System displays the retrieved data
       c. For each attribute:
          i. Domain expert identifies, normalizes the non-transformed values, and updates the dataset.
          ii. System modifies the values set and update the dataset
       d. Domain expert saves the processed data into repository
    3. **Output**: forwards the processed data to *Feature Selection*



경희대학교 KYUNG HEE UNIVERSITY UTAS

KCL2-SUC-02: Prepare lifelog and user profile data (3/4)

Missing Value Handler | Outlier Handler | **Transformation** | Features Selection



User-ID	Age	Gender	BMI	WeightStatus	CaloriesBurnedPerDay	Recommendation
1	34	M	26.5	Overweight	1250	ModerateActivity
2	22	M	22.8	Normal	1620	LightActivity
3	18	M	24.3	Normal	1600	ModerateActivity
4	34	F		Normal	1630	ModerateActivity
5	65	F	33.9	Obese	500	HeavyActivity
..	..	..	..	..	...	.
19	19	M	229.0	Overweight	1400	HeavyActivity
20	65	M	24.5	Normal	1650	ModerateActivity

> 1650 → High
1600 – 1650 → Normal
< 1600 → Low

CaloriesBurnedPerDay
Low
Normal
Low
Low
Low
...
Low
Normal

- Transformation
  - A function that maps the entire set of values of a given attribute to a new set of replacement values such that each old value can be identified with one of the new values
- Methods
  - **Discretization: Concept hierarchy climbing**
  - Smoothing: Remove noise from data
  - Attribute/feature construction
    - New attributes constructed from the given ones
  - Aggregation: Summarization, data cube construction
  - Normalization: Scaled to fall within a smaller, specified range

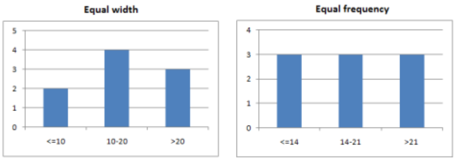
경희대학교 KYUNG HEE UNIVERSITY UTAS

KCL2-SUC-02: Prepare lifelog and user profile data (3/4)

Missing Value Handler | Outlier Handler | **Transformation** | Features Selection

- **Discretization:** Concept hierarchy climbing
  - Divide the range of a continuous attribute into intervals
  - Interval labels can then be used to replace actual data values
  - Discretization can be performed recursively on an attribute
- Typical methods:
  - **Binning**
    - Equal-width (distance) partitioning
      - Divides the range into N intervals of equal size: uniform grid
      - if A and B are the lowest and highest values of the attribute, the width of intervals will be: W = (B –A)/N.
    - Equal-depth (frequency) partitioning
      - Divides the range into N intervals, each containing approximately same number of samples
  - Histogram analysis
  - Clustering analysis
  - Correlation analysis

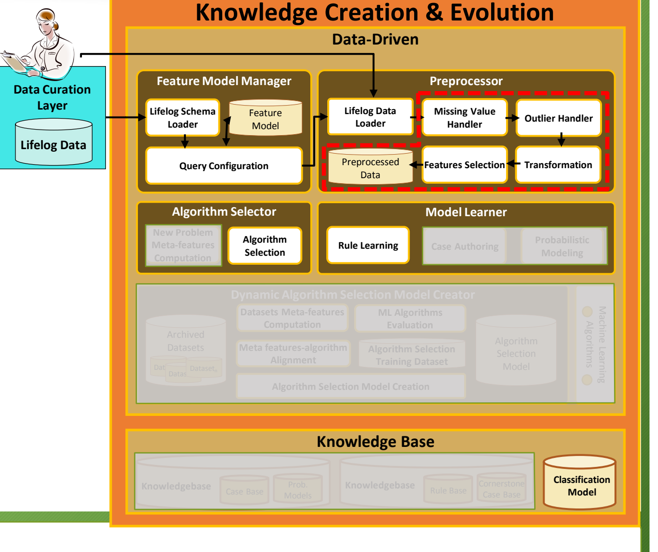
- **Data :** 0, 4, 12, 16, 16, 18, 24, 26, 28
- **Equal width**
  - Bin 1: 0, 4 [-,10)
  - Bin 2: 12, 16, 16, 18 [10,20)
  - Bin 3: 24, 26, 28 [20,+)
- **Equal frequency**
  - Bin 1: 0, 4, 12 [-, 14)
  - Bin 2: 16, 16, 18 [14, 21)
  - Bin 3: 24, 26, 28 [21,+)



KCL2-SUC-02: Prepare lifelog and user profile data (4/4)

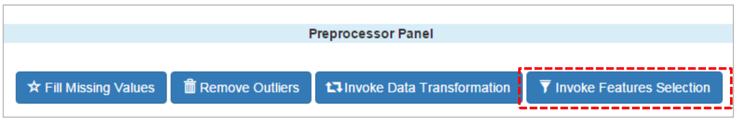
Missing Value Handler | Outlier Handler | Transformation | **Features Selection**

- **Objective**
  - To preprocess the data to generate models with high accuracy.
- **Methodology**
  - ***Feature Selection*** *(operation)*
    1. **Input**: processed data
    2. **Processing**:
       a. Domain expert loads the processed data
       b. System displays the retrieved data
       c. Domain expert applies the attributes filtration techniques (i.e. ranking)
       d. System computes the ranks for all attributes and displays to expert
       e. Domain expert select the highly ranked attributes (i.e. rank value >= 0.8)
       f. System filters the attributes based on selected attributes and displays to domain expert
       g. Domain expert saves the processed data into repository
    3. **Output**: processed data



KCL2-SUC-02: Prepare lifelog and user profile data (4/4)

Missing Value Handler | Outlier Handler | Transformation | **Features Selection**



User-ID	Age	Gender	BMI	WeightStatus	CaloriesBurnedPerDay	Recommendation
1	34	M	26.5	Overweight	1250	ModerateActivity
2	22	M	22.8	Normal	1620	LightActivity
3	18	M	24.3	Normal	1600	ModerateActivity
4	34	F		Normal	1630	ModerateActivity
5	65	F	33.9	Obese	500	HeavyActivity
..	..	..	..	..	…	..
19	19	M	229.0	Overweight	1400	HeavyActivity
20	65	M	24.5	Normal	1650	ModerateActivity

- Feature Selection
  - It is the process of selecting a subset of relevant features for use in model construction.
- Feature Selection Algorithms
  - **Filter Methods**
    - Chi Squared test
    - **Information Gain score**
    - Correlation Coefficient score
  - Wrapper Methods
  - Embedded Methods

$$Information\ Gain(S, F_i) = Entropy(S) - \sum_{v_i \epsilon V_{Fj}} \frac{|S_{vi}|}{|S|} * Entropy(S_{vi})$$

$$Entropy(S) = \sum_{i=1}^{c} -p_i * log_2(p_i)$$

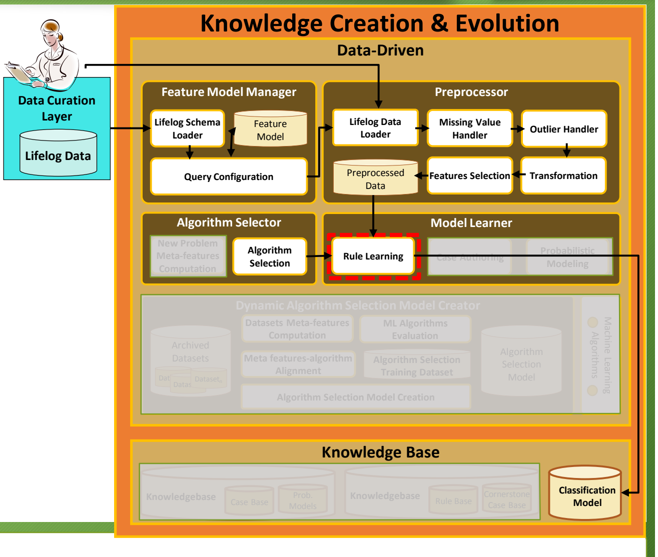
Where: p_i - proportion of the examples belonging to the i-th class.
S_{vi} – subset of S, for which feature F_j has value vi
V_{Fj} – set of all possible values of feature F_j

경희대학교 KYUNG HEE UNIVERSITY UTAS

KCL2-SUC-13: Learn classification model

Use Case

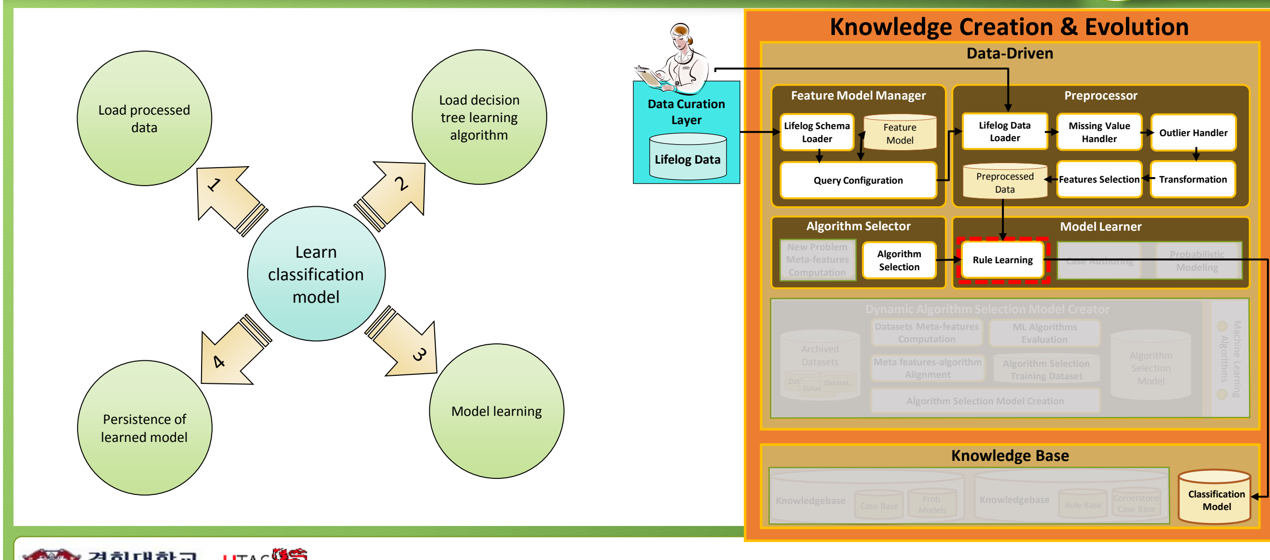
- **Objective**
  - To generate the classification model.
- **Methodology**
  - ***Rule Learning*** *(operation)*
    1. **Input**: processed data
    2. **Processing**:
       a. Domain expert loads the user profile lifelog processed data for selected domain
       b. System loads the corresponding processed data
       c. Domain expert selects the J48 learning algorithm
       d. System loads the J48 decision tree algorithm and generate the classification model
       e. Domain expert saves the model.
    3. **Output**: classification model



경희대학교 KYUNG HEE UNIVERSITY UTAS

KCL2-SUC-13: Learn classification model

Use Case Functions

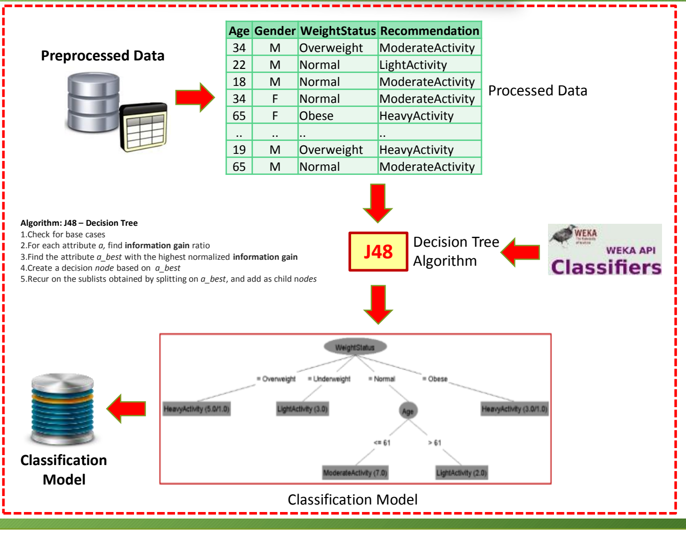


KCL2-SUC-13: Learn classification model

Implementation

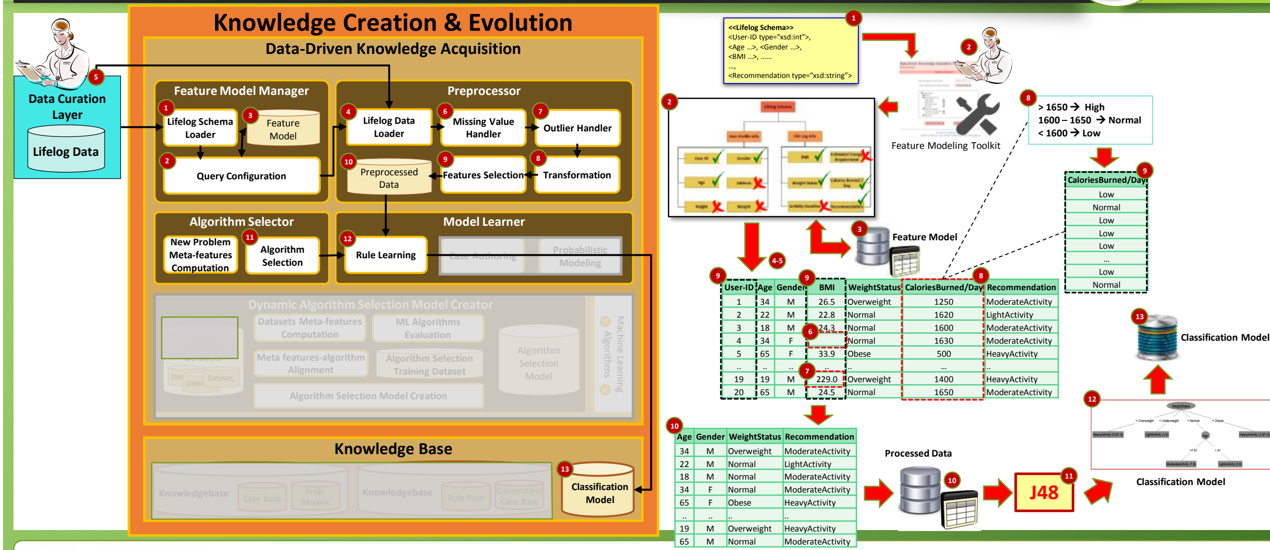
For creating Classification Model using J48 Algorithm, click *'Learn Model'* button.

Learn Model



Age	Gender	WeightStatus	Recommendation
34	M	Overweight	ModerateActivity
22	M	Normal	LightActivity
18	M	Normal	ModerateActivity
34	F	Normal	ModerateActivity
65	F	Obese	HeavyActivity
..	..	..	..
19	M	Overweight	HeavyActivity
65	M	Normal	ModerateActivity

Workflow - MM V2.0

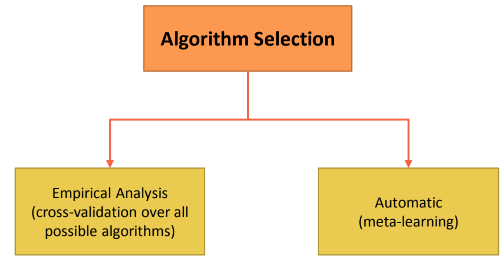


Dependencies, Issues and Challenges

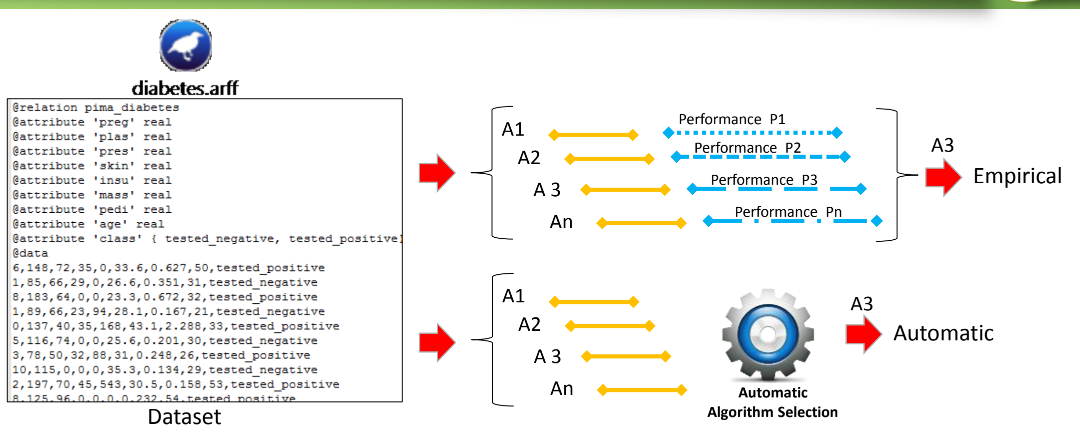
- The user profile and life log schema and data retrieval from DCL
- Feature model development
- Data preprocessing
- Model learning after configuration with available machine learning algorithms API's

Algorithm Selection

- The process of selecting an appropriate algorithm for learning a dataset
- Types of selecting appropriate algorithms
  - Empirical analysis
  - Automatic recommendation



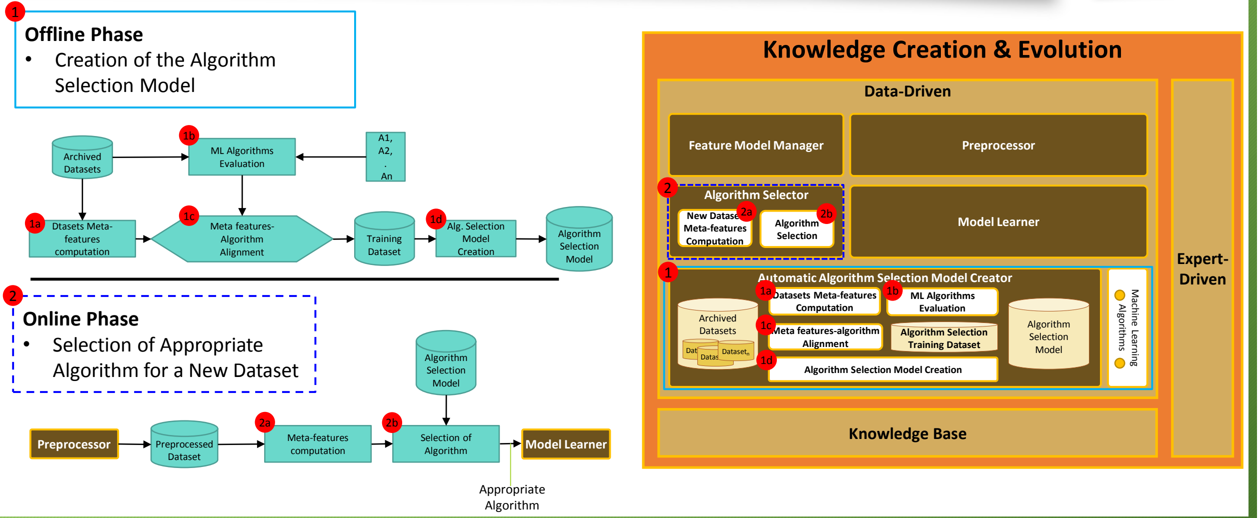
Algorithm Selection



경희대학교 KYUNG HEE UNIVERSITY UTAS

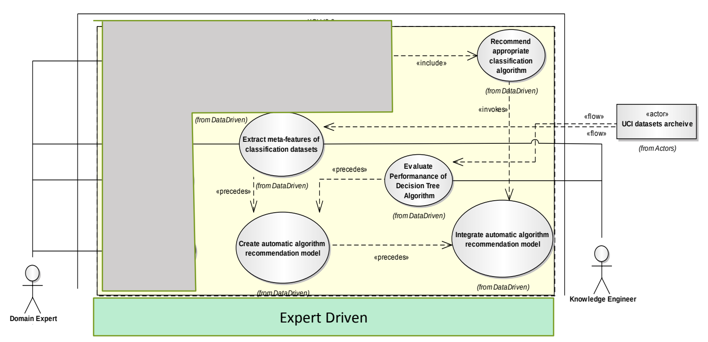
Automatic Algorithm Selection Process

Offline Phase | Online Phase



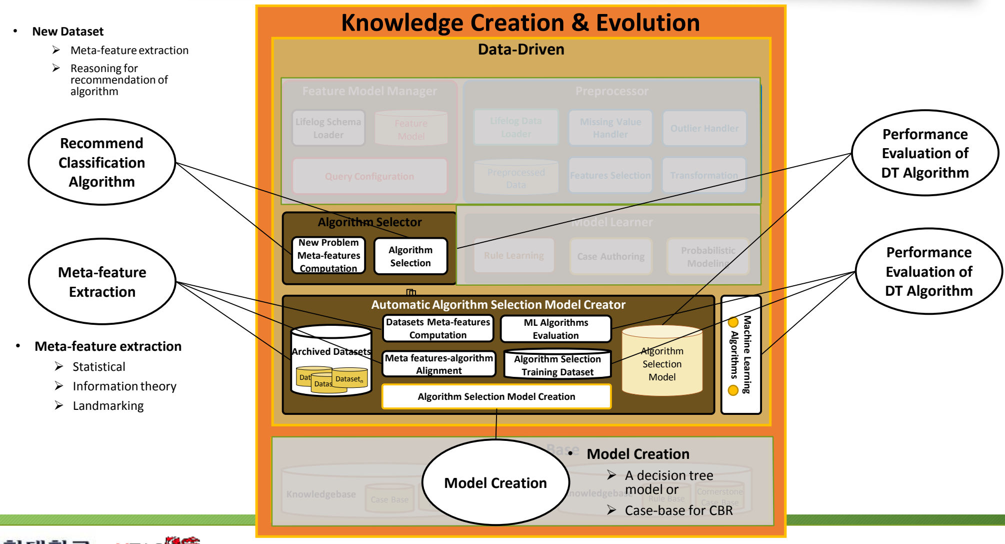
Use-case diagram of automatic algorithm selection

Use Case #ID	Use Case Name
KCL2-SUC-07	Extract meta-features of classification datasets
KCL2-SUC-08	Evaluate performance of classification algorithms
KCL2-SUC-09	Create automatic algorithm selection Model
KCL2-SUC-10	Integrate automatic algorithm recommendation model
KCL2-SUC-11	Recommend appropriate classification algorithm



경희대학교 KYUNG HEE UNIVERSITY UTAS

Use-case diagram of automatic algorithm selection



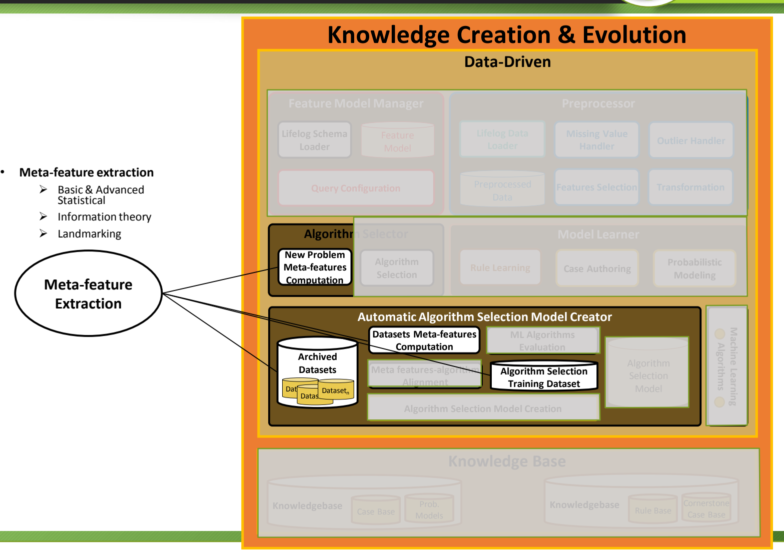
경희대학교 KYUNG HEE UNIVERSITY UTAS

Meta-Features Extraction

Use Case

Use-case (KCL2-SUC-07): *Extract meta-features of classification dataset*

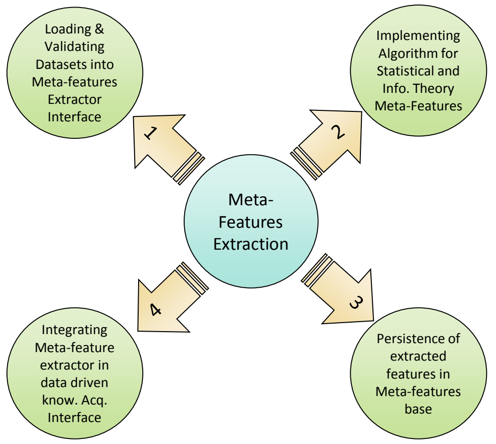
- **Objective**
  - Extraction of meta-characteristics
- **Methodology**
  - *Library used*: OpenML dataset characterization library is used
  - ***Dataset Meta-features computation*** *(operation)*
    1. **Input**: UCI and OpenMl classification datasets
    2. **Processing**:
       a. Takes each dataset form the local copy of the datasets
       b. Extracts basic and advanced statistical meta-feature and information theory features
       c. Stores the extracted features to meta-feature base
    3. **Output**:
       a. A set of meta-features

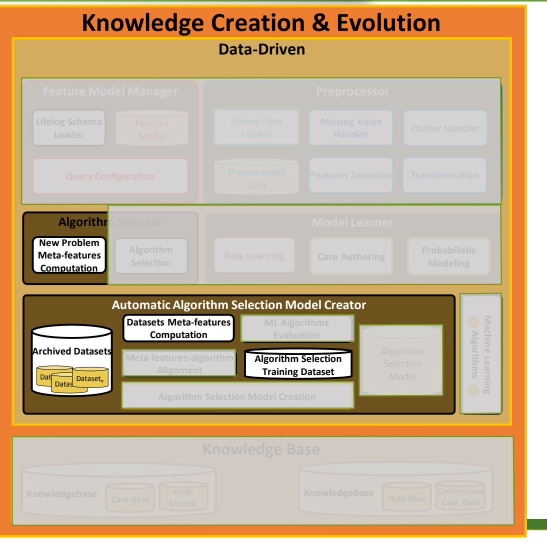


경희대학교 KYUNG HEE UNIVERSITY UTAS

Meta-Features Extraction

Use Case Features Functions





List of Meta-features

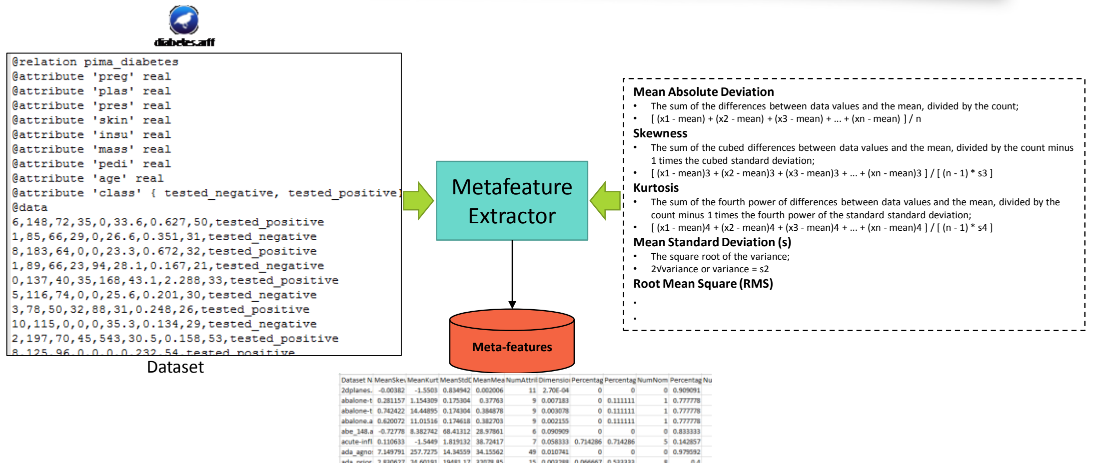
No.	Basic Statistical Features
Simp. Statistic 1	InstanceCount
Simp. Statistic 2	NumAttributes
Simp. Statistic 3	ClassCount
Simp. Statistic 4	PercentageOfBinaryAtts
Simp. Statistic 5	PercentageOfNominalAtts
Simp. Statistic 6	PercentageOfNumericAtts
Simp. Statistic 7	AttrWithOutlier.Prop
Simp. Statistic 8	MeanSkewnessOfNumericAtts
Simp. Statistic 9	MeanKurtosisOfNumericAtts
Simp. Statistic 10	MeanAbsCoef
Simp. Statistic 11	Dimensionality
Simp. Statistic 12	NumBinaryAtts
Simp. Statistic 13	NumNominalAtts
Simp. Statistic 14	NumNumericAtts
Simp. Statistic 15	NumMissingValues

No.	Advanced Statistical Features
Advanced Statistic 1	MeanStdDevOfNumericAtts
Advanced Statistic 2	MeanMeansOfNumericAtts
Advanced Statistic 3	NegativePercentage
Advanced Statistic 4	PositivePercentage
Advanced Statistic 5	DefaultAccuracy
Advanced Statistic 6	IncompleteInstanceCount
Advanced Statistic 7	PercentageOfMissingValues
Advanced Statistic 8	MinNominalAttDistinctValues
Advanced Statistic 9	MaxNominalAttDistinctValues
Advanced Statistic 10	StdvNominalAttDistinctValues
Advanced Statistic 11	MeanNominalAttDistinctValues

No.	Information Theory Features
InfTheory 1	ClassEntropy
InfTheory 2	MeanAttributeEntropy
InfTheory 3	MeanMutualInformation
InfTheory 4	EquivalentNumberOfAtts
InfTheory 5	NoiseToSignalRatio

Total 31 Meta-features

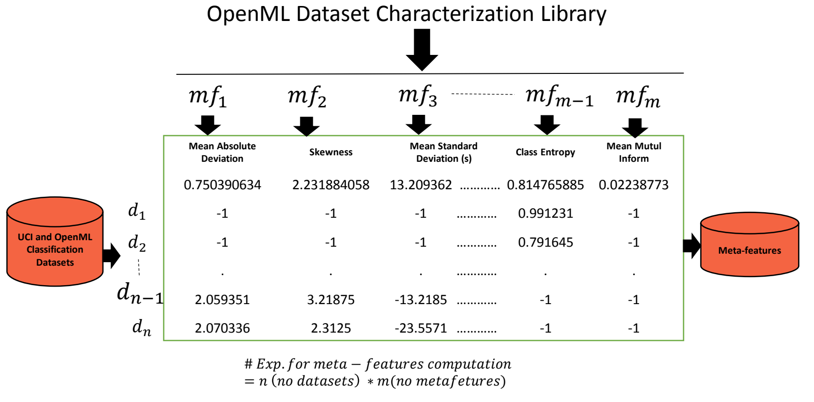
Meta-features Extraction (One Dataset)



경희대학교 KYUNG HEE UNIVERSITY UTAS

Meta-features Extraction (For All Datasets)

- Types of meta-features
  - Basic statistical (13)
  - Advanced statistical (11)
  - Information theory (5)
  - Complexity characteristics
  - Landmarking features
  - Model-based meta-features
- Types datasets
  - Classification datasets (80-100)
  - Clustering



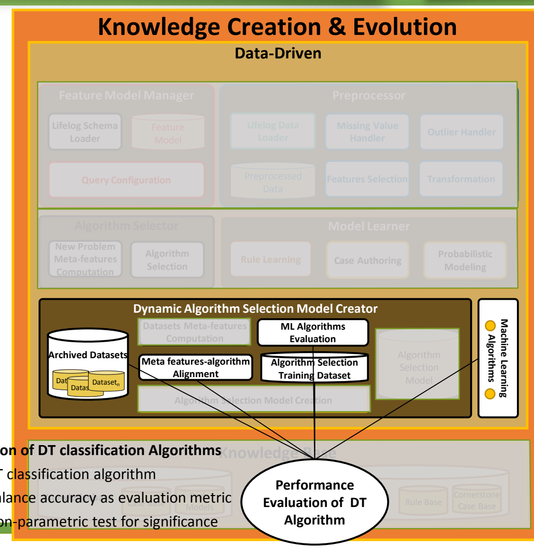
경희대학교 KYUNG HEE UNIVERSITY UTAS

Performance Evaluation of Classification Algorithm

Use-case (KCL2-SUC-08): *Evaluate decision tree classification algorithms*

- **Objective**
  - To find the most appropriate DT algorithm for the datasets
- **Methodology**
  - *Library used*: Weka, Excel Statistician Tool
  - ***ML algorithm evaluation*** *(operation)*

1. **Input**: UCI and OpenMl classification datasets, Weka DT algorithms
2. **Processing**:
   a. Takes each dataset form the local copy of the archeived datasets
   b. Use Weka experimenter and test the DT algorithm on the loaded dataset
   c. Obtain all the evaluation metrics
   d. Compute Balanced accuracy
   e. Perform parametric test for the significantly better DT algorithm
   f. Select the appropriate algorithm
   g. Align selected algorithm with the meta-features of that dataset
   h. Select appropriate algorithm as a class label
3. **Output**: Appropriate algorithm

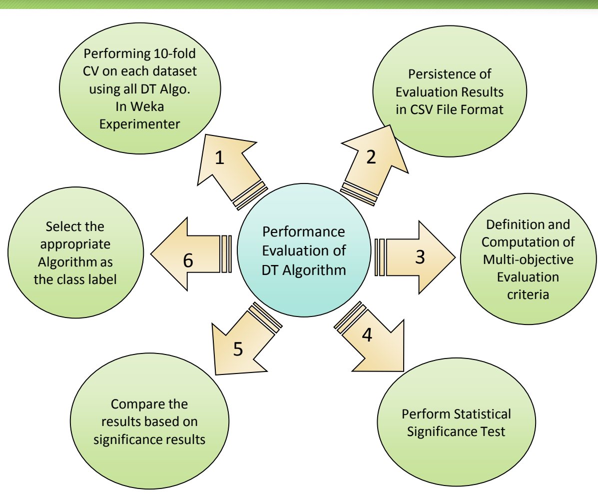


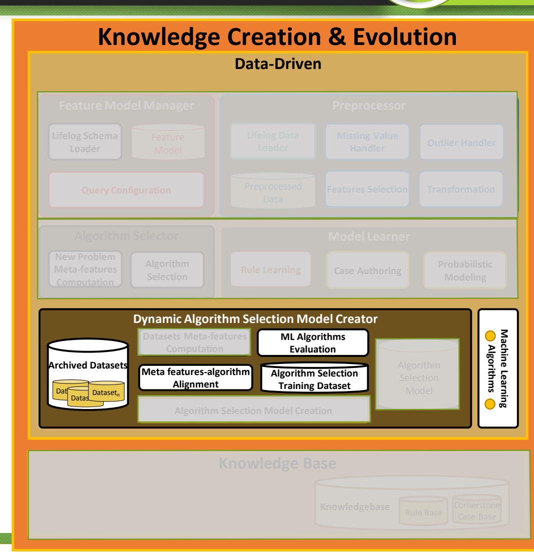
- **Evaluation of DT classification Algorithms**
  - DT classification algorithm
  - Balance accuracy as evaluation metric
  - Non-parametric test for significance

경희대학교 KYUNG HEE UNIVERSITY UTAS

Performance Evaluation of Classification Algorithm

Use Case Feature Functions





경희대학교 KYUNG HEE UNIVERSITY UTAS

Evaluation of Decision Tree Algorithms (One Dataset)

diabetes.arff

```
@relation pima_diabetes
@attribute 'preg' real
@attribute 'plas' real
@attribute 'pres' real
@attribute 'skin' real
@attribute 'insu' real
@attribute 'mass' real
@attribute 'pedi' real
@attribute 'age' real
@attribute 'class' { tested_negative, tested_positive}
@data
6,148,72,35,0,33.6,0.627,50,tested_positive
1,85,66,29,0,26.6,0.351,31,tested_negative
8,183,64,0,0,23.3,0.672,32,tested_positive
1,89,66,23,94,28.1,0.167,21,tested_negative
0,137,40,35,168,43.1,2.288,33,tested_positive
5,116,74,0,0,25.6,0.201,30,tested_negative
3,78,50,32,88,31,0.248,26,tested_positive
10,115,0,0,0,35.3,0.134,29,tested_negative
2,197,70,45,543,30.5,0.158,53,tested_positive
8,125,96,0,0,0,0.232,54,tested_positive
```

Dataset


```
(1) trees.BF | (2) trees (3) trees (4) trees (5) trees (6) trees (7) trees (8) trees
--------------------------------------------------------------------------------------
        73.14 |   74.49     74.62     74.47     76.18 v   70.25     74.46     74.56
--------------------------------------------------------------------------------------
Key:
(1) trees.BFTree
(2) trees.J48
(3) trees.J48graft
(4) trees.LADTree
(5) trees.RandomForest
(6) trees.RandomTree
(7) trees.REPTree
(8) trees.SimpleCart
```

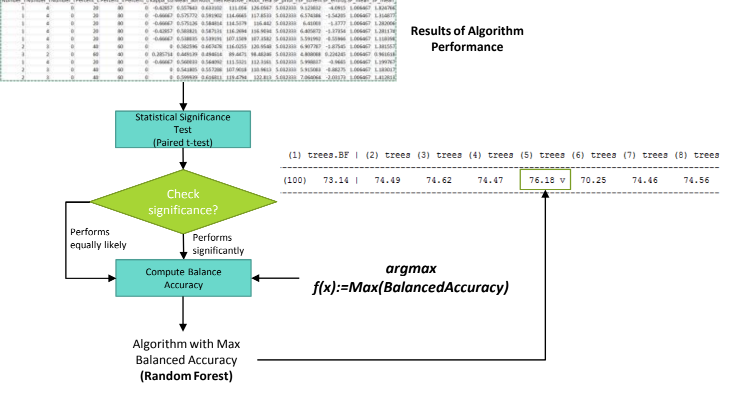
Comparison and Evaluation of the Results

Random Forest (winner)

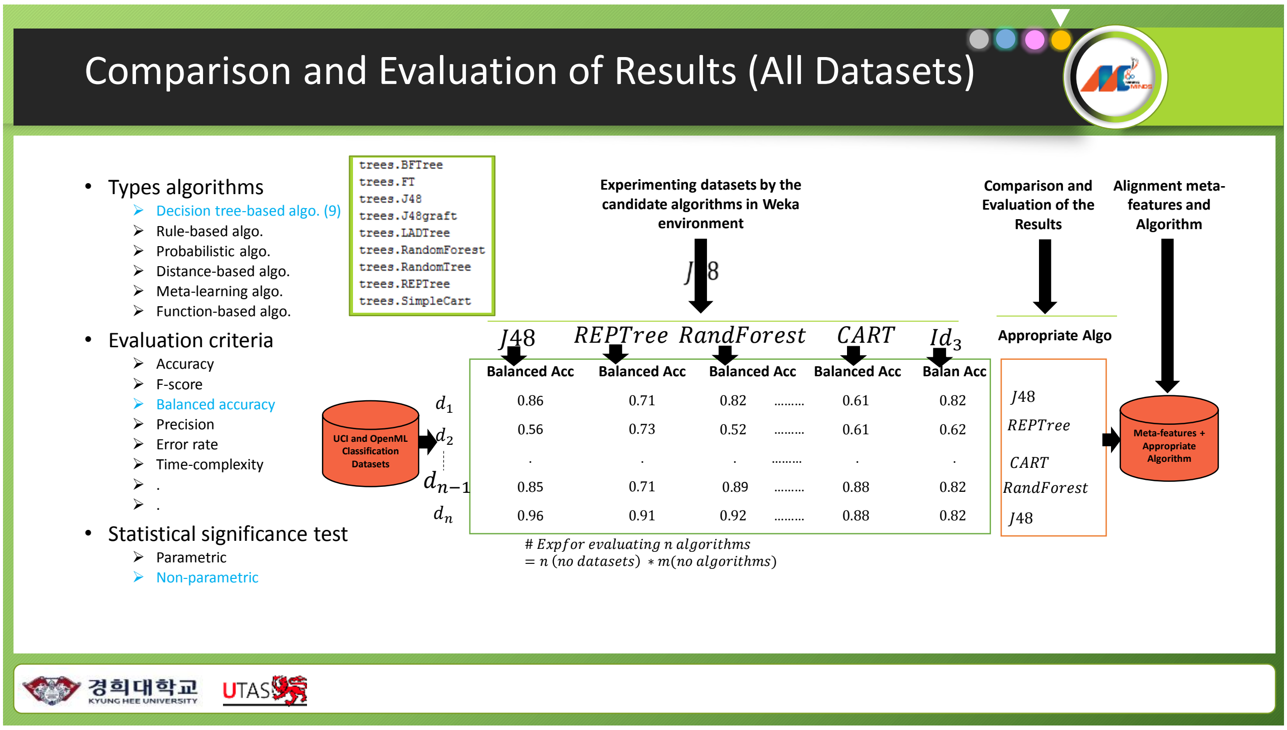
경희대학교 KYUNG HEE UNIVERSITY UTAS

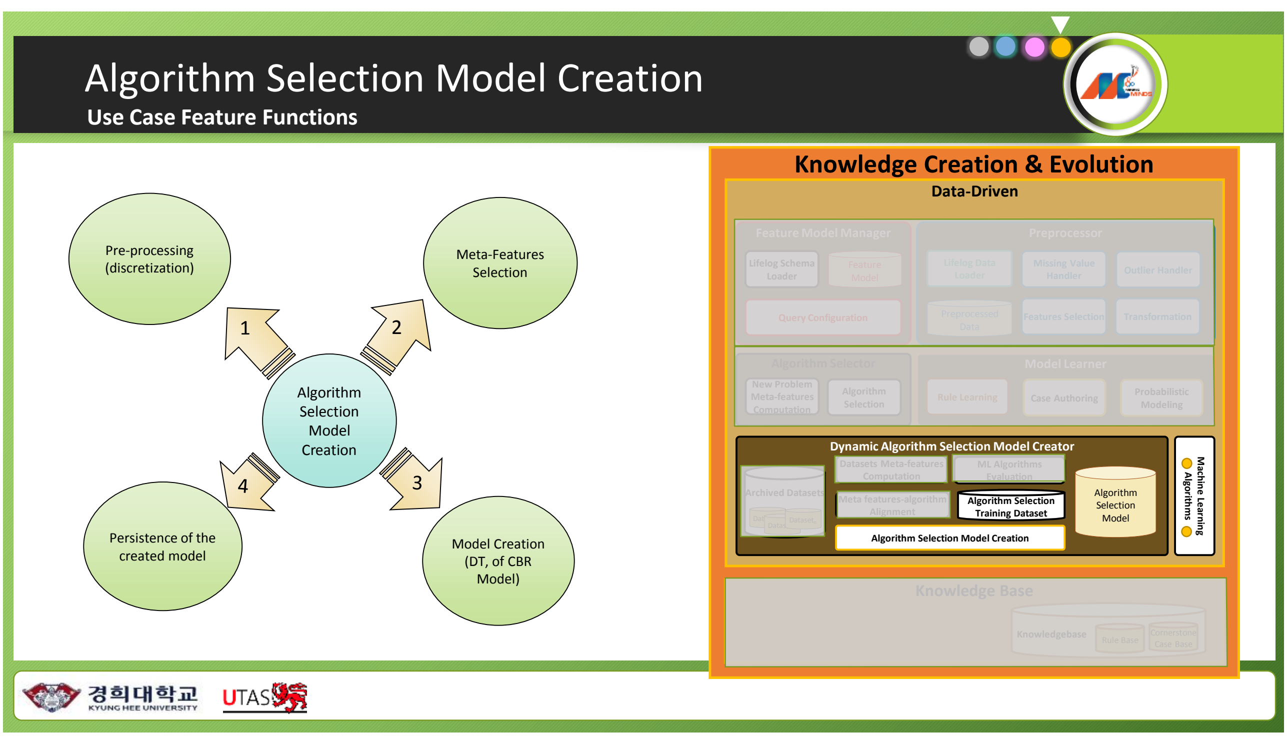
Comparison and Evaluation of Results (One Dataset)

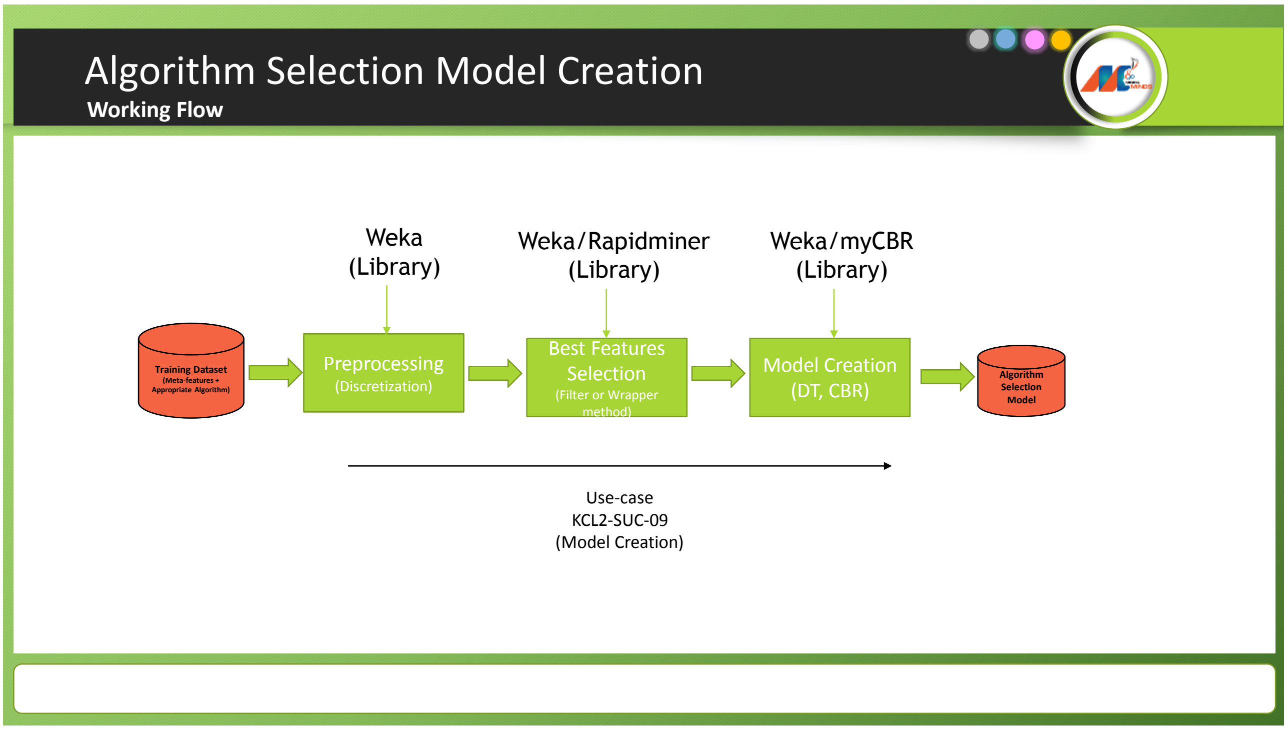
- Perform statistical significance test (Paired t-test)
  1. Perform significantly better
  2. Perform equally likely, cannot decide
  3. Perform significantly poor
- Drops algorithms that performs significantly poor
- Perform evaluation for balanced accuracy
- $Balanced\ Accuracy = \frac{Sensitivity+Specificity}{2} = \frac{0.5*TP}{TP+FN} + \frac{0.5*TN}{TN+FP}$
- Select the algorithm with maximum highest balanced accuracy value

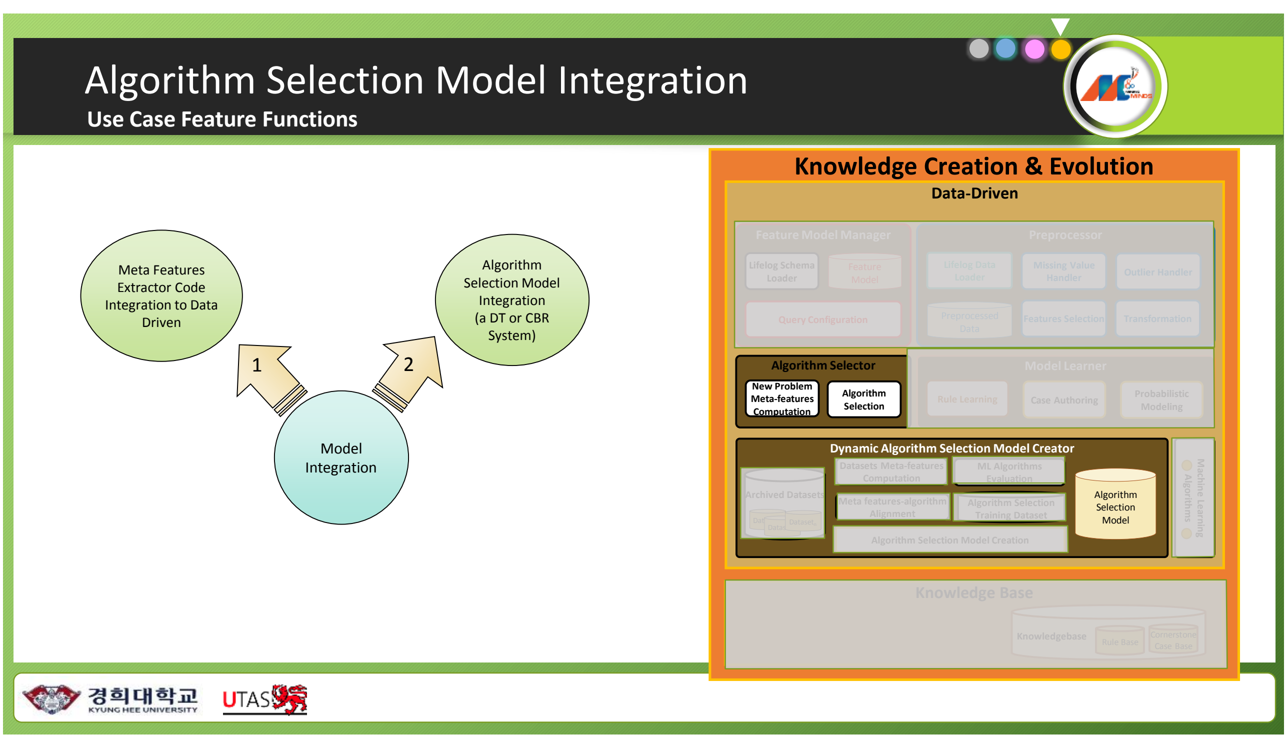


경희대학교 KYUNG HEE UNIVERSITY UTAS


Comparison and Evaluation of Results (All Datasets)
Types algorithms
Decision tree-based algo. (9)
Rule-based algo.
Probabilistic algo.
Distance-based algo.
Meta-learning algo.
Function-based algo.
trees.BFTree
trees.FT
trees.J48
trees.J48graft
trees.LADTree
trees.RandomForest
trees.RandomTree
trees.REPTree
trees.SimpleCart
Evaluation criteria
Accuracy
F-score
Balanced accuracy
Precision
Error rate
Time-complexity
Statistical significance test
Parametric
Non-parametric
Experimenting datasets by the candidate algorithms in Weka environment
Comparison and Evaluation of the Results
Alignment meta-features and Algorithm
J48
J48 REPTree RandForest CART Id3
Balanced Acc Balanced Acc Balanced Acc Balanced Acc Balan Acc
UCI and OpenML Classification Datasets
d1 0.86 0.71 0.82 .... 0.61 0.82
d2 0.56 0.73 0.52 .... 0.61 0.62
dn−1 0.85 0.71 0.89 .... 0.88 0.82
dn 0.96 0.91 0.92 .... 0.88 0.82
Exp for evaluating n algorithms = n (no datasets) * m(no algorithms)
Appropriate Algo
J48
REPTree
CART
RandForest
J48
Meta-features + Appropriate Algorithm
경희대학교
KYUNG HEE UNIVERSITY
UTAS

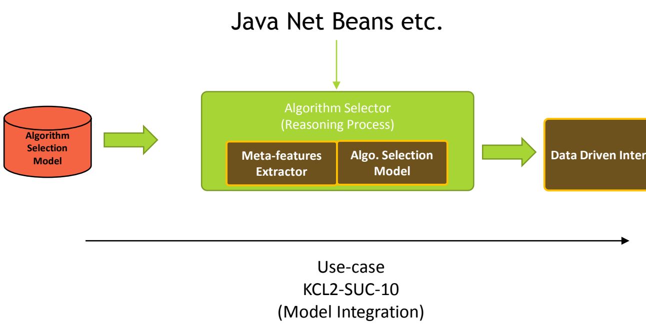

Algorithm Selection Model Creation
Use Case Feature Functions
Pre-processing (discretization)
Meta-Features Selection
Algorithm Selection Model Creation
Persistence of the created model
Model Creation (DT, of CBR Model)
1
2
3
4
Knowledge Creation & Evolution
Data-Driven
Dynamic Algorithm Selection Model Creator
Algorithm Selection Training Dataset
Algorithm Selection Model
Algorithm Selection Model Creation
Machine Learning Algorithms
경희대학교
KYUNG HEE UNIVERSITY
UTAS


Algorithm Selection Model Creation
Working Flow
Weka
(Library)
Weka/Rapidminer
(Library)
Weka/myCBR
(Library)
Training Dataset
(Meta-features + Appropriate Algorithm)
Preprocessing
(Discretization)
Best Features Selection
(Filter or Wrapper method)
Model Creation
(DT, CBR)
Algorithm Selection Model
Use-case
KCL2-SUC-09
(Model Creation)


Algorithm Selection Model Integration
Use Case Feature Functions
Meta Features Extractor Code Integration to Data Driven
Algorithm Selection Model Integration
(a DT or CBR System)
1
2
Model Integration
Knowledge Creation & Evolution
Data-Driven
Algorithm Selector
New Problem Meta-features Computation
Algorithm Selection
Dynamic Algorithm Selection Model Creator
Algorithm Selection Model
Knowledge Base
경희대학교
KYUNG HEE UNIVERSITY
UTAS

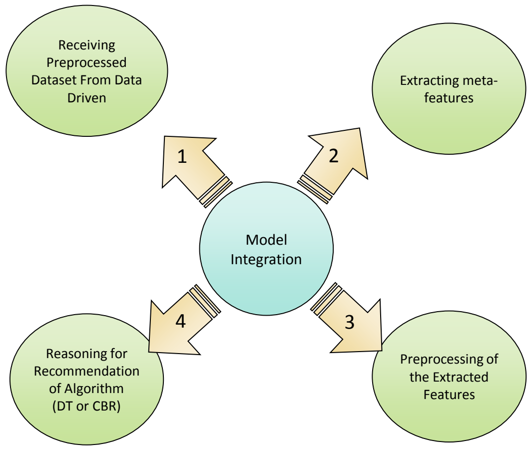
Algorithm Selection Model Integration

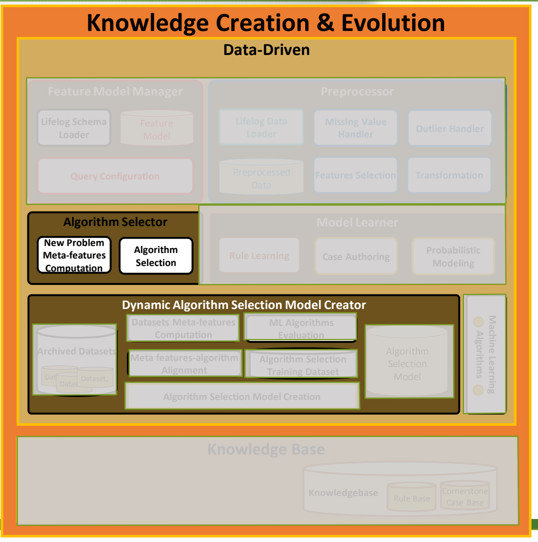
Tasks Flow



Algorithm Recommendation

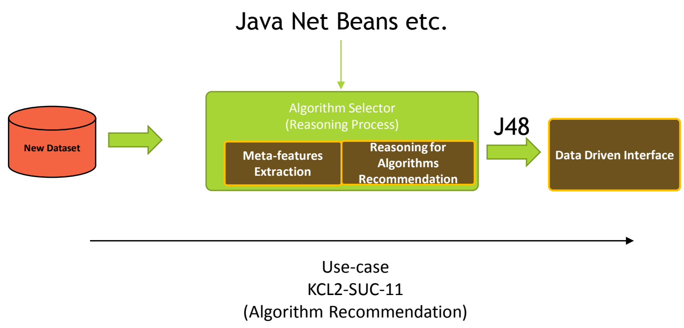
Use Case Feature Functions





경희대학교 KYUNG HEE UNIVERSITY UTAS

Algorithm Recommendation
Task Flow



Selected Tools and Techniques

- APIs:
  - Weka
  - OpenML
- Tools:
  - Weka
  - RapidMiner

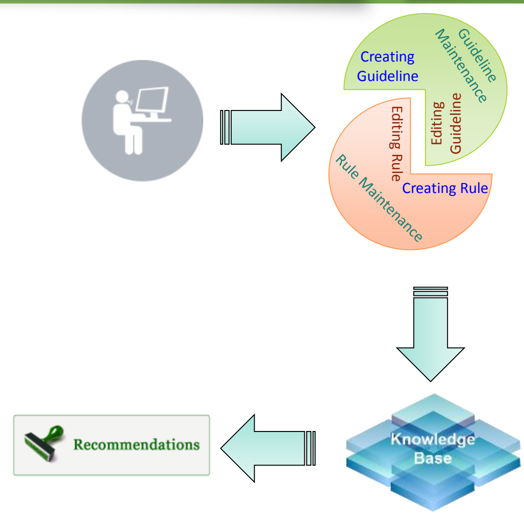
Contributions

- Support of *feature modeling* to select valid combination of features for optimal classification model
- Generating *classification model*
- Use of *multi meta-features learning* to exploit intrinsic behaviors of datasets for improved performance of the algorithm selection model
- Use of *multi-metric objective* function to evaluate algorithms performance and recommend appropriate algorithm
- *Automatic recommendation* of appropriate algorithm for a new dataset at run time

Expert Driven Knowledge Acquisition

Goal and Objectives

- Goal
  - Provide user-friendly environment to create recommendation's and alert's guidelines to transform experts knowledge into knowledge base
- Objectives
  - ❖Create easy-to-use Rule Editor to facilitate the domain experts to create knowledge rules using contextual selection of concepts from Intelli-sense window and/or domain model tree
  - ❖Providing user-friendly Guideline Editor to generate guidelines with the help of Intelli-sense and domain model tree selection as in Rule Editor
  - ❖Transformation of guidelines into knowledge rules and then to executable format to generate recommendations and alerts



경희대학교
KYUNG HEE UNIVERSITY

Motivations

1. Incorporating Guidelines for wellness domain

   Providing guideline editor to transform the guideline's knowledge into computer interpretable format.

2. Incorporating expert experiences into knowledge base

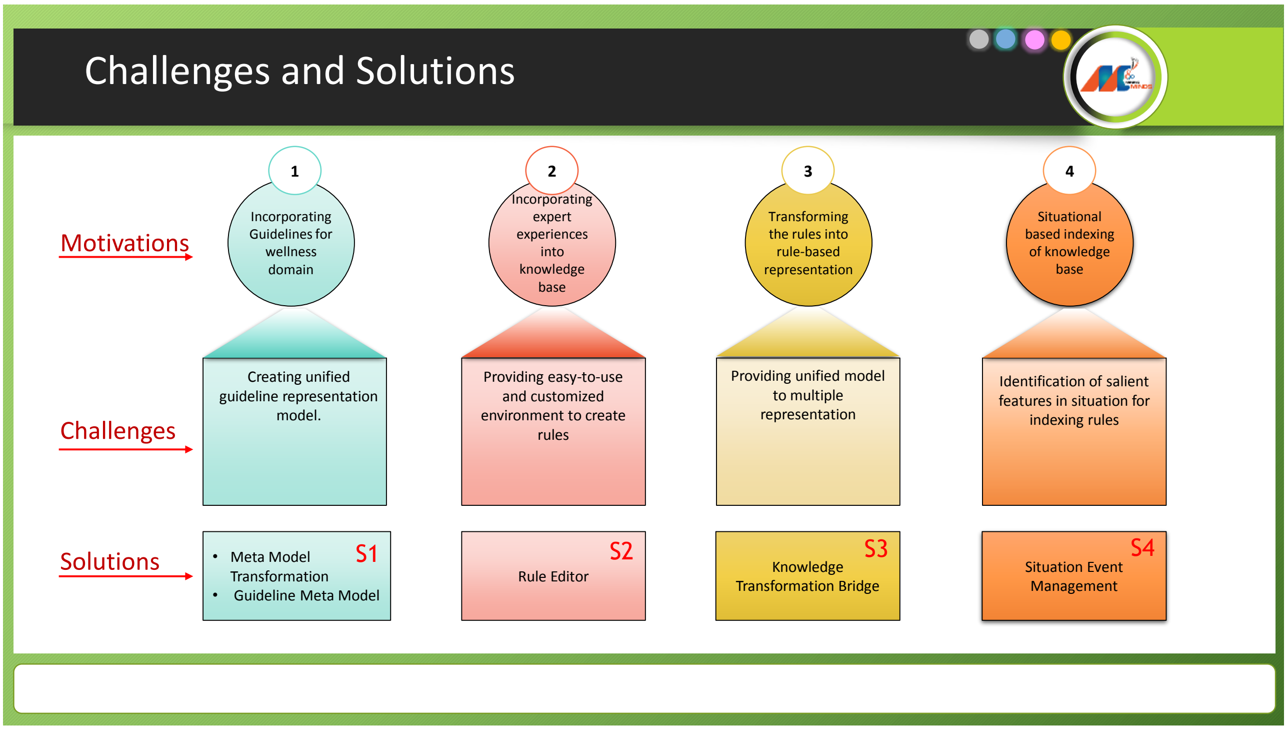
   Providing user-friendly environment to expert for transformation of their experiences into knowledge base

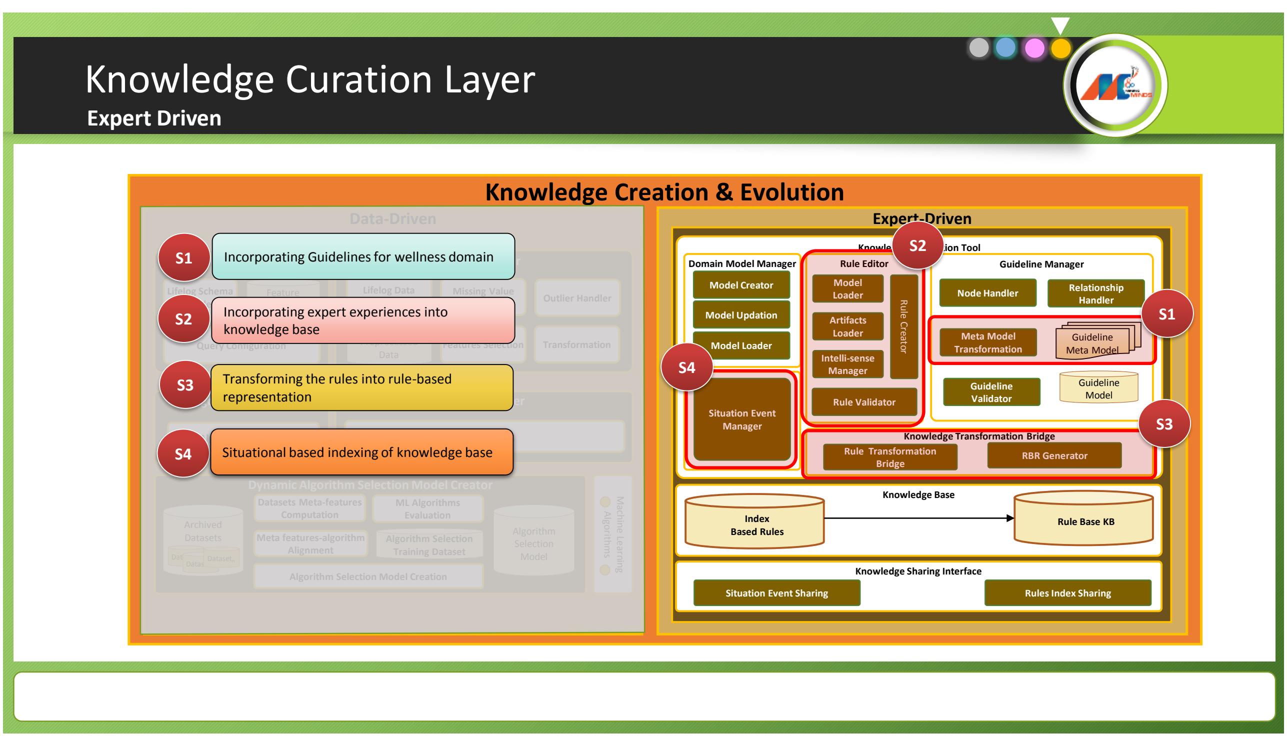
3. Transforming the rules into rule-based representation

   Providing transformation mechanism to represent the knowledge rules into multiple format.

4. Situational based indexing of knowledge base

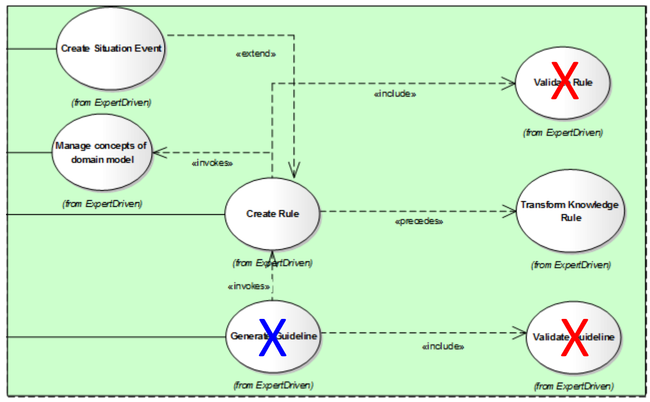
   Providing situation based indexing for knowledge base in order to classify diverse rules and enhance performance of reasoning


Challenges and Solutions
Motivations
Challenges
Solutions
1
Incorporating Guidelines for wellness domain
2
Incorporating expert experiences into knowledge base
3
Transforming the rules into rule-based representation
4
Situational based indexing of knowledge base
Creating unified guideline representation model.
Providing easy-to-use and customized environment to create rules
Providing unified model to multiple representation
Identification of salient features in situation for indexing rules
S1
• Meta Model Transformation
• Guideline Meta Model
S2
Rule Editor
S3
Knowledge Transformation Bridge
S4
Situation Event Management

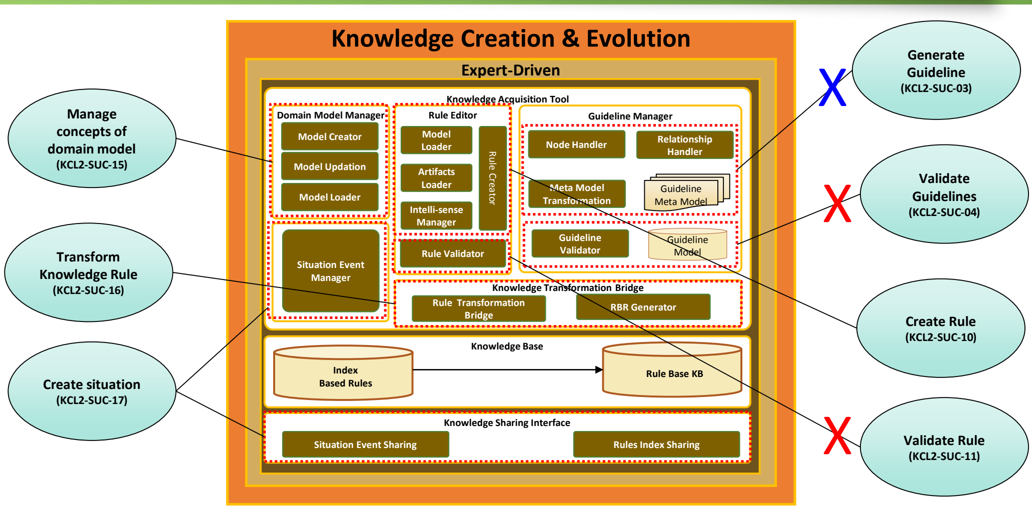

Knowledge Curation Layer
Expert Driven
Knowledge Creation & Evolution
Data-Driven
S1
Incorporating Guidelines for wellness domain
S2
Incorporating expert experiences into knowledge base
S3
Transforming the rules into rule-based representation
S4
Situational based indexing of knowledge base
Expert-Driven
Domain Model Manager
Model Creator
Model Updation
Model Loader
Rule Editor
Model Loader
Artifacts Loader
Intelli-sense Manager
Rule Creator
Rule Validator
Guideline Manager
Node Handler
Relationship Handler
Meta Model Transformation
Guideline Meta Model
Guideline Validator
Guideline Model
Situation Event Manager
Knowledge Transformation Bridge
Rule Transformation Bridge
RBR Generator
Knowledge Base
Index Based Rules
Rule Base KB
Knowledge Sharing Interface
Situation Event Sharing
Rules Index Sharing

Expert Driven Knowledge Acquisition-Use cases

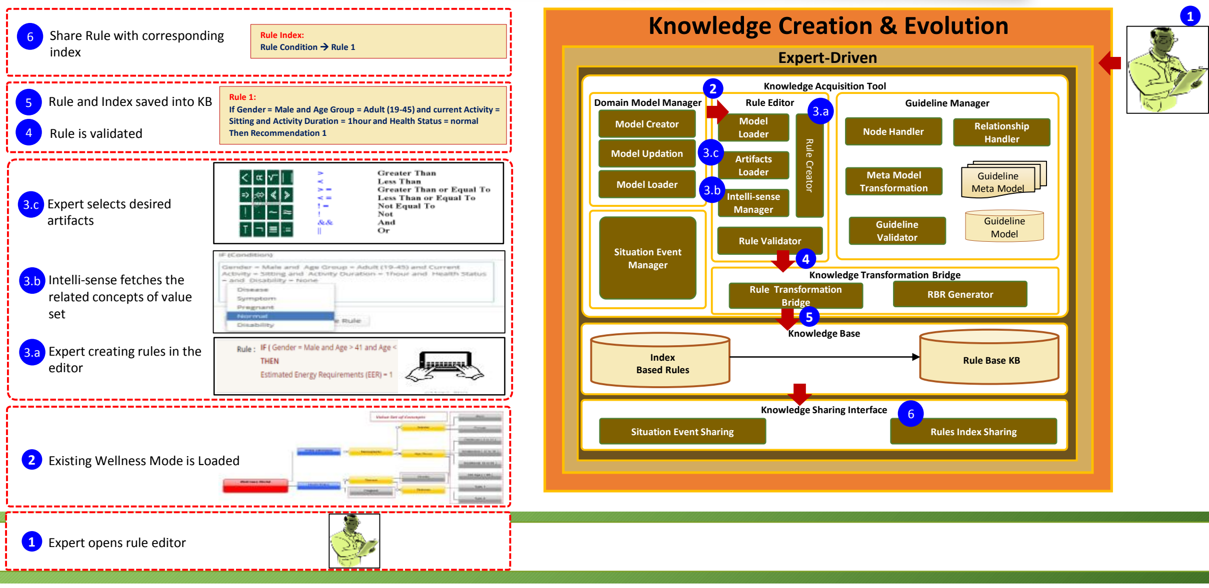
Use Case #ID	Description
KCL2-SUC-03	Generate Guideline
KCL2-SUC-04	Validate Guideline
KCL2-SUC-10	Create Rule
KCL2-SUC-11	Validate Rule
KCL2-SUC-15	Manage Concepts of Domain Model
KCL2-SUC-16	Transform Knowledge Rule
KCL2-SUC-17	Create Situation Event



Expert Driven Knowledge Acquisition-Use cases



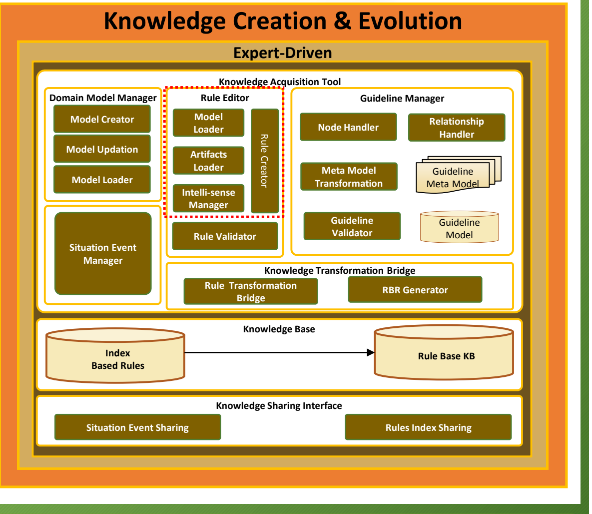
Expert Driven Knowledge Scenario for #2 path



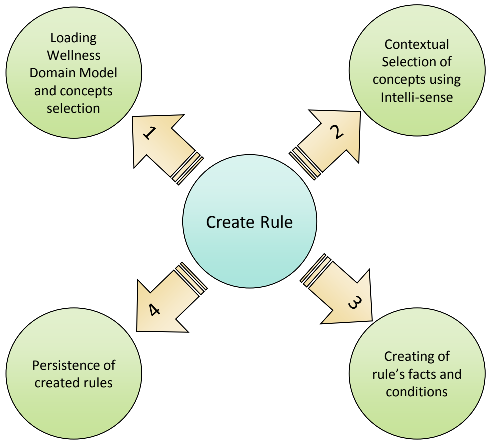
Methodology: Rule Editor

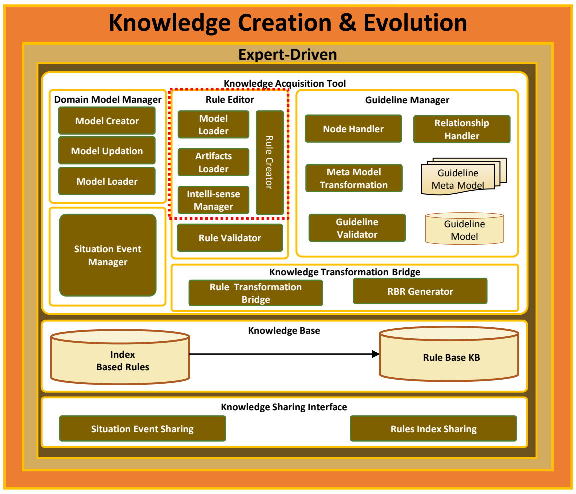
Use-case (KCL2-SUC-10): *Create Rule*

- **Objective**
  - ➢ Transform the expert's knowledge and guidelines into knowledge base rules
- **Methodology**
  - ➢ *Approach*: Model View Controller (MVC)
    - ➢ *Client side script: AngularJS*
    - ➢ *Server side script: Spring MVC*
  - ➢ ***Rule Editor*** *(operations)*
    1. **Input**: Concepts from wellness model/Intelli-sense, Expert's/Guideline knowledge
    2. **Processing**:
       a. Wellness domain model loads to the editor
       b. Domain experts create facts and conclusion of rules using Intelli-sense window and wellness model
       c. Domain experts select the required artifacts
       d. The created rule transform into plain rule format and store into knowledge base
    3. **Output**:
       a. Rule in plain rule format



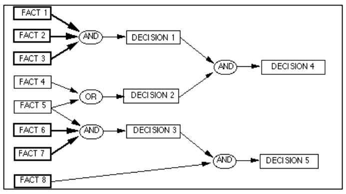
Use Case Functions: Rule Editor





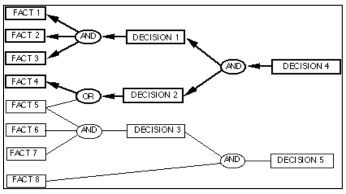
Rule Reasoning Strategies

Forward Chaining



Features
Works from facts to a conclusion (Data Driven)
Appropriate when all the facts are provided with the problem statement
Aims for finding conclusion
Appropriate for monitoring, planning, and interpretation applications
Breadth-first search

Backward Chaining



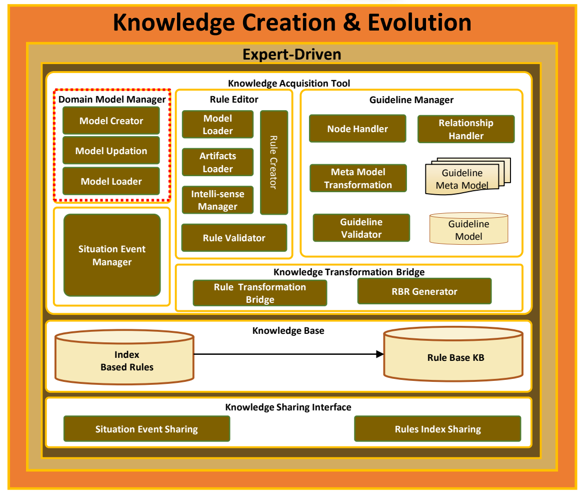
Features
Works from the conclusion to facts (Goal Driven)
Appropriate when the goal is given in the problem statement
Aims for finding necessary data
Appropriate for Diagnostic, prescription and debugging applications
Depth-first search

We will use Forward Chaining

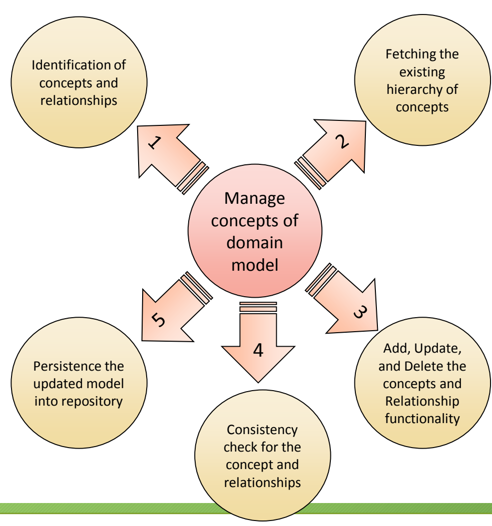
Methodology: Domain Model Manager

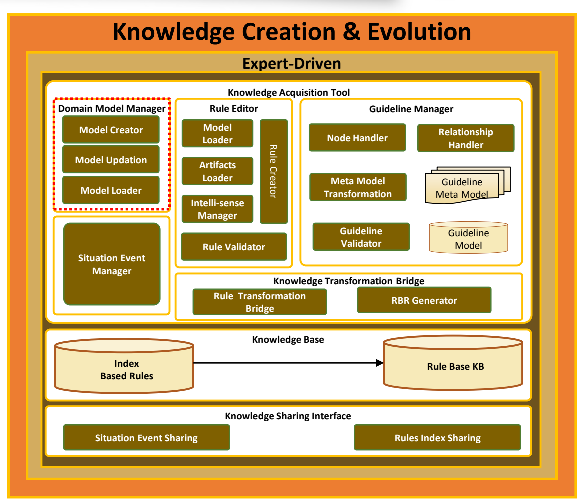
Use-case (KCL2-SUC-15): *Manage concepts of domain model*

- **Objective**
  - ➢ Provides user interface to manage wellness domain model with easy manner
- **Methodology**
  - ➢ *Approach*: Model View Controller (MVC)
    - ➢ *Client side script: AngularJS*
    - ➢ *Server side script: Spring MVC*
  - ➢ ***Domain Model Manager*** *(operations)*
    1. **Input**: Concepts and relationships
    2. **Processing**:
       a. Domain expert identify the concepts to add
       b. Identify the relationship of new concept with other existing concepts if exists
       c. Add concepts and create relationships
       d. Store the updated model into repository
    3. **Output**:
       a. Updated wellness model

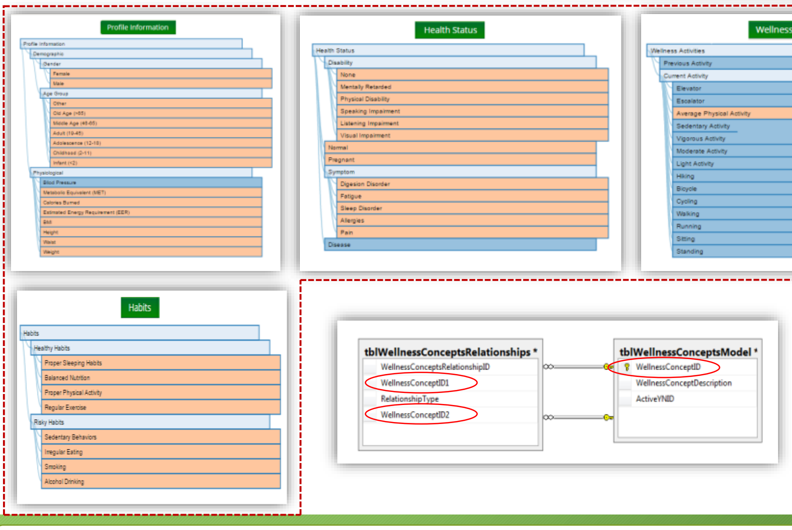


Use case functions: Domain Model Manager





Existing Wellness Model and Schema

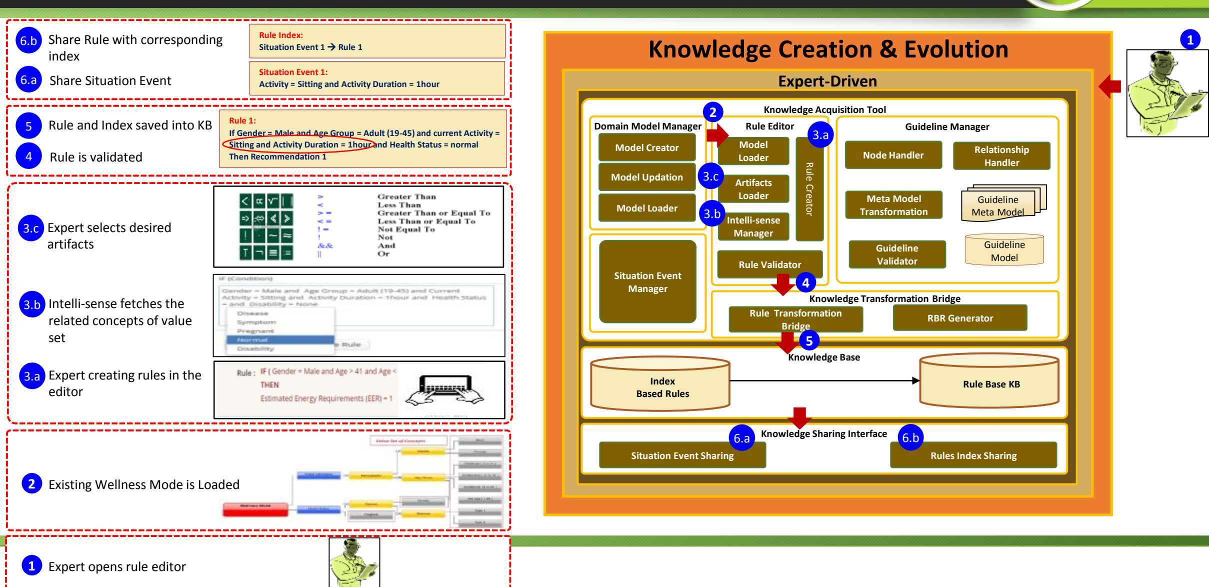


WellnessConceptID	WellnessConceptDescription	ActiveYNID
54	Risky Habits	Yest
55	Alcohol Drinking	Yes
56	Smoking	Yes

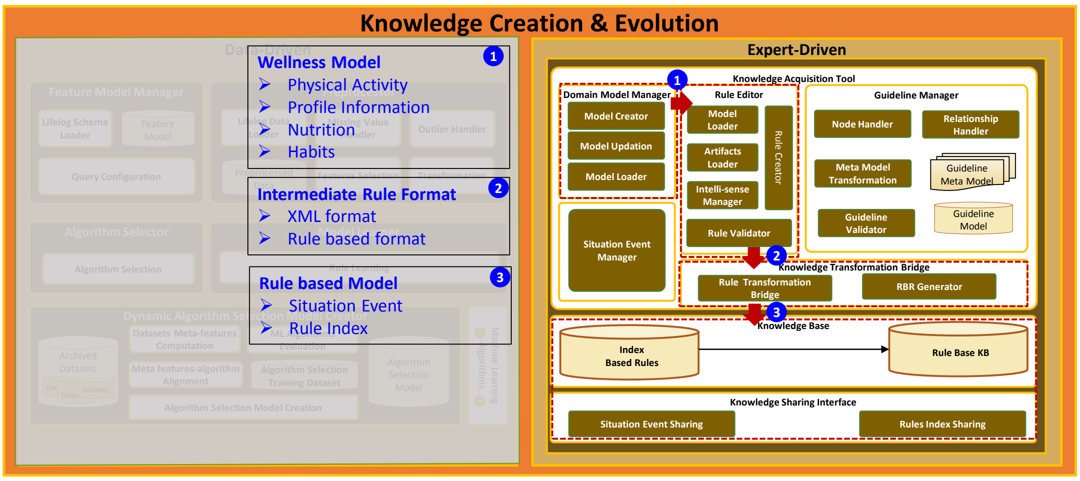
WellnessConcept RelationshipID	WellnessConceptID 1	Relationship Type	WellnessConceptID 2
60	55 (Alcohol Drinking)	IsA	54 (Risky Habits)
61	56 (Smoking)	IsA	54 (Risky Habits)

Expert Driven Knowledge Scenario for #2 path

Situation Event Base



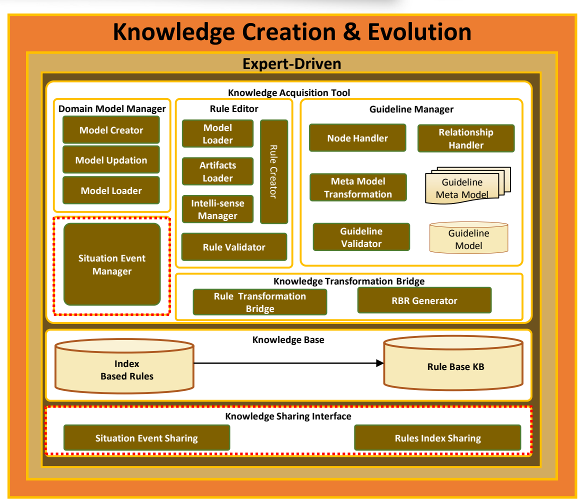
Expert Driven Rule Creation Workflow for #2 path



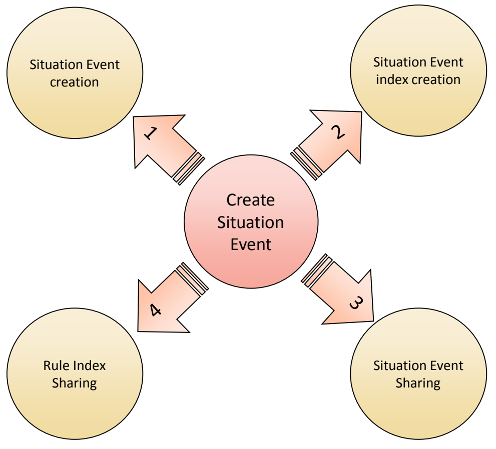
Methodology: Situation Event Manager

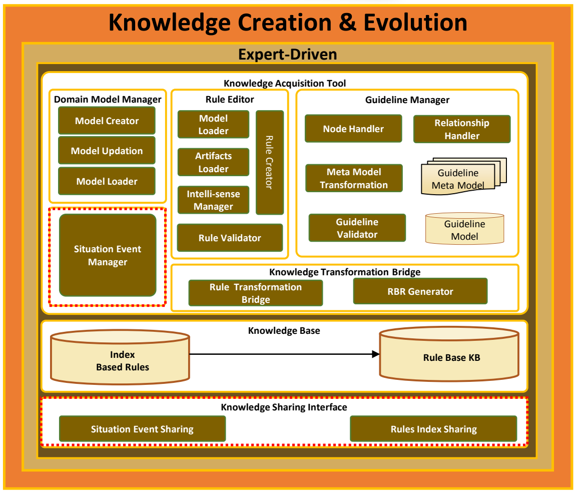
Use-case (KCL2-SUC-15): *Create Situation Event*

- **Objective**
  - To include set of associated recommendation rules and associate rules into index
- **Methodology**
  - *Approach*: Model View Controller (MVC)
    - *Server side script: Spring MVC*
  - ***Situation Event Manager** (operations)*
    1. **Input**: Plain rule in XML format
    2. **Processing**:
       a. Domain expert selects salient features (indicating as event) from conditions of the rule
       b. System creates situation event with selected salient features
       c. Assign index to the situation and associate to the rule
       d. Store the index with associated rule in knowledge base index
    3. **Output**:
       a. Indexed rules
       b. Situation Event: JSON

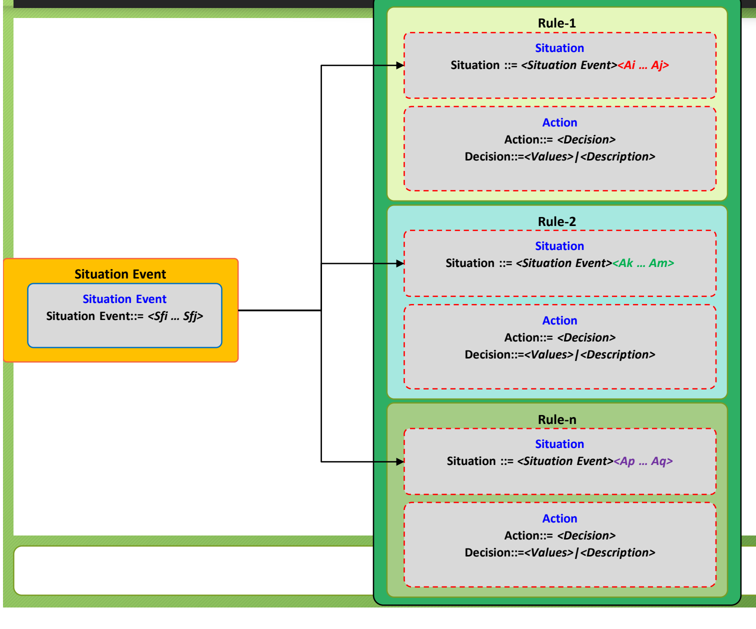


Use case functions: Situation Event Manager and Knowledge Sharing Interface

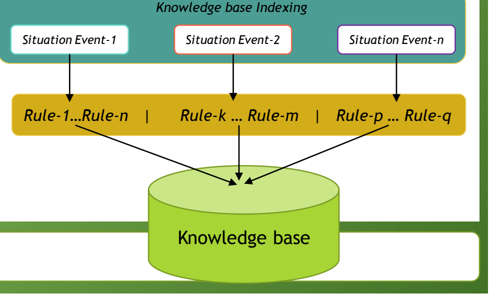


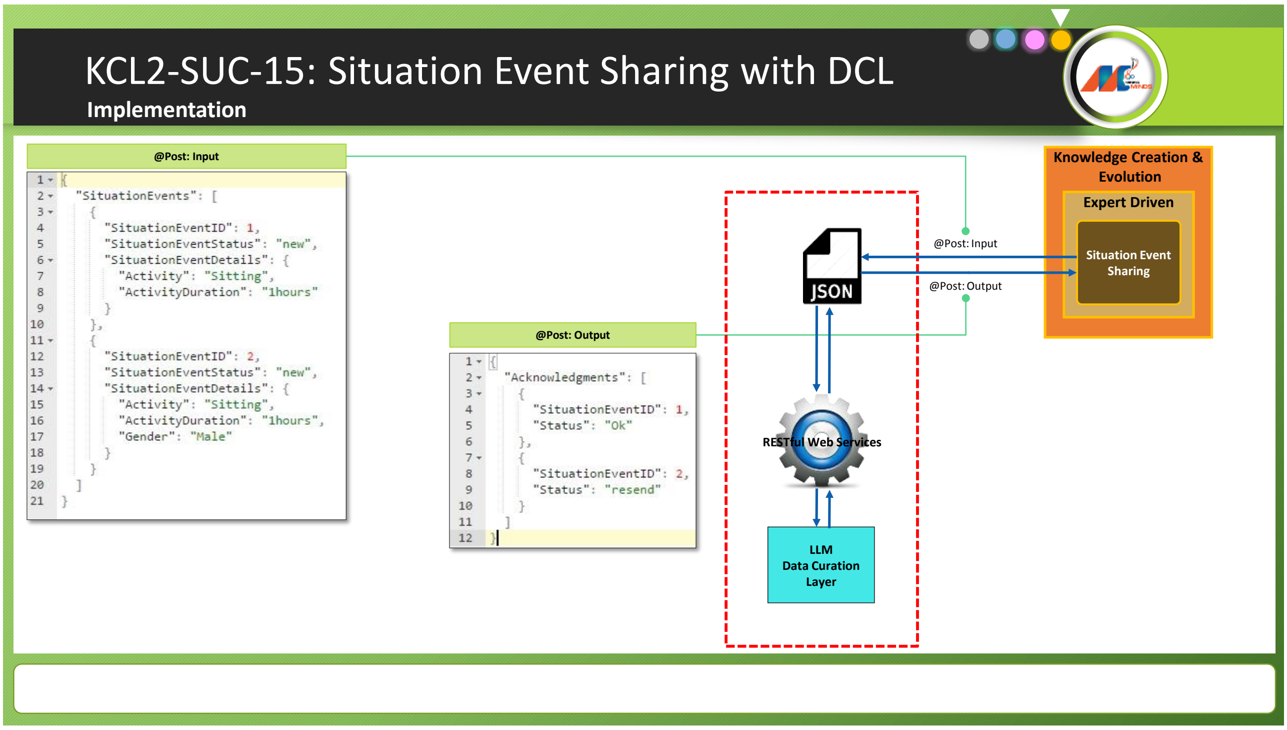


Situation, Situation Event and Rule

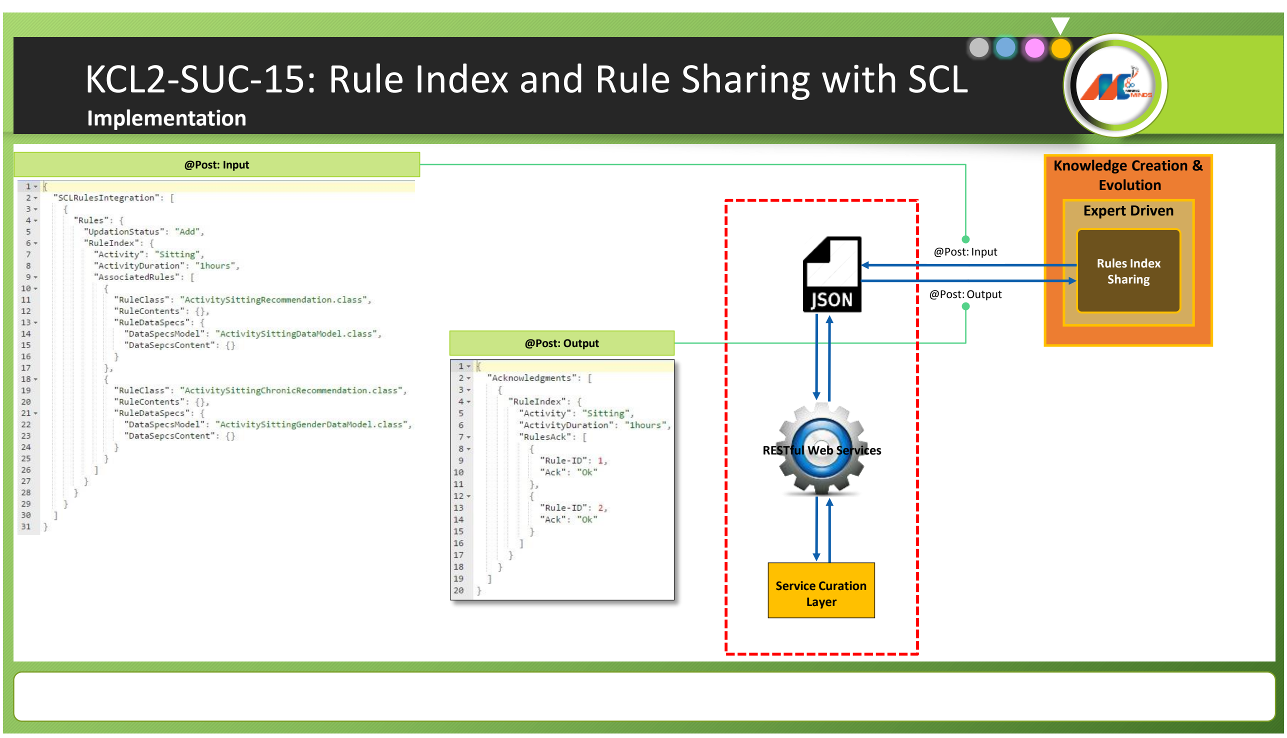


- *Situation* constitute part of rule which includes data elements(condition) required to process rule.
- *Situation Event* is part of *Situation* which includes *salient features* of data elements(condition).
- *Rule* is part of knowledge base which includes Situation and Action (decision)





KCL2-SUC-15: Situation Event Sharing with DCL
Implementation
@Post: Input
{
"SituationEvents": [
{
"SituationEventID": 1,
"SituationEventStatus": "new",
"SituationEventDetails": {
"Activity": "Sitting",
"ActivityDuration": "1hours"
}
},
{
"SituationEventID": 2,
"SituationEventStatus": "new",
"SituationEventDetails": {
"Activity": "Sitting",
"ActivityDuration": "1hours",
"Gender": "Male"
}
}
]
}
@Post: Output
{
"Acknowledgments": [
{
"SituationEventID": 1,
"Status": "Ok"
},
{
"SituationEventID": 2,
"Status": "resend"
}
]
}
JSON
RESTful Web Services
LLM
Data Curation
Layer
@Post: Input
@Post: Output
Knowledge Creation & Evolution
Expert Driven
Situation Event Sharing

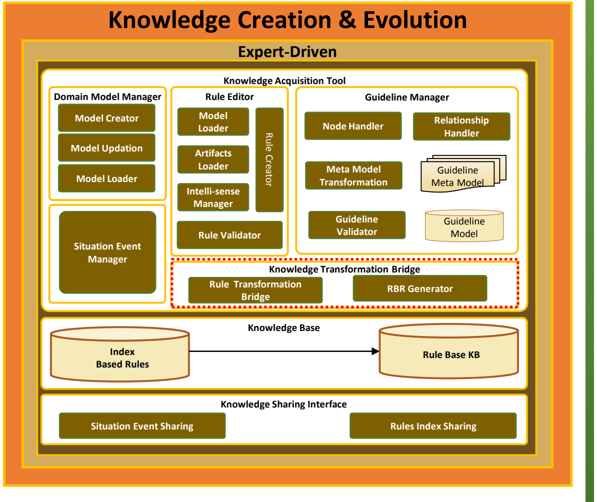


KCL2-SUC-15: Rule Index and Rule Sharing with SCL
Implementation
@Post: Input
{
"SCLRulesIntegration": [
{
"Rules": {
"UpdationStatus": "Add",
"RuleIndex": {
"Activity": "Sitting",
"ActivityDuration": "1hours",
"AssociatedRules": [
{
"RuleClass": "ActivitySittingRecommendation.class",
"RuleContents": {},
"RuleDataSpecs": {
"DataSpecsModel": "ActivitySittingDataModel.class",
"DataSepcsContent": {}
}
},
{
"RuleClass": "ActivitySittingChronicRecommendation.class",
"RuleContents": {},
"RuleDataSpecs": {
"DataSpecsModel": "ActivitySittingGenderDataModel.class",
"DataSepcsContent": {}
}
}
]
}
}
}
]
}
@Post: Output
{
"Acknowledgments": [
{
"RuleIndex": {
"Activity": "Sitting",
"ActivityDuration": "1hours",
"RulesAck": [
{
"Rule-ID": 1,
"Ack": "Ok"
},
{
"Rule-ID": 2,
"Ack": "Ok"
}
]
}
}
]
}
JSON
RESTful Web Services
Service Curation
Layer
@Post: Input
@Post: Output
Knowledge Creation & Evolution
Expert Driven
Rules Index Sharing

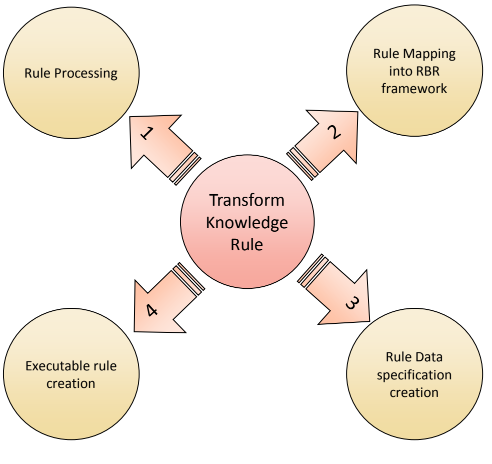
Methodology: Knowledge Transformation Bridge

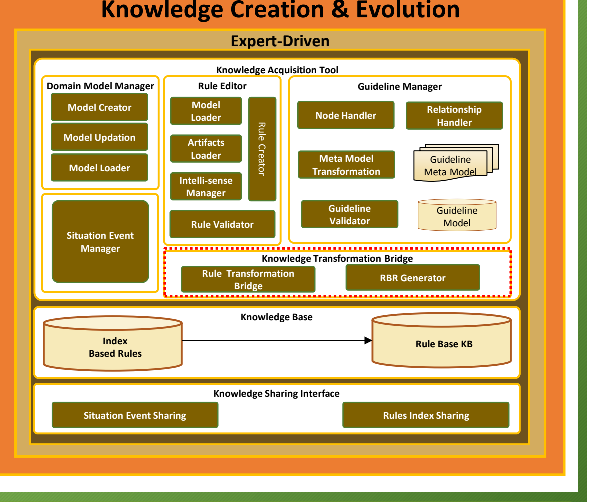
Use-case (KCL2-SUC-15): *Transform knowledge Rule*

- **Objective**
  - ➢ Transform the created rules and guidelines into executable format to generate recommendations
- **Methodology**
  - ➢ *Approach*: Template based code generations
    - ➢ *Server side script: Core Java*
  - ➢ ***Knowledge Transformation Bridge*** *(operations)*
    1. **Input**: Plain rule in XML format
    2. **Processing**:
       a. The system identifies appropriate representation model
       b. Fetch the artifacts and controls of the selected representation model
       c. Transform the rule into selected representation model using its artifacts, controls and syntax
       d. Store the created/updated rule into repository
    3. **Output**:
       a. Executable rules into knowledge base



Use case functions: Knowledge Transformation Bridge





Use case functions: Knowledge Transformation Bridge
[Rule Processing: BNF]

- Backus–Naur Form (BNF) of Rule

<Rule> ::= if <condition> then <conclusion>

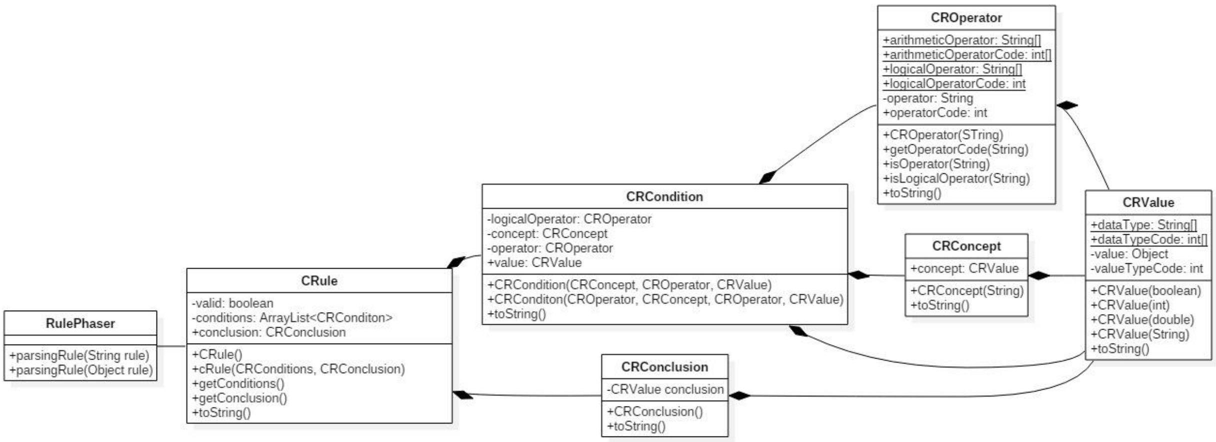
<condition> ::= <DomainModelConcept><comparisonOperator><DMCValue>
[<logicalOperator><condition>]

<comparisonOperator> ::= > | < | >= | <= | = | !=

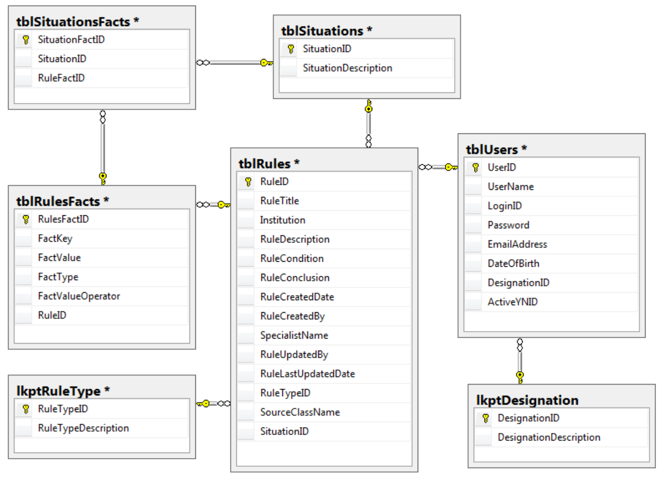
<logicalOperator> ::= AND | OR

<conclusion> ::= <DomainModelConcept> | <DMCValue> |
<DomainModelConcept>[<text>]<DMCValue> | <text>

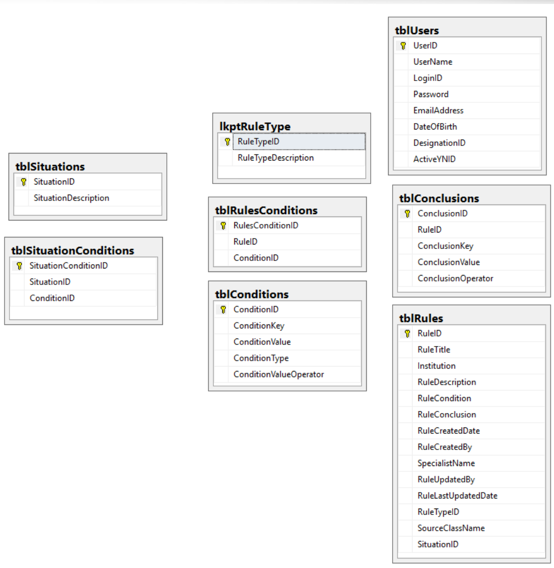
Use case functions: Knowledge Transformation Bridge
[Rule Processing: Class Model]



Use case functions: Knowledge Transformation Bridge
[Rule Mapping into RBR framework]



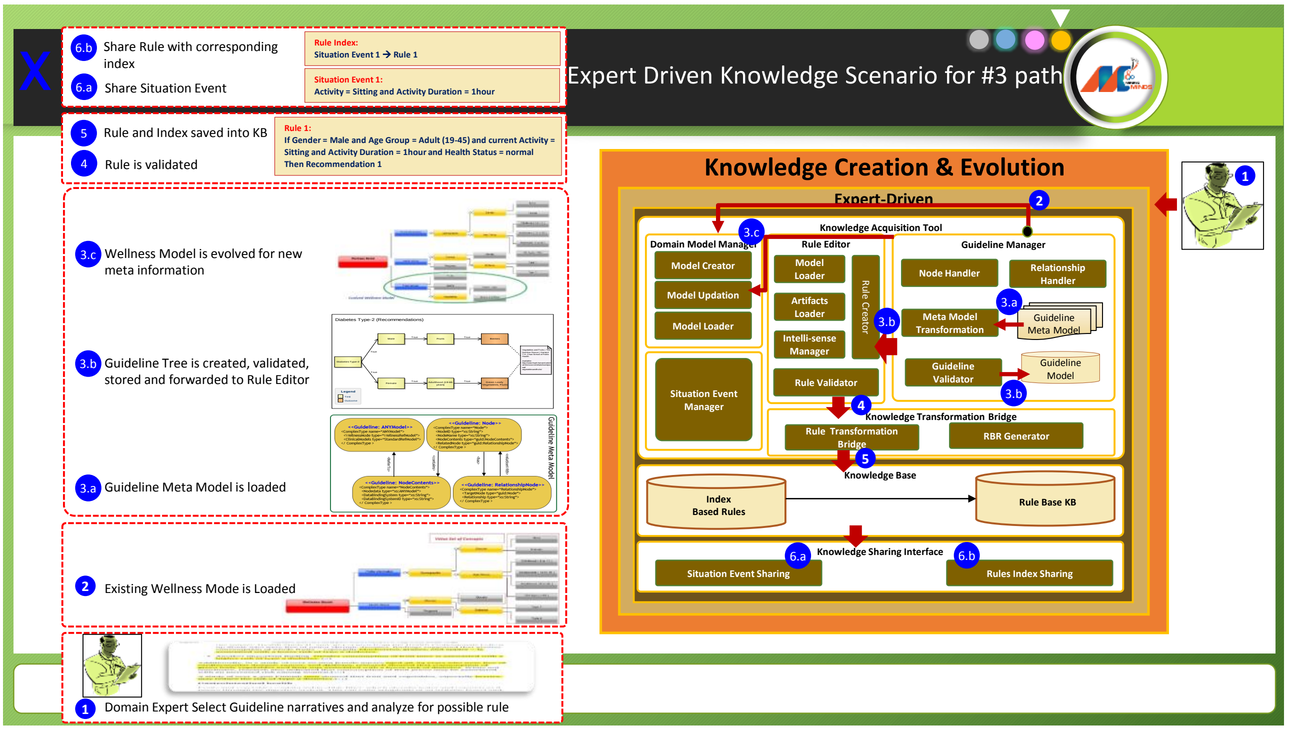
Use case functions: Knowledge Transformation Bridge
[Rule Mapping into RBR framework]



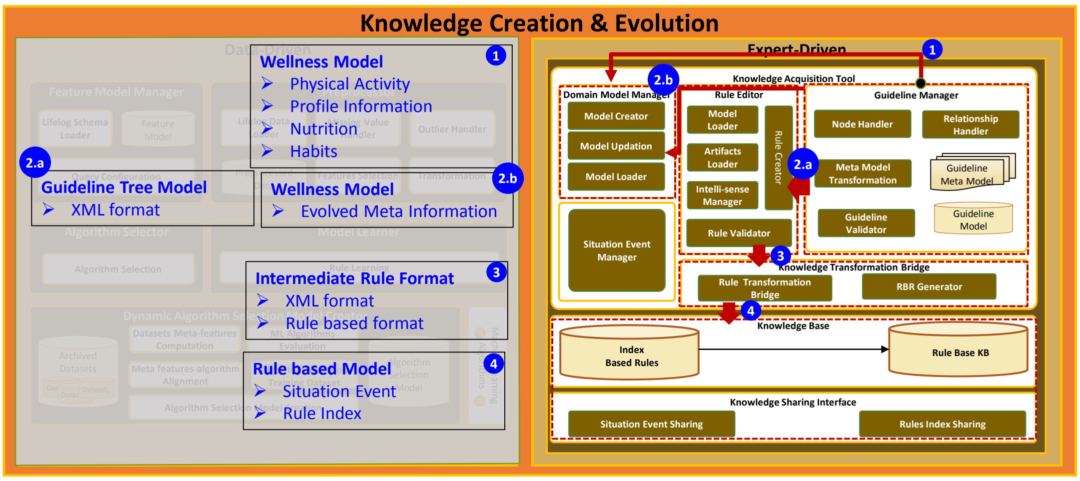
Situation Based Reasoning

	RuleID	ConditionKey	ConditionValueOperator	ConditionValue	ConditionType	ConclusionKey	ConclusionValue
1	2	Current Activity	=	Sitting	String	Activity	Standing
2	2	Current Activity	=	Sitting	String	Activity	Stretching
3	2	Current Activity	=	Sitting	String	Activity Duration	20-30s
4	2	Current Activity	=	Sitting	String	Description	Take a break! Get out of your chair or couch every 15 minutes and merely stand up and then sit back down Or take stretching for 20-30 seconds.
5	2	Activity Duration	=	15m	String	Activity	Standing
6	2	Activity Duration	=	15m	String	Activity	Stretching
7	2	Activity Duration	=	15m	String	Activity Duration	20-30s
8	2	Activity Duration	=	15m	String	Description	Take a break! Get out of your chair or couch every 15 minutes and merely stand up and then sit back down Or take stretching for 20-30 seconds.

```sql
ALTER PROCEDURE [dbo].[usp_SituationBasedReasoner]
        @situationEventData varchar(max) = null,
        @siuationEventCount int = 0

AS
BEGIN
        SET NOCOUNT ON;
        DECLARE @situationQuery nvarchar(max)
        IF @situationEventData IS NOT null
        BEGIN
                SET @situationQuery = 'DECLARE cur_sitEvent CURSOR FOR
                SELECT s.SituationID , COUNT (s.SituationID)
                FROM tblSituations s, tblSituationConditions sc, tblConditions c
                WHERE s.SituationID = sc.SituationID AND sc.ConditionID = c.ConditionID AND
                ( ' + @situationEventData + ' ) GROUP BY s.SituationID'

                Execute sp_executesql @situationQuery
                OPEN cur_sitEvent
                DECLARE @sitID numeric(18,0), @sitConCount int, @winnersitID numeric(18,0)
                SET @winnersitID = 0
                FETCH NEXT FROM cur_sitEvent
                INTO @sitID, @sitConCount

                WHILE @@FETCH_STATUS = 0
                BEGIN
                        IF @sitConCount = @siuationEventCount
                        BEGIN
                                DECLARE @countSE int
                                SELECT @countSE = COUNT(*)
                                FROM tblSituations s, tblSituationConditions sc
                                WHERE s.SituationID = sc.SituationID and s.SituationID =
                                        @sitID
                                IF @siuationEventCount = @countSE
                                BEGIN
                                        SET @winnersitID = @sitID
                                        BREAK
                                END
                        END
                        FETCH NEXT FROM cur_sitEvent
                        INTO @sitID, @sitConCount
                END
                SELECT r.RuleID, c.ConditionKey, c.ConditionValueOperator,
                c.ConditionValue, c.ConditionType , rcon.ConclusionKey,
                rcon.ConclusionValue
                FROM tblRules r, tblRulesConditions rc, tblConditions c , tblConclusions
                        rcon
                WHERE r.RuleID = rc.RuleID And rc.ConditionID = c.ConditionID And r.RuleID
                        = rcon.RuleID
                AND r.SituationID = @winnersitID
                Close cur_sitEvent
                Deallocate cur_sitEvent
        END
END
```



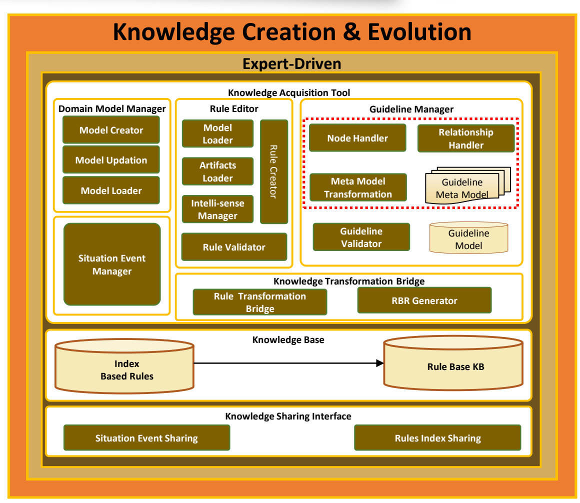
Expert Driven Rule Creation Workflow for #3 path X



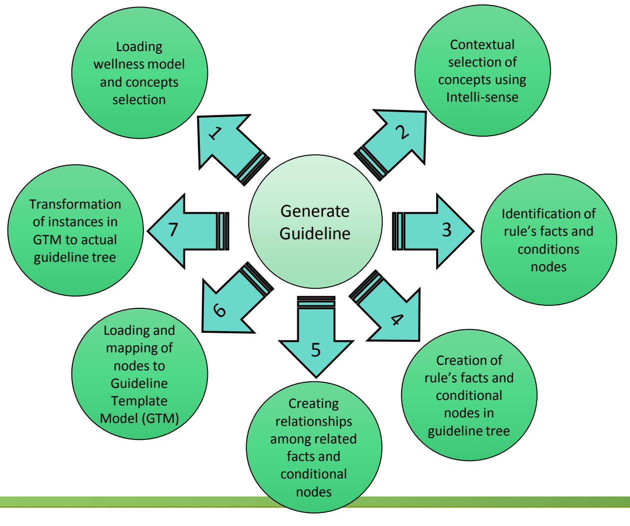
Methodology: Guideline Manager

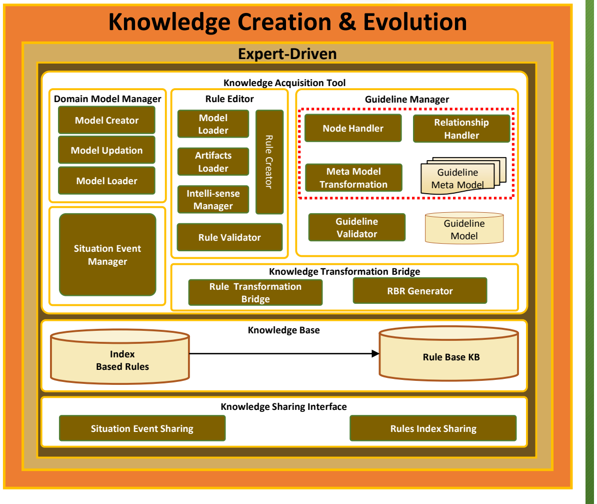
Use-case (KCL2-SUC-03): *Generate Guideline*

- **Objective**
  - Generate domain guidelines in tree format with easy manner
- **Methodology**
  - *Approach*: Model View Controller (MVC*)*
    - *Client side script: AngularJS*
    - *Server side script: Spring MVC*
  - ***Guideline Manager** (operations)*
    1. **Input**: Concepts from wellness model, Expert's/Guideline knowledge
    2. **Processing**:
       a. Domain expert generate tree nodes using Intelli-sense window and wellness model
       b. Make relationships among the nodes according to expert's knowledge
       c. Load guideline meta model and map the created nodes and relationships
       d. Transform the mapped nodes and relationships into tree in form of guideline meta model
    3. **Output**:
       a. Guideline Tree in graphical view
       b. *Guideline Tree in XML (Guideline Meta Model)*

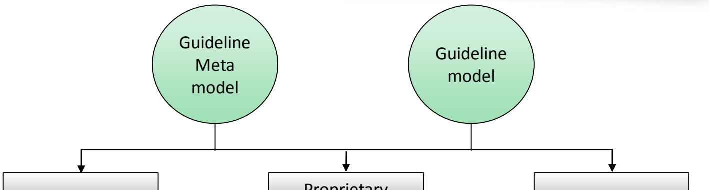


Use case functions: Guideline Manager





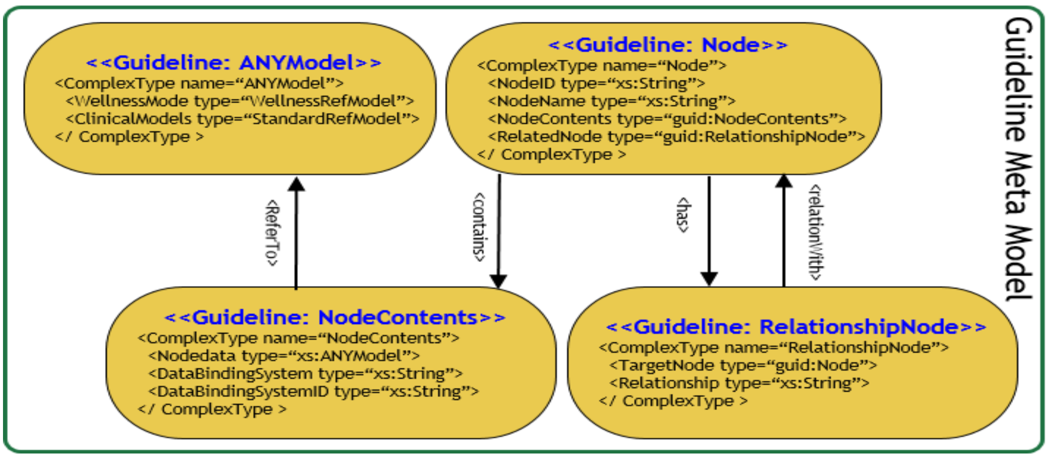
Guideline Representation Model



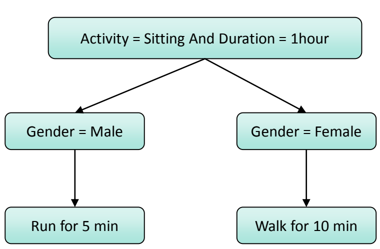
Features \ Representations	XML	Proprietary (RDB, File format)	JSON
Easy Syntactic Validation	Exists	Does not exist	Does not exist
Easily shareable	Exists	Does not exist	Exists
Support in visualization	Exists	Partially exists	Exists
Easy transformation	Exists	Does not exist	Exists

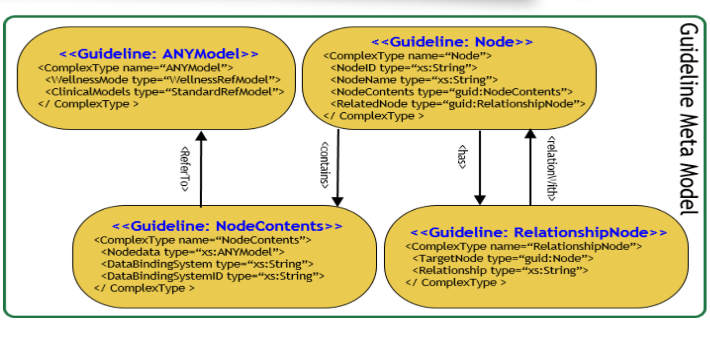
Exists Partially exists Does not exist

Guideline Template Model



Guideline Template Model



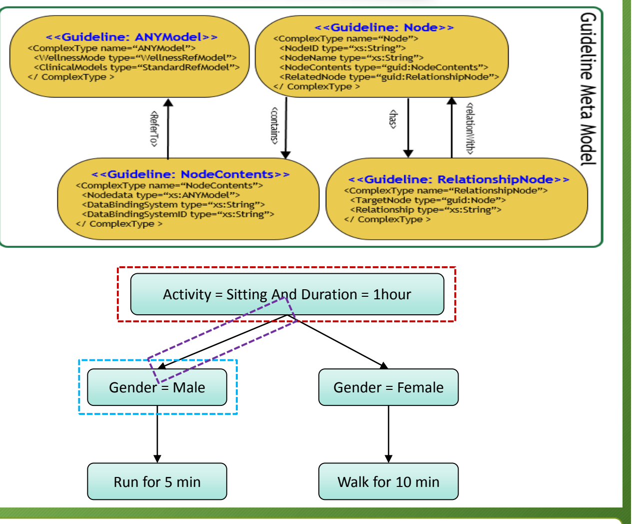


Conditional Nodes : Activity = Sitting And Duration = 1hour | Gender = Female | Gender = Male

Decisional Nodes : Run for 5 min | Walk for 10 min

Guideline Template Model

```
<Guideline>
  <Node>
    <NodeID = Activity1 />
    <NodeName = rootNode />
    <NodeType = conditional />
    <NodeContents>
      <WellnessModel:Activity>
        <ActivityName> Sitting </ActivityName>
        <ActivityDuration> 1hour </ActivityDuration>
      </WellnessModel:Activity>
    </NodeContents>
    <RelatedNode>
      <TargetNode>Gender1</TargetNode>
      <Relationship>Child</Relationship>
    </RelatedNode>
  </Node>
  <Node>
    <NodeID = Gender1 />
    <NodeName = genderMale />
    <NodeType = conditional />
    <NodeContents>
      <WellnessModel:Gender>Male</WellnessModel:Gender>
</NodeContents>
    <RelatedNode>
      <TargetNode>Activity1</TargetNode>
      <Relationship>Parent</Relationship>
    </RelatedNode>
  </Node>
……..
</Guideline>
```



Contributions

- Provide user-friendly Rule Editor to domain experts
- Provide easy-to-use Guideline Editor to transform the guidelines from text to computer executable guidelines and rules
- Provide uniform Guideline Template Model (GTM)
- Provide wellness model manager to create wellness model by experts
- Index based rules (Situation enabled) generation and sharing for different services

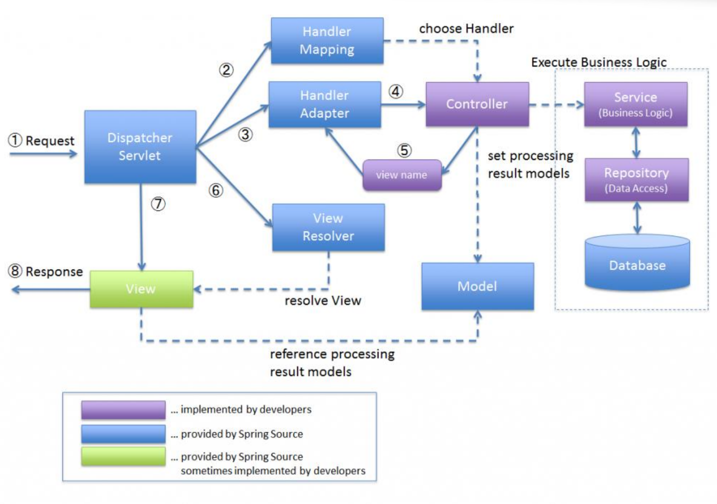
경희대학교 KYUNG HEE UNIVERSITY UTAS

Selected Tools and Techniques

- Language
  - Java
- Frameworks:
  - Spring MVC
  - AngularJS
  - Hibernate
- Database:
  - MS SQL Server
- IDEs:
  - Eclipse

Request-response flow including services with Spring MVC

Spring MVC Request Lifecycle Representing Backend Services

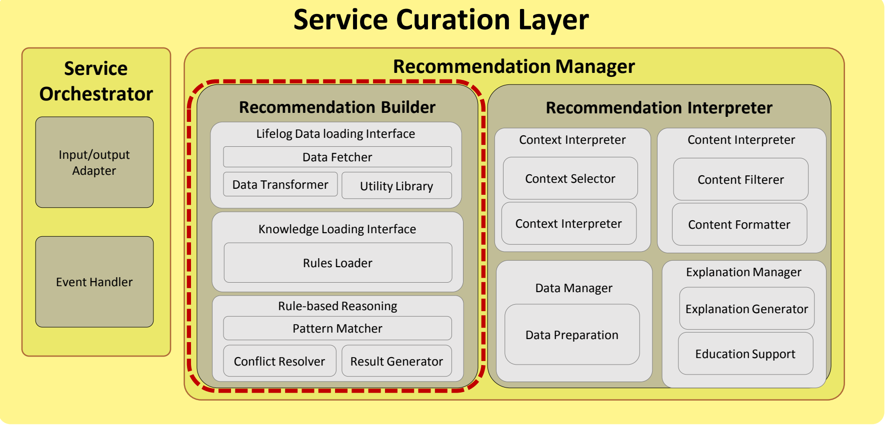


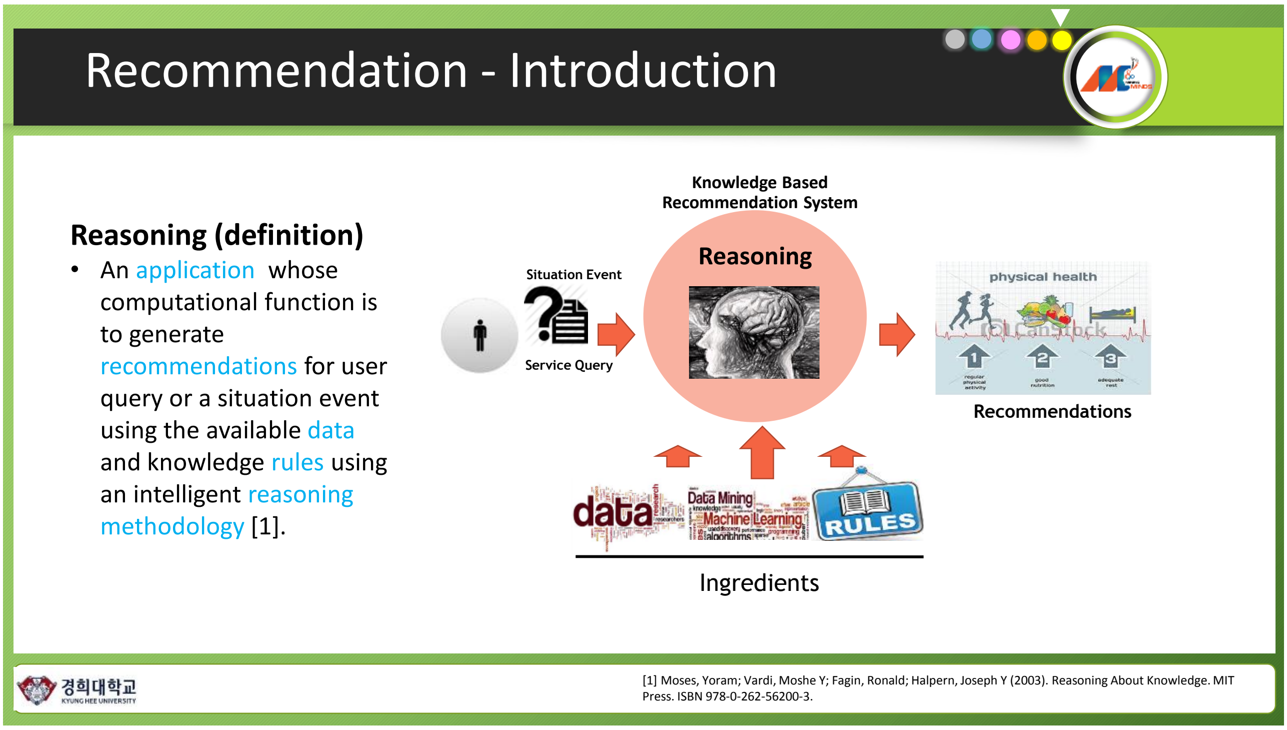
Mining Minds

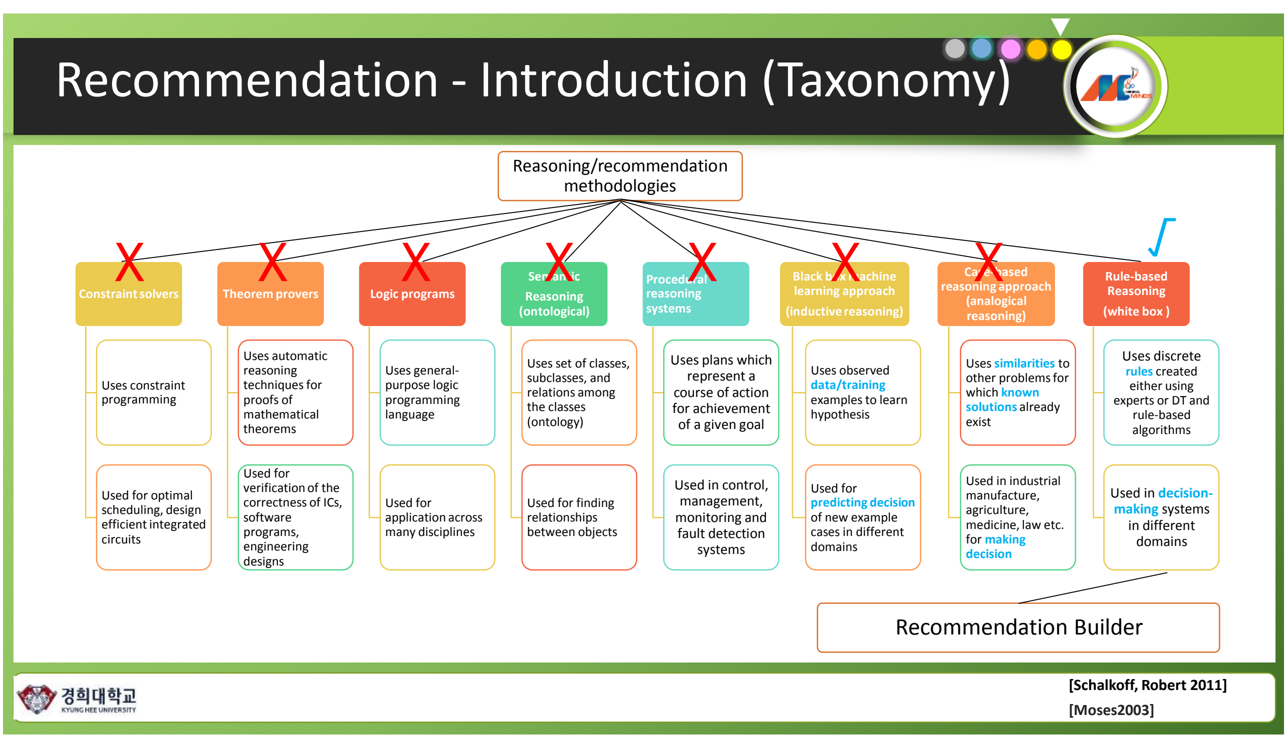
Recommendation Builder

경희대학교 KYUNG HEE UNIVERSITY · KITECH 한국생산기술연구원 · POSTECH 포항공과대학교 · U2 system User & Usability system · UTAS · 녹십자헬스케어 · TAPACROSS

Service Curation Layer

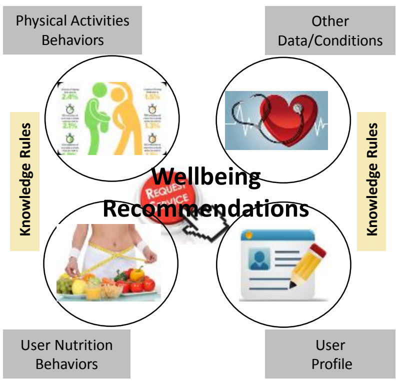



Recommendation - Introduction
Reasoning (definition)
An application whose computational function is to generate recommendations for user query or a situation event using the available data and knowledge rules using an intelligent reasoning methodology [1].
Knowledge Based Recommendation System
Situation Event
Service Query
Reasoning
physical health
Recommendations
data
Data Mining
Machine Learning
RULES
Ingredients
경희대학교
KYUNG HEE UNIVERSITY
[1] Moses, Yoram; Vardi, Moshe Y; Fagin, Ronald; Halpern, Joseph Y (2003). Reasoning About Knowledge. MIT Press. ISBN 978-0-262-56200-3.


Recommendation - Introduction (Taxonomy)
Reasoning/recommendation methodologies
Constraint solvers
Theorem provers
Logic programs
Semantic Reasoning (ontological)
Procedural reasoning systems
Black box machine learning approach (inductive reasoning)
Case-based reasoning approach (analogical reasoning)
Rule-based Reasoning (white box)
Uses constraint programming
Uses automatic reasoning techniques for proofs of mathematical theorems
Uses general-purpose logic programming language
Uses set of classes, subclasses, and relations among the classes (ontology)
Uses plans which represent a course of action for achievement of a given goal
Uses observed data/training examples to learn hypothesis
Uses similarities to other problems for which known solutions already exist
Uses discrete rules created either using experts or DT and rule-based algorithms
Used for optimal scheduling, design efficient integrated circuits
Used for verification of the correctness of ICs, software programs, engineering designs
Used for application across many disciplines
Used for finding relationships between objects
Used in control, management, monitoring and fault detection systems
Used for predicting decision of new example cases in different domains
Used in industrial manufacture, agriculture, medicine, law etc. for making decision
Used in decision-making systems in different domains
Recommendation Builder
경희대학교
KYUNG HEE UNIVERSITY
[Schalkoff, Robert 2011]
[Moses2003]

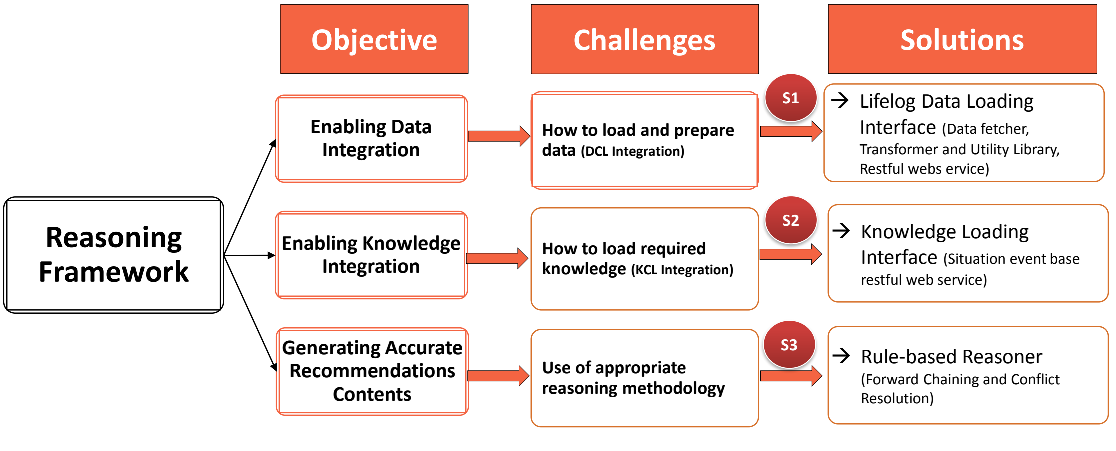
Recommendation Builder - Goal and Objectives

- Goal
  - Generating accurate contents for wellbeing recommendations using domain knowledge, and users' personal profile and lifelog data
- Objectives
  - ❖Enabling data integration for recommendation generation
  - ❖Enabling knowledge integration, for reasoning purpose
  - ❖Generating accurate recommendations contents, using individual's personal profile and lifelog data along with the knowledge



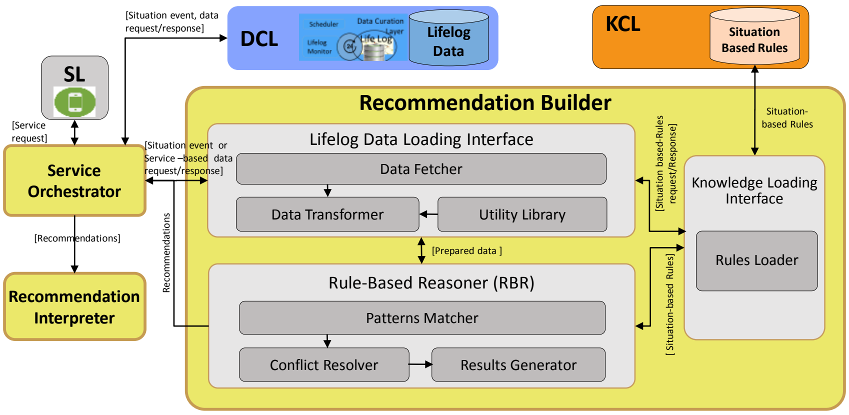
경희대학교 KYUNG HEE UNIVERSITY

Objectives – Challenges - Solutions



경희대학교 KYUNG HEE UNIVERSITY

Recommendation Builder - Architecture



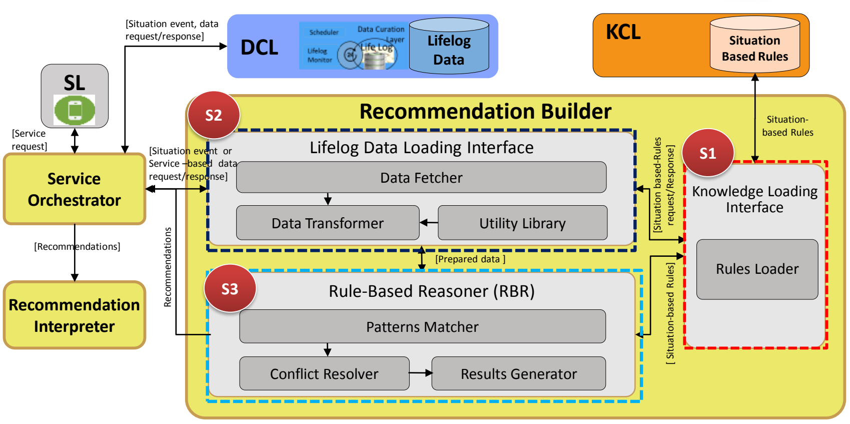
경희대학교 KYUNG HEE UNIVERSITY

Recommendation Builder – Solutions for Challanges

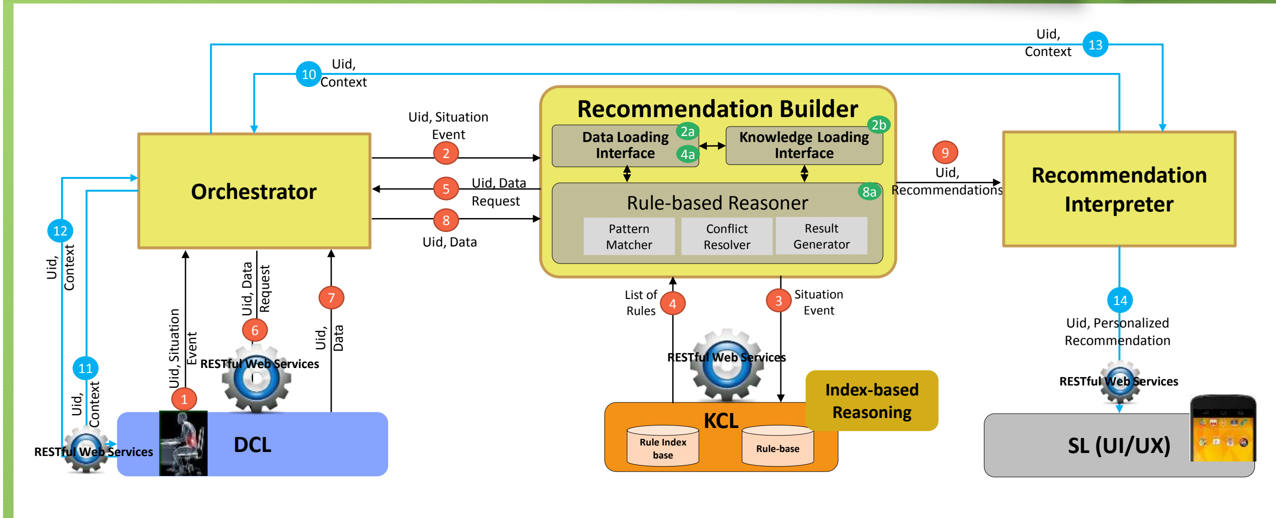
S1: Situation event/Service request-based Knowledge Integration

S2: Utility Library-based Data Transformation

S3: Rule-based Reasoning with Forward Chaining and Conflict Resolution Algorithms



Recommendation Builder - Communication View



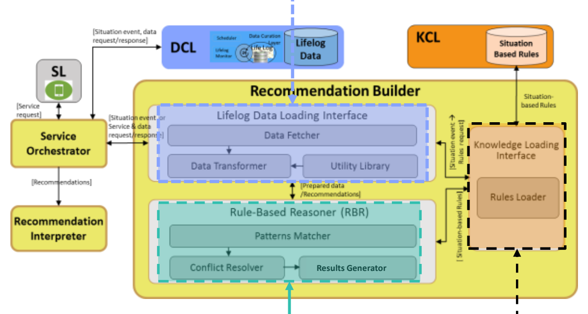
경희대학교 KYUNG HEE UNIVERSITY

Recommendation Builder – Abstract Concept

- Data integration
  - Objective
    - DCL data loading for new instance generation
  - Method
    - Restful web service -- data transformation → utility library
- Knowledge integration
  - Objective
    - KCL Knowledge loading
  - Method
    - Restful web service -- JSON
- Reasoning
  - Pattern matcher
    - Objective
      - Matching rules against new instance
    - Method
      - Pattern matching forward chaining algorithm
  - Conflict resolver
    - Objective
      - Selecting final rule from a set of matched rule
    - Method
      - Maximum specificity conflict resolving algorithm
  - Results generator
    - Objective
      - Communicating the results of fired rule with Interpreter
    - Method
      - Complex object preparation – recommendation packet

Loading and Preparing Data

Loading Knowledge Rules



Rules matching, conflict resolving and recommendations generation

Recommendation Builder – Abstract Concept

Data Loading Interface

Lifelog Data Loading Interface

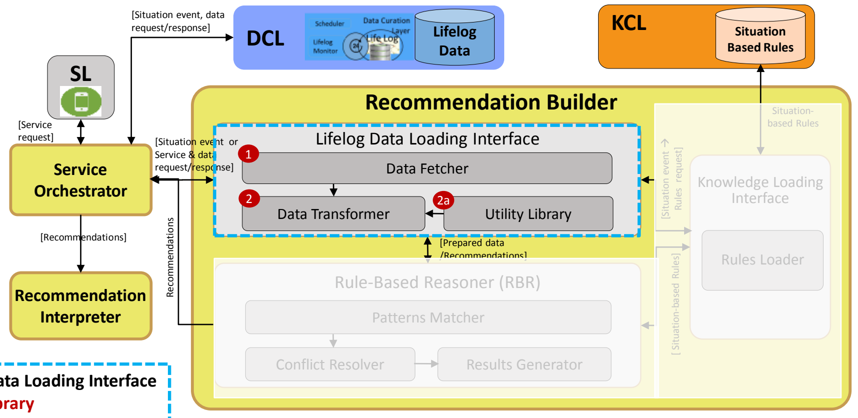
Data Fetcher

1. Receives requests from SL and DCL for recommendations
2. Parse request (service contract basis)
3. Request Rules for the service using knowledge loading interface
4. Requests life-log data from DCL

Lifelog Data Loading Interface

Data Transformer

- Parse rules conditions part
- Conformance check of condition attributes and data
- Transformation of data to the condition formats using Utility Library functions



Lifelog Data Loading Interface

Utility Library

- Utilities functions are defined and implemented for abstracted conditions

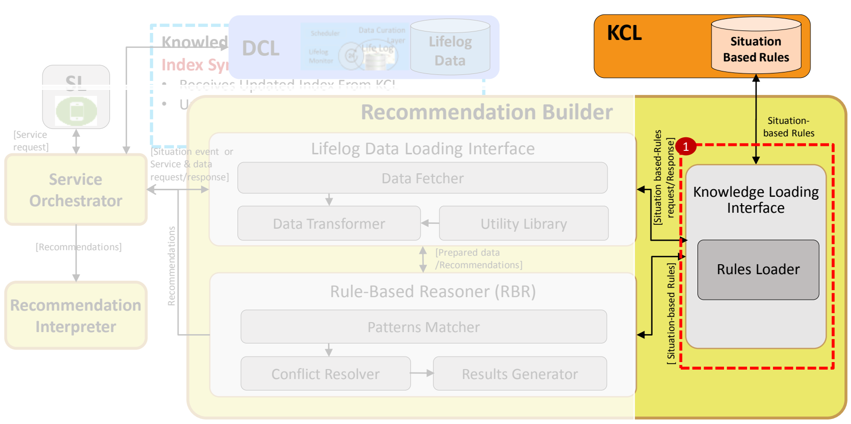
Recommendation Builder – Abstract Concept

Knowledge Loading Interface

Knowledge Loading Interface

Rules Loader

1. Takes situation event as a request from data fetcher
2. Send rules request to KCL as a restful web service request
3. Receives Rules as a restful web service response from KCL
4. Provides Rules to Reasoner and Data Loading Interface



Recommendation Builder – Abstract Concept

Rule-based Reasoner

1 Rule-based Reasoner

Pattern Matcher

1. Perform data types conversion
2. Matches rules's conditions against each data (prepared data) from DCL
3. Uses forward chaining algorithm for matching
4. Returns list of matched rules

2 Rule-based Reasoner

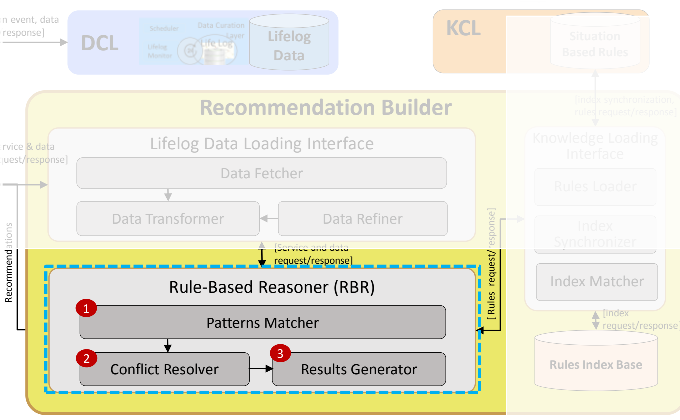
Conflict Resolver

- Receives list of matched rules from pattern matcher
- Performs conflict resolution using maximum specify algorithm
- Returns final set of resolved rules

3 Rule-based Reasoner

Results Generator

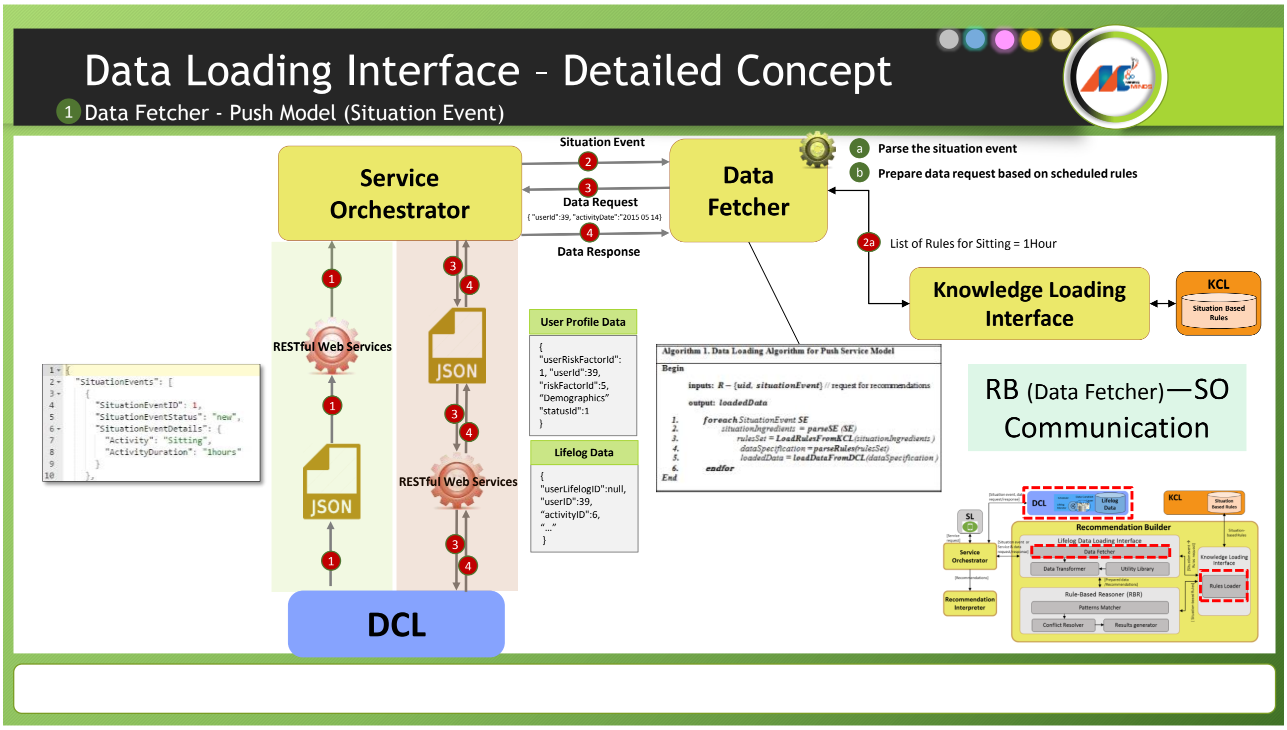
- Prepare the final results, i.e., conclusion part of the rule
- Bind user id and service id with the conclusion
- Provides recommendations to interpreter as an object model

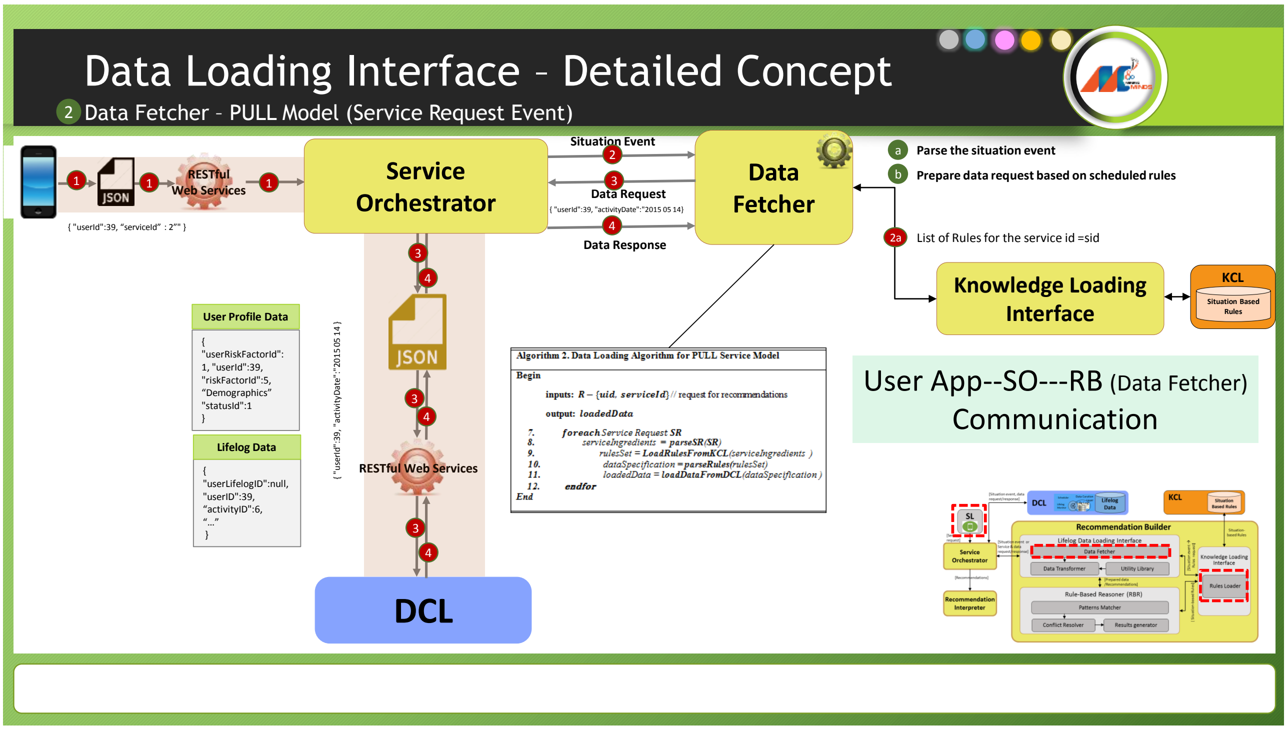


Data Loading Interface – Detailed Concept

Data Integration

(Lifelog Data Loading Interface


Data Loading Interface – Detailed Concept
1 Data Fetcher - Push Model (Situation Event)
Service Orchestrator
Situation Event
2
3
Data Request
{ "userId":39, "activityDate":"2015 05 14}
4
Data Response
Data Fetcher
a Parse the situation event
b Prepare data request based on scheduled rules
2a List of Rules for Sitting = 1Hour
Knowledge Loading Interface
KCL
Situation Based Rules
RESTful Web Services
JSON
User Profile Data
{
"userRiskFactorId": 1, "userId":39,
"riskFactorId":5,
"Demographics"
"statusId":1
}
Lifelog Data
{
"userLifelogID":null,
"userID":39,
"activityID":6,
"..."
}
Algorithm 1. Data Loading Algorithm for Push Service Model
RB (Data Fetcher)—SO Communication
DCL
Recommendation Builder
Service Orchestrator
Recommendation Interpreter


Data Loading Interface – Detailed Concept
2 Data Fetcher - PULL Model (Service Request Event)
JSON
RESTful Web Services
{ "userId":39, "serviceId" : 2"" }
Service Orchestrator
Situation Event
Data Request
{ "userId":39, "activityDate":"2015 05 14}
Data Response
Data Fetcher
a Parse the situation event
b Prepare data request based on scheduled rules
2a List of Rules for the service id =sid
Knowledge Loading Interface
KCL
Situation Based Rules
User Profile Data
{
"userRiskFactorId": 1, "userId":39,
"riskFactorId":5,
"Demographics"
"statusId":1
}
Lifelog Data
{
"userLifelogID":null,
"userID":39,
"activityID":6,
"..."
}
Algorithm 2. Data Loading Algorithm for PULL Service Model
User App--SO---RB (Data Fetcher) Communication
DCL
Recommendation Builder
Service Orchestrator
Recommendation Interpreter

Data Loading Interface – Detailed Concept

Data Transformation & Utility Kibrary

Example Rule:
If Activity = Sitting, ActivityDuration >=1h, Gender = Male, AgeGroup = Adult, WeightStatus = Overweight →
"RuleConclusion": "TAKE REST! Please have walk for 10 minutes outdoor !!!"

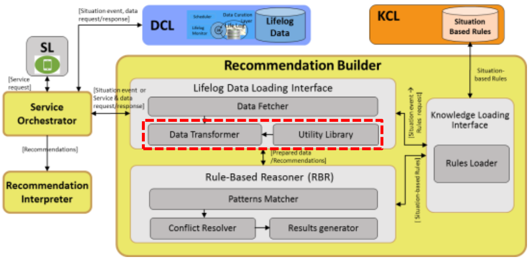
Data Fetcher → **Data Transformer** ← **Utility Library**

Algorithm 3. Data Transformation and Preparation Using Utility Library

```
Begin
    inputs: loadedData < lifeLogData, ProfData >,utilLibrary
    output: AbsCond, utilLibraryFunc
        foreach AbsCond of the Rule R
            if (AbsCond ϵ utilLibrary)
            AbsCond = transform(loadedData)
            Else
            utilLibrary = updateUtilLib (loadedData)
            endif
        endfor
End
```

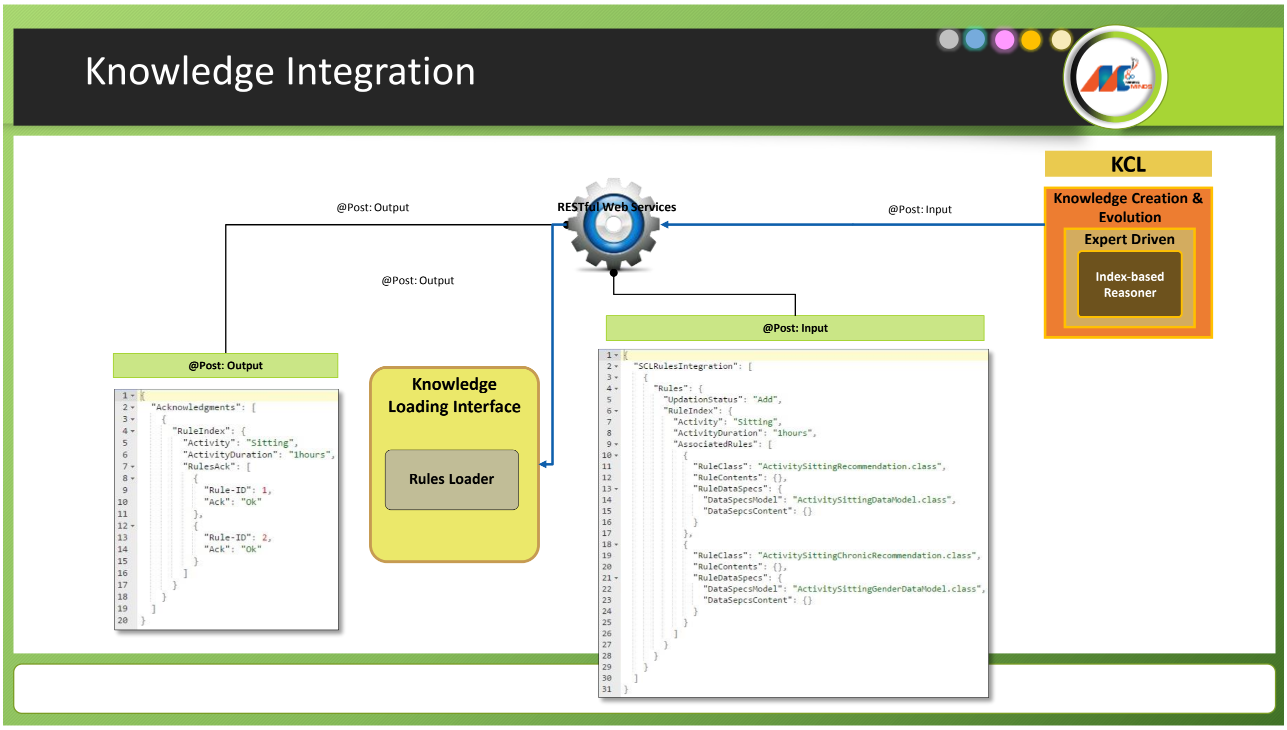
Loaded Data from Lifelog
- Gender: Male
- DoB: 1985
- Height: 6ft
- Weight: 89Kg
- Activity: Sitting
- ActivityDuration: >=1h

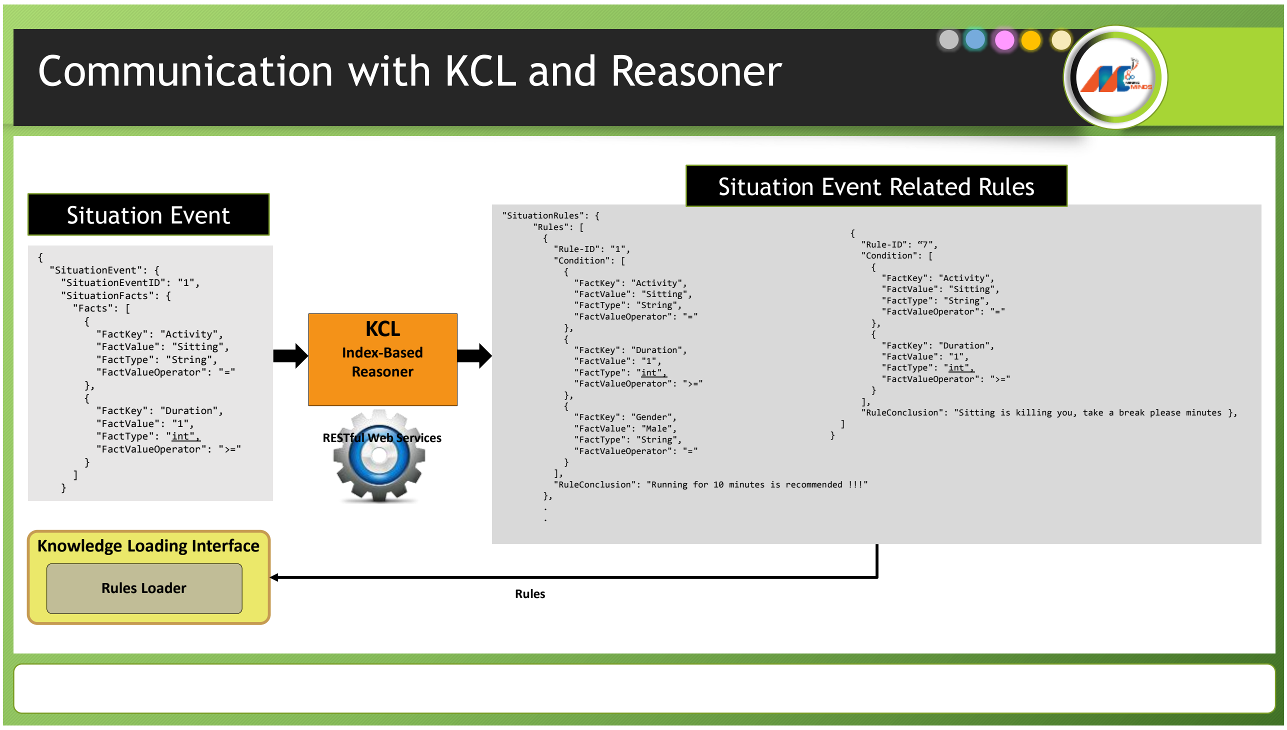
Transformation
- **Rule Parsing - Only Condition Part**
  - **(Activity, Gender, AgeGroup, WeightStatus)**
- **Conformance check**
  - (AgeGroup, WeightStatus)
- **Refinement/computation (using Utility Library)**
  - Dob → **AgeGroup** (transformation)
  - Height, Weight→ **WeightStatus** (transformation)
    - Height, Weight → BMI
    - BMI → WeightStatus



Knowledge Loading Interface – Detailed Concept

Knowledge Integration
(Knowledge Loading Interface)


Knowledge Integration
KCL
Knowledge Creation & Evolution
Expert Driven
Index-based Reasoner
@Post: Output
RESTful Web Services
@Post: Input
@Post: Output
@Post: Output
Knowledge Loading Interface
Rules Loader
@Post: Input
{
"Acknowledgments": [
{
"RuleIndex": {
"Activity": "Sitting",
"ActivityDuration": "1hours",
"RulesAck": [
{
"Rule-ID": 1,
"Ack": "Ok"
},
{
"Rule-ID": 2,
"Ack": "Ok"
}
]
}
}
]
}
{
"SCLRulesIntegration": [
{
"Rules": {
"UpdationStatus": "Add",
"RuleIndex": {
"Activity": "Sitting",
"ActivityDuration": "1hours",
"AssociatedRules": [
{
"RuleClass": "ActivitySittingRecommendation.class",
"RuleContents": {},
"RuleDataSpecs": {
"DataSpecsModel": "ActivitySittingDataModel.class",
"DataSepcsContent": {}
}
},
{
"RuleClass": "ActivitySittingChronicRecommendation.class",
"RuleContents": {},
"RuleDataSpecs": {
"DataSpecsModel": "ActivitySittingGenderDataModel.class",
"DataSepcsContent": {}
}
}
]
}
}
}
]
}


Communication with KCL and Reasoner
Situation Event
{
"SituationEvent": {
"SituationEventID": "1",
"SituationFacts": {
"Facts": [
{
"FactKey": "Activity",
"FactValue": "Sitting",
"FactType": "String",
"FactValueOperator": "="
},
{
"FactKey": "Duration",
"FactValue": "1",
"FactType": "int",
"FactValueOperator": ">="
}
]
}
KCL
Index-Based Reasoner
RESTful Web Services
Situation Event Related Rules
"SituationRules": {
"Rules": [
{
"Rule-ID": "1",
"Condition": [
{
"FactKey": "Activity",
"FactValue": "Sitting",
"FactType": "String",
"FactValueOperator": "="
},
{
"FactKey": "Duration",
"FactValue": "1",
"FactType": "int",
"FactValueOperator": ">="
},
{
"FactKey": "Gender",
"FactValue": "Male",
"FactType": "String",
"FactValueOperator": "="
}
],
"RuleConclusion": "Running for 10 minutes is recommended !!!"
},
.
.
{
"Rule-ID": "7",
"Condition": [
{
"FactKey": "Activity",
"FactValue": "Sitting",
"FactType": "String",
"FactValueOperator": "="
},
{
"FactKey": "Duration",
"FactValue": "1",
"FactType": "int",
"FactValueOperator": ">="
}
],
"RuleConclusion": "Sitting is killing you, take a break please minutes },
]
}
Knowledge Loading Interface
Rules Loader
Rules

Rule-based Reasoning

Recommendation Contents Generation
(Rule-based Reasoner)

Rule-based Reasoning

Pattern Matcher Matching Strategies

RBR Strategies (Matching)

Forward chaining (data driven)

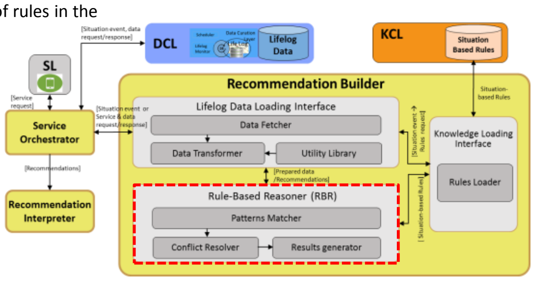
- Works from *facts (conditions) to a conclusion*.
- Matches *data* against '*conditions*' of rules in the rule-base.

Backward chaining (goal-driven)

- Works from the *conclusion to the facts (conditions)*.
- Matches a *goal* against '*conclusions*' of rules in the rule-base.

Mixed chaining (forward-backward or backward-forward)

- Some rules are used for *chaining forwards*, and others for *chaining backwards*.
- System first chains in one direction, then switches to the other direction (use of meta-rules).



Rule-based Reasoning
Choice of Appropriate Matching Strategies

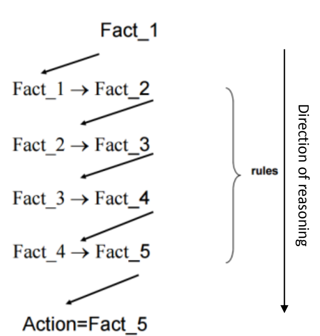
Forward chaining	Backward chaining
• Appropriate when all the facts are provided with the problem statement	• Appropriate when the goal is given in the problem statement
• Appropriate when there are many possible goals or there isn't any sensible way to guess what the goal is at the beginning of the reasoning.	• Appropriate when goal can sensibly be guessed at the beginning of the reasoning
• Appropriate for monitoring, planning, and interpretation applications	• Appropriate for Diagnostic, prescription and debugging applications
• Starts from data/request	• Starts from conclusion/decision
• Aims for finding conclusion(s)	• Aims for finding necessary data (reasons of decision)
• Bottom-up reasoning	• Top-down reasoning
• Breadth-first search	• Depth-first search
• Flow is from facts/conditions to conclusion	• Flow is from consequent to conditions

Our Choice is Forward Chaining

경희대학교 KYUNG HEE UNIVERSITY

http://www.ijetae.com/files/Volume2Issue10/IJETAE_1012_48.pdf

Rule-based Reasoning
Forward Chaining Strategies



While (**goal** is not reached)

Foreach rule in the **Rulebase**

1. bind **Data** by matching to **Facts** (**conditions**)

2. if all **Facts** are matched, generate **Action**

(**Conclusion**)

3. **Repeat** 1. and 2. for all matching/instantiation

alternatives

Endfor

Continue

경희대학교 KYUNG HEE UNIVERSITY

Rule-based Reasoning
Forward Chaining Example

R#1: If gender=male, user_age=25, height=6.5 → **BMI** (<18, <19-25>, <25-30>, >30)

R#2: If BMI=X → **Weight Status** (UW, OW, N, OB)

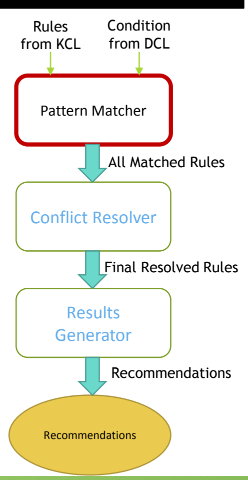
R#3: If user_weight_Status= OW (overweight), health_Condition=Normal → **GoalForWeightLoss** = Burn 2000 Cal

R#4: If goal=2000 Cal, user_Interest=walking, phy_act_Routines=Active, diet_Routines=regular → **Brisk walk for 30 minutes**

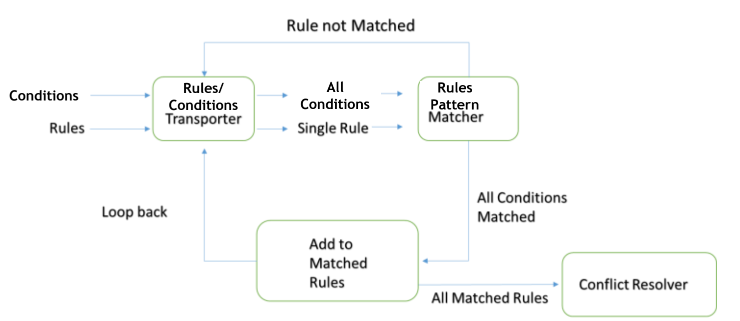
경희대학교
KYUNG HEE UNIVERSITY

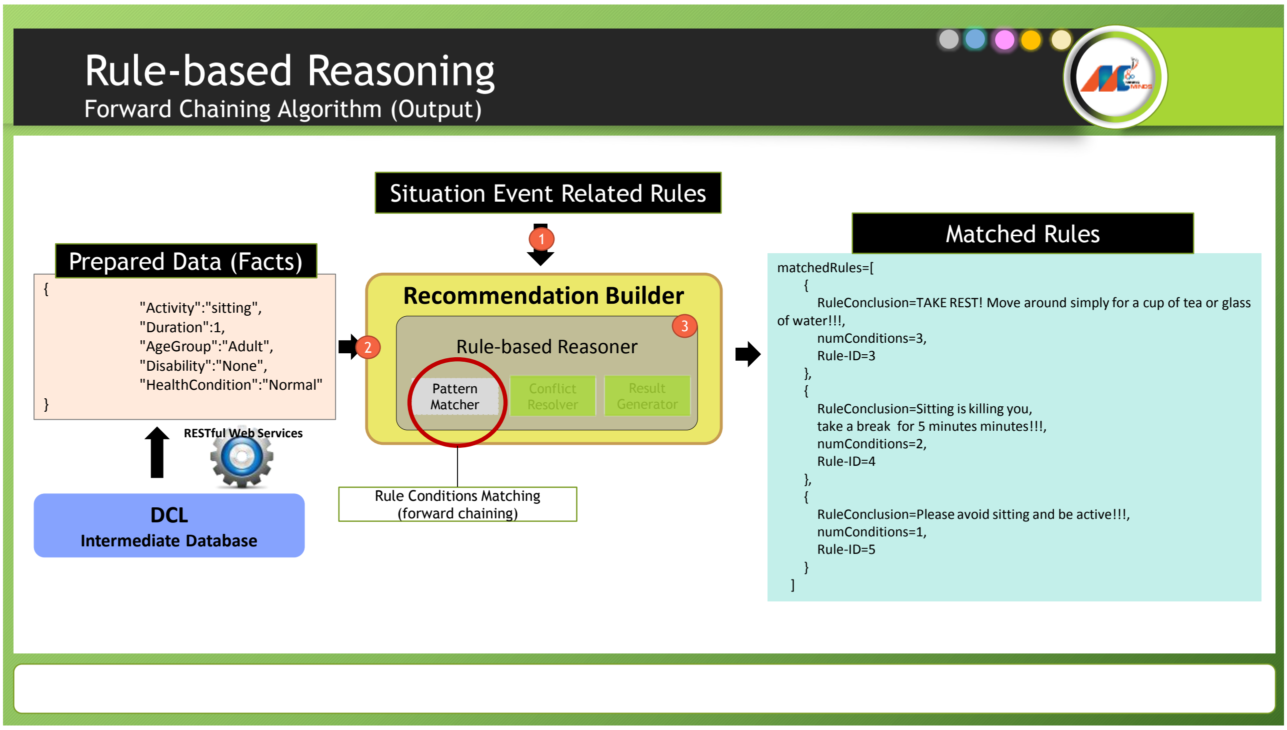
Rule-based Reasoning
Forward Chaining Algorithm

Recommendation Builder

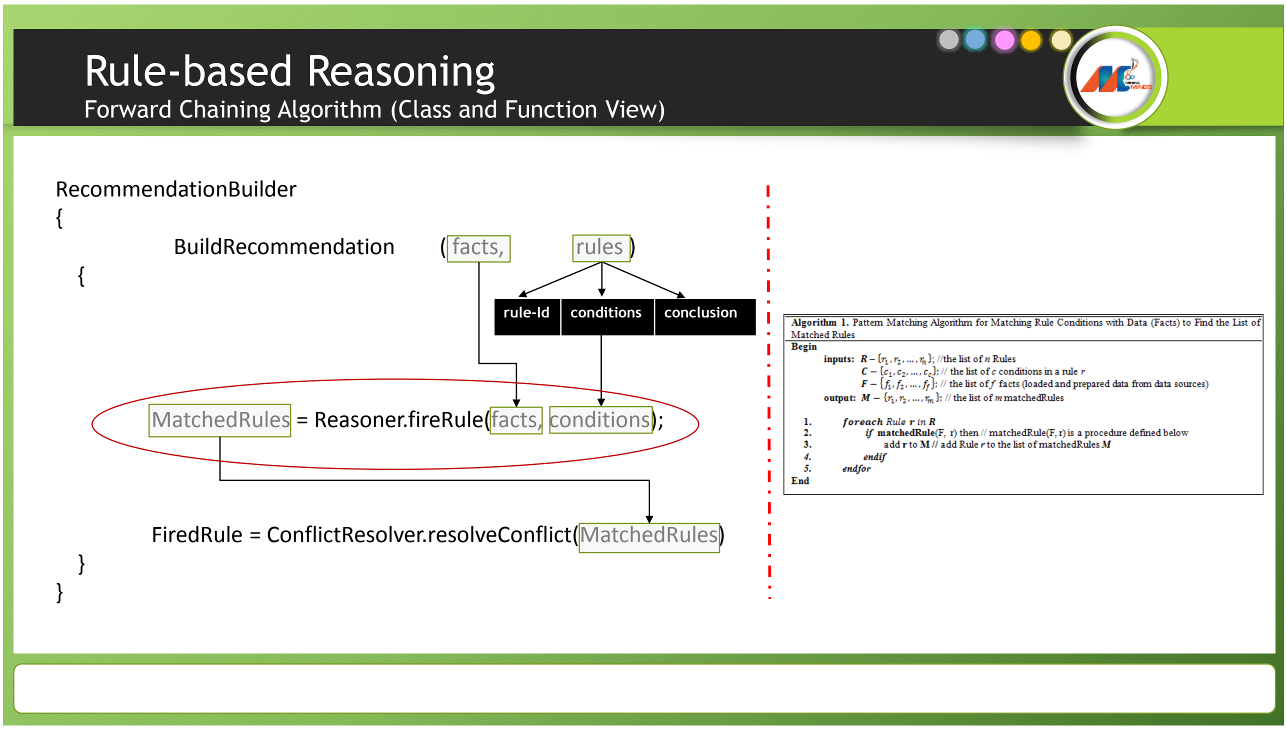


Rules Pattern Matching Algorithm

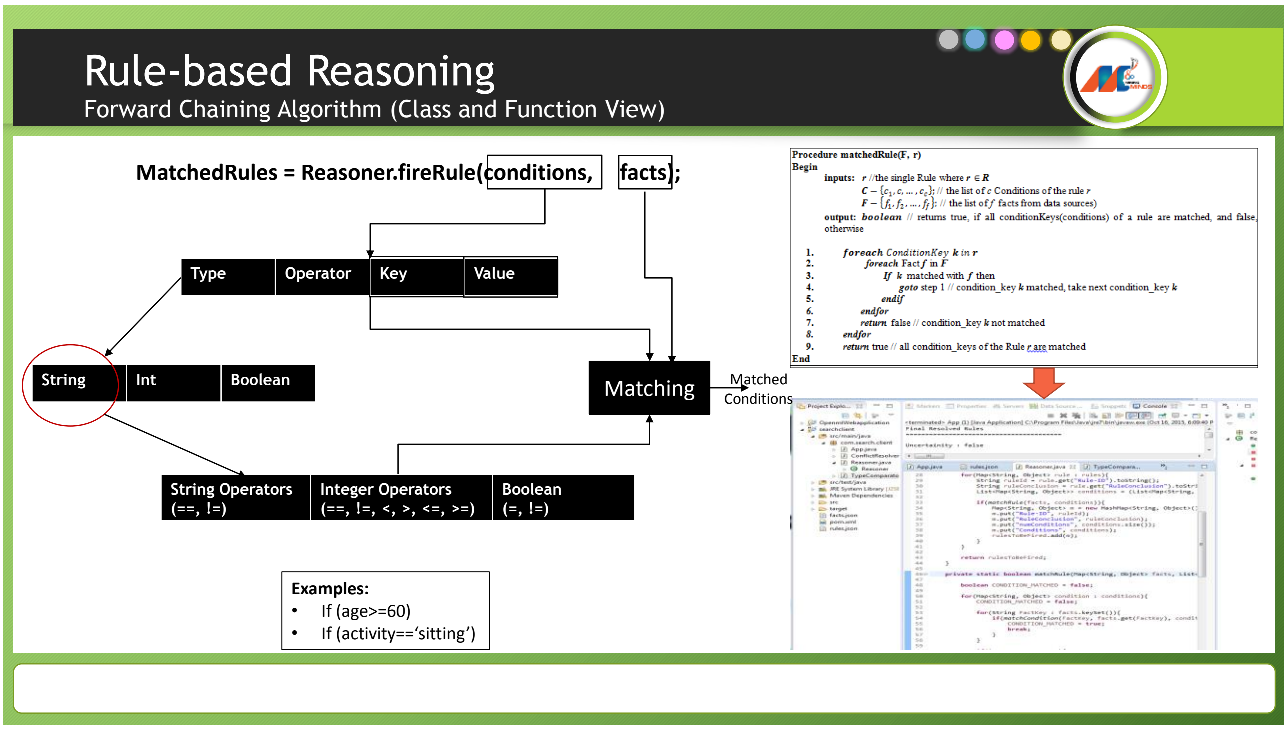


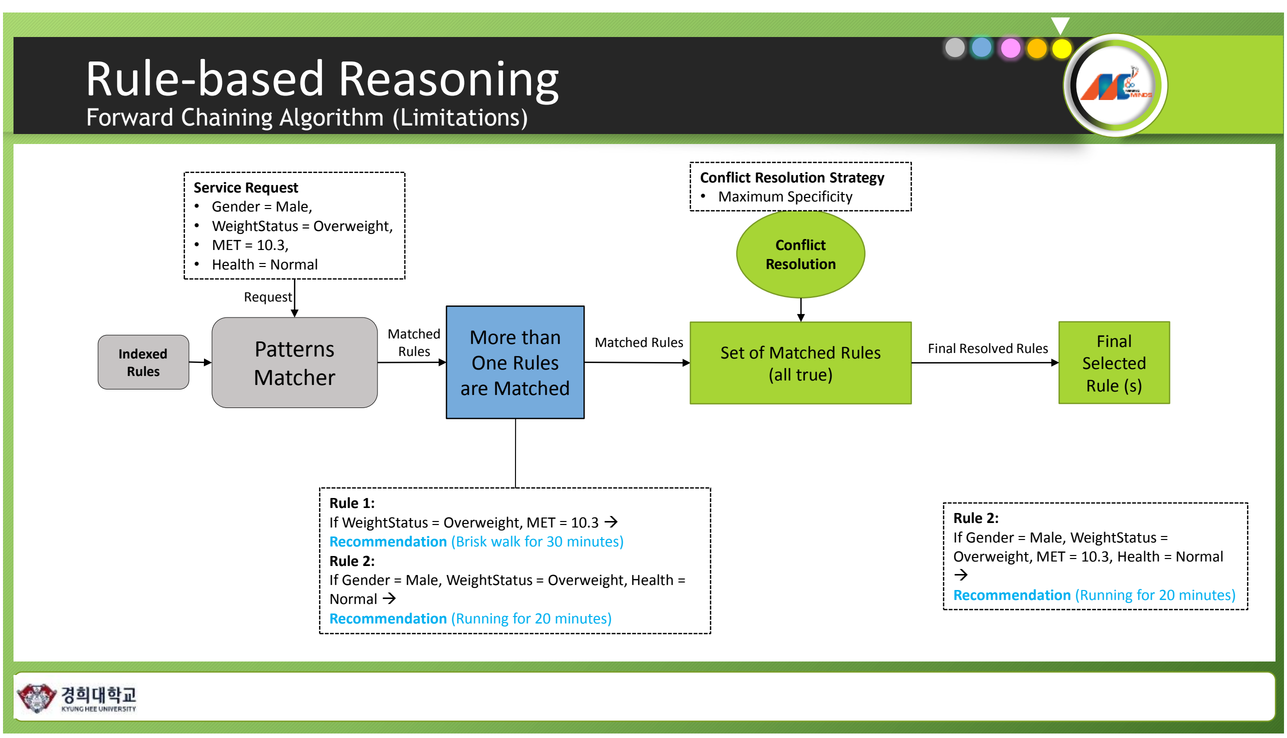


Rule-based Reasoning
Forward Chaining Algorithm (Output)
Situation Event Related Rules
Prepared Data (Facts)
{
"Activity":"sitting",
"Duration":1,
"AgeGroup":"Adult",
"Disability":"None",
"HealthCondition":"Normal"
}
RESTful Web Services
DCL
Intermediate Database
Recommendation Builder
Rule-based Reasoner
Pattern Matcher
Conflict Resolver
Result Generator
Rule Conditions Matching (forward chaining)
Matched Rules
matchedRules=[
{
RuleConclusion=TAKE REST! Move around simply for a cup of tea or glass of water!!!,
numConditions=3,
Rule-ID=3
},
{
RuleConclusion=Sitting is killing you, take a break for 5 minutes minutes!!!,
numConditions=2,
Rule-ID=4
},
{
RuleConclusion=Please avoid sitting and be active!!!,
numConditions=1,
Rule-ID=5
}
]



Rule-based Reasoning
Forward Chaining Algorithm (Class and Function View)
RecommendationBuilder
{
BuildRecommendation (facts, rules)
{
rule-Id
conditions
conclusion
MatchedRules = Reasoner.fireRule(facts, conditions);
FiredRule = ConflictResolver.resolveConflict(MatchedRules)
}
}
Algorithm 1. Pattern Matching Algorithm for Matching Rule Conditions with Data (Facts) to Find the List of Matched Rules
Begin
inputs: R – {r1, r2, ..., rn}; //the list of n Rules
C – {c1, c2, ..., cc}; // the list of c conditions in a rule r
F – {f1, f2, ..., ff}; // the list of f facts (loaded and prepared data from data sources)
output: M – {r1, r2, ..., rm}; // the list of m matchedRules
1. foreach Rule r in R
2. if matchedRule(F, r) then // matchedRule(F, r) is a procedure defined below
3. add r to M // add Rule r to the list of matchedRules M
4. endif
5. endfor
End


Rule-based Reasoning
Forward Chaining Algorithm (Class and Function View)
MatchedRules = Reasoner.fireRule(conditions, facts);
Type
Operator
Key
Value
String
Int
Boolean
Matching
Matched Conditions
String Operators (==, !=)
Integer Operators (==, !=, <, >, <=, >=)
Boolean (=, !=)
Examples:
If (age>=60)
If (activity=='sitting')
Procedure matchedRule(F, r)
Begin
inputs: r //the single Rule where r ∈ R
C – {c1, c, ... , cc}; // the list of c Conditions of the rule r
F – {f1, f2, ... , ff}; // the list of f facts from data sources)
output: boolean // returns true, if all conditionKeys(conditions) of a rule are matched, and false, otherwise
1. foreach ConditionKey k in r
2. foreach Fact f in F
3. If k matched with f then
4. goto step 1 // condition_key k matched, take next condition_key k
5. endif
6. endfor
7. return false // condition_key k not matched
8. endfor
9. return true // all condition_keys of the Rule r are matched
End


Rule-based Reasoning
Forward Chaining Algorithm (Limitations)
Service Request
Gender = Male,
WeightStatus = Overweight,
MET = 10.3,
Health = Normal
Request
Indexed Rules
Patterns Matcher
Matched Rules
More than One Rules are Matched
Matched Rules
Conflict Resolution Strategy
Maximum Specificity
Conflict Resolution
Set of Matched Rules (all true)
Final Resolved Rules
Final Selected Rule (s)
Rule 1:
If WeightStatus = Overweight, MET = 10.3 →
Recommendation (Brisk walk for 30 minutes)
Rule 2:
If Gender = Male, WeightStatus = Overweight, Health = Normal →
Recommendation (Running for 20 minutes)
Rule 2:
If Gender = Male, WeightStatus = Overweight, MET = 10.3, Health = Normal →
Recommendation (Running for 20 minutes)
경희대학교
KYUNG HEE UNIVERSITY

Conflict Resolution

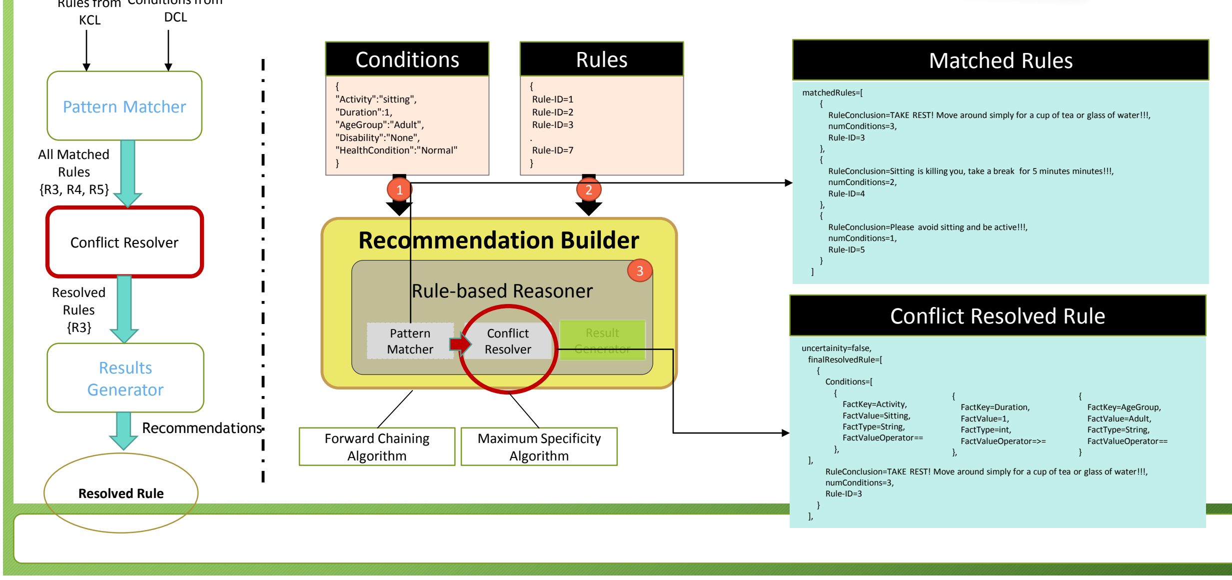
Strategies

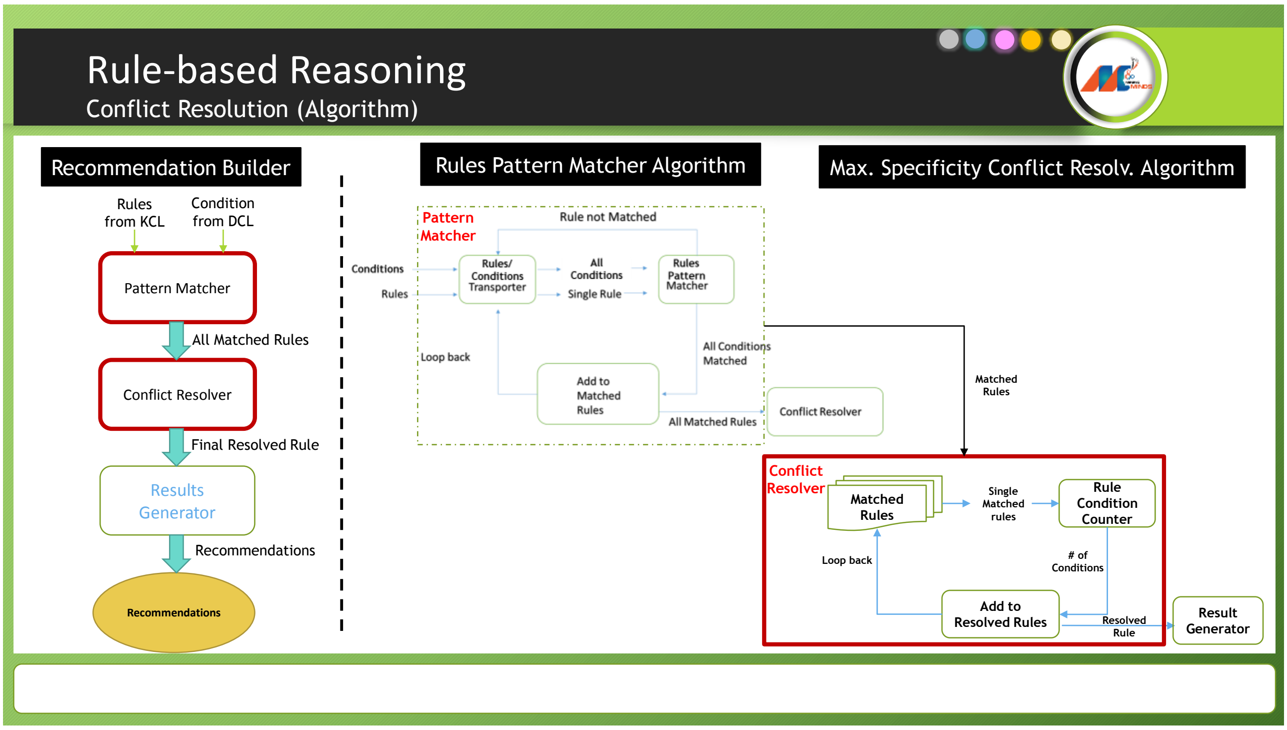
- Specificity or Maximum Specificity
  - based on number of condition attributes matched
  - choose the rule with the most/least matches for condition attributes
- Priority-based approach
  - arrange condition attributes in priority queue
  - use rule dealing with highest priority condition attributes
- Explicit /meta-rules for conflict resolution
  - rule based system within a rule based system
  - use meta-rules to resolve the conflict

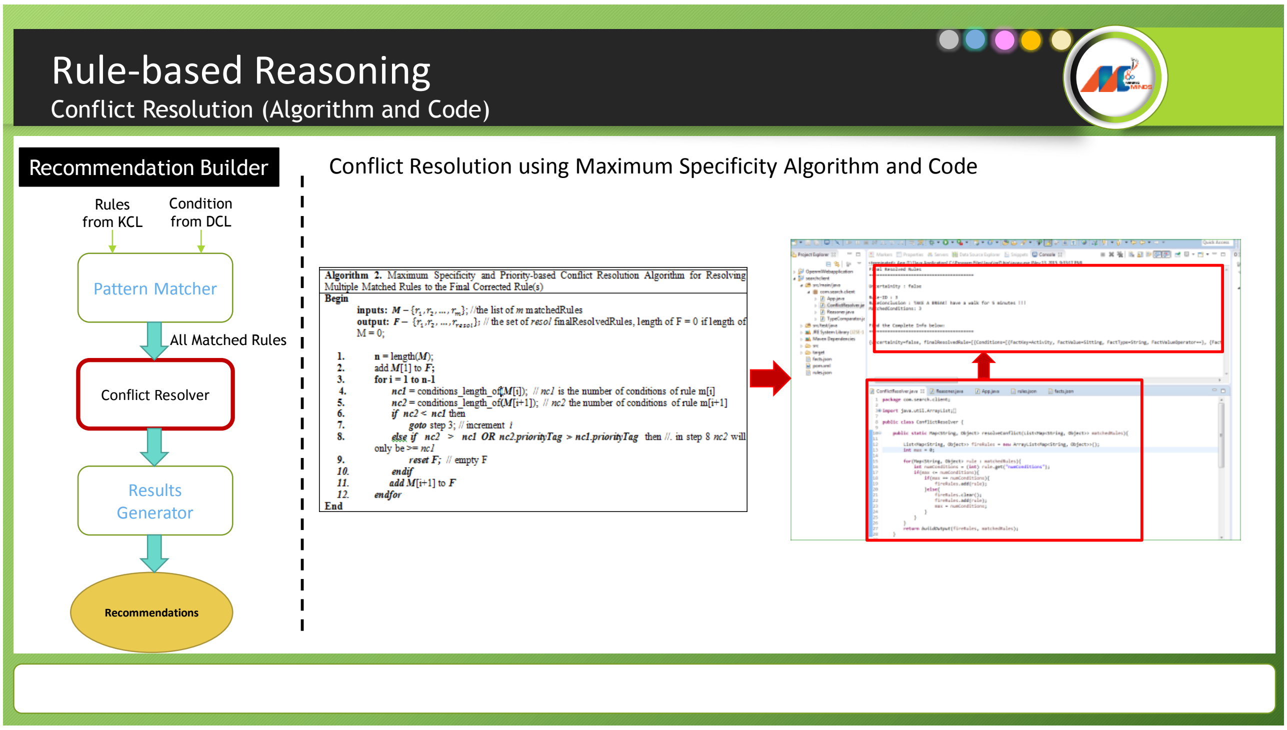
- Fire All Selected Rules
  - Executes all the matched rules
- Context Limiting
  - partition rule base into disjoint subsets
  - doing this we can have subsets and we may also have preconditions
- Execution Time
  - Chose the one with faster execution
- Physically ordering of rules
  - hard to add rules based on their physical order
- Random
  - Randomly pick one rule for execution

Rule-based Reasoning

Conflict Resolution (Examples)

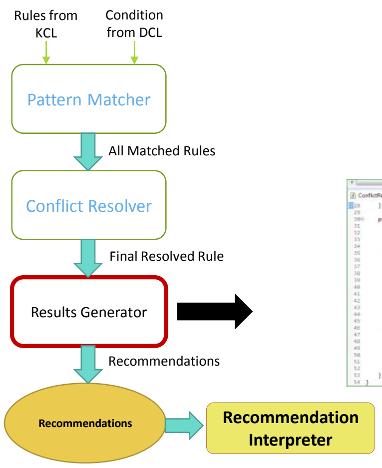


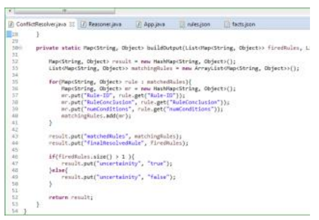

Rule-based Reasoning
Conflict Resolution (Algorithm)
Recommendation Builder
Rules from KCL
Condition from DCL
Pattern Matcher
All Matched Rules
Conflict Resolver
Final Resolved Rule
Results Generator
Recommendations
Recommendations
Rules Pattern Matcher Algorithm
Pattern Matcher
Rule not Matched
Conditions
Rules
Rules/ Conditions Transporter
All Conditions
Single Rule
Rules Pattern Matcher
Loop back
All Conditions Matched
Add to Matched Rules
All Matched Rules
Conflict Resolver
Max. Specificity Conflict Resolv. Algorithm
Matched Rules
Conflict Resolver
Matched Rules
Single Matched rules
Rule Condition Counter
Loop back
# of Conditions
Add to Resolved Rules
Resolved Rule
Result Generator


Rule-based Reasoning
Conflict Resolution (Algorithm and Code)
Recommendation Builder
Rules from KCL
Condition from DCL
Pattern Matcher
All Matched Rules
Conflict Resolver
Results Generator
Recommendations
Conflict Resolution using Maximum Specificity Algorithm and Code
Algorithm 2. Maximum Specificity and Priority-based Conflict Resolution Algorithm for Resolving Multiple Matched Rules to the Final Corrected Rule(s)
Begin
inputs: M – {r1, r2, ..., rm}; //the list of m matchedRules
output: F – {r1, r2, ..., rresol}; // the set of resol finalResolvedRules, length of F = 0 if length of M = 0;
1. n = length(M);
2. add M[1] to F;
3. for i = 1 to n-1
4. nc1 = conditions_length_of(M[i]); // nc1 is the number of conditions of rule m[i]
5. nc2 = conditions_length_of(M[i+1]); // nc2 the number of conditions of rule m[i+1]
6. if nc2 < nc1 then
7. goto step 3; // increment i
8. else if nc2 > nc1 OR nc2.priorityTag > nc1.priorityTag then //. in step 8 nc2 will only be >= nc1
9. reset F; // empty F
10. endif
11. add M[i+1] to F
12. endfor
End

Rule-based Reasoning

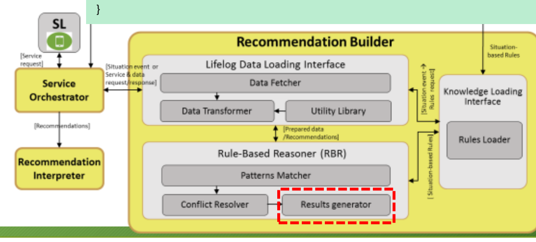
Result Generator





{
ruleConclusion: "TAKE A BREAK! Bend your waist forward at least 4 times to stretch your body! Your heart will thank you."
conclusionList: [
2]0: {
conclusionKey: "Activity"
conclusionValue: "Stretching"
conclusionOperator: "="
ruleID: 4
conclusionID: 8
}
-
1: {
conclusionKey: "Times"
conclusionValue: "4"
conclusionOperator: "="
ruleID: 4
conclusionID: 9
}

Recommendation will be Shared in Complex Object Form



경희대학교
KYUNG HEE UNIVERSITY

Tools and Technologies

- Language: Java
- Java Framework: Spring
- Web Server: Glassfish
- NetBean IDE/Eclipse
- Communications
  - Orchestrator-Recommendation builder (*Restful Services*)
  - Recommendation Builder - KCL (*Restful Services*)

경희대학교
KYUNG HEE UNIVERSITY

Contributions

- An indexed rules-based reasoning engine for reliable recommendations using rule-based forward chaining strategy
- A flexible recommendation framework with loosely coupled data and knowledge for diverse services
- Provisioning of non-conflicting accurate recommendations using specificity conflict resolution strategy

경희대학교
KYUNG HEE UNIVERSITY

Conclusions

- A rules based reasoning engine is implemented using rule-based forward chaining strategy
- Lifelog data integration and transformation for recommendations
- Rules conflict resolution with maximum specificity technique for accurate recommendations

Thank You!

Appendix

Demo Scenarios

Recommendation Builder Demo Scenarios

MM V2.0_SCL_T5_V0.3_20151016_Rahman Ali

- **Scenario 1**
  - Pattern Matcher – Forward Chaining Algorithm
    - Sub-scenario 1
      - Only one rule matched
    - Sub-scenario 2
      - More than 1 rules matched
    - Sub-scenario 3
      - No rule matched

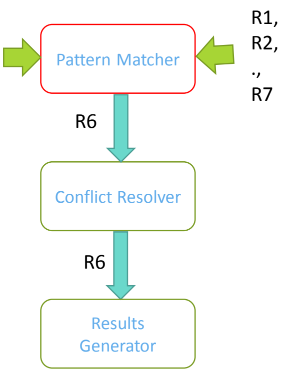
- **Scenario 2**
  - Conflict Resolver – Maximum Specificity Algorithm
    - Sub-scenario 1
      - More than 1 rules matched → Resolved to → Single rule
    - Sub-scenario 2 - Uncertainty
      - More than 1 rules matched → Resolved to → More than 1 rules

Recommendation Builder Demo Scenarios

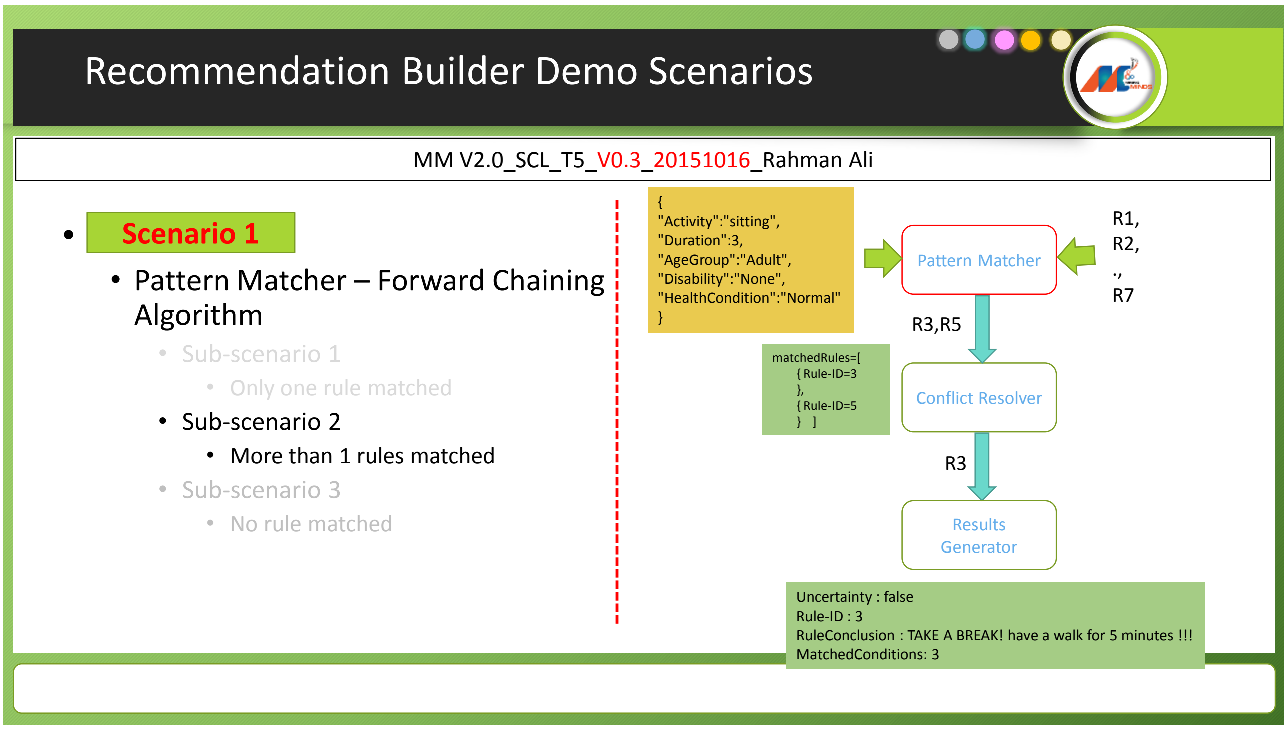
MM V2.0_SCL_T5_V0.3_20151016_Rahman Ali

- **Scenario 1**
  - Pattern Matcher – Forward Chaining Algorithm
    - Sub-scenario 1
      - Only one rule matched
    - Sub-scenario 2
      - More than 1 rules matched
    - Sub-scenario 3
      - No rule matched

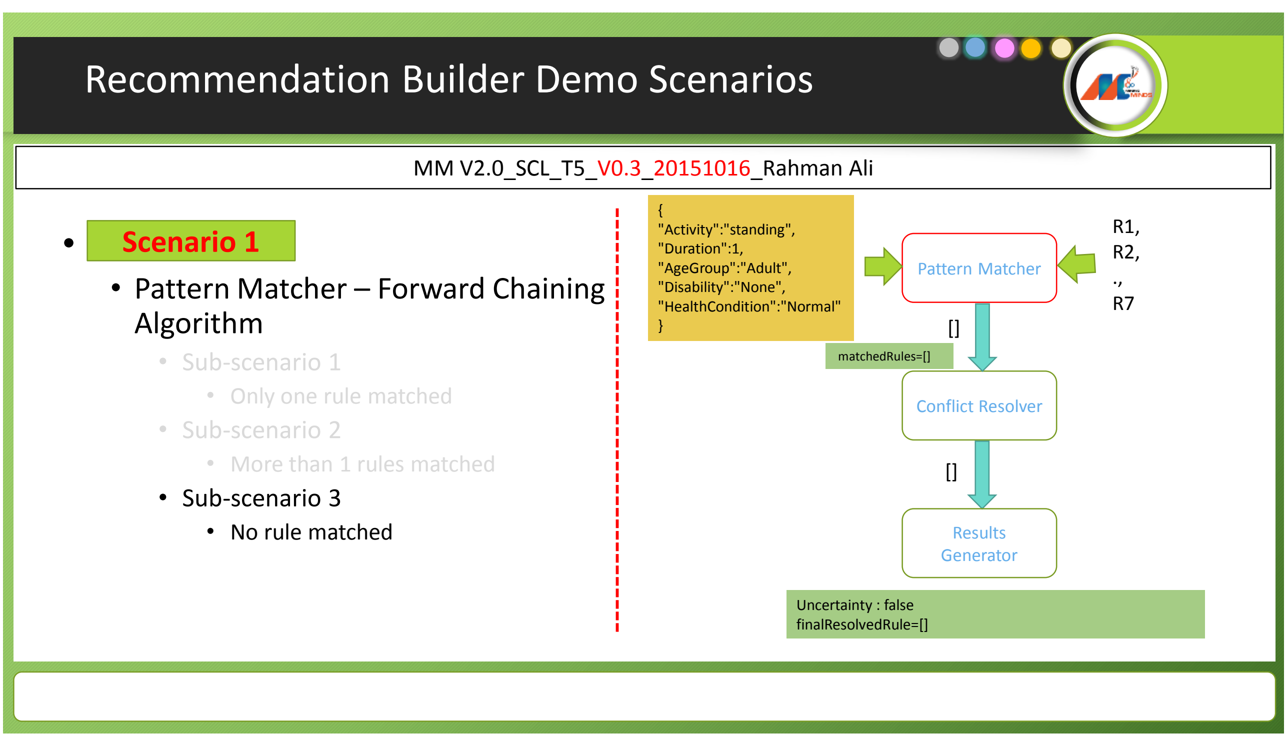
{
"Activity":"LyingDown",
"Duration":6,
"Chronic":"Yes"
}



Rule-ID : 6
RuleConclusion : Walking for atleast 10 minutes after prolonged lying down adds more to your health !!!
MatchedConditions: 3



Recommendation Builder Demo Scenarios
MM V2.0_SCL_T5_V0.3_20151016_Rahman Ali
Scenario 1
Pattern Matcher – Forward Chaining Algorithm
Sub-scenario 1
Only one rule matched
Sub-scenario 2
More than 1 rules matched
Sub-scenario 3
No rule matched
{ "Activity":"sitting", "Duration":3, "AgeGroup":"Adult", "Disability":"None", "HealthCondition":"Normal" }
Pattern Matcher
R1, R2, ., R7
R3,R5
matchedRules=[{ Rule-ID=3 }, { Rule-ID=5 }]
Conflict Resolver
R3
Results Generator
Uncertainty : false
Rule-ID : 3
RuleConclusion : TAKE A BREAK! have a walk for 5 minutes !!!
MatchedConditions: 3



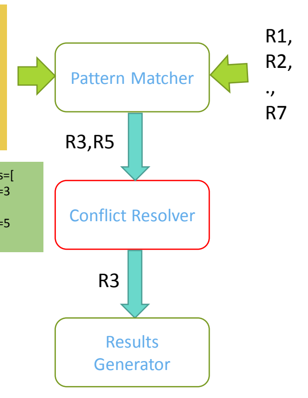
Recommendation Builder Demo Scenarios
MM V2.0_SCL_T5_V0.3_20151016_Rahman Ali
Scenario 1
Pattern Matcher – Forward Chaining Algorithm
Sub-scenario 1
Only one rule matched
Sub-scenario 2
More than 1 rules matched
Sub-scenario 3
No rule matched
{ "Activity":"standing", "Duration":1, "AgeGroup":"Adult", "Disability":"None", "HealthCondition":"Normal" }
Pattern Matcher
R1, R2, ., R7
[]
matchedRules=[]
Conflict Resolver
[]
Results Generator
Uncertainty : false
finalResolvedRule=[]

Recommendation Builder Demo Scenarios

MM V2.0_SCL_T5_V0.3_20151016_Rahman Ali

- **Scenario 2**
  - Conflict Resolver – Maximum Specificity Algorithm
    - Sub-scenario 1
      - More than 1 rules matched → Resolved to → Single rule
    - Sub-scenario 2 - Uncertainty
      - More than 1 rules matched → Resolved to → More than 1 rules

{
"Activity":"sitting",
"Duration":3,
"AgeGroup":"Adult",
"Disability":"None",
"HealthCondition":"Normal"
}



matchedRules=[
{ Rule-ID=3
},
{ Rule-ID=5
}]

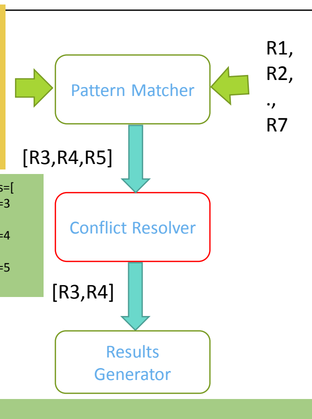
Uncertainty : false
Rule-ID : 3
RuleConclusion : TAKE A BREAK! have a walk for 5 minutes !!!
MatchedConditions: 3

Recommendation Builder Demo Scenarios

MM V2.0_SCL_T5_V0.3_20151016_Rahman Ali

- **Scenario 2**
  - Conflict Resolver – Maximum Specificity Algorithm
    - Sub-scenario 1
      - More than 1 rules matched → Resolved to → Single rule
    - Sub-scenario 2 - Uncertainty
      - More than 1 rules matched → Resolved to → More than 1 rules

{
"Activity":"sitting",
"Duration":3,
"AgeGroup":"Adult",
"Disability":"None",
"HealthCondition":"Normal",
"WeightStatus":"Overweight"
}



matchedRules=[
{ Rule-ID=3
},
{ Rule-ID=4
},
{ Rule-ID=5
}]

Uncertainty : true
Rule-ID : 3
RuleConclusion : TAKE A BREAK! have a walk for 5 minutes !!!
MatchedConditions: 3
Rule-ID : 4
RuleConclusion : Too much sitting will kill you, please take a break and have brisk walk for 15 minutes !!!
MatchedConditions: 3

Reasoner Input Cases Scenarios

MM V2.0_SCL_T5_V0.3_20151016_Rahman Ali

```
{
"Activity":"Standing",
"Duration":1,
"Chronic":"Yes"
}
```

→ No Match

```
{
"Activity":"LyingDown",
"Duration":6,
"Chronic":"Yes"
}
```

→ R6 → R6

```
{
"Activity":"sitting",
"Duration":3,
"AgeGroup":"Adult",
"Disability":"None",
"HealthCondition":"Normal",
"WeightStatus":"Overweight"
}
```

R3, R4, R5 → R3,R4

```
{
"Activity":"sitting",
"Duration":3,
"AgeGroup":"Adult",
"Disability":"None",
"HealthCondition":"Normal"
}
```

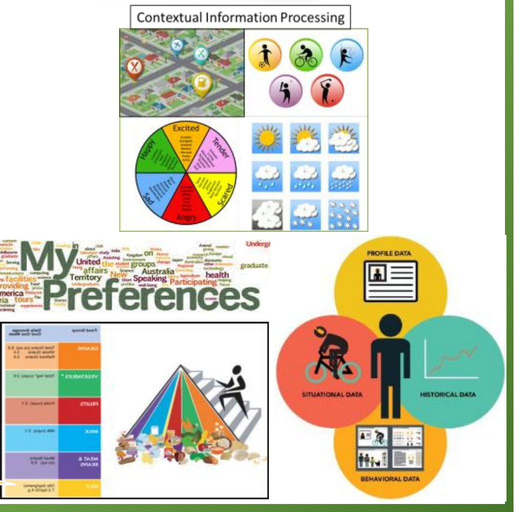
→ R3, R5 → R3

Recommendation Interpreter

Service Curation Layer 3.0

Overview

- **Personalization** is a key element in **Recommender Systems**
- **Personalization** consists of tailoring a service or a product to accommodate specific needs of **individuals**
- **Contextual Information** combined with **User Preferences** enable **Personalization**
- **Recommendation Interpreter** performs interpretation according to the contextual information and preferences of the user in order to deliver the appropriate recommendations at right time

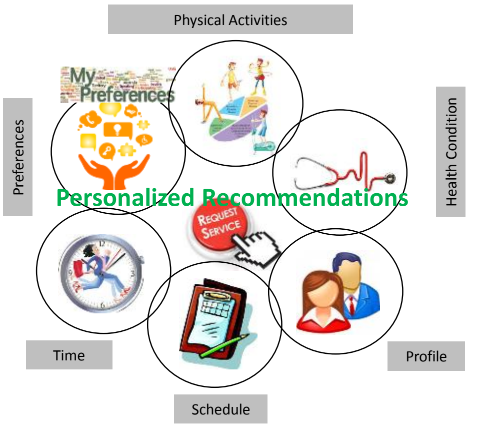


경희대학교 KYUNG HEE UNIVERSITY

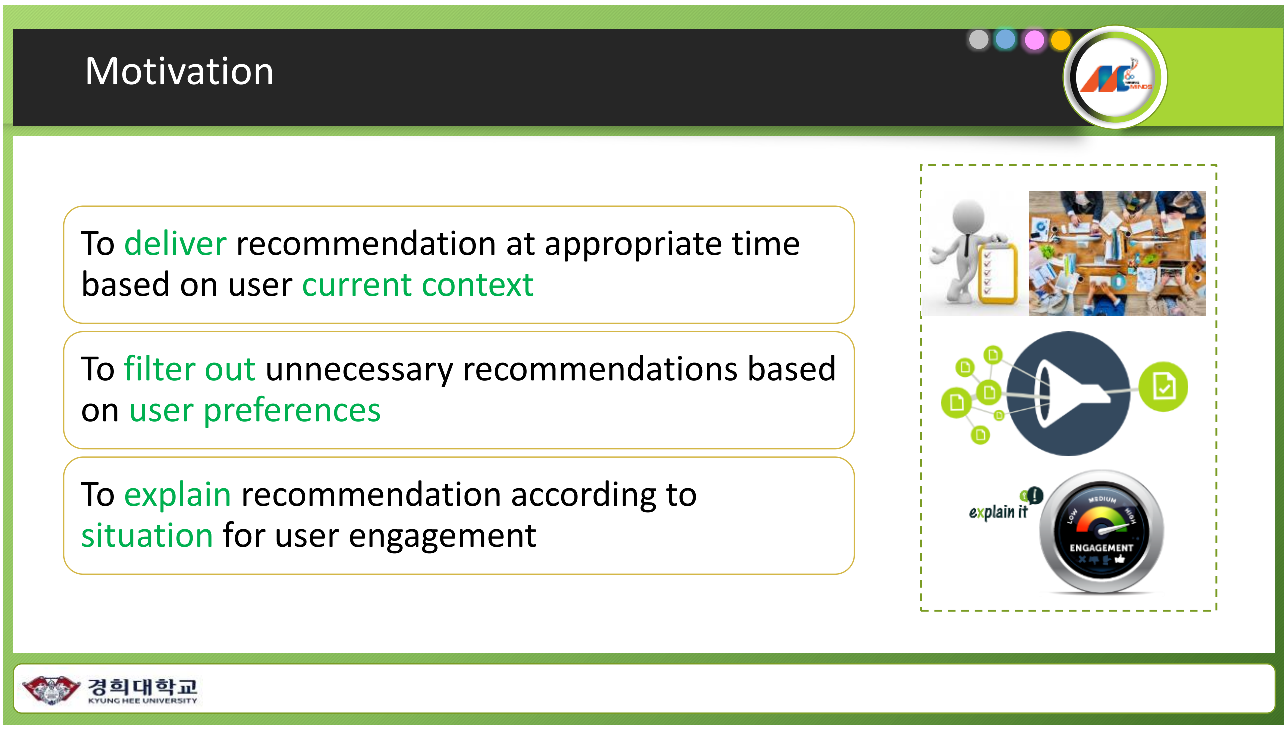
http://www.quora.com/What-is-the-definition-of-personalization

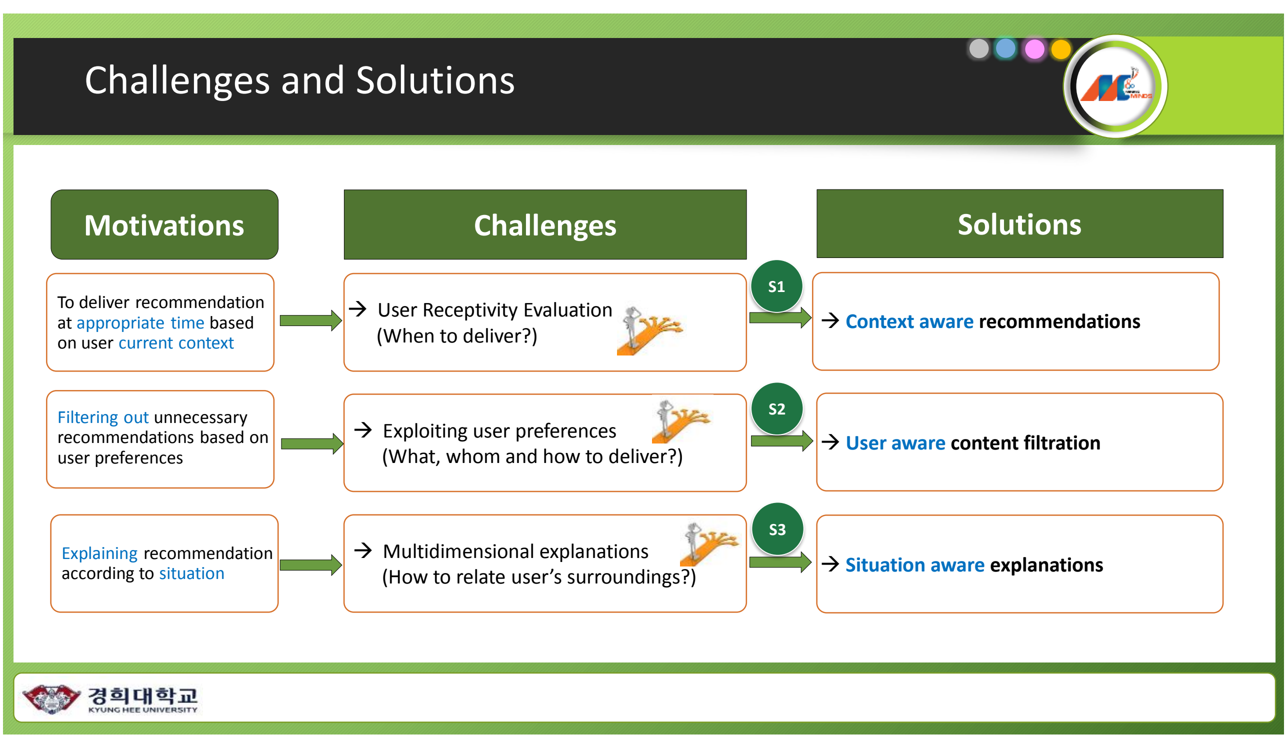
Goal and Objectives

- Goal
  - Providing **context-aware** and **personalized** wellbeing recommendations
- Objectives
  - Interpreting recommendations to address
    - **Receptiveness** of the user for recommendation
    - **Preferences** of the user for recommendation
    - User friendly **explanation** of the recommendation



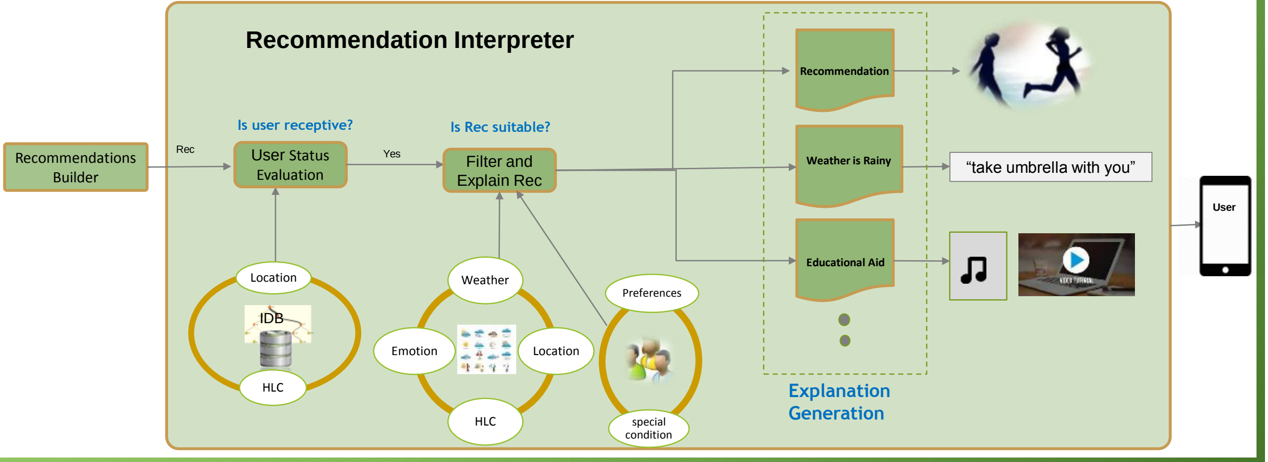
경희대학교 KYUNG HEE UNIVERSITY


Motivation
To deliver recommendation at appropriate time based on user current context
To filter out unnecessary recommendations based on user preferences
To explain recommendation according to situation for user engagement
explain it
ENGAGEMENT
경희대학교
KYUNG HEE UNIVERSITY


Challenges and Solutions
Motivations
Challenges
Solutions
To deliver recommendation at appropriate time based on user current context
→ User Receptivity Evaluation (When to deliver?)
S1
→ Context aware recommendations
Filtering out unnecessary recommendations based on user preferences
→ Exploiting user preferences (What, whom and how to deliver?)
S2
→ User aware content filtration
Explaining recommendation according to situation
→ Multidimensional explanations (How to relate user's surroundings?)
S3
→ Situation aware explanations
경희대학교
KYUNG HEE UNIVERSITY

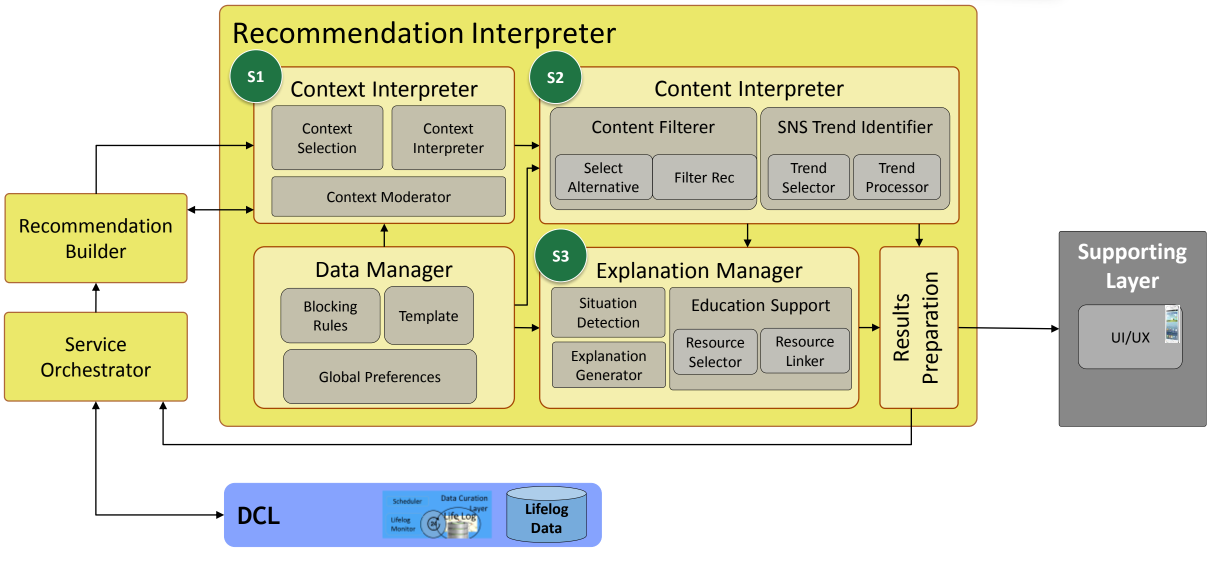
Conceptual View

Deliver the recommendation according to present context and preferences

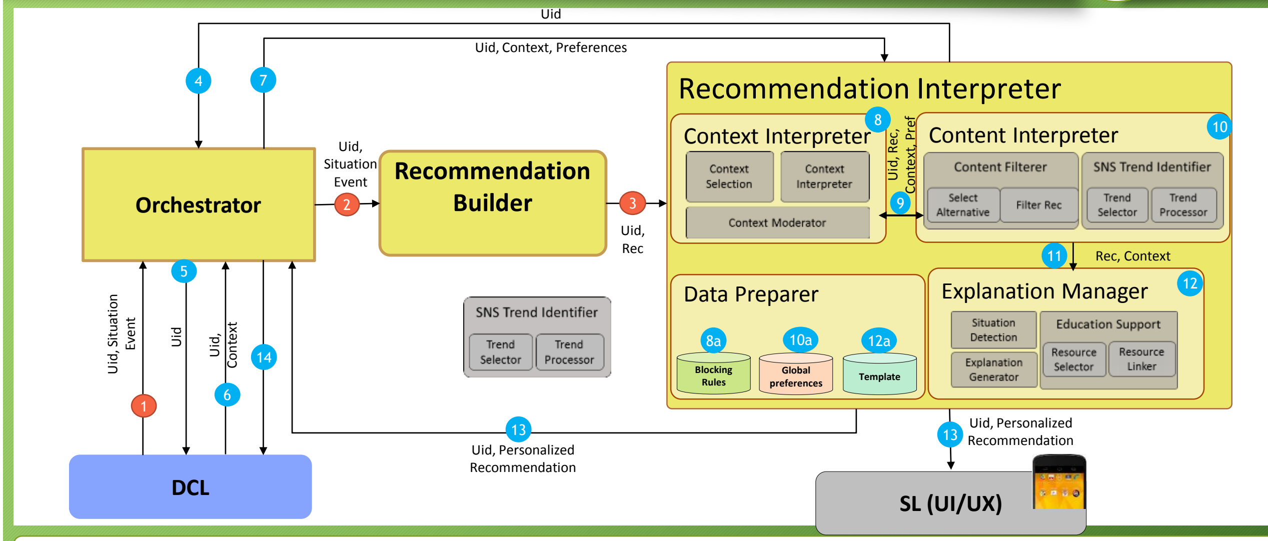


경희대학교
KYUNG HEE UNIVERSITY

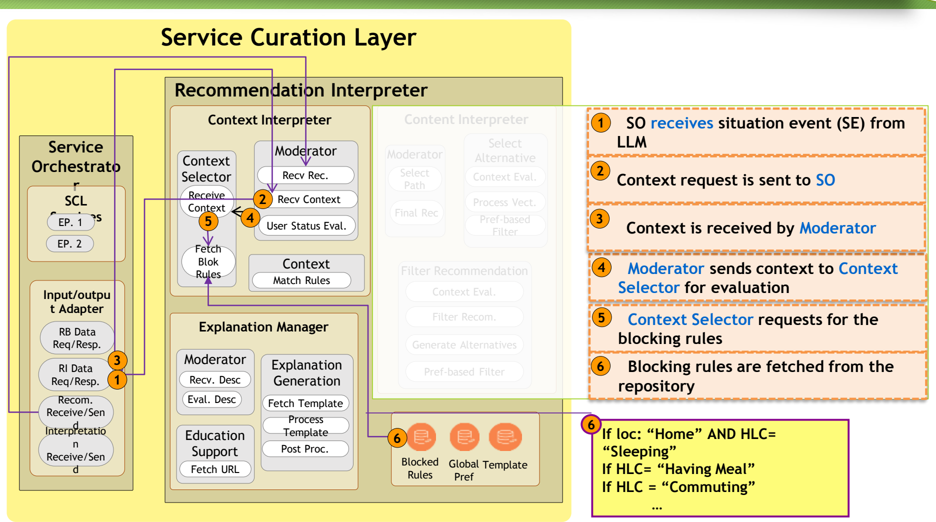
Component Architecture

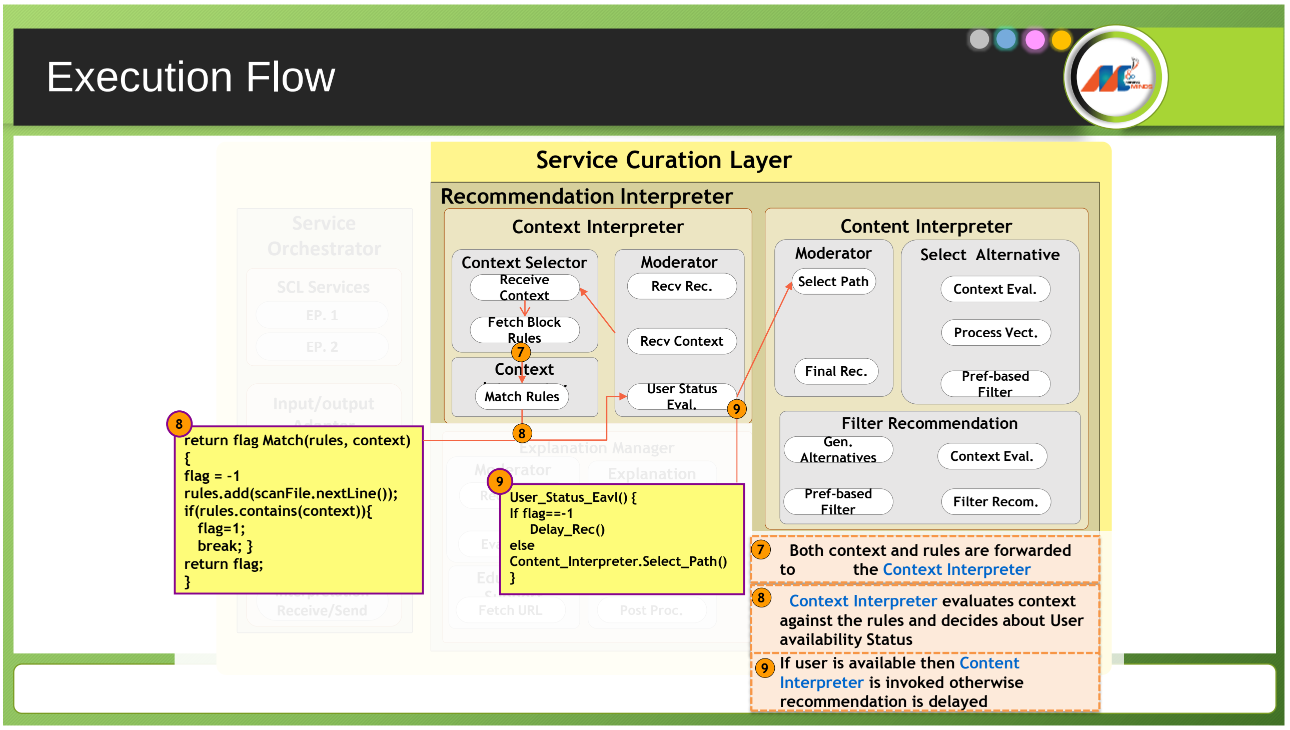


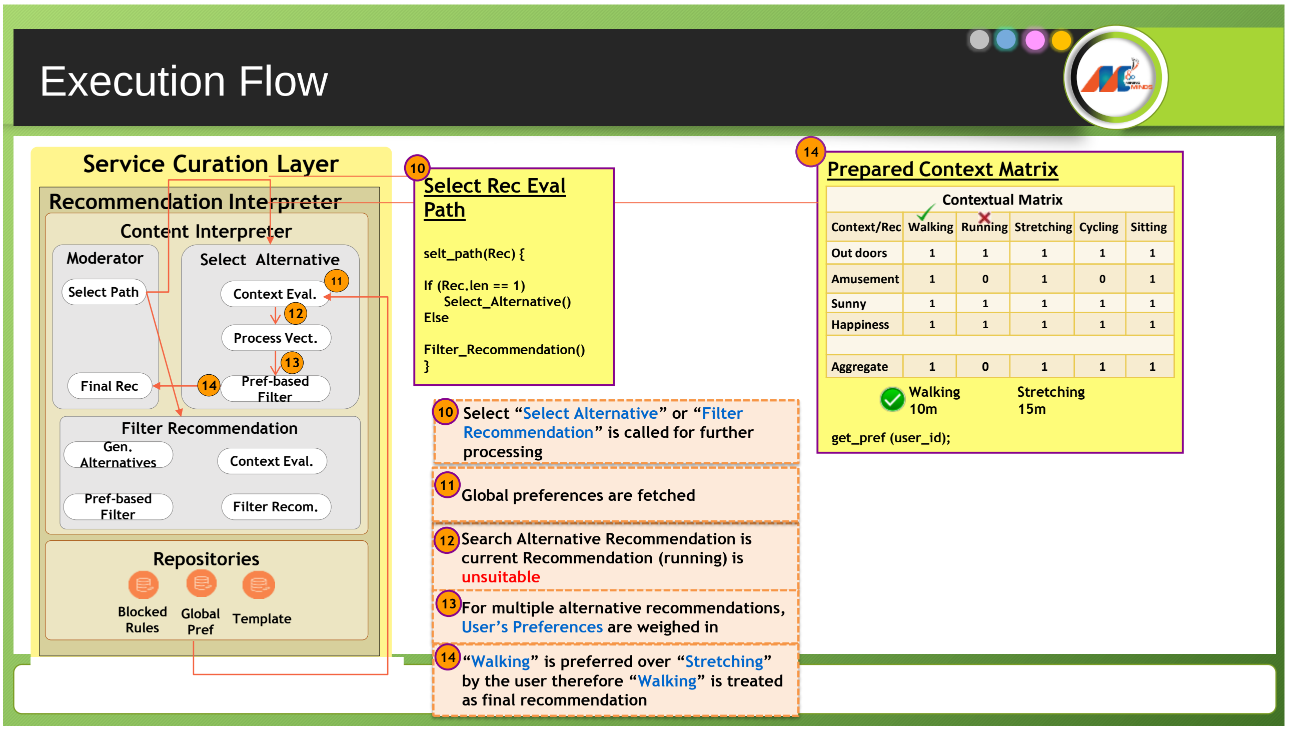
Complete Communication Workflow of Interpreter

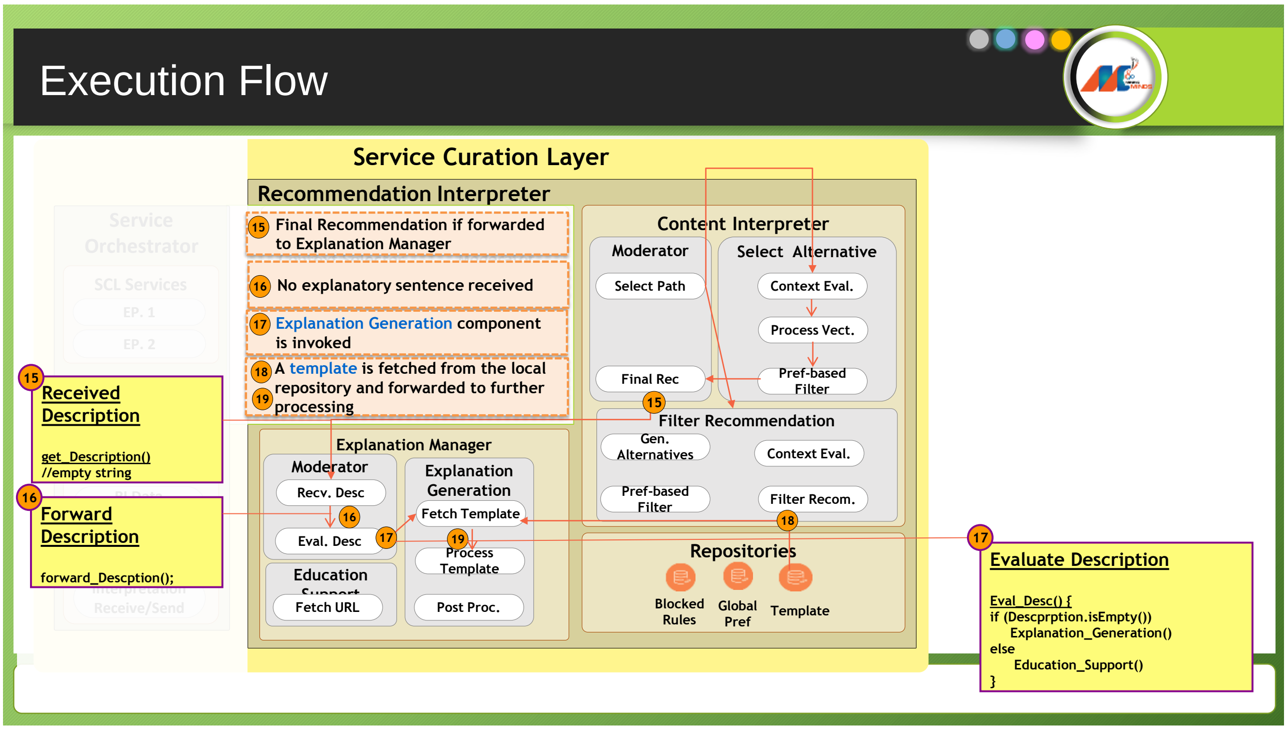


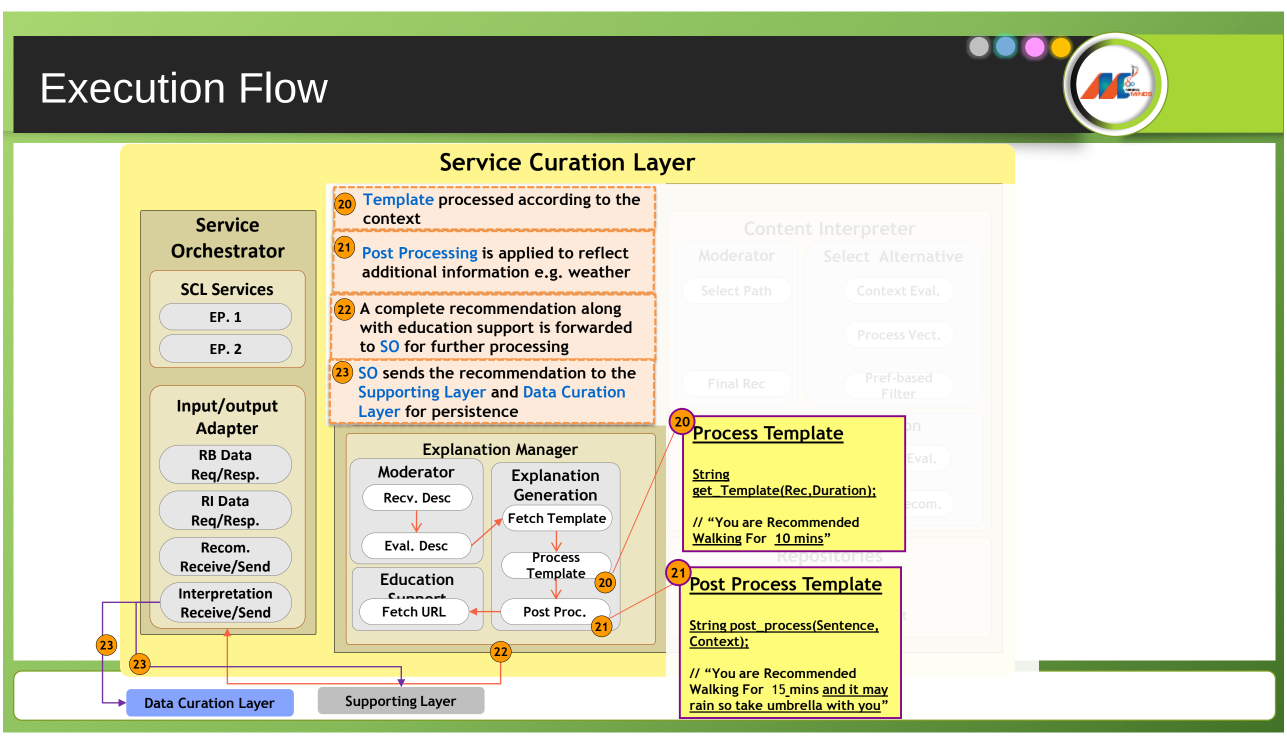
Execution Flow



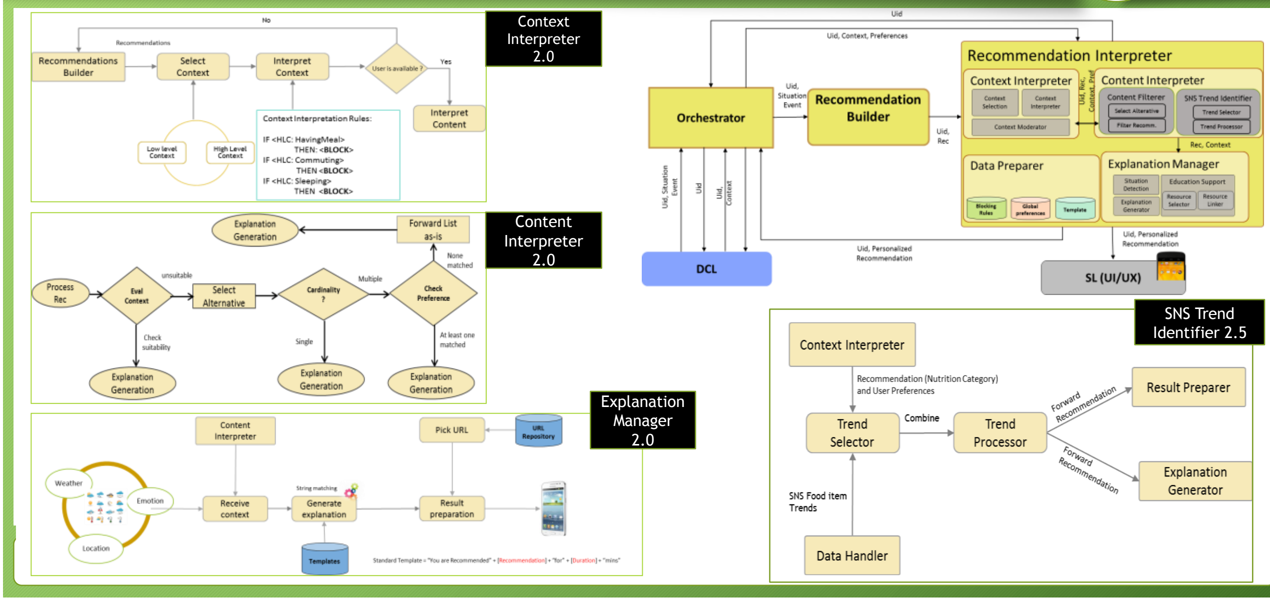

Execution Flow
Service Curation Layer
Recommendation Interpreter
Context Interpreter
Context Selector
Receive Context
Fetch Block Rules
7
Context
Match Rules
Moderator
Recv Rec.
Recv Context
User Status Eval.
9
8
Content Interpreter
Moderator
Select Path
Final Rec.
Select Alternative
Context Eval.
Process Vect.
Pref-based Filter
Filter Recommendation
Gen. Alternatives
Context Eval.
Pref-based Filter
Filter Recom.
8
return flag Match(rules, context)
{
flag = -1
rules.add(scanFile.nextLine());
if(rules.contains(context)){
flag=1;
break; }
return flag;
}
9
User_Status_Eavl() {
If flag==-1
Delay_Rec()
else
Content_Interpreter.Select_Path()
}
7 Both context and rules are forwarded to the Context Interpreter
8 Context Interpreter evaluates context against the rules and decides about User availability Status
9 If user is available then Content Interpreter is invoked otherwise recommendation is delayed


Execution Flow
Service Curation Layer
Recommendation Interpreter
Content Interpreter
Moderator
Select Path
Final Rec
Select Alternative
Context Eval.
11
12
Process Vect.
13
14
Pref-based Filter
Filter Recommendation
Gen. Alternatives
Context Eval.
Pref-based Filter
Filter Recom.
Repositories
Blocked Rules
Global Pref
Template
10
Select Rec Eval Path
selt_path(Rec) {
If (Rec.len == 1)
Select_Alternative()
Else
Filter_Recommendation()
}
10 Select "Select Alternative" or "Filter Recommendation" is called for further processing
11 Global preferences are fetched
12 Search Alternative Recommendation is current Recommendation (running) is unsuitable
13 For multiple alternative recommendations, User's Preferences are weighed in
14 "Walking" is preferred over "Stretching" by the user therefore "Walking" is treated as final recommendation
14
Prepared Context Matrix
Contextual Matrix
Context/Rec | Walking | Running | Stretching | Cycling | Sitting
Out doors | 1 | 1 | 1 | 1 | 1
Amusement | 1 | 0 | 1 | 0 | 1
Sunny | 1 | 1 | 1 | 1 | 1
Happiness | 1 | 1 | 1 | 1 | 1
Aggregate | 1 | 0 | 1 | 1 | 1
Walking 10m
Stretching 15m
get_pref (user_id);

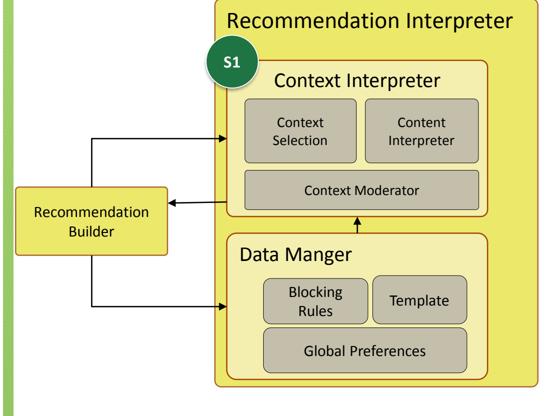

Execution Flow
Service Curation Layer
Recommendation Interpreter
15 Final Recommendation if forwarded to Explanation Manager
16 No explanatory sentence received
17 Explanation Generation component is invoked
18 A template is fetched from the local repository and forwarded to further
19 processing
Content Interpreter
Moderator
Select Path
Final Rec
Select Alternative
Context Eval.
Process Vect.
Pref-based Filter
Filter Recommendation
Gen. Alternatives
Context Eval.
Pref-based Filter
Filter Recom.
Explanation Manager
Moderator
Recv. Desc
Eval. Desc
Explanation Generation
Fetch Template
Process Template
Education Support
Fetch URL
Post Proc.
Repositories
Blocked Rules
Global Pref
Template
15 Received Description
get_Description()
//empty string
16 Forward Description
forward_Descption();
17 Evaluate Description
Eval_Desc() {
if (Descprption.isEmpty())
Explanation_Generation()
else
Education_Support()
}


Execution Flow
Service Curation Layer
20 Template processed according to the context
21 Post Processing is applied to reflect additional information e.g. weather
22 A complete recommendation along with education support is forwarded to SO for further processing
23 SO sends the recommendation to the Supporting Layer and Data Curation Layer for persistence
Service Orchestrator
SCL Services
EP. 1
EP. 2
Input/output Adapter
RB Data Req/Resp.
RI Data Req/Resp.
Recom. Receive/Send
Interpretation Receive/Send
Explanation Manager
Moderator
Recv. Desc
Eval. Desc
Explanation Generation
Fetch Template
Process Template
Post Proc.
Education Support
Fetch URL
Data Curation Layer
Supporting Layer
20 Process Template
String get_Template(Rec,Duration);
// "You are Recommended Walking For 10 mins"
21 Post Process Template
String post_process(Sentence, Context);
// "You are Recommended Walking For 15 mins and it may rain so take umbrella with you"

Recommendation Interpreter– Main Components

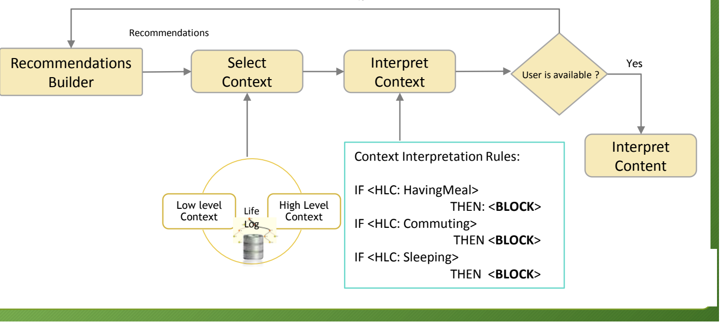


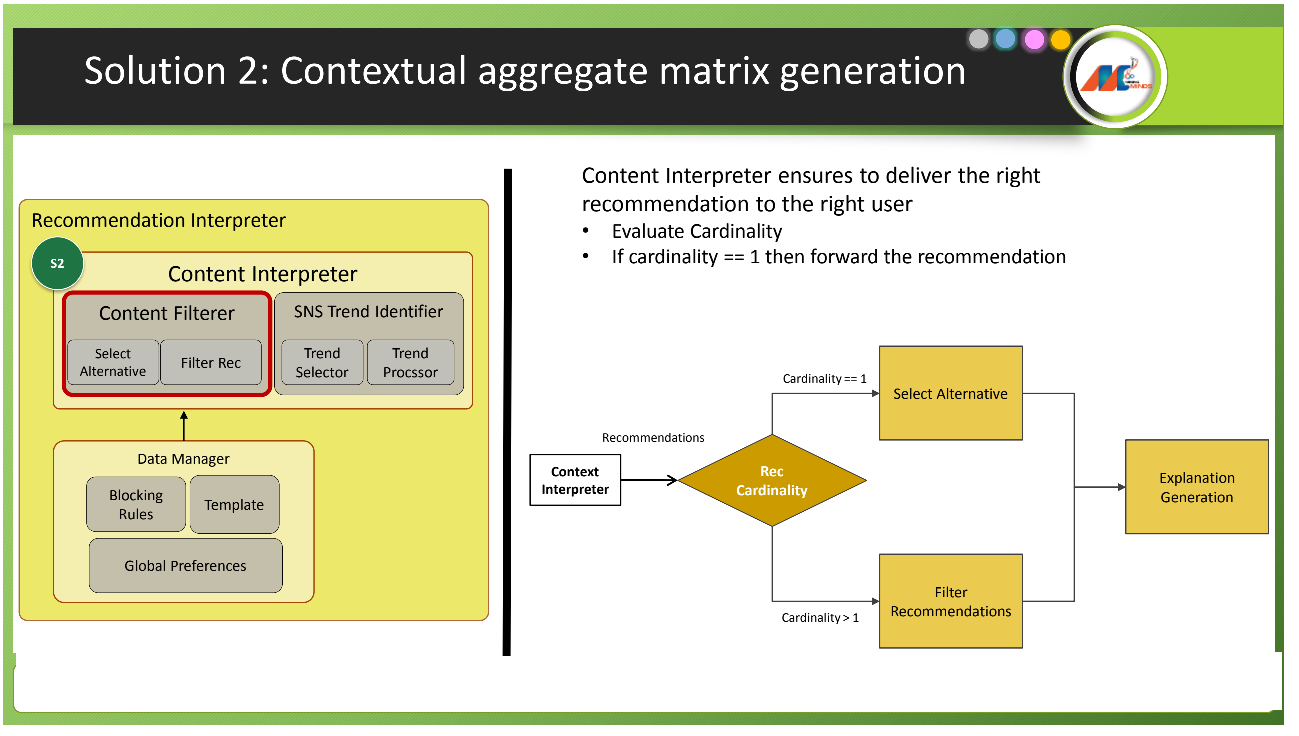
Solution 1: User availability rule interpretation

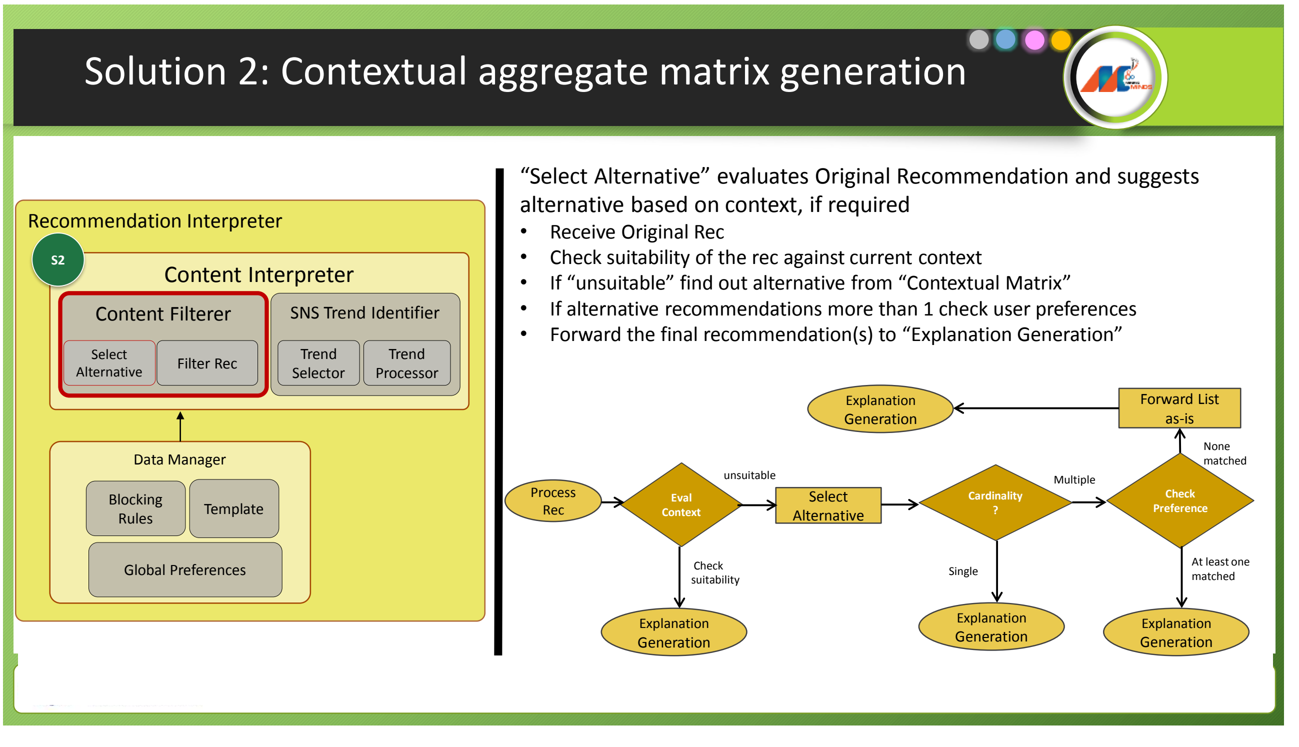


Context Interpreter ensures to deliver the recommendation at right time

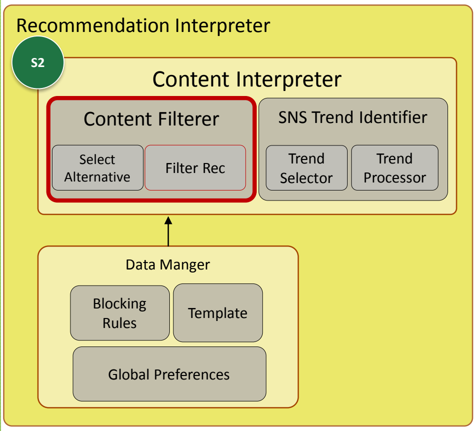
1. selects context one by one from lifelog
2. Interpret the selected context
3. If user is available, the recommendation are forwarded for next component
4. Else it inform the recommendation builder about the reason of not delivering the recommendation




Solution 2: Contextual aggregate matrix generation
Recommendation Interpreter
S2
Content Interpreter
Content Filterer
Select Alternative
Filter Rec
SNS Trend Identifier
Trend Selector
Trend Procssor
Data Manager
Blocking Rules
Template
Global Preferences
Content Interpreter ensures to deliver the right recommendation to the right user
• Evaluate Cardinality
• If cardinality == 1 then forward the recommendation
Context Interpreter
Recommendations
Rec Cardinality
Cardinality == 1
Select Alternative
Cardinality > 1
Filter Recommendations
Explanation Generation

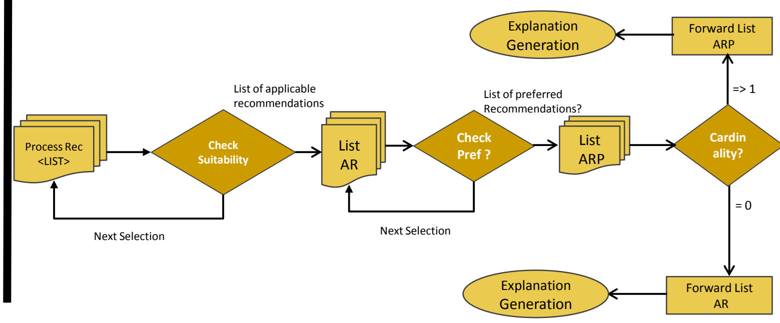

Solution 2: Contextual aggregate matrix generation
Recommendation Interpreter
S2
Content Interpreter
Content Filterer
Select Alternative
Filter Rec
SNS Trend Identifier
Trend Selector
Trend Processor
Data Manager
Blocking Rules
Template
Global Preferences
"Select Alternative" evaluates Original Recommendation and suggests alternative based on context, if required
• Receive Original Rec
• Check suitability of the rec against current context
• If "unsuitable" find out alternative from "Contextual Matrix"
• If alternative recommendations more than 1 check user preferences
• Forward the final recommendation(s) to "Explanation Generation"
Explanation Generation
Forward List as-is
None matched
Process Rec
Eval Context
unsuitable
Select Alternative
Cardinality ?
Multiple
Check Preference
Check suitability
Single
At least one matched
Explanation Generation
Explanation Generation
Explanation Generation

Solution 2: Contextual aggregate matrix generation

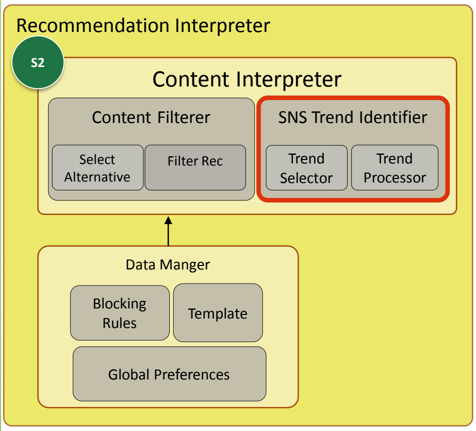


"Filter Rec" is evoked when multiple recommendations are received

- Receive list of recommendations
- Check suitability of each recommendation against the current context
- Store the applicable recommendations in AR_List
- Check user preferences
- Forward the final recommendation(s) to "Explanation Generation"

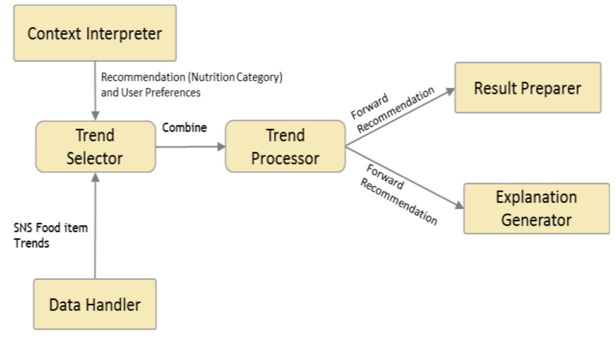


Solution 2: SNS Trend Identifier

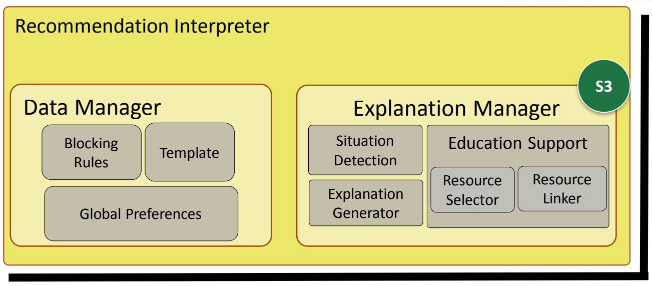


Food is recommended based on required nutrition

- Receiver nutrient category recommendation and user preferences
- Receiver latest food item trends from SNS
- Combine both aforementioned information
- Check user preferences
- Forward recommended food items to Result Preparer and Explanation Generation

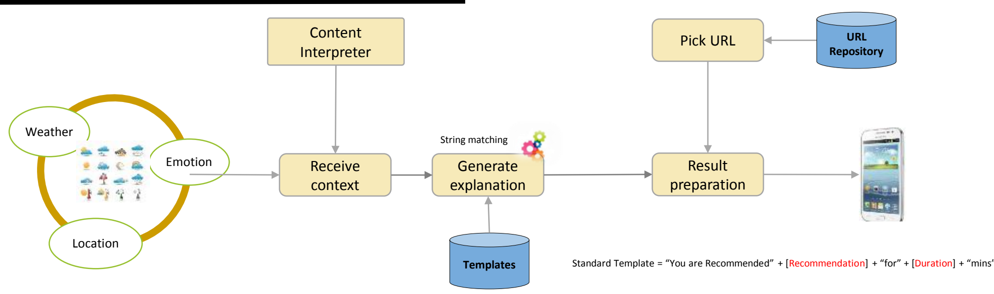


Solution 3: Template-based explanation



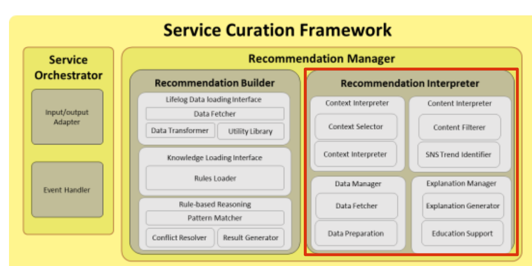
Explanation Manager explains the recommendation according to the situation

- Receive context
- Evaluate explanation requirement
- Select template
- Get recommendation specific URL
- Attached the link to resources
- Prepare the results



Uniqueness and Contributions

- The ability to make context-aware recommendations
  - Context-aware user Interruptibility
  - Situation based recommendation adaptability
  - SNS-based nutritious food recommendation
- Recommendation enrichment by embedding explanatory and educational nuggets
  - Explanatory note embedding with the recommendation
  - Audio-visual aids for recommendation adaptability

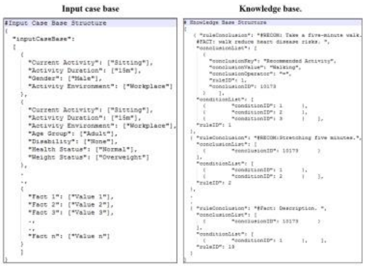


Evaluation Environment

- Evaluation Matrix

Criteria	Explanation
Execution Time	• Average execution time of the knowledge-base reasoning
Accuracy	• Accuracy of the knowledge-base reasoning • Accuracy of the recommendation interpreter

- Reasoner Evaluation Data Input cases



- Recommendation Interpreter Evaluation Questionnaire

Questionnaire Snapshot

Name:

Recommendation Evaluation

Guidelines: You are given physical activity based recommendations by two recommender systems i.e. System 1 (S1) and System 2 (S2). You are requested to assume yourself in a given context; in terms of location, high level context, weather and emotion.
Please tick (✓) mark each suitable recommendation of System 2. You may tick 1 or more recommendations of **System 2**. Cross (✕) those recommendations of **System 2** which are not suitable in the given context.
Note: Recommendations labeled "Not-to-Interrupt" indicate that the respective system decided to withhold the recommendation for that particular user's scenario.

No.	System 1	User's Context (Location, High Level Context, Weather, Emotion)	System 2
1	Running	Office, Office Work, Sunny, Neutral	Walking or Stretching
2	Stretching	Office, Inactivity, Sunny, Neutral	Stretching
3	Cycling	Home, House Work, Windy, Happiness	Walking or Stretching
4	Walking	Home, Inactivity, Rainy, Neutral	Walking
5	Walking	Home, Having Meal, Cloudy, Sadness	Not-to-Interrupt (because HLC context)

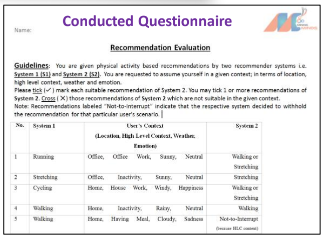
Accuracy Evaluation for Recommendation Interpreter

- <u>Experimental Setup</u>
  - Standalone system for RI
  - Questionnaire items: 40
  - Number of participants: 40
  - Survey conducted with the following population characteristics

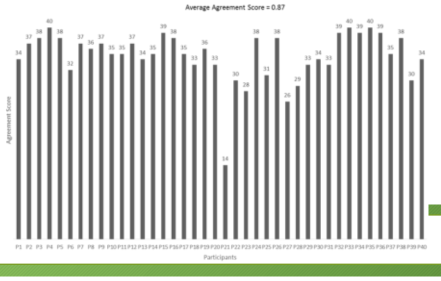
Ethnicity	Gender Male	Gender Female	Total
South Korea	9	1	10
Pakistan	8	1	9
Vietnam	3	2	5
India	1	1	2
China	2	0	2
Spain	1	1	2
Uzbekistan	0	2	2
Yemen	2	0	2
Ecuador	0	2	2
Bangladesh	4	0	4
Total	30	10	40

Sample table for recording questionnaire results

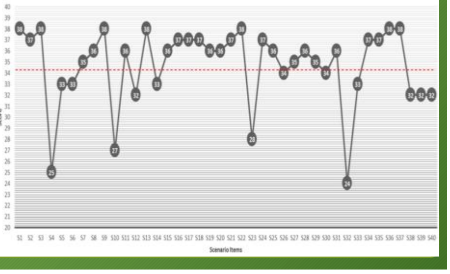
Scenario Items	P1	P2	…	P39	P40
1	+	+	…	-	+
2	+	+	…	+	+
3	+	+	…	+	+
⋮	⋮	⋮	…	⋮	⋮
38	+	+	…	-	+
39	+	+	…	+	+
40	+	+	…	+	+



<u>Result Summary Experiment 1</u> (Participant-Agreement)
Based on meta-accuracy scores the proposed system achieved 87% accuracy



<u>Result Summary Experiment 2</u> (Item-Participant score)
24 Scenarios (out of 40) achieved favorable results



Summary

- RI performs contextual information processing for situation-aware recommendation by deciding on factors such as user's interruptibility and contextual suitability of the recommendation
- RI enriches the contents of the generated recommendation by embedding audio-visual aids and relevant nuggets of contextual information in terms of current weather conditions, food trends and user's emotional state
- RI provides following features:
  - Delivering recommendation at right time --- user interruptibility << Context Interpreter>>
  - Delivering preference-in-context personalized recommendation << Content Interpreter>>
  - Delivering contextually explained recommendation <<Explanation Manager>>

Summary

- SCL handles dynamically both push and pull models for situation-aware recommendation generation
- SCL provides reliable recommendations using rule-based reasoning with forward chaining mechanism
- SCL provides a flexible reasoning framework with loosely coupled data and knowledge that supports diverse services
- SCL performs contextual information processing for situation-aware recommendation by deciding on factors such as user's interruptibility and contextual suitability of the recommendation
- SCL enriches the contents of the generated recommendation by embedding audio-visual aids and relevant nuggets of contextual information in terms of current weather conditions, food trends and user's emotional state

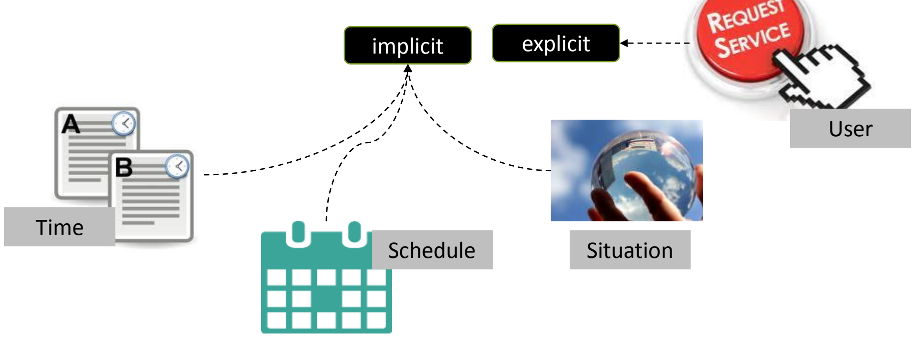
경희대학교 KYUNG HEE UNIVERSITY

Mining Minds

SERVICE ORCHESTRATOR

Introduction

- Orchestrator enable communication of SCL with other layers and user application
- It allows implicit or explicit communications

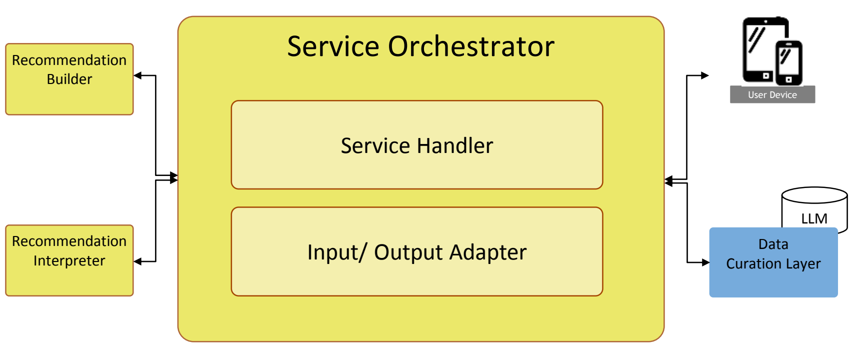


Motivation

- A common gateway of communication
- Communication abstraction for intra-layer modules
- Provide guidance to intra-layer logical modules fulfilling service requirements

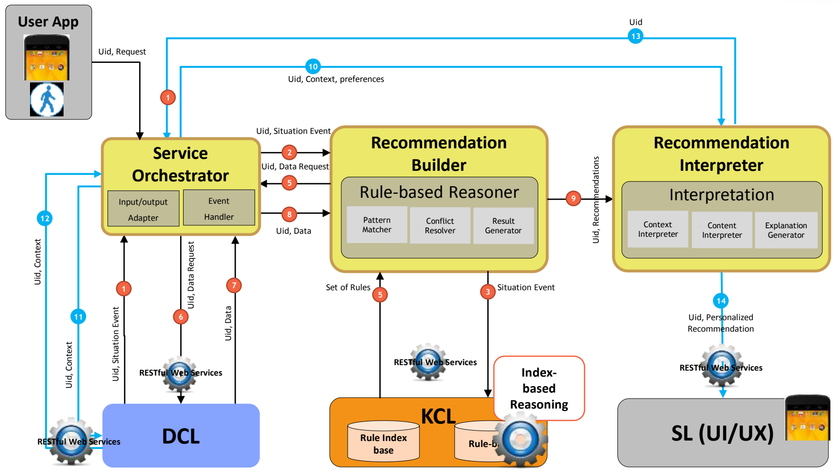
경희대학교
KYUNG HEE UNIVERSITY

Service Orchestrator - Architecture

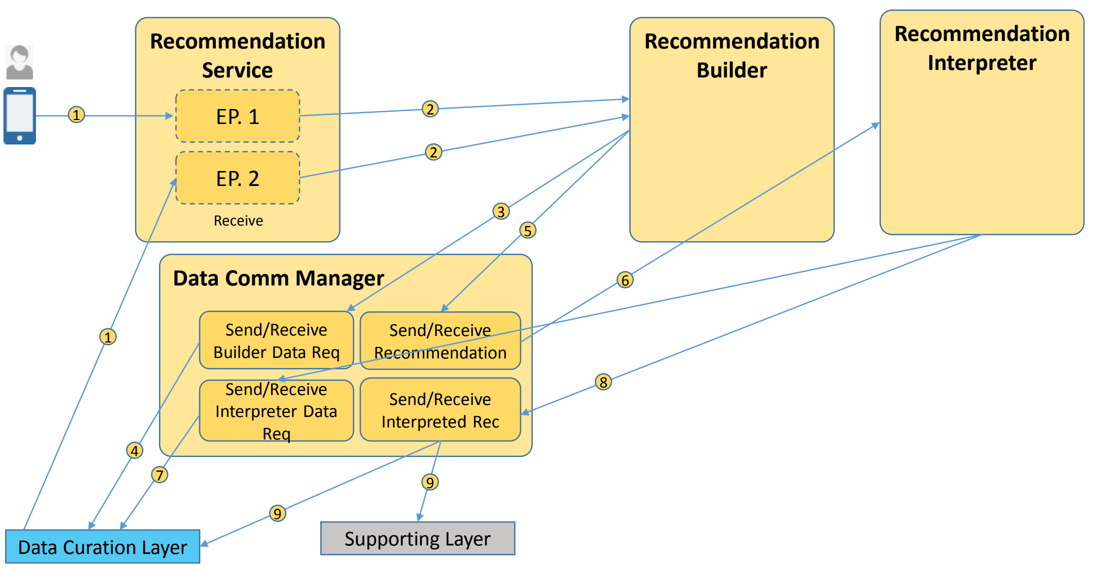


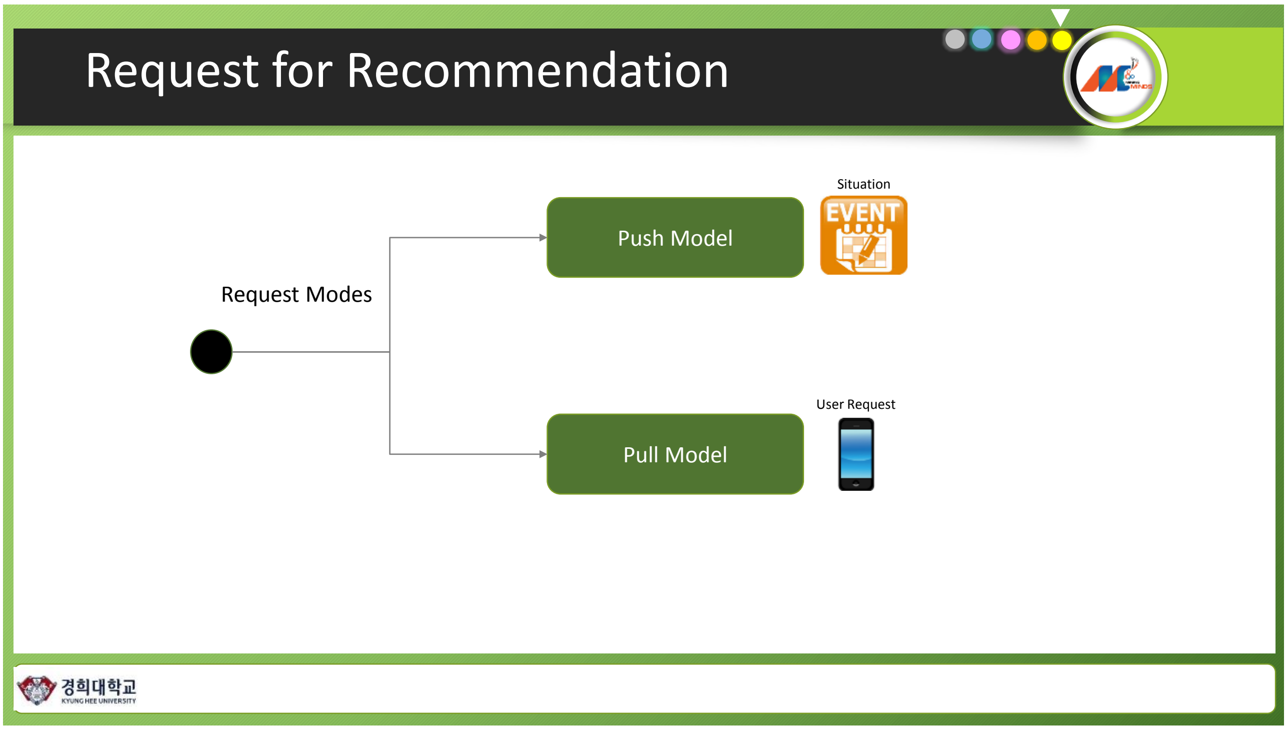
경희대학교
KYUNG HEE UNIVERSITY

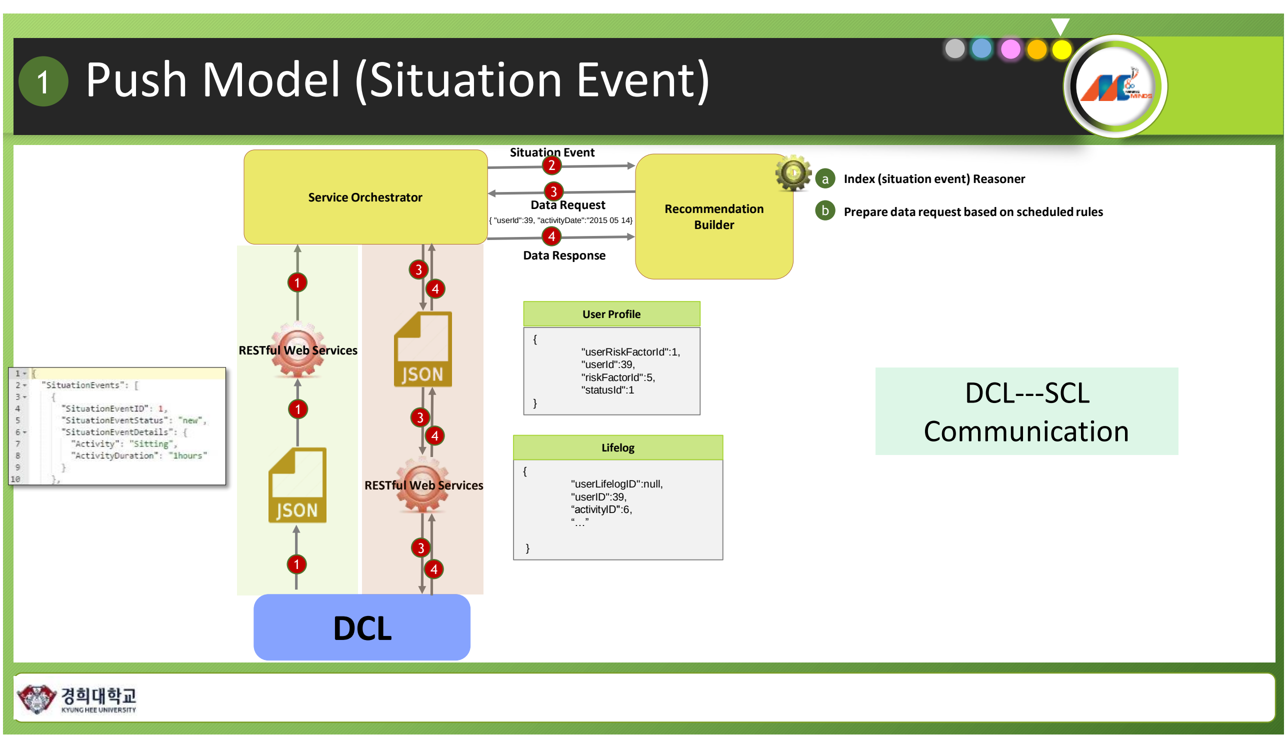
SCL Communication- Service Orchestrator



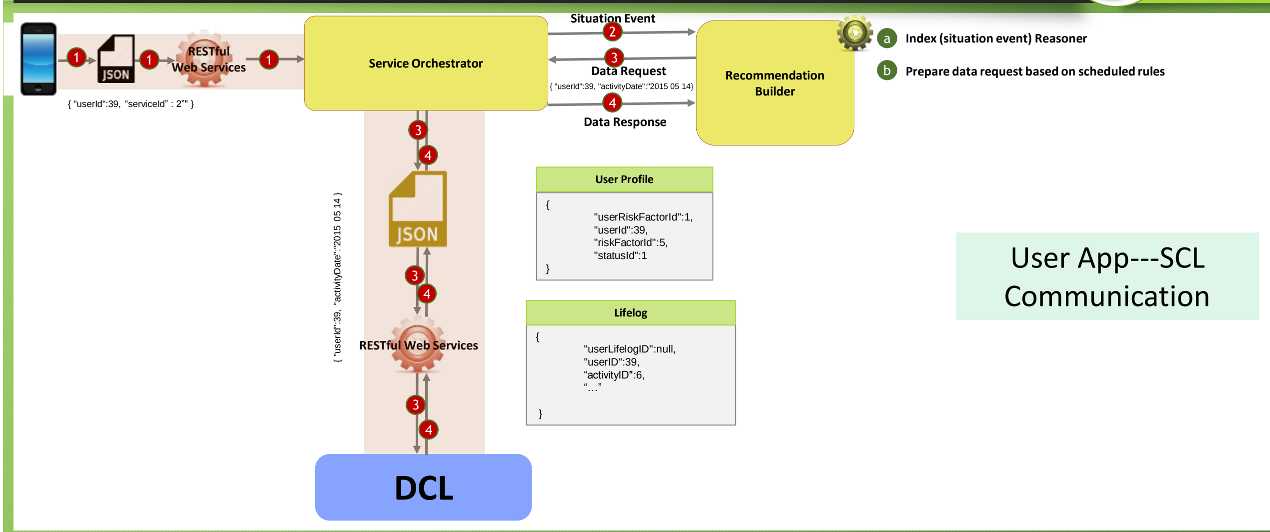
Service Orchestrator




Request for Recommendation
Situation
Push Model
Request Modes
User Request
Pull Model
경희대학교
KYUNG HEE UNIVERSITY


1
Push Model (Situation Event)
Situation Event
Service Orchestrator
Data Request
Data Response
Recommendation Builder
a Index (situation event) Reasoner
b Prepare data request based on scheduled rules
RESTful Web Services
JSON
RESTful Web Services
JSON
"SituationEvents": [
{
"SituationEventID": 1,
"SituationEventStatus": "new",
"SituationEventDetails": {
"Activity": "Sitting",
"ActivityDuration": "1hours"
User Profile
{
"userRiskFactorId":1,
"userId":39,
"riskFactorId":5,
"statusId":1
}
Lifelog
{
"userLifelogID":null,
"userID":39,
"activityID":6,
"..."
}
DCL---SCL Communication
DCL
경희대학교
KYUNG HEE UNIVERSITY

1 Pull Model (User Request)



경희대학교 KYUNG HEE UNIVERSITY

Contributions

- Handling the situation based on dynamically generated events from DCL
- Handling user request explicitly made for the recommendation
- Handles the communication of data required for SCL services

경희대학교 KYUNG HEE UNIVERSITY

Supporting Layer (SL)

경희대학교 KYUNG HEE UNIVERSITY KITECH 한국생산기술연구원 POSTECH 포항공과대학교 UTAS 녹십자헬스케어 TAPACROSS U2 system User & Usability system

Background

- Complex Large unstructured data sources are becoming a norm and extracting important and hidden information from it is becoming the need of today
- Additionally displaying them interactively with interesting graphics and adaptive UI is necessary.

- Analytics can identify abnormal behavior and different attributes related to the user through visualization and summarization to bring context.
- Adaptive User interface generation based on user information, context of user & device at run time

경희대학교 KYUNG HEE UNIVERSITY

Motivation for Supporting Layer

- Use of smartphones as primary devices
  - Personalized and adaptive user interface of the mobile phone application is very important in today's world.
  - User experience should be evolved constantly based on profile and performance metrics.
- Complex Large unstructured data sources
  - Complex datasets have hidden information which must be extracted for additional and complex decision making.
  - Visualization and trends are necessary for complex data to find abnormal information.

Hidden Information

경희대학교 KYUNG HEE UNIVERSITY

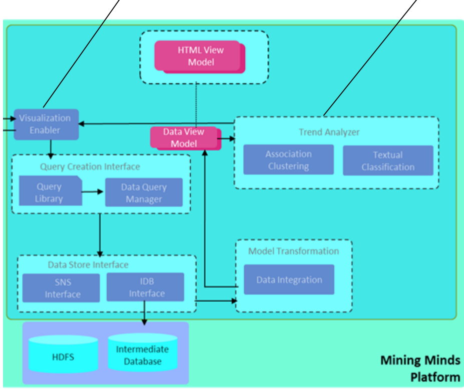
Supporting Layer as a Framework (SLF)

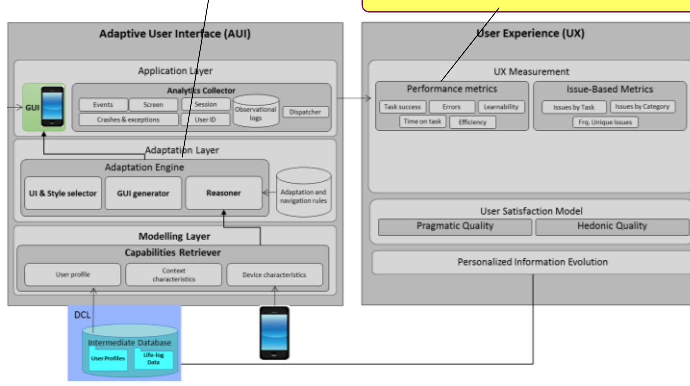
- Creating graphs and facts with respect to clustering
- Timeline and intuitive way to display SNS and trends

- Trend analysis extracts data facts
- Clustering for visualization

- Adaptation engine to create new rules and change existing rules with respect to user behavior and profile

- UX measurement in terms of task success, errors and time completion
- User satisfaction modelling with respect to performance/issues metrics

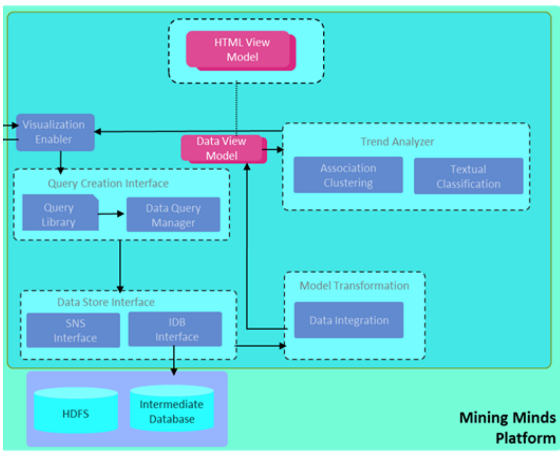




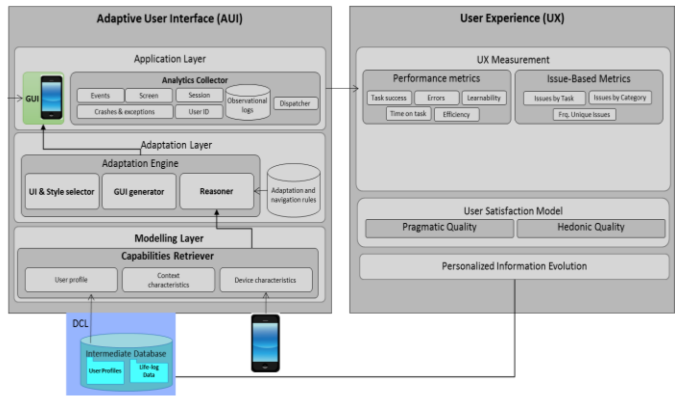
경희대학교 KYUNG HEE UNIVERSITY

SL Architecture

Descriptive Analytics



UI/UX



Related Work (Descriptive Analytics)

	Systems	Limitations
Clustering	• The automated extraction of interrelated data objects from ERP systems is discussed but without using a graph model and for the single analytical goal of process mining	• No parameter type classification • No distribution of data ranges and rendering information
	• Gradoop (Graph analytics on Hadoop) analyze graph data for business intelligence and social network analysis.	• Only graph analytics and focus on trends based on images • It stores graph formats only
Summarization	• Radoop is a big data analytics solution for Hadoop which computes the jobs on the cluster using ensemble learning	• It is based on complex machine learning techniques • Less information as it is being developed into a product.

Rudolf, Michael, et al. "SynopSys: large graph analytics in the SAP HANA database through summarization." *First International Workshop on Graph Data Management Experiences and Systems*. ACM, 2013.
Junghanns, Martin, et al. "GRADOOP: Scalable Graph Data Management and Analytics with Hadoop." *arXiv preprint arXiv:1506.00548* (2015).
radoop.eu

Limitations of Existing Work (Descriptive Analytics)

- No query library for mapping the queries, lack of which results in inefficient data retrieval in case of big data.
- Lack of Parameter filtration which makes it difficult to group and summarize the data.
- No model transformation for preserving semantics of data and integration of heterogeneous data

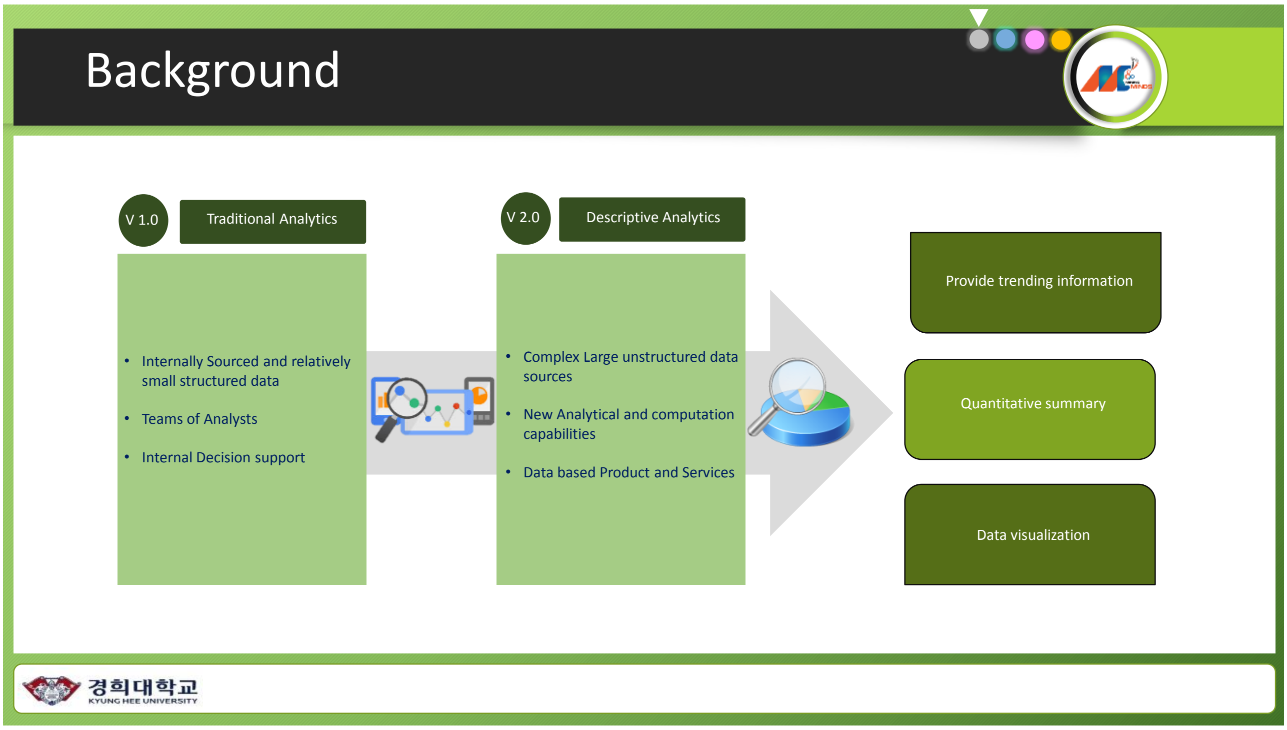
Overview of existing adaptive systems (UI/UX)

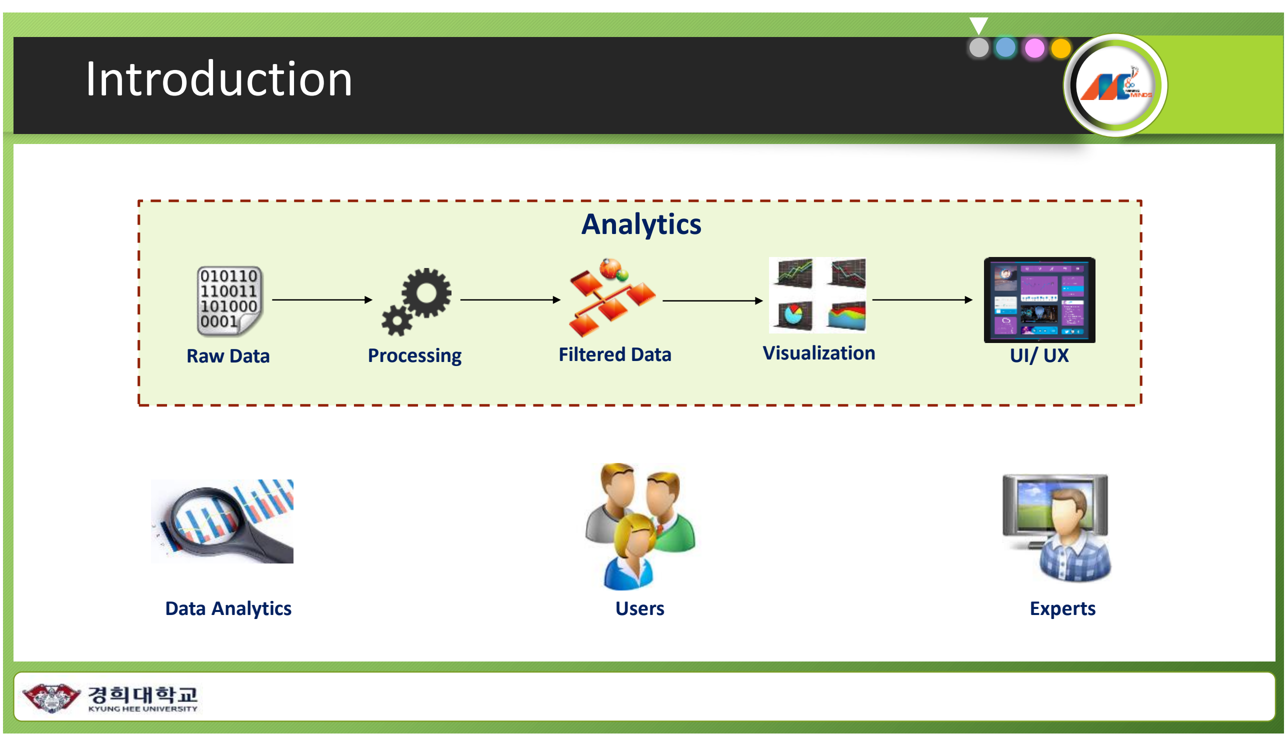
Existing Systems	Descriptions	Pros:	Cons:
Doppelgänger [20]	It was intended to produce a personalized, printed Newspaper for the user	• Sharing of user information among several applications. • Diverse types of sensors contribute to an extensive user model. • Unobtrusive user modelling.	• No systematic feedback mechanism
Flexcel [21]	It enhances Microsoft Excel by an adaptive User Interface	• Users have control over their own user profiles	• Some of the user dialogues for adaptability seem very complex
Lumière-Project[22]	It led to the later MS-Office assistant	• It combined the temporal reasoning and Bayesian user models in order to manage the uncertainty of recognizing user goals from a stream of user actions over time.	• It only focus on recognizing user goals in order to provide appropriate
Lifestyle Finder[25]	It gives the user suggestions for interesting websites	• User profiling and clustering is based on publically available demographic mass data	• Adaptation covers only the selection of content • User modelling covers aspects such as purchasing history, lifestyle characteristics and survey responses
Supple[26]	It is an application that adapts the display of objects considering window size and user preferences	• Run-time rendering of the user interface • Information about the user is collected by analyzing user tracking	• Adaptation focuses on layout and selection of appropriate controls and display elements • It does not address accessibility issues • It does not provide an authoring tool
MYUI [19]	It generates individualized user interfaces and performs adaptations to diverse user needs, devices and environmental conditions during run time.	• Toolkit: supports industrial developers and designers to easily create self-adaptive applications • Explicit and implicit data collection about user for user modeling • Run-time rendering of user interface.	• No feedback functionality • Manual setting for platform device category

Limitations of existing work (UI/UX)

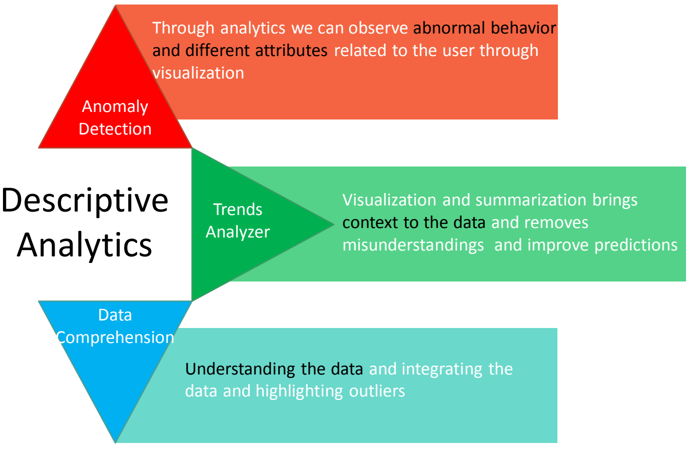
- The existing user models are not comprehensive
  - Lacks behavioral & physiological measurement for adaptive UI
  - Environmental variables
- Lack addressing accessibility issues
- No feedback functionality
- Lack of user experience for adaptive UI

Descriptive Analytics


Background
V 1.0
Traditional Analytics
Internally Sourced and relatively small structured data
Teams of Analysts
Internal Decision support
V 2.0
Descriptive Analytics
Complex Large unstructured data sources
New Analytical and computation capabilities
Data based Product and Services
Provide trending information
Quantitative summary
Data visualization
경희대학교
KYUNG HEE UNIVERSITY


Introduction
Analytics
010110
110011
101000
0001
Raw Data
Processing
Filtered Data
Visualization
UI/ UX
Data Analytics
Users
Experts
경희대학교
KYUNG HEE UNIVERSITY

Motivation



Workflow of Descriptive Analytics

Graph Data

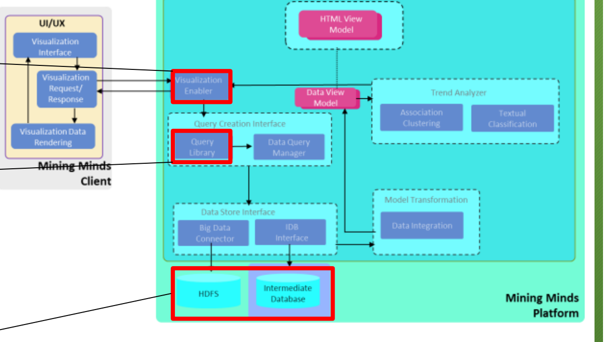
Id	X-axis	Y-axis	Multiple series	Graph
1	Yes	No	No	Pie
2	Yes	Yes	Yes	Line
3	Yes	Yes	Yes	Bar
4	Yes	Yes	No	Bubble

Query Library

Return Attribute	Conditional Attributes	Temporal Attribute
Food	Age	Days
Activity	Duration	DateTime
*.activity	uid	None
Activity	GPS/Location	Days

Data Store

uid	ActId	DateTime	Activity	Duration	accX	accY	accZ
2	2	20/08/1513: 00:03	Walking	5Min	1213.23 1123	1213.23 1123	1213.23 1123
29	2	20/08/1510: 00:03	Walking	45m24s	14334.1 23	14334.1 23	14334.1 23
30	4	20/08/1511: 00:03	Sitting	2hr	1232	1232	1232
23	2	21/08/1518: 00:03	Walking	10min	1213.23 1123	1213.23 1123	1213.23 1123
32	4	21/08/1512: 00:03	Sitting	2m24s	14334.1 23	14334.1 23	14334.1 23
30	3	21/08/1513: 00:03	running	4m	1232.84 1	1232.84 1	1232.84 1



Workflow of Descriptive Analytics

Graph Data

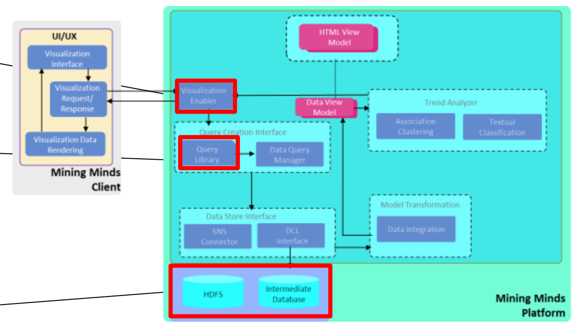
Id	X-axis	Y-axis	Multiple series	Graph
1	Yes	No	No	Pie
2	Yes	Yes	Yes	Line
3	Yes	Yes	Yes	Bar
4	Yes	Yes	No	Bubble

Query Library

Return Attribute	Conditional Attributes	Temporal Attribute
Food	Age	Days
Activity	Duration	DateTime
*.activity	uid	None
Activity	GPS/Location	Days

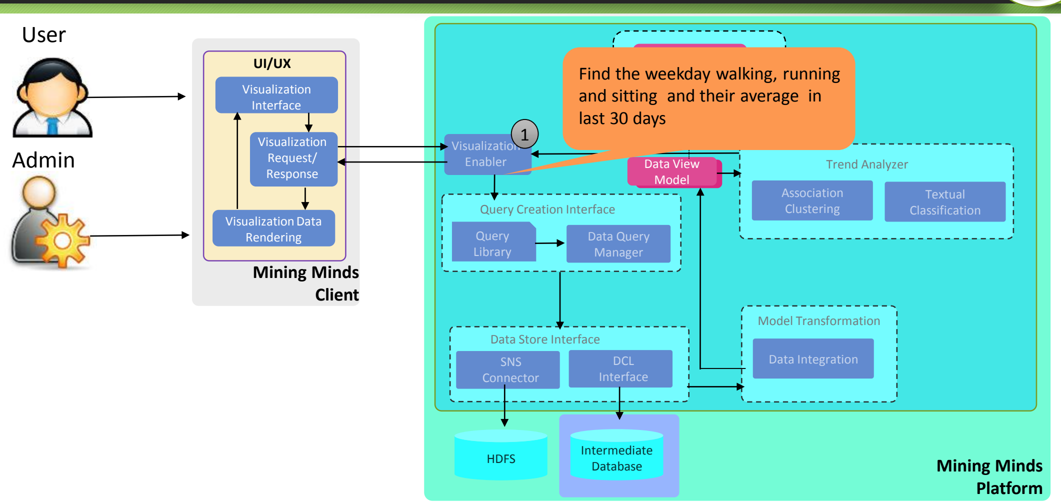
Data Store

uid	ActId	DateTime	Activity	Duration	accX	accY	accZ
2	2	20/08/1513: 00:03	Walking	5Min	1213.23 1123	1213.23 1123	1213.23 1123
29	2	20/08/1510: 00:03	Walking	45m24s	14334.1 23	14334.1 23	14334.1 23
30	4	20/08/1511: 00:03	Sitting	2hr	1232	1232	1232
23	2	21/08/1518: 00:03	Walking	10min	1213.23 1123	1213.23 1123	1213.23 1123
32	4	21/08/1512: 00:03	Sitting	2m24s	14334.1 23	14334.1 23	14334.1 23
30	3	21/08/1513: 00:03	running	4m	1232.84 1	1232.84 1	1232.84 1

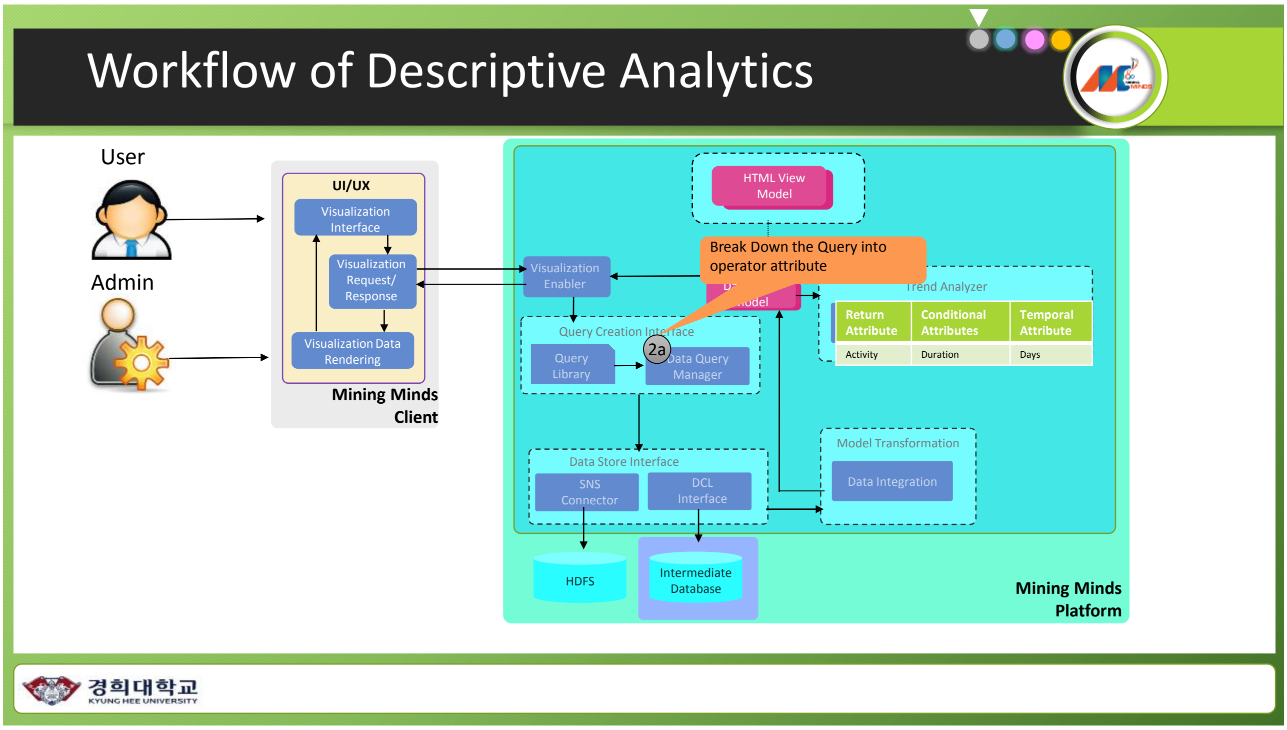


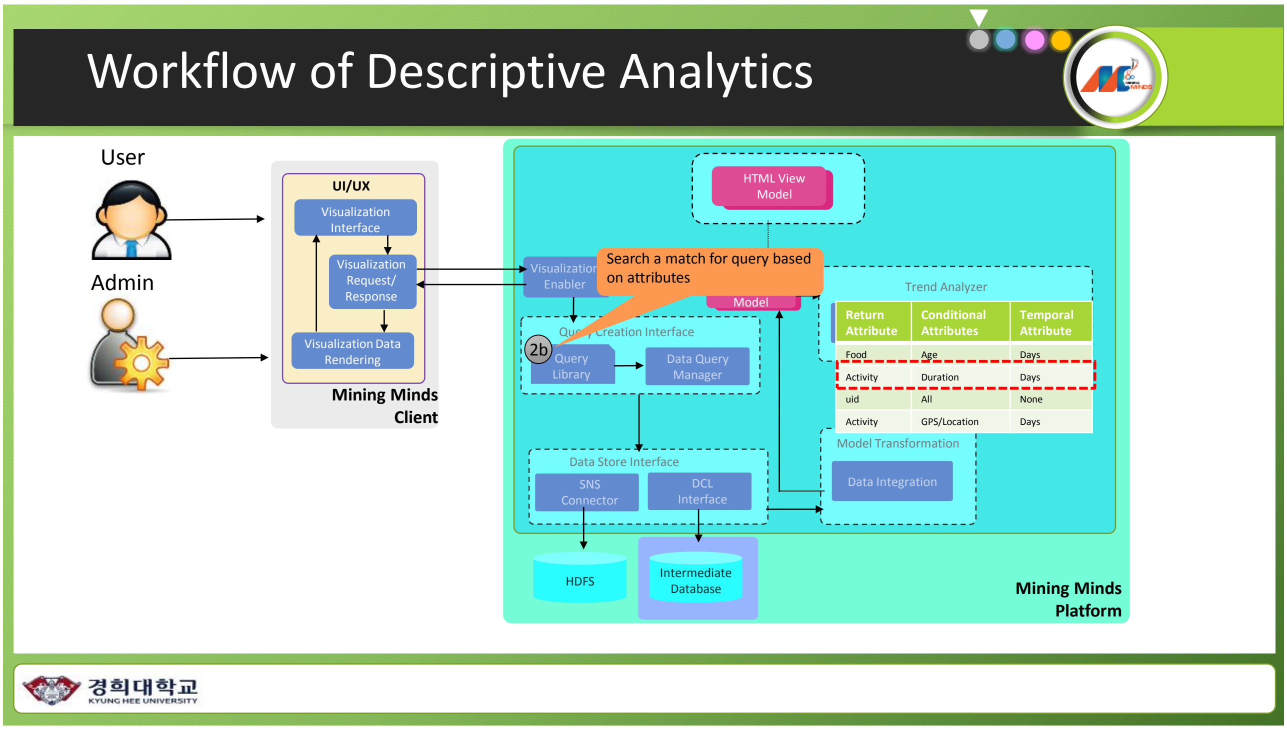
경희대학교
KYUNG HEE UNIVERSITY

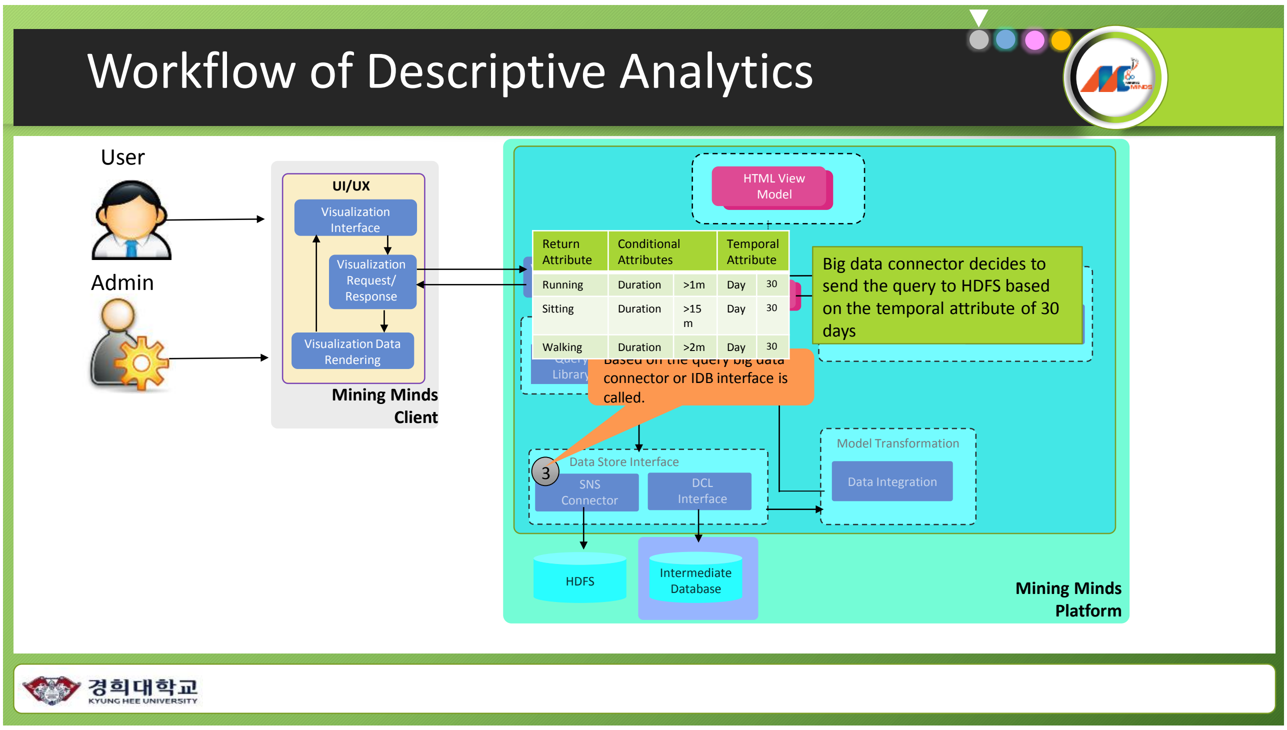
Workflow of Descriptive Analytics

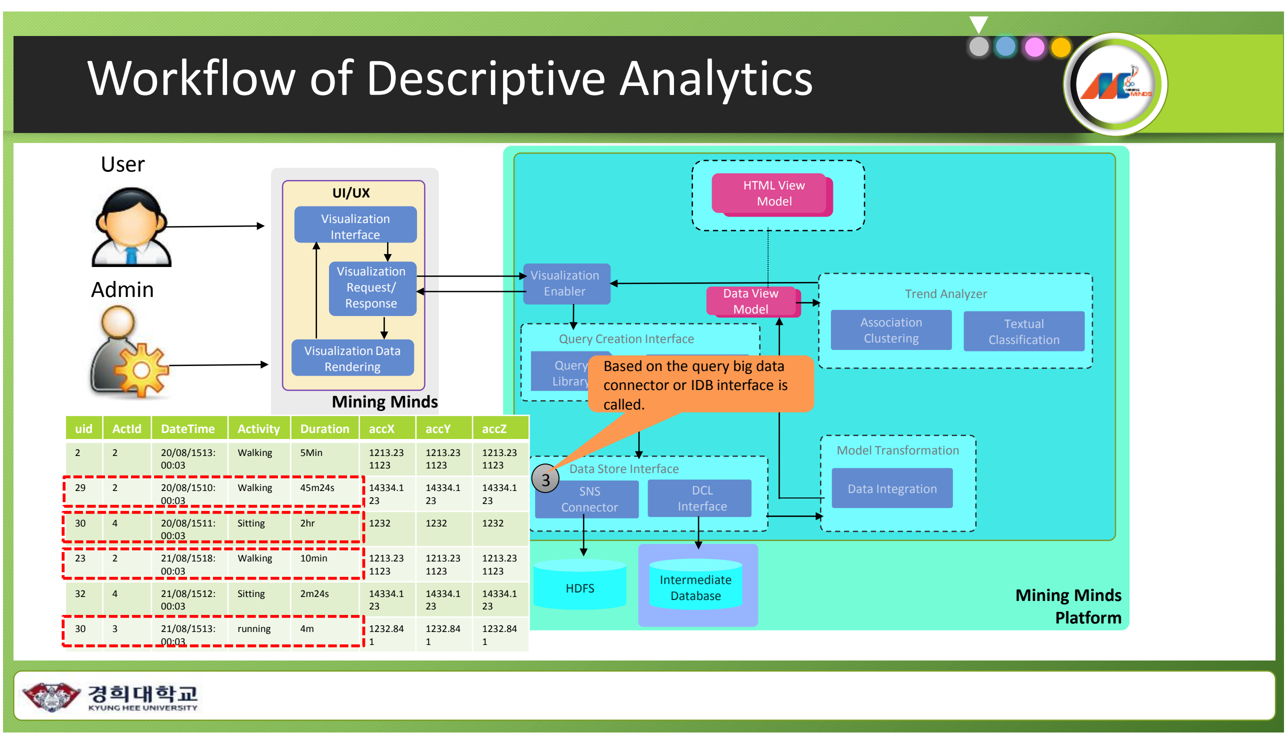


경희대학교
KYUNG HEE UNIVERSITY

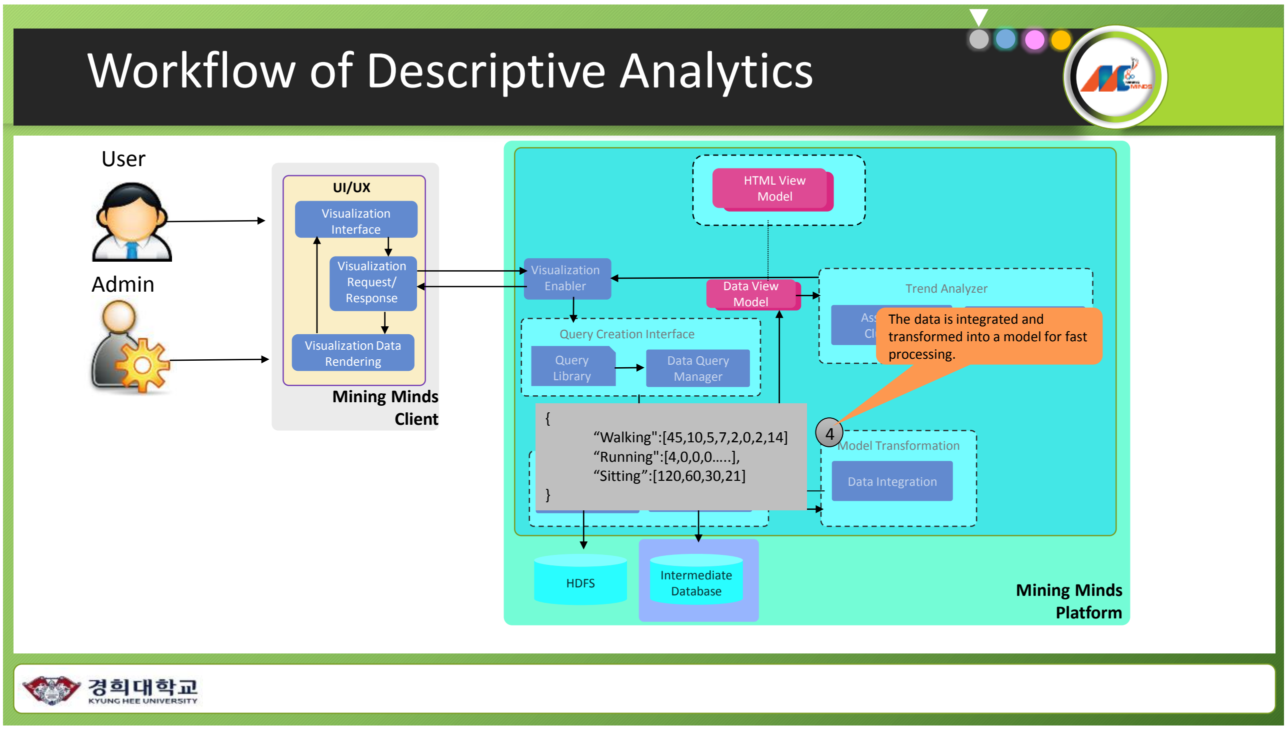

Workflow of Descriptive Analytics
User
Admin
UI/UX
Visualization Interface
Visualization Request/ Response
Visualization Data Rendering
Mining Minds Client
HTML View Model
Break Down the Query into operator attribute
Visualization Enabler
Trend Analyzer
Return Attribute
Conditional Attributes
Temporal Attribute
Activity
Duration
Days
Query Creation Interface
2a
Query Library
Data Query Manager
Model Transformation
Data Integration
Data Store Interface
SNS Connector
DCL Interface
HDFS
Intermediate Database
Mining Minds Platform
경희대학교
KYUNG HEE UNIVERSITY

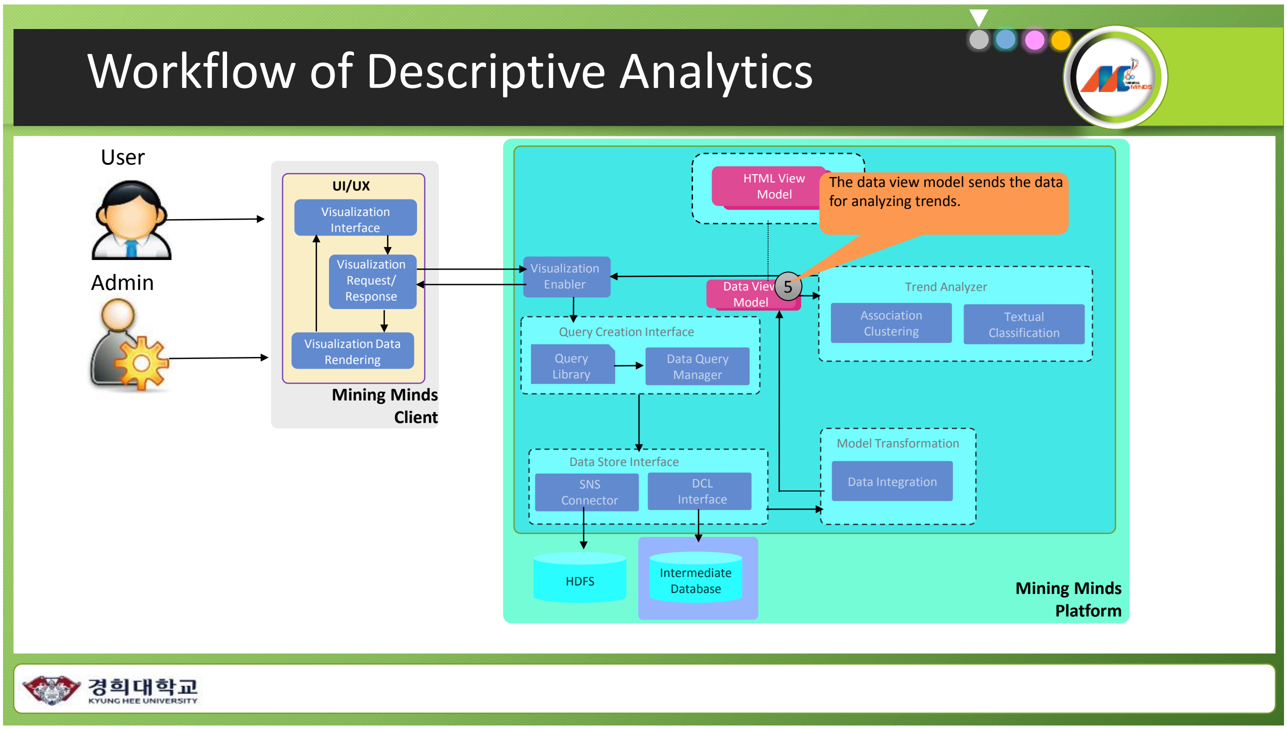

Workflow of Descriptive Analytics
User
Admin
UI/UX
Visualization Interface
Visualization Request/ Response
Visualization Data Rendering
Mining Minds Client
HTML View Model
Search a match for query based on attributes
Visualization Enabler
Model
Trend Analyzer
Return Attribute
Conditional Attributes
Temporal Attribute
Food
Age
Days
Activity
Duration
Days
uid
All
None
Activity
GPS/Location
Days
Query Creation Interface
2b
Query Library
Data Query Manager
Model Transformation
Data Integration
Data Store Interface
SNS Connector
DCL Interface
HDFS
Intermediate Database
Mining Minds Platform
경희대학교
KYUNG HEE UNIVERSITY

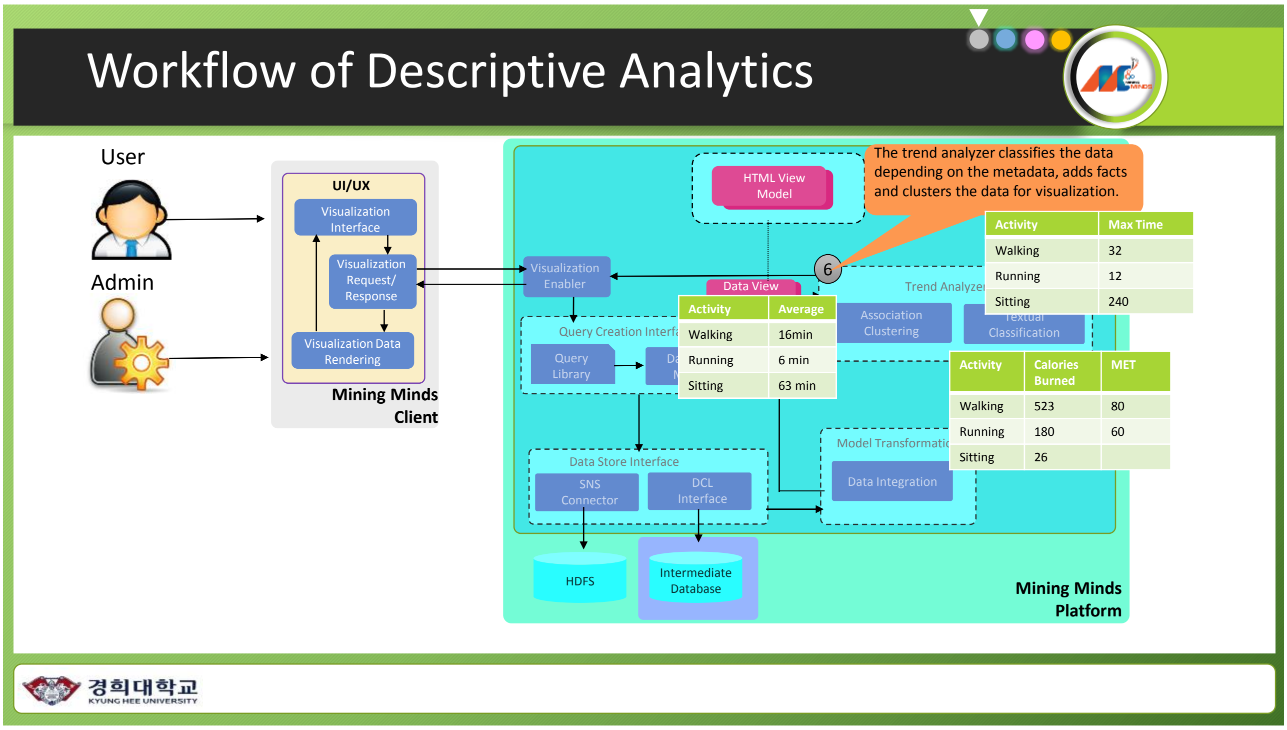




uid	ActId	DateTime	Activity	Duration	accX	accY	accZ
2	2	20/08/1513:00:03	Walking	5Min	1213.231123	1213.231123	1213.231123
29	2	20/08/1510:00:03	Walking	45m24s	14334.123	14334.123	14334.123
30	4	20/08/1511:00:03	Sitting	2hr	1232	1232	1232
23	2	21/08/1518:00:03	Walking	10min	1213.231123	1213.231123	1213.231123
32	4	21/08/1512:00:03	Sitting	2m24s	14334.123	14334.123	14334.123
30	3	21/08/1513:00:03	running	4m	1232.841	1232.841	1232.841


Workflow of Descriptive Analytics
User
Admin
UI/UX
Visualization Interface
Visualization Request/ Response
Visualization Data Rendering
Mining Minds Client
HTML View Model
Visualization Enabler
Data View Model
Trend Analyzer
The data is integrated and transformed into a model for fast processing.
Query Creation Interface
Query Library
Data Query Manager
{
"Walking":[45,10,5,7,2,0,2,14]
"Running":[4,0,0,0.....],
"Sitting":[120,60,30,21]
}
4
Model Transformation
Data Integration
HDFS
Intermediate Database
Mining Minds Platform
경희대학교
KYUNG HEE UNIVERSITY


Workflow of Descriptive Analytics
User
Admin
UI/UX
Visualization Interface
Visualization Request/ Response
Visualization Data Rendering
Mining Minds Client
HTML View Model
The data view model sends the data for analyzing trends.
Visualization Enabler
Data View Model
5
Trend Analyzer
Association Clustering
Textual Classification
Query Creation Interface
Query Library
Data Query Manager
Data Store Interface
SNS Connector
DCL Interface
Model Transformation
Data Integration
HDFS
Intermediate Database
Mining Minds Platform
경희대학교
KYUNG HEE UNIVERSITY

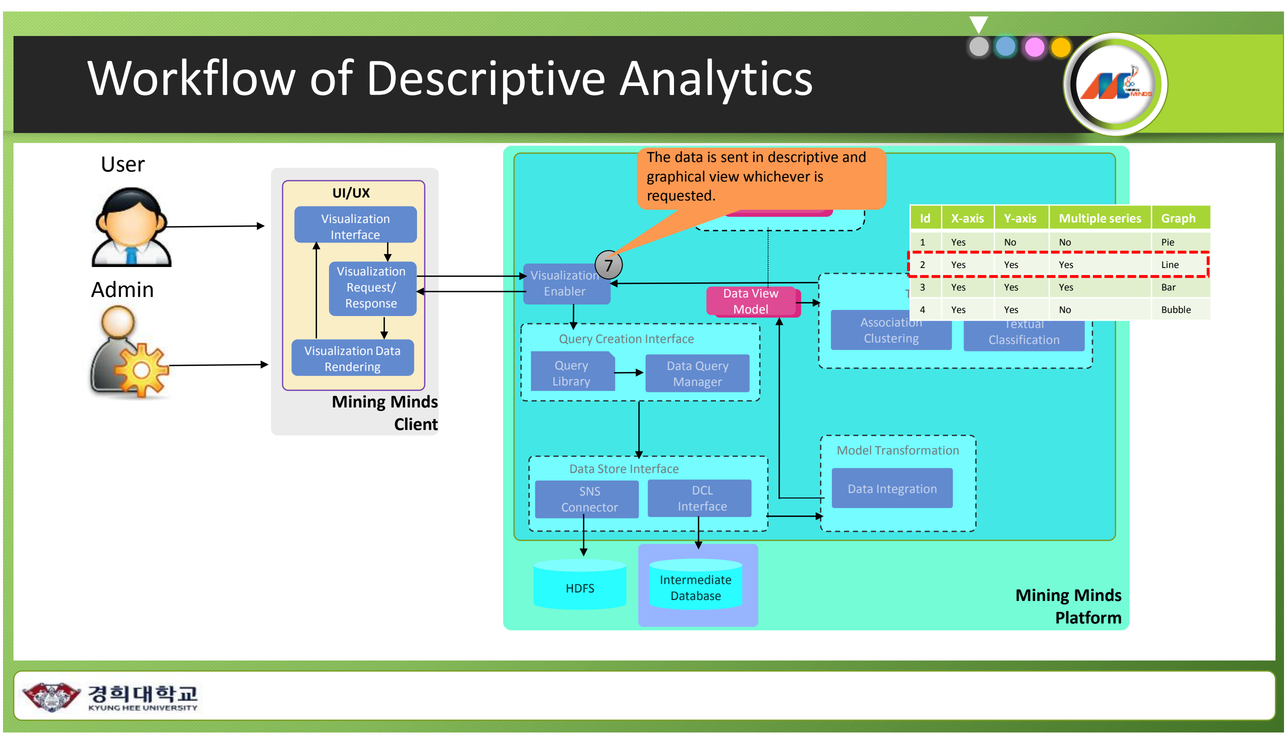


Workflow of Descriptive Analytics
User
Admin
UI/UX
Visualization Interface
Visualization Request/ Response
Visualization Data Rendering
Mining Minds Client
HTML View Model
The trend analyzer classifies the data depending on the metadata, adds facts and clusters the data for visualization.
6
Visualization Enabler
Data View
Query Creation Interface
Query Library
Trend Analyzer
Association Clustering
Textual Classification
Model Transformation
Data Integration
Data Store Interface
SNS Connector
DCL Interface
HDFS
Intermediate Database
Mining Minds Platform
경희대학교
KYUNG HEE UNIVERSITY

Activity	Max Time
Walking	32
Running	12
Sitting	240

Activity	Average
Walking	16min
Running	6 min
Sitting	63 min

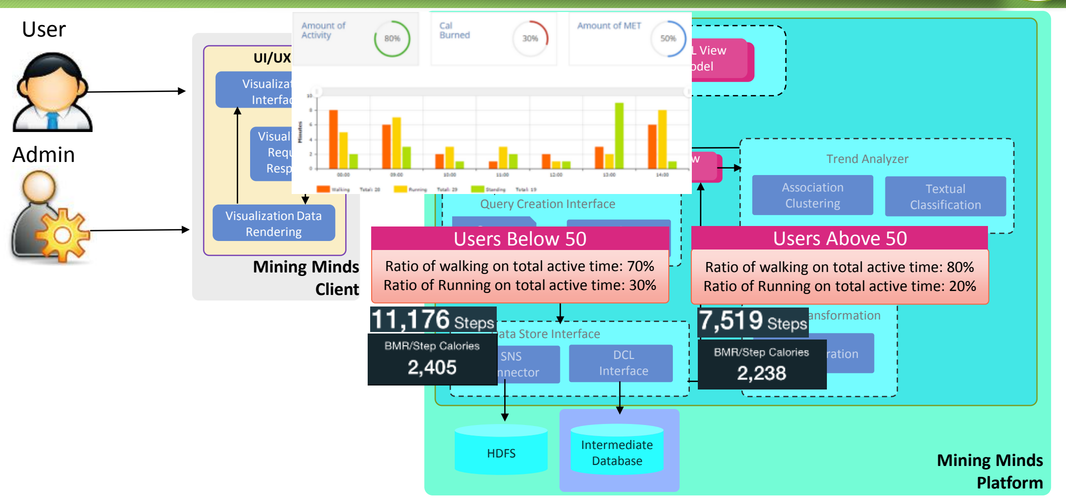
Activity	Calories Burned	MET
Walking	523	80
Running	180	60
Sitting	26	



Workflow of Descriptive Analytics
User
Admin
UI/UX
Visualization Interface
Visualization Request/ Response
Visualization Data Rendering
Mining Minds Client
The data is sent in descriptive and graphical view whichever is requested.
7
Visualization Enabler
Data View Model
Query Creation Interface
Query Library
Data Query Manager
Association Clustering
Textual Classification
Model Transformation
Data Integration
Data Store Interface
SNS Connector
DCL Interface
HDFS
Intermediate Database
Mining Minds Platform
경희대학교
KYUNG HEE UNIVERSITY

Id	X-axis	Y-axis	Multiple series	Graph
1	Yes	No	No	Pie
2	Yes	Yes	Yes	Line
3	Yes	Yes	Yes	Bar
4	Yes	Yes	No	Bubble

Workflow of Descriptive Analytics



KYUNG HEE UNIVERSITY

Contributions

- A query library for efficient data retrieval from big data repository.
- Model Transformation module to transform the data from HDFS and integrate it semantically
- Data Store interface for bringing unstructured (Big Data) and structured data (Intermediate).
- Trend analyzer to classify parameters and associate and cluster them for further insights
  - Focus on grouping and association techniques

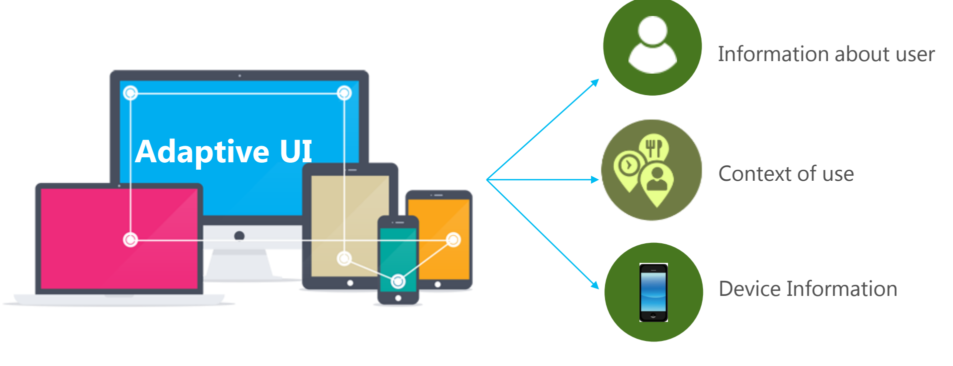
KYUNG HEE UNIVERSITY

Supporting Layer

경희대학교 KYUNG HEE UNIVERSITY U2 system User & Usability system

Introduction [1/2]

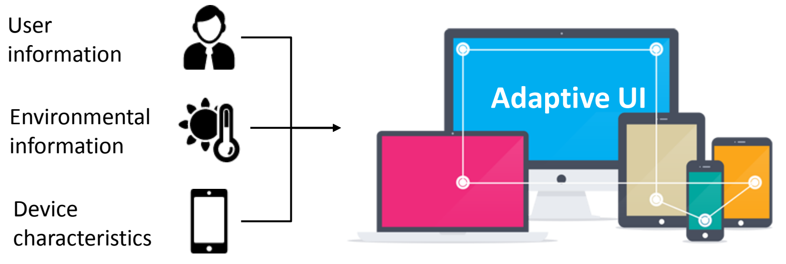
An Adaptive User interface is a User Interface(UI) which **adapts**, its **layout** and **elements** to individual users based on information acquired about its **user(s)**, the **context of use** and its **environment**.



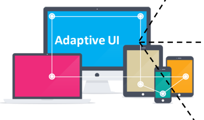
경희대학교 KYUNG HEE UNIVERSITY

Introduction [2/2]

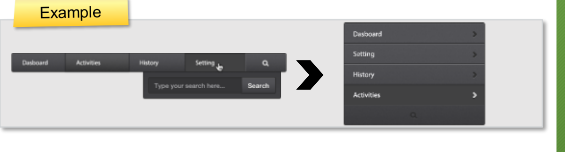
- In order to achieve a **satisfactory adaptation**, it is necessary to have several inputs such as



Granularity levels- UI adaptation at different level



Navigation

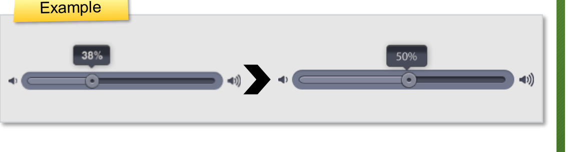


- ❑ To guide a user to their specific goal within the system by altering the way the system is navigated based on certain factors of the user

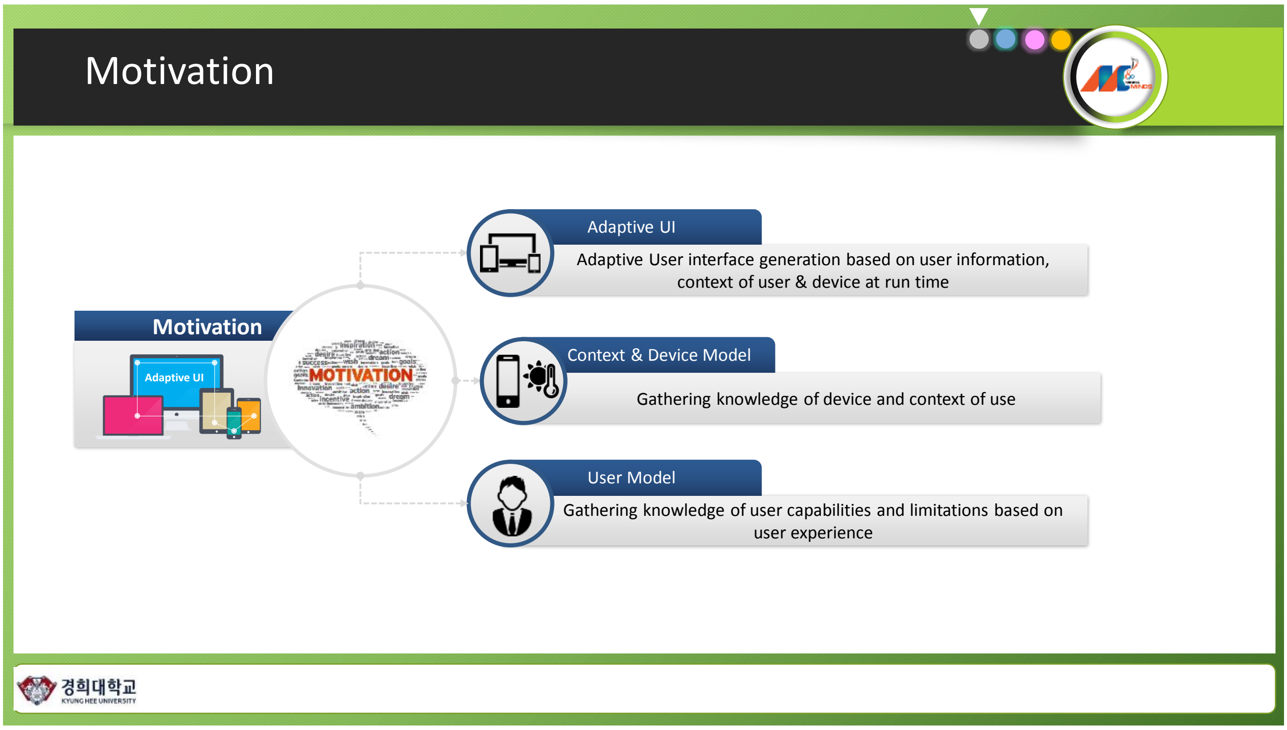
Presentation

- ❑ It concern the design of the interface, e.g. font size or color, layout of user interface elements

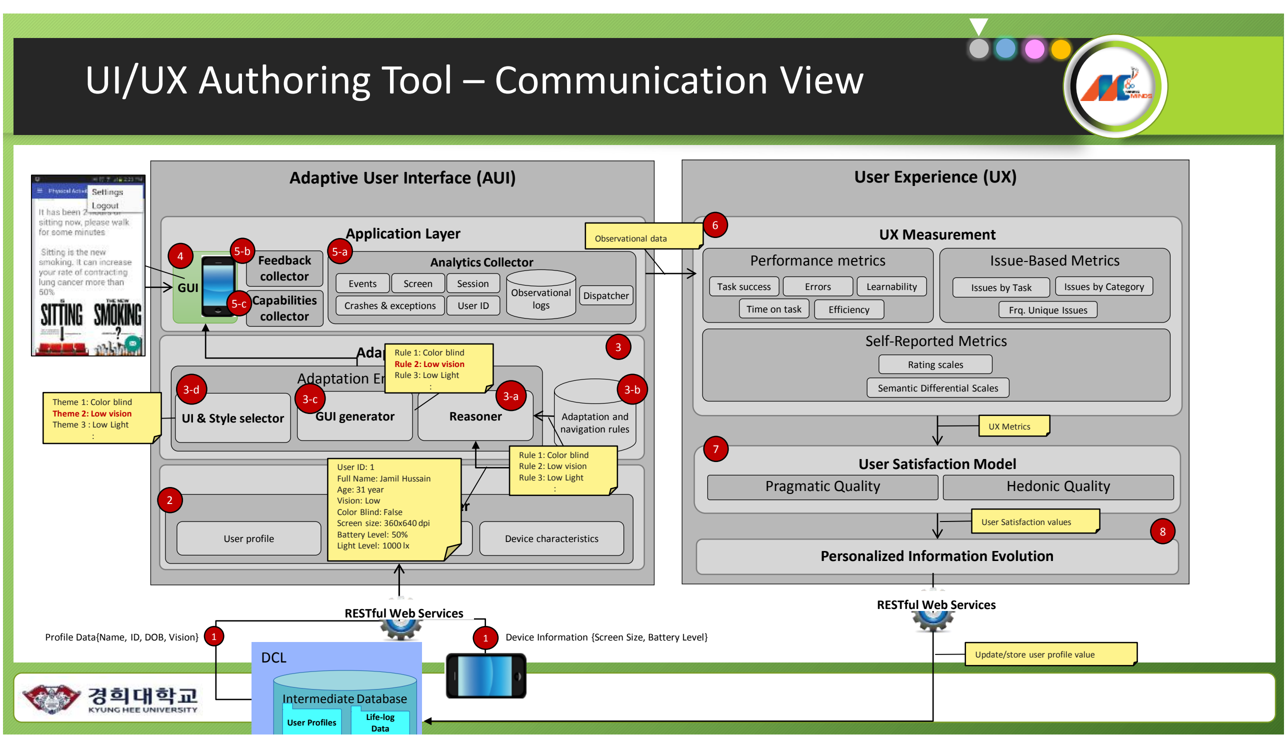
Content



- ❑ Adaptations that deal with the context of the situation occurred.



Motivation
Motivation
Adaptive UI
MOTIVATION
Adaptive UI
Adaptive User interface generation based on user information, context of user & device at run time
Context & Device Model
Gathering knowledge of device and context of use
User Model
Gathering knowledge of user capabilities and limitations based on user experience
경희대학교
KYUNG HEE UNIVERSITY



UI/UX Authoring Tool – Communication View
Adaptive User Interface (AUI)
Application Layer
Observational data
Feedback collector
Capabilities collector
GUI
Analytics Collector
Events
Screen
Session
Crashes & exceptions
User ID
Observational logs
Dispatcher
Rule 1: Color blind
Rule 2: Low vision
Rule 3: Low Light
Theme 1: Color blind
Theme 2: Low vision
Theme 3 : Low Light
UI & Style selector
GUI generator
Reasoner
Adaptation and navigation rules
User ID: 1
Full Name: Jamil Hussain
Age: 31 year
Vision: Low
Color Blind: False
Screen size: 360x640 dpi
Battery Level: 50%
Light Level: 1000 lx
User profile
Device characteristics
User Experience (UX)
UX Measurement
Performance metrics
Task success
Errors
Learnability
Time on task
Efficiency
Issue-Based Metrics
Issues by Task
Issues by Category
Frq. Unique Issues
Self-Reported Metrics
Rating scales
Semantic Differential Scales
UX Metrics
User Satisfaction Model
Pragmatic Quality
Hedonic Quality
User Satisfaction values
Personalized Information Evolution
RESTful Web Services
RESTful Web Services
Profile Data{Name, ID, DOB, Vision}
Device Information {Screen Size, Battery Level}
Update/store user profile value
DCL
Intermediate Database
User Profiles
Life-log Data
경희대학교
KYUNG HEE UNIVERSITY

UI/UX Authoring Tool – Overall Concept

UX Measurement

- ❑ It deals with metrics that reveal something about the user experience- about the personal experience/ interaction of user with product or system like
  - ✓ effectiveness (being able to complete a task),
  - ✓ efficiency (the amount of effort required to complete the task),
  - ✓ satisfaction (the degree to which the user was happy with his or her experience while performing the task)

Analytics Collector

- ❑ User interaction behavioral data collection
  - ✓ To measure the Pragmatic qualities

Feedback Collector

- ❑ Responsible for display the prompt feedback form based on implicit collected data then the user feedback data can be send to self-reporting component to in order to find out (1) the hedonic and (2) pragmatic quality

Capabilities collector

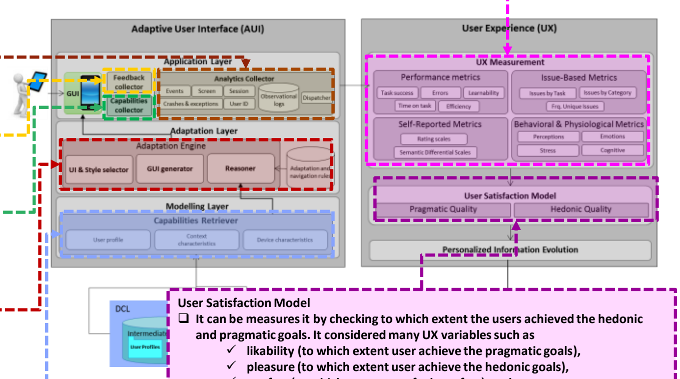
- ❑ Responsible to collect information about user perception, cognitive and motor skill using different game/ techniques

Adaptation Engine

- ❑ Reasoner
  - ✓ Input data
  - ✓ Rules
  - ✓ Fire rules based on collected data
- ❑ GUI Generator render UI based on fired rules

Capabilities Retriever

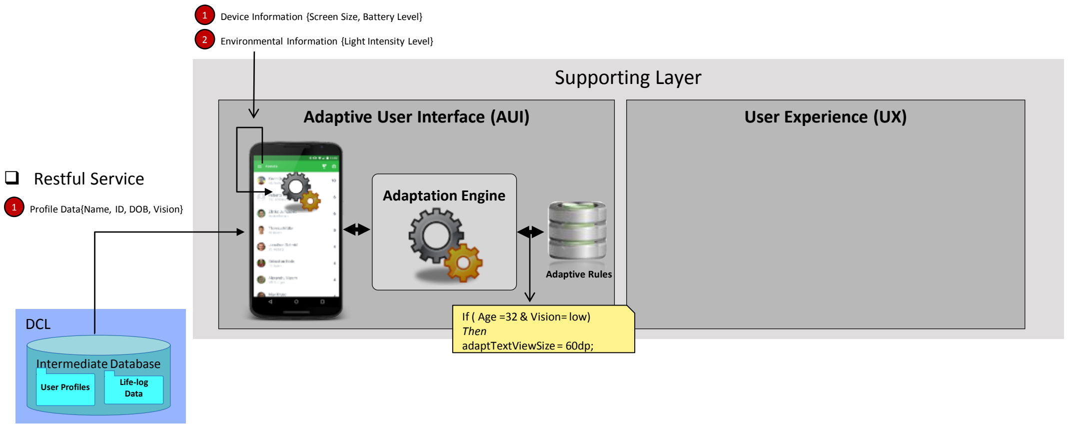
- ❑ It collect data for adaptation engine such as
  - ✓ User Profile data
  - ✓ Environmental data
  - ✓ Device information

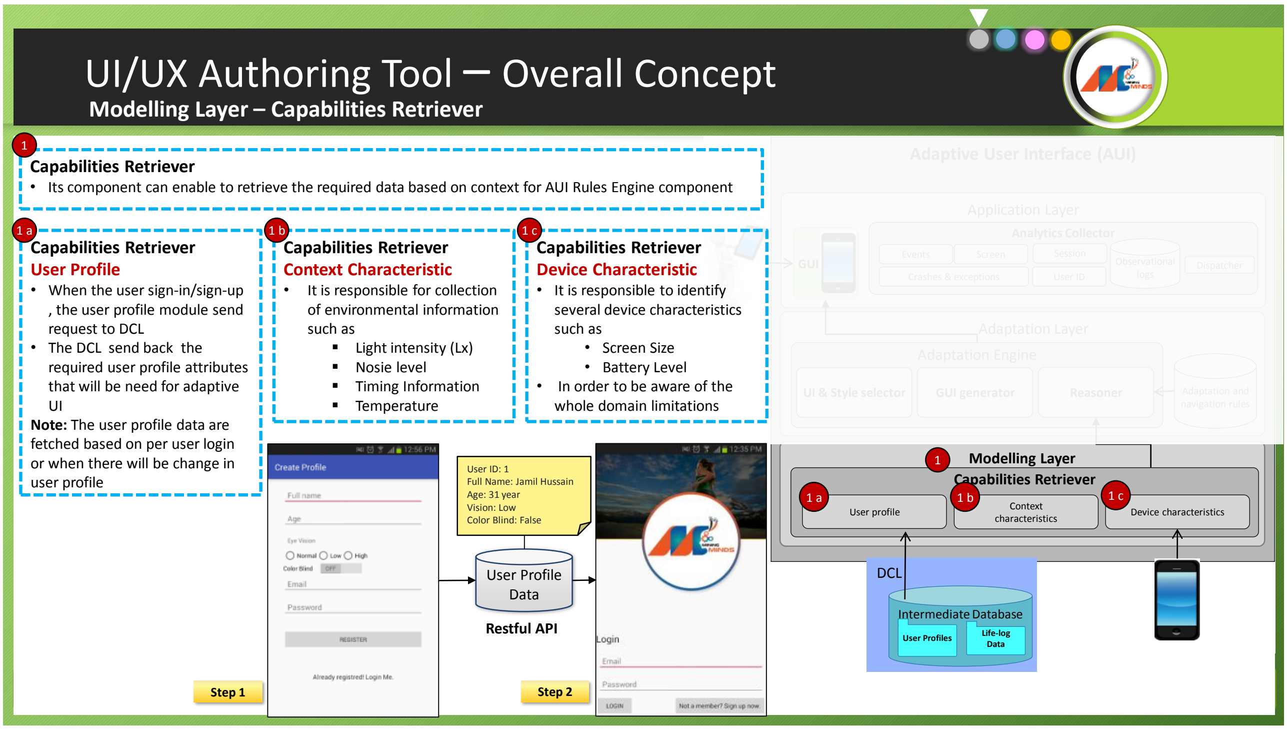


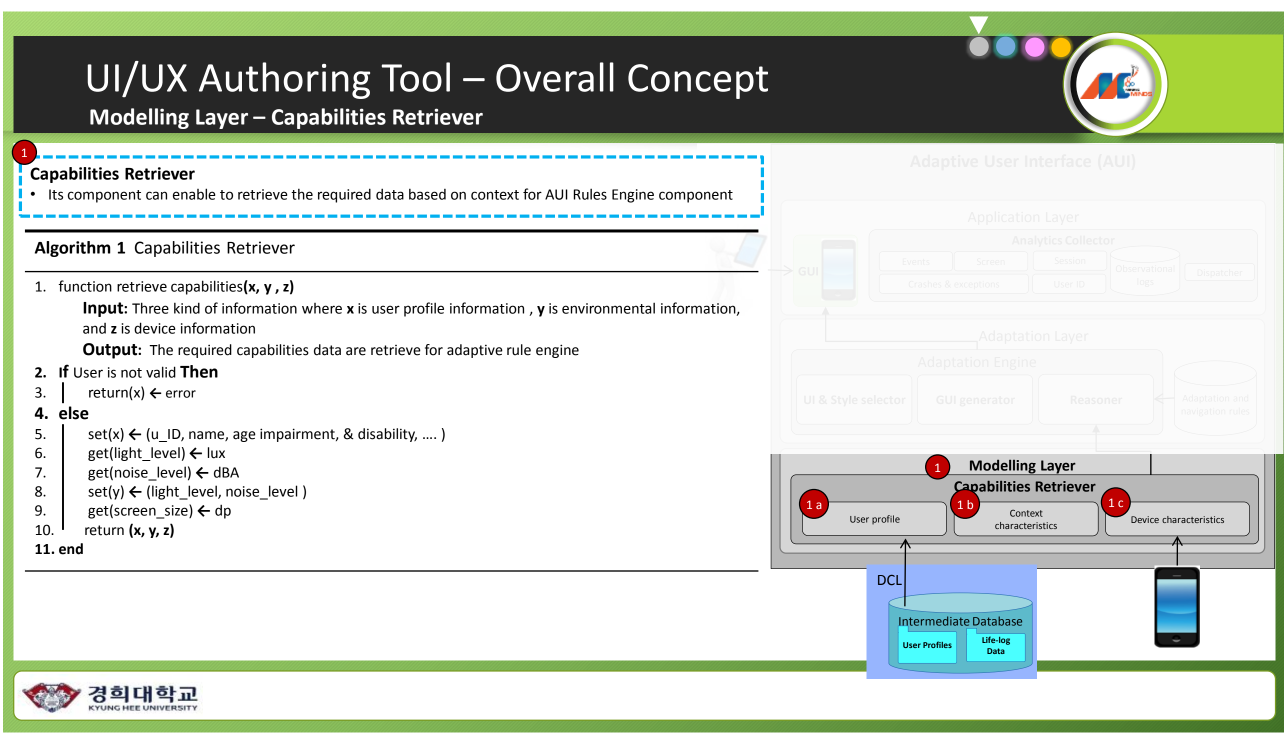
User Satisfaction Model

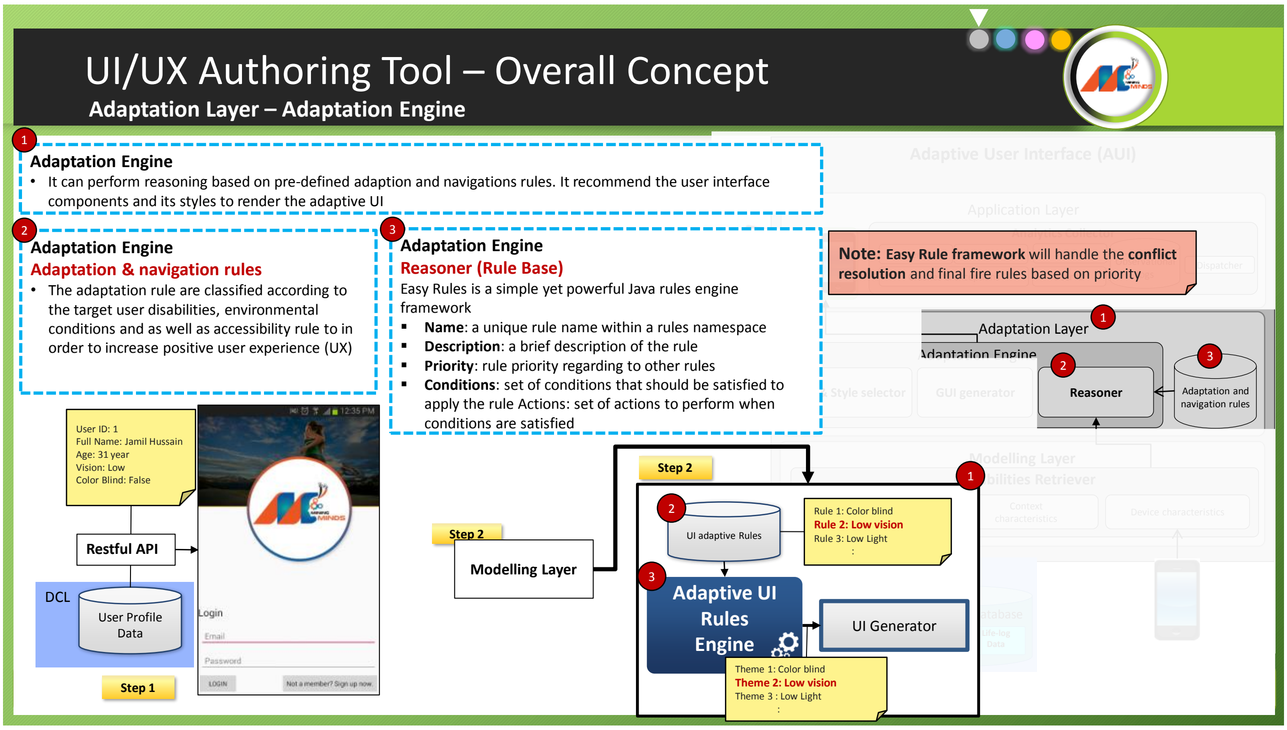
- ❑ It can be measures it by checking to which extent the users achieved the hedonic and pragmatic goals. It considered many UX variables such as
  - ✓ likability (to which extent user achieve the pragmatic goals),
  - ✓ pleasure (to which extent user achieve the hedonic goals),
  - ✓ comfort (to which extent user feel comfort), and
  - ✓ trust (to which extent user satisfy with overall system).

UI/UX Authoring Tool – Overall Concept of AUI

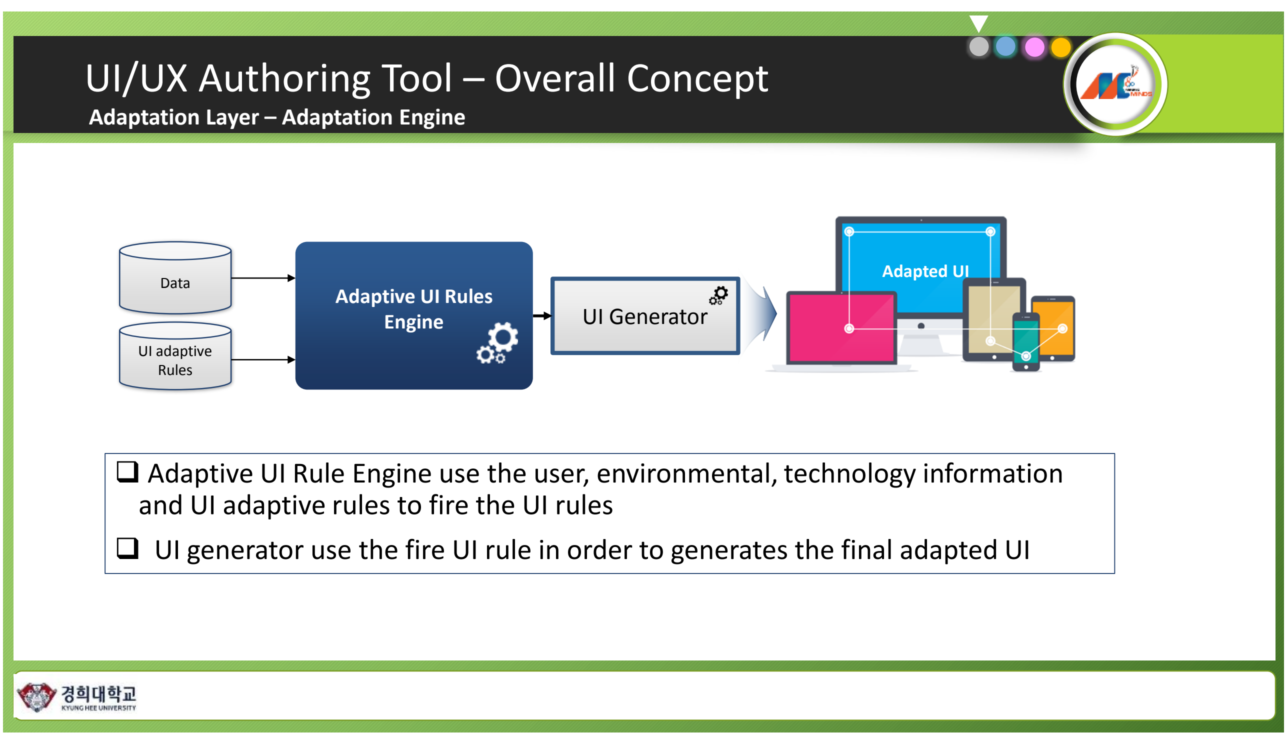



UI/UX Authoring Tool – Overall Concept
Modelling Layer – Capabilities Retriever
1
Capabilities Retriever
• Its component can enable to retrieve the required data based on context for AUI Rules Engine component
1 a
Capabilities Retriever
User Profile
• When the user sign-in/sign-up , the user profile module send request to DCL
• The DCL send back the required user profile attributes that will be need for adaptive UI
Note: The user profile data are fetched based on per user login or when there will be change in user profile
1 b
Capabilities Retriever
Context Characteristic
• It is responsible for collection of environmental information such as
▪ Light intensity (Lx)
▪ Nosie level
▪ Timing Information
▪ Temperature
1 c
Capabilities Retriever
Device Characteristic
• It is responsible to identify several device characteristics such as
• Screen Size
• Battery Level
• In order to be aware of the whole domain limitations
Create Profile
REGISTER
User ID: 1
Full Name: Jamil Hussain
Age: 31 year
Vision: Low
Color Blind: False
User Profile Data
Restful API
Login
Step 1
Step 2
Adaptive User Interface (AUI)
Application Layer
Adaptation Layer
Adaptation Engine
UI & Style selector
GUI generator
Reasoner
Modelling Layer
Capabilities Retriever
User profile
Context characteristics
Device characteristics
DCL
Intermediate Database
User Profiles
Life-log Data


UI/UX Authoring Tool – Overall Concept
Modelling Layer – Capabilities Retriever
1
Capabilities Retriever
• Its component can enable to retrieve the required data based on context for AUI Rules Engine component
Algorithm 1 Capabilities Retriever
1. function retrieve capabilities(x, y , z)
Input: Three kind of information where x is user profile information , y is environmental information, and z is device information
Output: The required capabilities data are retrieve for adaptive rule engine
2. If User is not valid Then
3. return(x) ← error
4. else
5. set(x) ← (u_ID, name, age impairment, & disability, ….)
6. get(light_level) ← lux
7. get(noise_level) ← dBA
8. set(y) ← (light_level, noise_level)
9. get(screen_size) ← dp
10. return (x, y, z)
11. end
Adaptive User Interface (AUI)
Application Layer
Adaptation Layer
Adaptation Engine
UI & Style selector
GUI generator
Reasoner
Modelling Layer
Capabilities Retriever
User profile
Context characteristics
Device characteristics
DCL
Intermediate Database
User Profiles
Life-log Data
경희대학교
KYUNG HEE UNIVERSITY



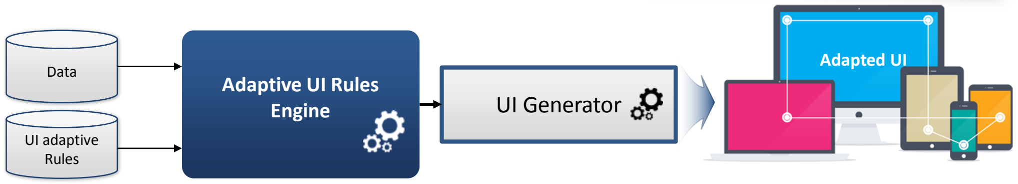
UI/UX Authoring Tool – Overall Concept
Adaptation Layer – Adaptation Engine
1
Adaptation Engine
• It can perform reasoning based on pre-defined adaption and navigations rules. It recommend the user interface components and its styles to render the adaptive UI
2
Adaptation Engine
Adaptation & navigation rules
• The adaptation rule are classified according to the target user disabilities, environmental conditions and as well as accessibility rule to in order to increase positive user experience (UX)
3
Adaptation Engine
Reasoner (Rule Base)
Easy Rules is a simple yet powerful Java rules engine framework
▪ Name: a unique rule name within a rules namespace
▪ Description: a brief description of the rule
▪ Priority: rule priority regarding to other rules
▪ Conditions: set of conditions that should be satisfied to apply the rule Actions: set of actions to perform when conditions are satisfied
Note: Easy Rule framework will handle the conflict resolution and final fire rules based on priority
Adaptation Layer
Reasoner
Adaptation and navigation rules
User ID: 1
Full Name: Jamil Hussain
Age: 31 year
Vision: Low
Color Blind: False
Restful API
DCL
User Profile Data
Step 1
Login
Email
Password
LOGIN
Not a member? Sign up now
Step 2
Modelling Layer
Step 2
UI adaptive Rules
Rule 1: Color blind
Rule 2: Low vision
Rule 3: Low Light
Adaptive UI Rules Engine
UI Generator
Theme 1: Color blind
Theme 2: Low vision
Theme 3 : Low Light

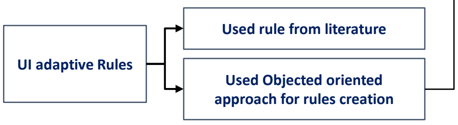


UI/UX Authoring Tool – Overall Concept
Adaptation Layer – Adaptation Engine
Data
UI adaptive Rules
Adaptive UI Rules Engine
UI Generator
Adapted UI
❑ Adaptive UI Rule Engine use the user, environmental, technology information and UI adaptive rules to fire the UI rules
❑ UI generator use the fire UI rule in order to generates the final adapted UI
경희대학교
KYUNG HEE UNIVERSITY

UI/UX Authoring Tool – Overall Concept

Adaptation Layer – Adaptation Engine





Note: In MM we will created rule Authoring tool, so that UX expert will be able to create new rule, update and delete the existing rules.

Easy Rules : Easy Rules is a simple yet powerful Java rules engine framework

First, define your rule..

```
@Rule (name = "my awesome rule" )
public class MyRule {
  @Condition
  public boolean when() {
    return true;
  }
  @Action
  public void then() {
    System.out.println("Easy Rules rocks!");
  }
}
```

Then, fire it!

```
public class Test {
  public static void main(String[] args) {
    // create a rules engine
    RulesEngine rulesEngine =
               aNewRulesEngine().build();
    //register the rule
    rulesEngine.registerRule(new MyRule());
    //fire rules
    rulesEngine.fireRules();
  }
}
```


http://www.easyrules.org/index.html

UI/UX Authoring Tool – Overall Concept

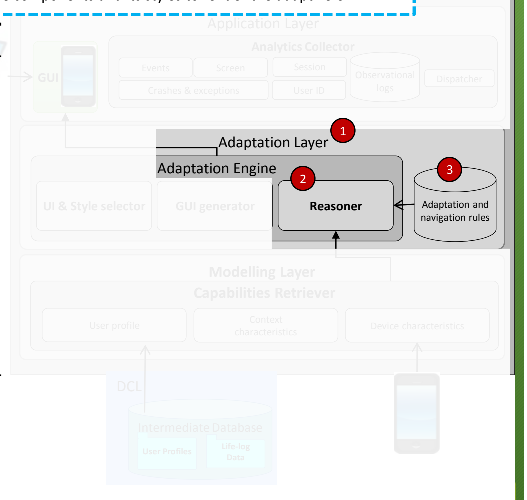
Adaptation Layer – Adaptation Engine

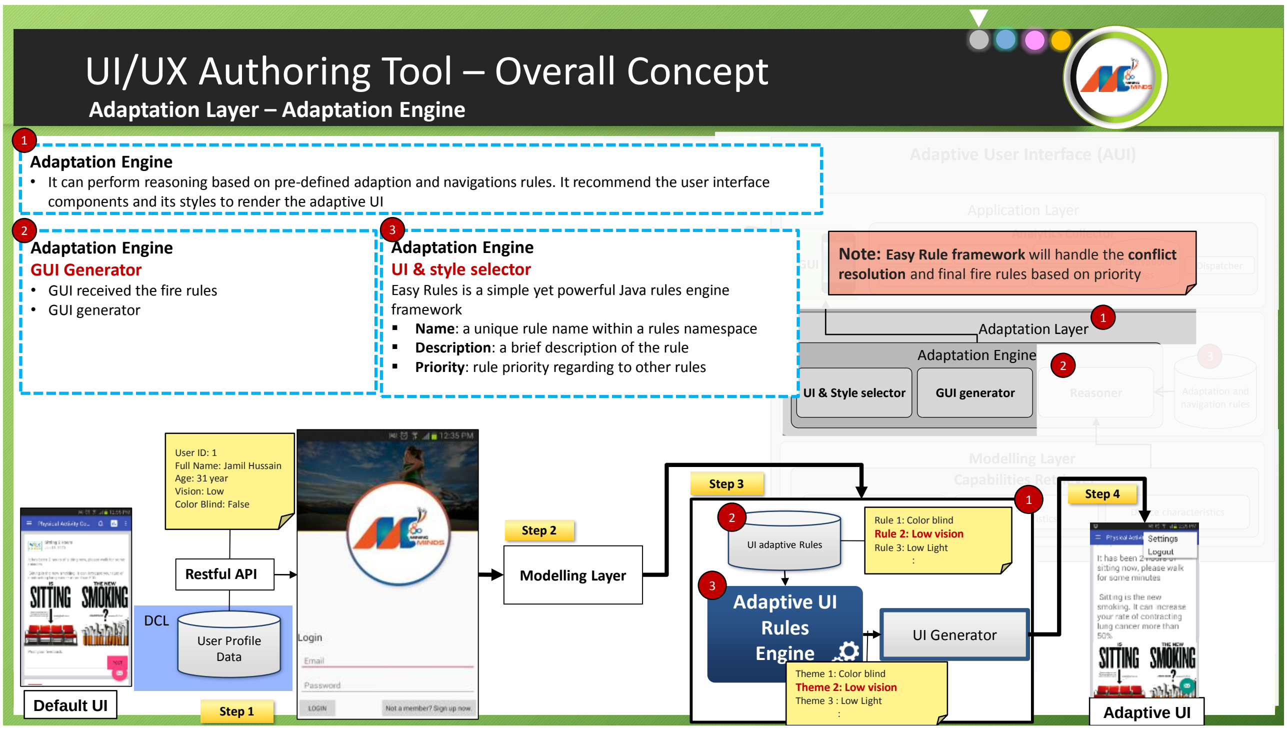
1 **Adaptation Engine**

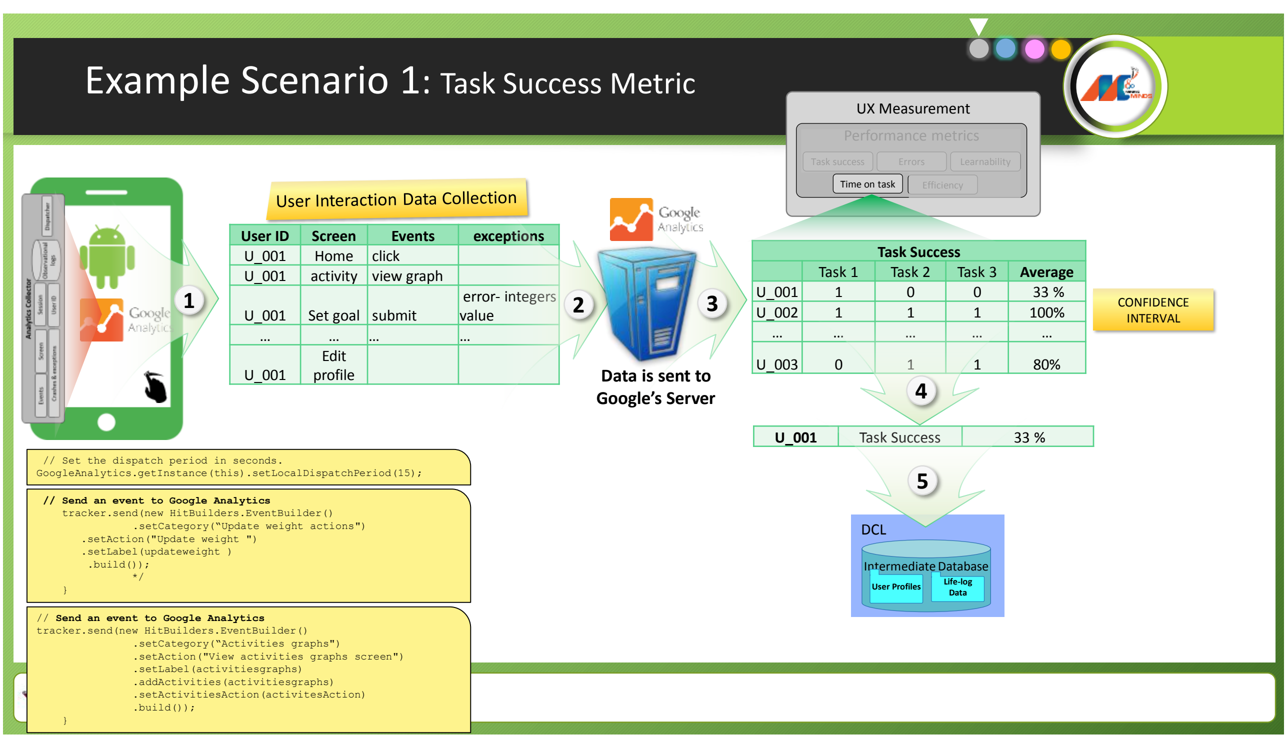
- It can perform reasoning based on pre-defined adaption and navigations rules. It recommend the user interface components and its styles to render the adaptive UI

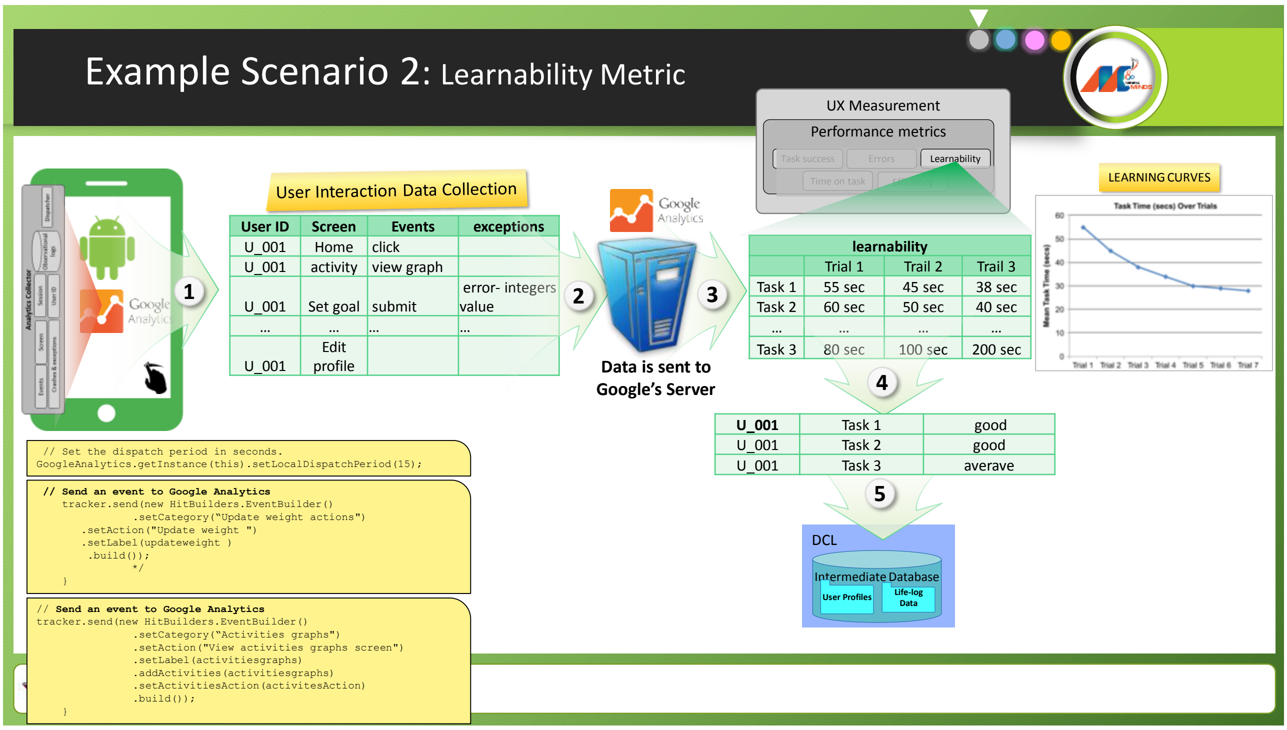
Algorithm 2 Adaptation Engine

1. Function reasoning(**x, y**)
   Input: Two kind of information where **x** is retrieved information, **y** is adaptation rule sets
   Output: Fire the final rule so that GUI generator can render the GUI based on that rule
2. **If** User is not valid **Then**
3. return(x) ← error
4. **else**
5. set(x) ← (user profile, environmental information, device information)
6. set(y) ← load all rules from adaptation and navigation rules from the local DB
7. **foreach** input ϵ y **do**
8. registered all rules to rule engine
9. **end**
10. fire_rule ← fire rules based on matching
11. **return** (fire_rule)
12. **end**

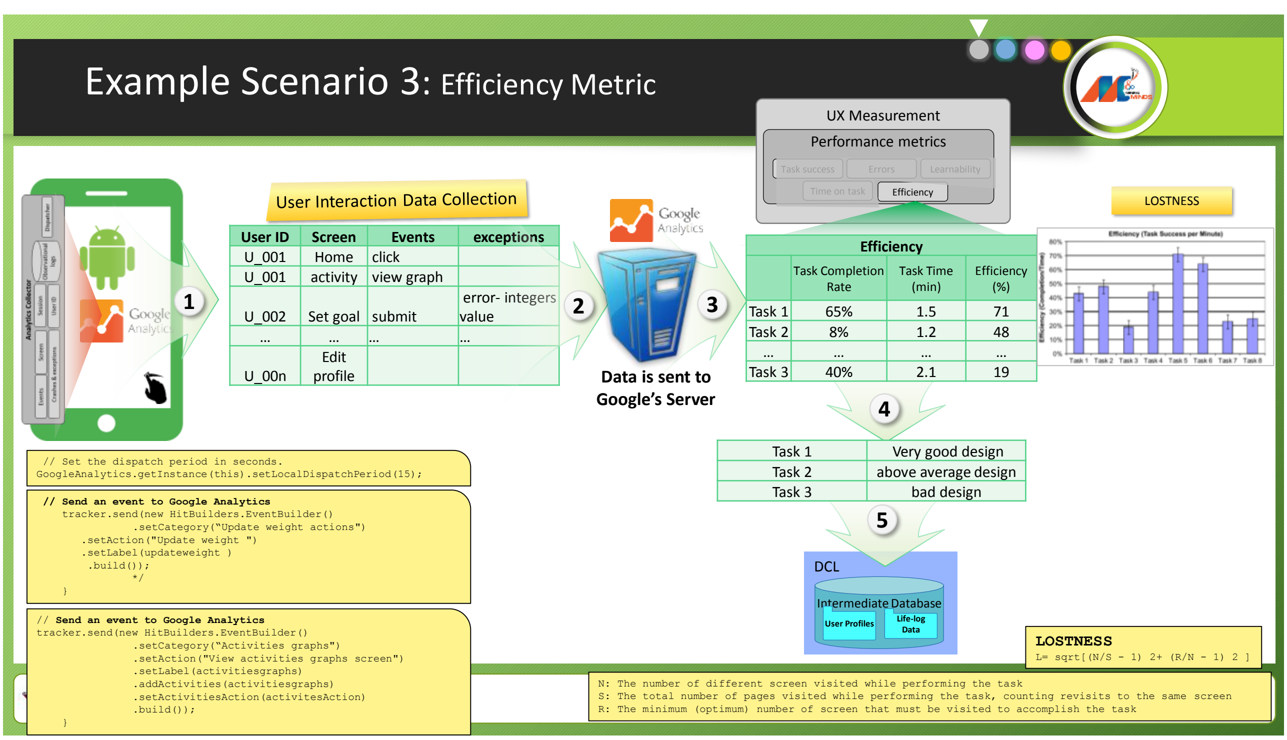



UI/UX Authoring Tool – Overall Concept
Adaptation Layer – Adaptation Engine
1
Adaptation Engine
• It can perform reasoning based on pre-defined adaption and navigations rules. It recommend the user interface components and its styles to render the adaptive UI
2
Adaptation Engine
GUI Generator
• GUI received the fire rules
• GUI generator
3
Adaptation Engine
UI & style selector
Easy Rules is a simple yet powerful Java rules engine framework
▪ Name: a unique rule name within a rules namespace
▪ Description: a brief description of the rule
▪ Priority: rule priority regarding to other rules
Note: Easy Rule framework will handle the conflict resolution and final fire rules based on priority
Adaptation Layer
Adaptation Engine
UI & Style selector
GUI generator
User ID: 1
Full Name: Jamil Hussain
Age: 31 year
Vision: Low
Color Blind: False
Restful API
DCL
User Profile Data
Default UI
Step 1
Login
Step 2
Modelling Layer
Step 3
UI adaptive Rules
Rule 1: Color blind
Rule 2: Low vision
Rule 3: Low Light
Adaptive UI Rules Engine
UI Generator
Theme 1: Color blind
Theme 2: Low vision
Theme 3 : Low Light
Step 4
Adaptive UI


Example Scenario 1: Task Success Metric
UX Measurement
Performance metrics
Task success
Errors
Learnability
Time on task
Efficiency
User Interaction Data Collection
User ID | Screen | Events | exceptions
U_001 | Home | click
U_001 | activity | view graph
error- integers value
U_001 | Set goal | submit
Edit profile
1
2
3
Data is sent to Google's Server
Task Success
Task 1 | Task 2 | Task 3 | Average
U_001 | 1 | 0 | 0 | 33 %
U_002 | 1 | 1 | 1 | 100%
U_003 | 0 | 1 | 1 | 80%
CONFIDENCE INTERVAL
4
U_001 | Task Success | 33 %
5
DCL
Intermediate Database
User Profiles
Life-log Data
// Set the dispatch period in seconds.
GoogleAnalytics.getInstance(this).setLocalDispatchPeriod(15);
// Send an event to Google Analytics
tracker.send(new HitBuilders.EventBuilder()
.setCategory("Update weight actions")
.setAction("Update weight ")
.setLabel(updateweight)
.build());
// Send an event to Google Analytics
tracker.send(new HitBuilders.EventBuilder()
.setCategory("Activities graphs")
.setAction("View activities graphs screen")
.setLabel(activitiesgraphs)
.addActivities(activitiesgraphs)
.setActivitiesAction(activitesAction)
.build());

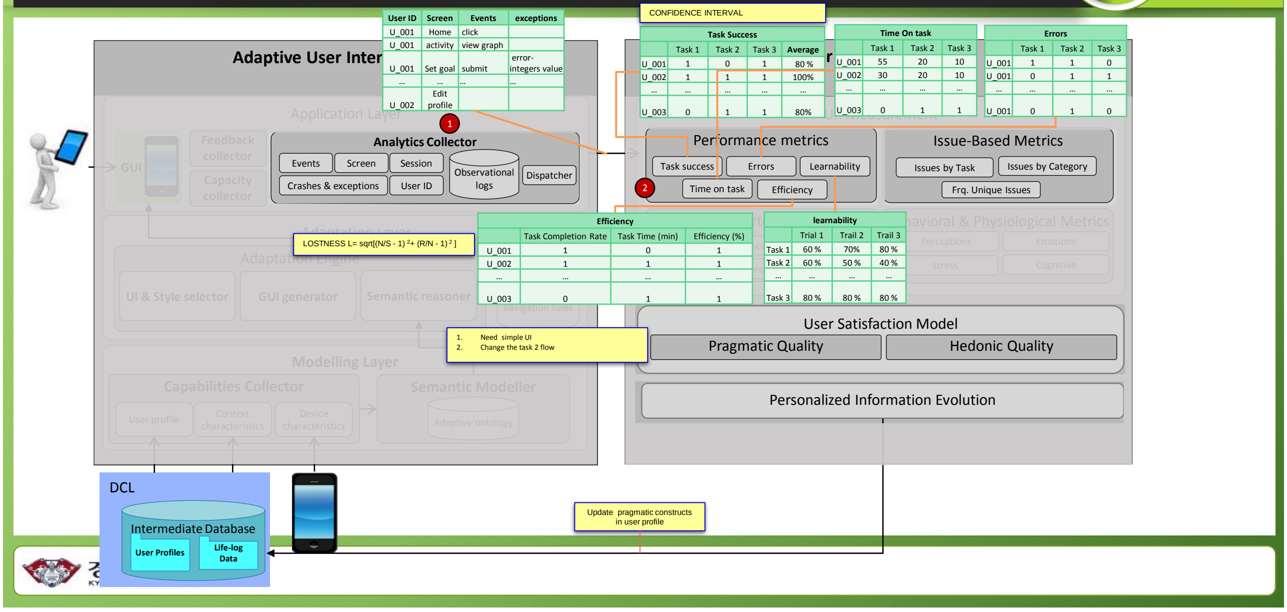


Example Scenario 2: Learnability Metric
UX Measurement
Performance metrics
Task success
Errors
Learnability
Time on task
LEARNING CURVES
Task Time (secs) Over Trials
Mean Task Time (secs)
Trial 1 Trial 2 Trial 3 Trial 4 Trial 5 Trial 6 Trial 7
User Interaction Data Collection
User ID | Screen | Events | exceptions
U_001 | Home | click
U_001 | activity | view graph
U_001 | Set goal | submit | error- integers value
U_001 | Edit profile
Data is sent to Google's Server
learnability
Trial 1 | Trail 2 | Trail 3
Task 1 | 55 sec | 45 sec | 38 sec
Task 2 | 60 sec | 50 sec | 40 sec
Task 3 | 80 sec | 100 sec | 200 sec
U_001 | Task 1 | good
U_001 | Task 2 | good
U_001 | Task 3 | averave
DCL
Intermediate Database
User Profiles
Life-log Data
// Set the dispatch period in seconds.
GoogleAnalytics.getInstance(this).setLocalDispatchPeriod(15);
// Send an event to Google Analytics
tracker.send(new HitBuilders.EventBuilder()
.setCategory("Update weight actions")
.setAction("Update weight ")
.setLabel(updateweight)
.build());
*/
}
// Send an event to Google Analytics
tracker.send(new HitBuilders.EventBuilder()
.setCategory("Activities graphs")
.setAction("View activities graphs screen")
.setLabel(activitiesgraphs)
.addActivities(activitiesgraphs)
.setActivitiesAction(activitesAction)
.build());
}



Example Scenario 3: Efficiency Metric
UX Measurement
Performance metrics
Task success
Errors
Learnability
Time on task
Efficiency
LOSTNESS
Efficiency (Task Success per Minute)
Task 1 Task 2 Task 3 Task 4 Task 5 Task 6 Task 7 Task 8
User Interaction Data Collection
User ID | Screen | Events | exceptions
U_001 | Home | click
U_001 | activity | view graph
U_002 | Set goal | submit | error- integers value
U_00n | Edit profile
Data is sent to Google's Server
Efficiency
Task Completion Rate | Task Time (min) | Efficiency (%)
Task 1 | 65% | 1.5 | 71
Task 2 | 8% | 1.2 | 48
Task 3 | 40% | 2.1 | 19
Task 1 | Very good design
Task 2 | above average design
Task 3 | bad design
DCL
Intermediate Database
User Profiles
Life-log Data
LOSTNESS
L= sqrt[(N/S – 1) 2+ (R/N – 1) 2]
N: The number of different screen visited while performing the task
S: The total number of pages visited while performing the task, counting revisits to the same screen
R: The minimum (optimum) number of screen that must be visited to accomplish the task
// Set the dispatch period in seconds.
GoogleAnalytics.getInstance(this).setLocalDispatchPeriod(15);
// Send an event to Google Analytics
tracker.send(new HitBuilders.EventBuilder()
.setCategory("Update weight actions")
.setAction("Update weight ")
.setLabel(updateweight)
.build());
*/
}
// Send an event to Google Analytics
tracker.send(new HitBuilders.EventBuilder()
.setCategory("Activities graphs")
.setAction("View activities graphs screen")
.setLabel(activitiesgraphs)
.addActivities(activitiesgraphs)
.setActivitiesAction(activitesAction)
.build());
}

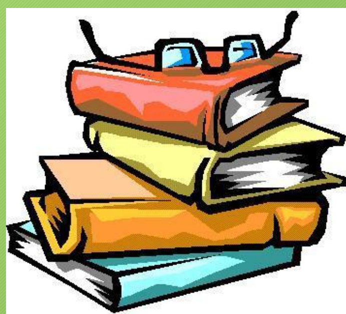
Example Scenario 4: All UX Measurement Metrics



Contributions

- Rules for adaptive UI
- User experience measurement toolkit development
- User Experience (UX) can be improved with Adaptive UI
- Adaptive UI can improved accessibility
- Adaptive UI can improved users' performance and satisfaction
- Continuous evolution of UI with contextual information change

경희대학교
KYUNG HEE UNIVERSITY



REFERENCES

References- Descriptive Analytics

1. [Glatz2014] Glatz, Eduard, et al. "Visualizing big network traffic data using frequent pattern mining and hypergraphs." *Computing* 96.1 (2014): 27-38.
2. [Gehlenborg2010] Gehlenborg, Nils, et al. "Visualization of omics data for systems biology." Nature methods 7 (2010): S56-S68.
3. [Bennett2009] Bennett, Jeremy S., "Massive model visualization: An investigation into spatial partitioning" (2009). Graduate Theses and Dissertations. Paper 10546.
4. [Dancy2008] Elizabeth Dancy, Uliyana Markova "Autonomic computing in Canadian academia"
5. [Dietrich2007] Dietrich, Andreas, Enrico Gobbetti, and Sung-Eui Yoon. "Massive-model rendering techniques: a tutorial." IEEE Computer Graphics and Applications27.6 (2007): 20-34.
6. [Shi2004] Shi, Rui, et al. "An adaptive image content representation and segmentation approach to automatic image annotation." *Image and Video Retrieval*. Springer Berlin Heidelberg, 2004. 545-554.
7. [McAdams2000] McAdams, Daniel A., and Kristin L. Wood. "Tuning parameter tolerance design: foundations, methods, and measures." Research in Engineering Design 12.3 (2000): 152-162.
8. Chang, Ya-Ting, and Shih-Wei Sun. "A realtime interactive visualization system for knowledge transfer from social media in a big data." *Information, Communications and Signal Processing (ICICS) 2013 9th International Conference on*. IEEE, 2013.
9. Kandogan, E., "Just-in-time annotation of clusters, outliers, and trends in point-based data visualizations," *Visual Analytics Science and Technology (VAST), 2012 IEEE Conference on* , vol., no., pp.73,82, 14-19 Oct. 2012

References-UI/IX Authoring Tool

1. J. Orwant. Doppelg¨ anger–a user modeling system. PhD thesis, Massachusetts Institute of Technology, 1991.
2. G. Brajnik and C. Tasso. A shell for developing non-monotonic user modeling systems. International Journal of Human-Computer Studies, 40(1):31–62, 1994.
3. A. Paiva and J. Self. Tagus—a user and learner modeling workbench. User Mod- eling and User-Adapted Interaction, 4(3):197–226, 1994.
4. J. Kay. The um toolkit for cooperative user modelling. User Modeling and User- Adapted Interaction, 4(3):149–196, 1994
5. W. Pohl. Logic-based representation and reasoning for user modeling shell systems. User Modeling and User-Adapted Interaction, 9(3):217–282, 1999.
6. G. Fischer. User modeling in human–computer interaction. User modeling and user-adapted interaction, 11(1):65–86, 2001. ISSN 0924-1868.
7. A. Kobsa. Generic user modeling systems. User modeling and user-adapted interaction, 11(1):49–63, 2001.
8. P. Gregor, A. F. Newell, and M. Zajicek. Designing for dynamic diversity: inter- faces for older people. In Proceedings of the fifth international ACM conference on Assistive technologies, page 151–156, 2002.
9. S. Gauch, J. Chaffee, and A. Pretschner. Ontology-based personalized search and browsing. Web Intelligence and Agent Systems, 1(3):219–234, 2003. URL
10. L. Razmerita, A. Angehrn, and A. Maedche. Ontology-based user mod- eling for knowledge management systems. In User Modeling 2003, page 213–217. Springer, 2003.
11. M. Hatala and R.Wakkary. Ontology-based user modeling in an augmented audio reality system for museums. User Modeling and User-Adapted Interaction, 15 (3-4):339–380, 2005. URL
12. F. Pereira. A triple user characterization model for video adaptation and quality of experience evaluatio. In Multimedia Signal Processing, 2005 IEEE 7th Workshop on, page 1–4, 2005. URL
13. D. Heckmann, T. Schwartz, B.s Brandherm, M. Schmitz, and M. vonWilamowitz- Moellendorff. Gumo–the general user model ontology. In User Modeling 2005, pages 428–432. Springer, 2005.

경희대학교
KYUNG HEE UNIVERSITY

References-UI/UX Authoring Tool

14. U. Persad, P. Langdon, and J. Clarkson. Characterising user capabilities to support inclusive design evaluation. Universal Access in the Information Society, 6(2): 119–135, 2007
15. P. Gregor, A. F. Newell, and M. Zajicek. Designing for dynamic diversity: inter- faces for older people. In Proceedings of the fifth international ACM conference on Assistive technologies, page 151–156, 2002
16. R. Casas, R. Blasco Mar´ ın, A. Robinet, A. Delgado, A. Yarza, J. Mcginn, R. Picking, and V. Grout. User modelling in ambient intelligence for elderly and dis- abled people. Computers Helping People with Special Needs, page 114–122, 2008. URL
17. C. Evers, R. Kniewel, K. Geihs, and L. Schmidt. Achieving user participation for adaptive applications. In Ubiquitous Computing and Ambient Intelligence, Lec- ture Notes in Computer Science, pages 200–207. Springer Berlin Heidelberg, Jan- uary 2012. ISBN
18. K. L. Skillen, L. Chen, C. D. Nugent, M. P. Donnelly, W. Burns, and I. Solheim. Ontological user profile modeling for context-aware application personalization. In Ubiquitous Computing and Ambient Intelligence 261–268. Springer, 2012. URL
19. http://myui.eu/
20. http://www.ibm.com/developerworks/library/wa-uihealth/
21. Bongartz, Sara, et al. "Adaptive User Interfaces for Smart Environments with the Support of Model-Based Languages." Ambient Intelligence. Springer Berlin Heidelberg, 2012. 33-48.
22. Jason, Bronwin, Andre Calitz, and Jean Greyling. "The evaluation of an adaptive user interface model." Proceedings of the 2010 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists. ACM, 2010.
23. http://www.w3.org/standards/techs/ccpp#w3c_all
24. Uaprofindex. http://www.w3.org/wiki/UAProfIndex
25. Wurfl - the mobile device database. http://wurfl.sourceforge.net/

경희대학교
KYUNG HEE UNIVERSITY

Section 8

Stencils for
Mining Minds Version 3.0 Layers Description

Mining Minds Platform

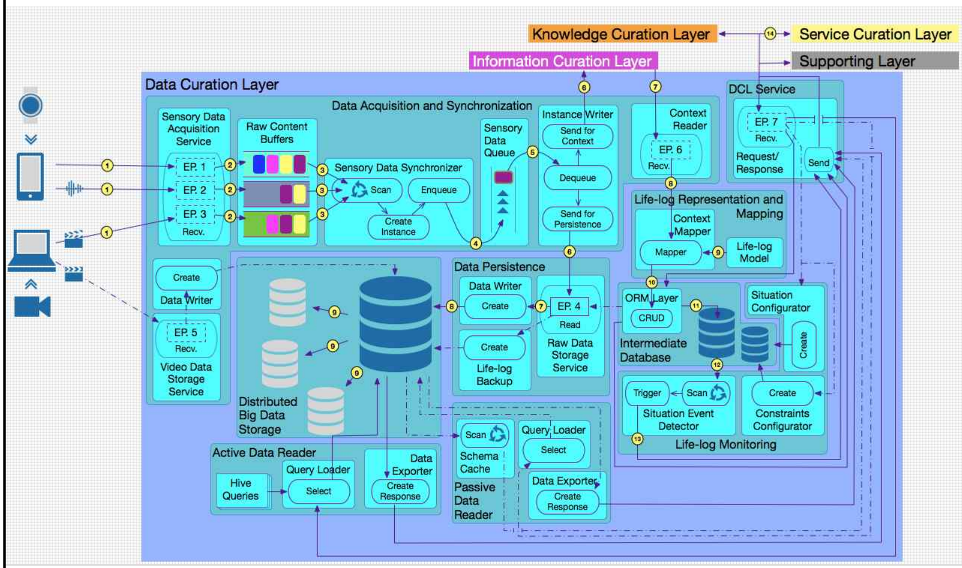
STENCILS for Layers Micro-Animation

Professor Sungyoung Lee

Ubiquitous Computing Laboratory
Kyung Hee University, Korea

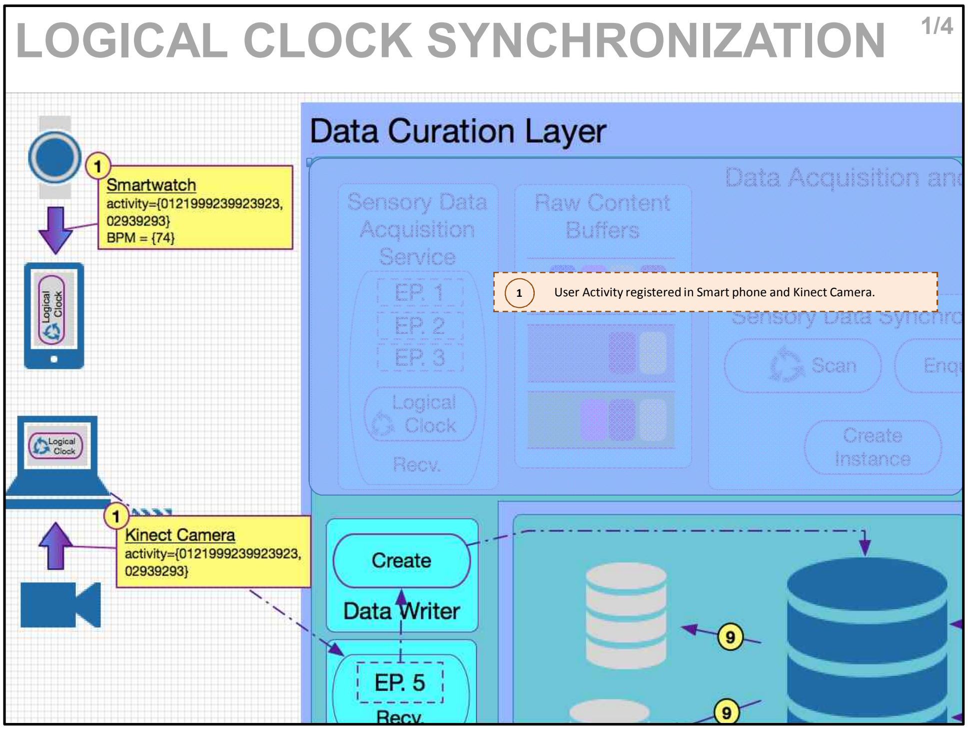
STENCILS for DCL Micro-Animation

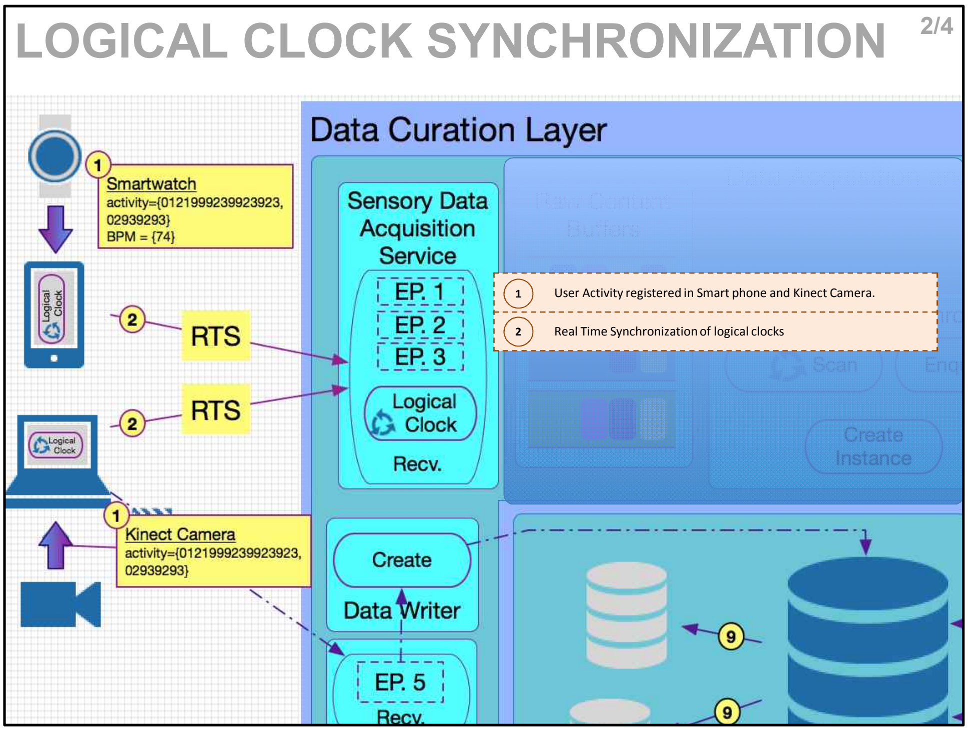
DCL Detailed Architecture

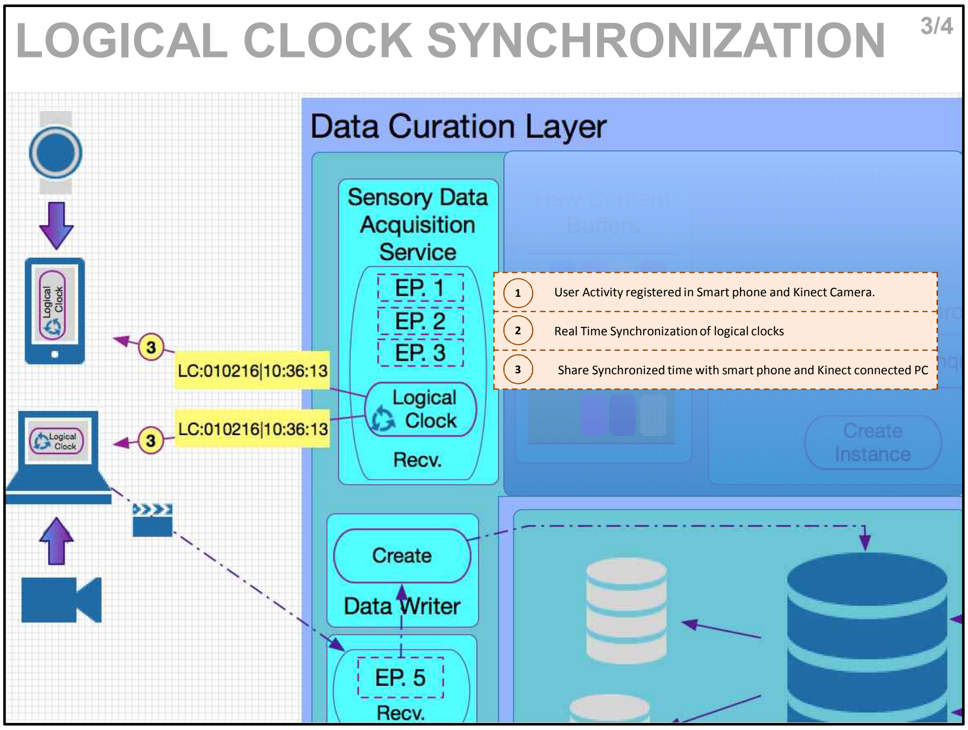


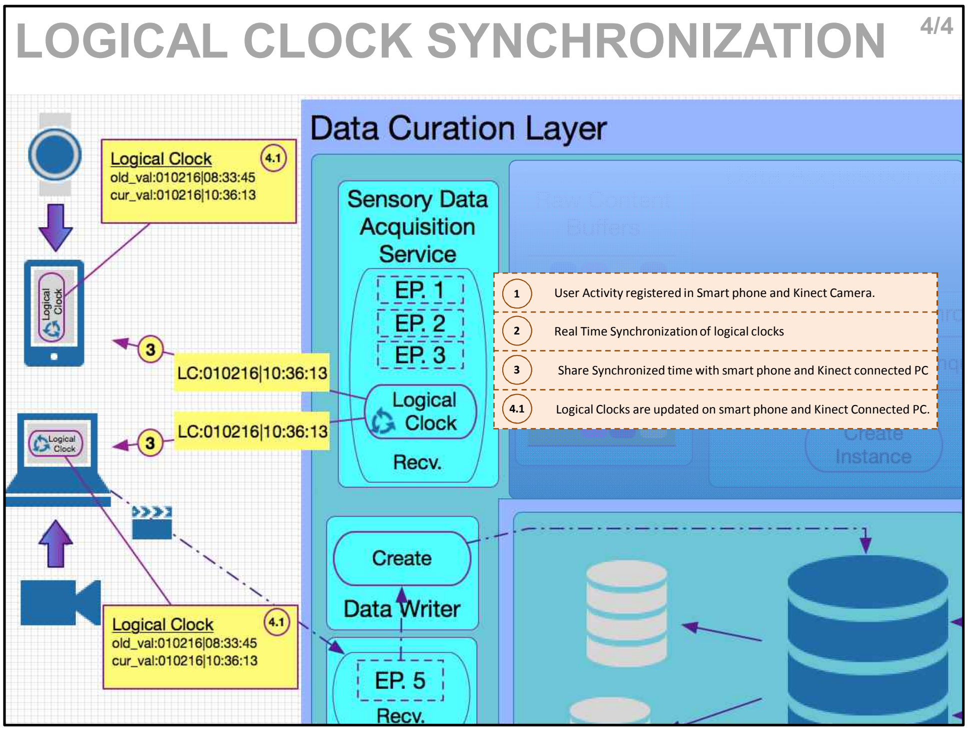
Scenario 1: Real-time

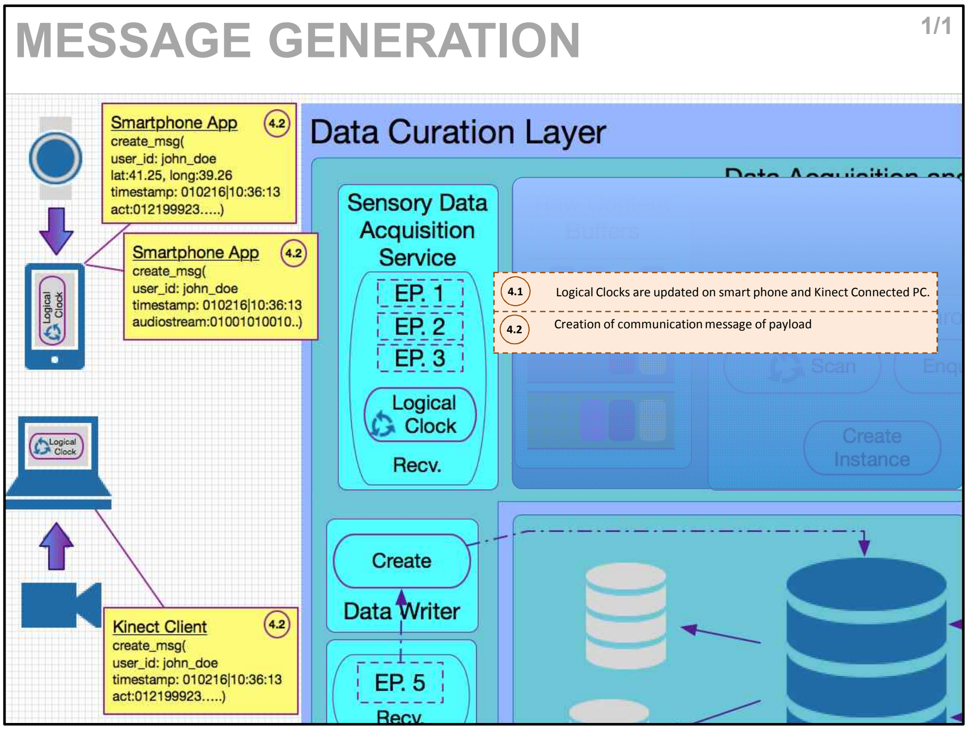
Desc. : Raw-sensory data from multimodal data sources is **acquisitioned**, and **persisted**. identified context on the sensory data is **mapped** to user life-log and **monitored** for situation detection.

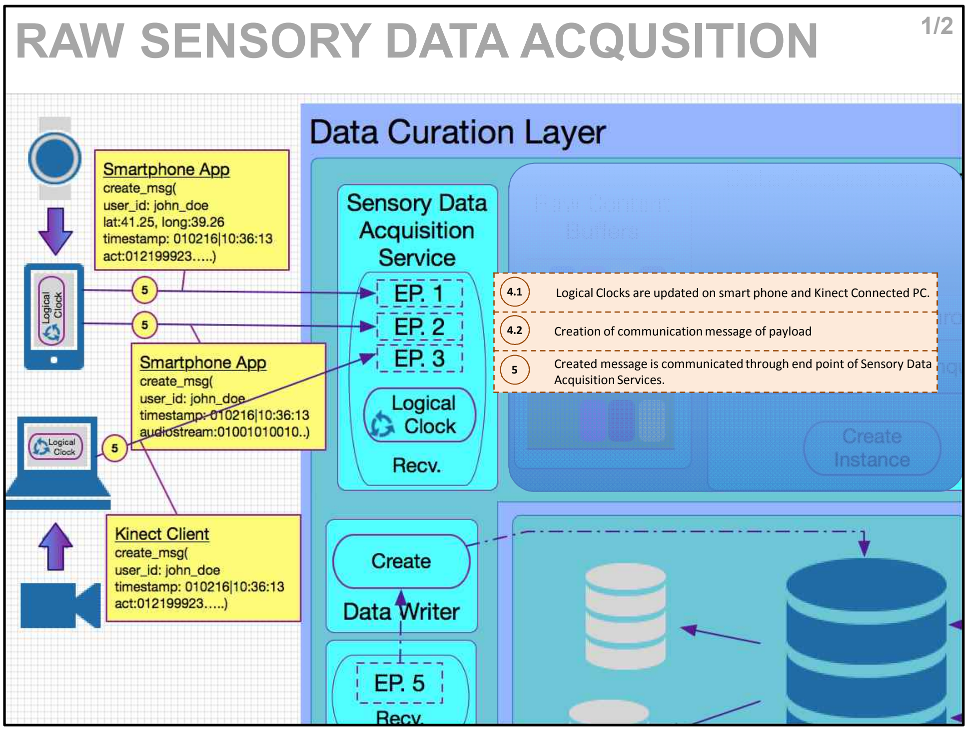

LOGICAL CLOCK SYNCHRONIZATION
1/4
Data Curation Layer
Smartwatch
activity={0121999239923923, 02939293}
BPM = {74}
Logical Clock
Kinect Camera
activity={0121999239923923, 02939293}
User Activity registered in Smart phone and Kinect Camera.
Create
Data Writer
EP. 5
Recv.

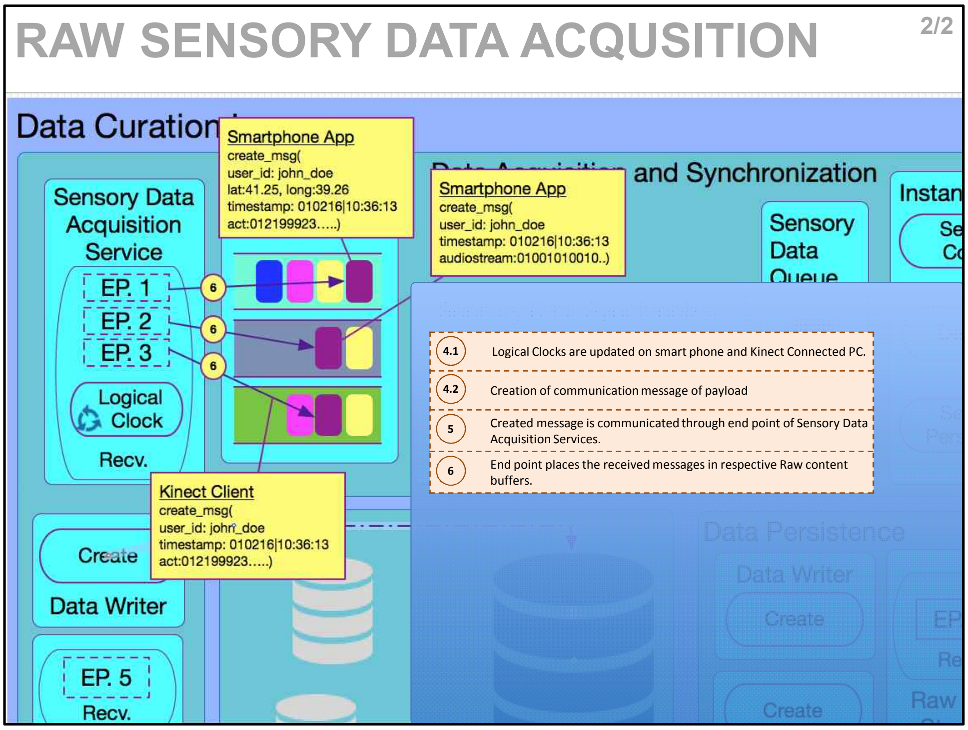

LOGICAL CLOCK SYNCHRONIZATION
2/4
Data Curation Layer
Smartwatch
activity={0121999239923923, 02939293}
BPM = {74}
RTS
RTS
Sensory Data Acquisition Service
EP. 1
EP. 2
EP. 3
Logical Clock
Recv.
User Activity registered in Smart phone and Kinect Camera.
Real Time Synchronization of logical clocks
Kinect Camera
activity={0121999239923923, 02939293}
Create
Data Writer
EP. 5
Recv.

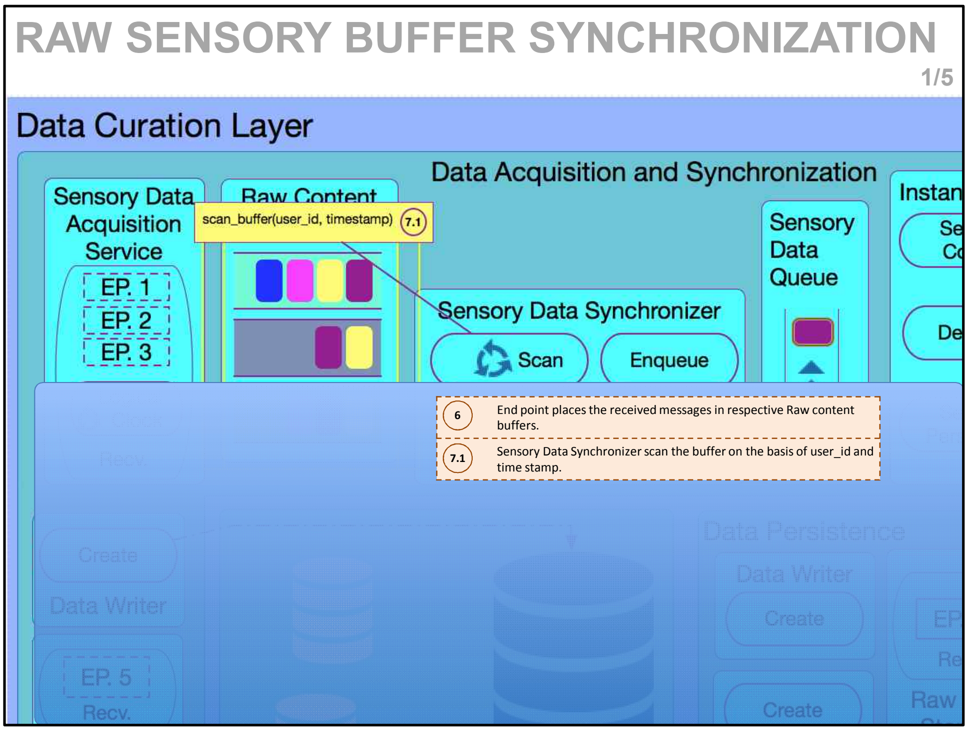

LOGICAL CLOCK SYNCHRONIZATION
3/4
Data Curation Layer
Sensory Data Acquisition Service
EP. 1
EP. 2
EP. 3
Logical Clock
Recv.
1 User Activity registered in Smart phone and Kinect Camera.
2 Real Time Synchronization of logical clocks
3 Share Synchronized time with smart phone and Kinect connected PC
LC:010216|10:36:13
LC:010216|10:36:13
3
3
Logical Clock
Logical Clock
Create Instance
Create
Data Writer
EP. 5
Recv.


LOGICAL CLOCK SYNCHRONIZATION
4/4
Data Curation Layer
Logical Clock 4.1
old_val:010216|08:33:45
cur_val:010216|10:36:13
Sensory Data Acquisition Service
EP. 1
EP. 2
EP. 3
Logical Clock
Recv.
1 User Activity registered in Smart phone and Kinect Camera.
2 Real Time Synchronization of logical clocks
3 Share Synchronized time with smart phone and Kinect connected PC
4.1 Logical Clocks are updated on smart phone and Kinect Connected PC.
LC:010216|10:36:13
LC:010216|10:36:13
3
3
Logical Clock
Logical Clock
Create Instance
Create
Data Writer
EP. 5
Recv.
Logical Clock 4.1
old_val:010216|08:33:45
cur_val:010216|10:36:13


MESSAGE GENERATION
1/1
Data Curation Layer
Smartphone App
create_msg(
user_id: john_doe
lat:41.25, long:39.26
timestamp: 010216|10:36:13
act:012199923.....)
Smartphone App
create_msg(
user_id: john_doe
timestamp: 010216|10:36:13
audiostream:01001010010..)
Kinect Client
create_msg(
user_id: john_doe
timestamp: 010216|10:36:13
act:012199923.....)
Logical Clock
Sensory Data Acquisition Service
EP. 1
EP. 2
EP. 3
Logical Clock
Recv.
4.1 Logical Clocks are updated on smart phone and Kinect Connected PC.
4.2 Creation of communication message of payload
Create
Data Writer
EP. 5
Recv.
Scan
Create Instance

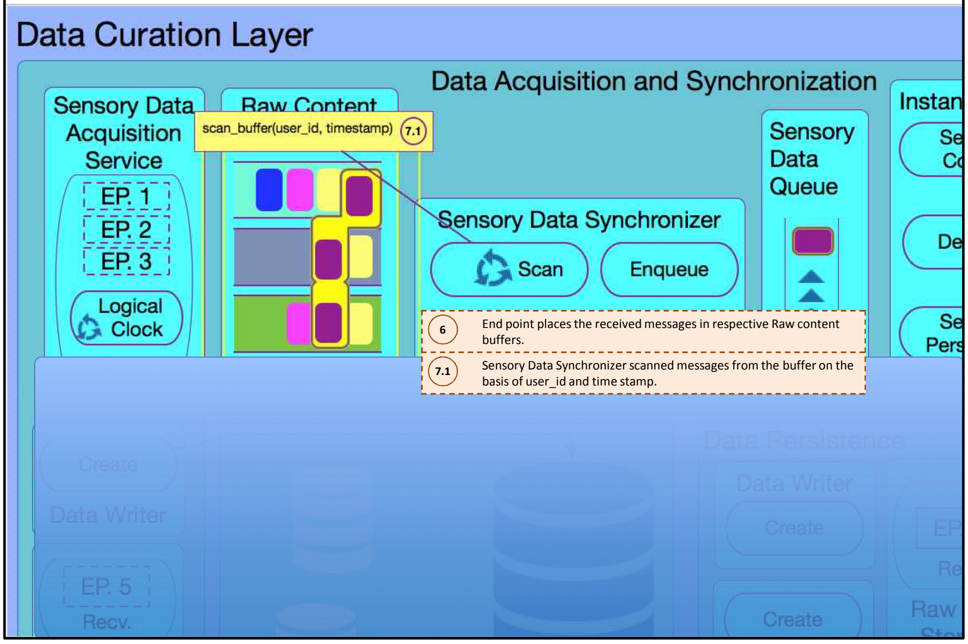

RAW SENSORY DATA ACQUSITION
1/2
Data Curation Layer
Smartphone App
create_msg(
user_id: john_doe
lat:41.25, long:39.26
timestamp: 010216|10:36:13
act:012199923.....)
Smartphone App
create_msg(
user_id: john_doe
timestamp: 010216|10:36:13
audiostream:01001010010..)
Kinect Client
create_msg(
user_id: john_doe
timestamp: 010216|10:36:13
act:012199923.....)
Logical Clock
5
Sensory Data Acquisition Service
EP. 1
EP. 2
EP. 3
Logical Clock
Recv.
4.1 Logical Clocks are updated on smart phone and Kinect Connected PC.
4.2 Creation of communication message of payload
5 Created message is communicated through end point of Sensory Data Acquisition Services.
Create
Data Writer
EP. 5
Recv.
Create Instance


RAW SENSORY DATA ACQUSITION
2/2
Data Curation
Smartphone App
create_msg(
user_id: john_doe
lat:41.25, long:39.26
timestamp: 010216|10:36:13
act:012199923…..)
and Synchronization
Smartphone App
create_msg(
user_id: john_doe
timestamp: 010216|10:36:13
audiostream:01001010010..)
Sensory Data Acquisition Service
EP. 1
EP. 2
EP. 3
Logical Clock
Recv.
Sensory Data Queue
Instan
4.1 Logical Clocks are updated on smart phone and Kinect Connected PC.
4.2 Creation of communication message of payload
5 Created message is communicated through end point of Sensory Data Acquisition Services.
6 End point places the received messages in respective Raw content buffers.
Kinect Client
create_msg(
user_id: john_doe
timestamp: 010216|10:36:13
act:012199923…..)
Create
Data Writer
EP. 5
Recv.
Data Persistence
Data Writer
Create
Raw


RAW SENSORY BUFFER SYNCHRONIZATION
1/5
Data Curation Layer
Data Acquisition and Synchronization
Sensory Data Acquisition Service
Raw Content
scan_buffer(user_id, timestamp) 7.1
EP. 1
EP. 2
EP. 3
Sensory Data Synchronizer
Scan
Enqueue
Sensory Data Queue
Instan
6 End point places the received messages in respective Raw content buffers.
7.1 Sensory Data Synchronizer scan the buffer on the basis of user_id and time stamp.
Create
Data Writer
EP. 5
Recv.
Data Persistence
Data Writer
Create
Raw

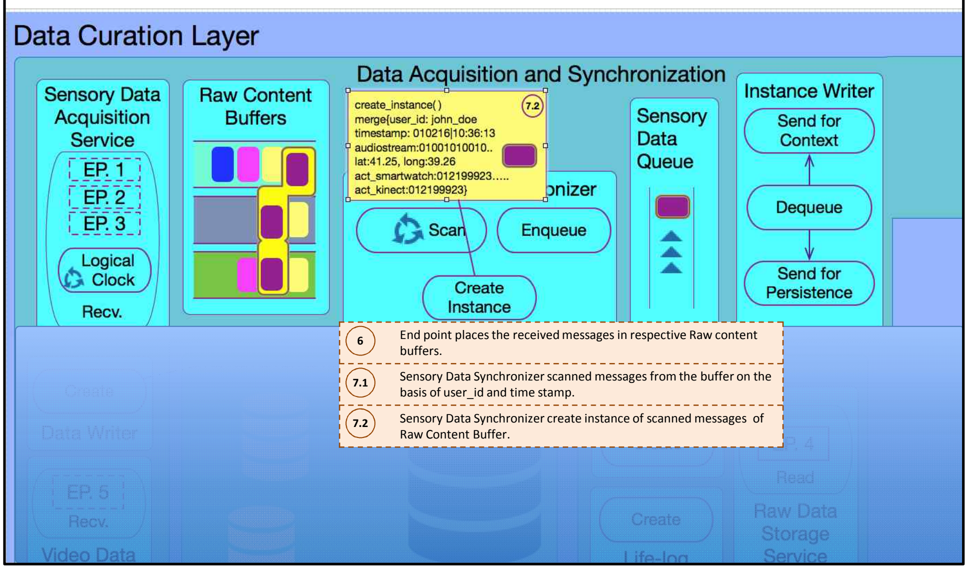
RAW SENSORY BUFFER SYNCHRONIZATION

2/5



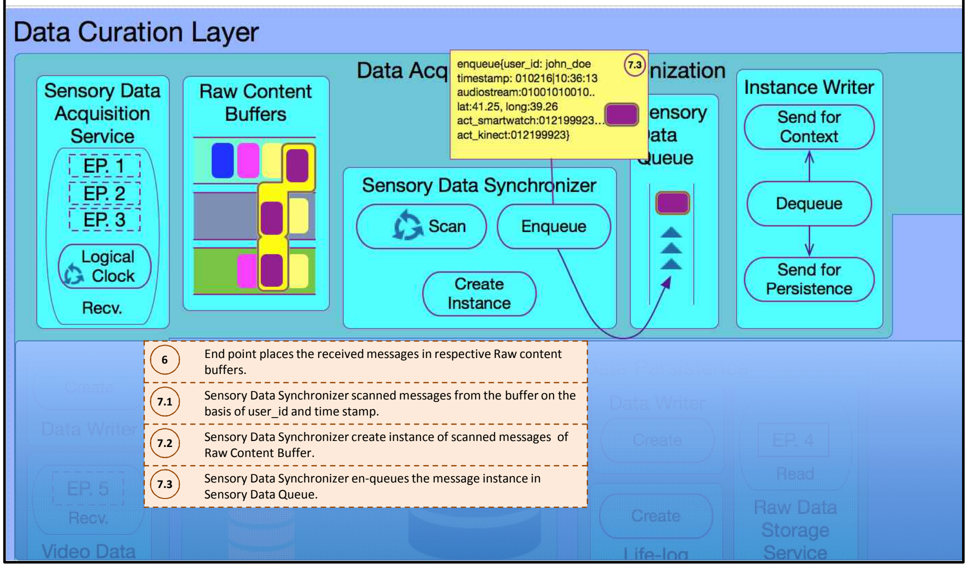
RAW SENSORY BUFFER SYNCHRONIZATION

3/5



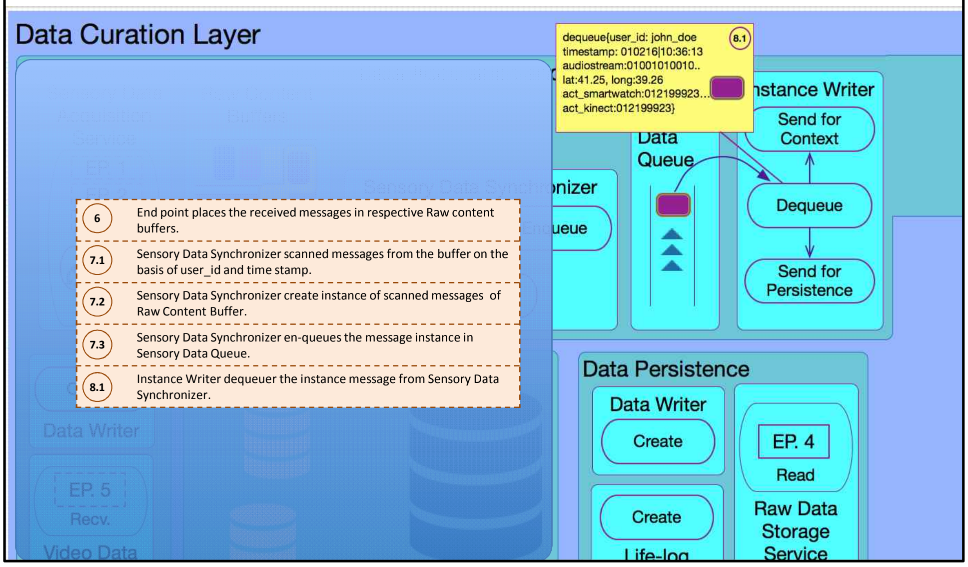
RAW SENSORY BUFFER SYNCHRONIZATION

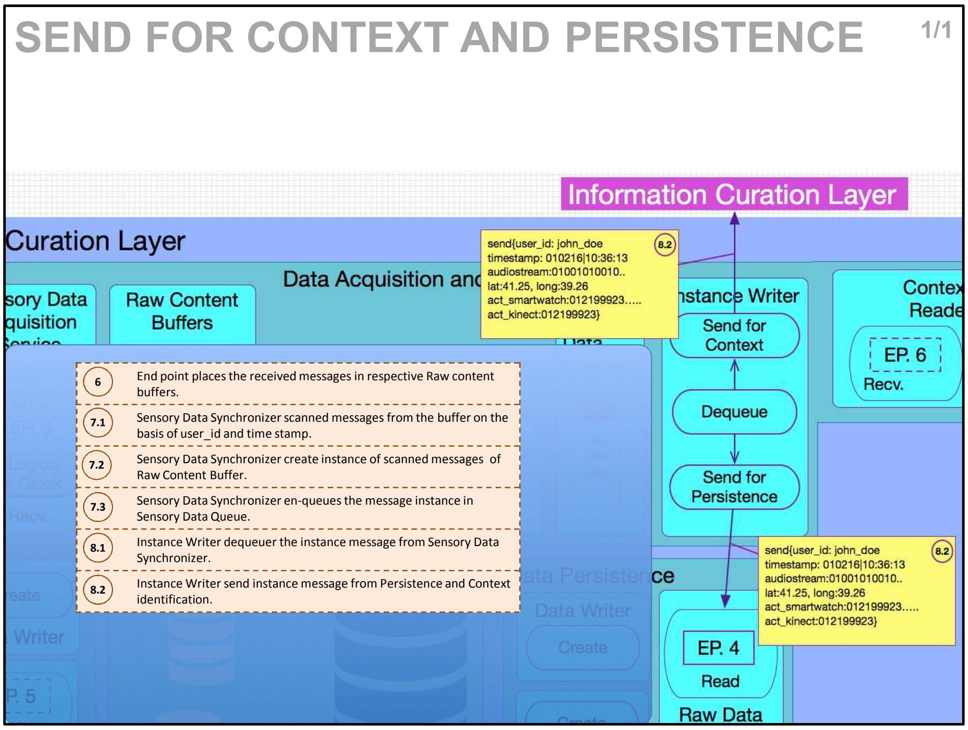
4/5

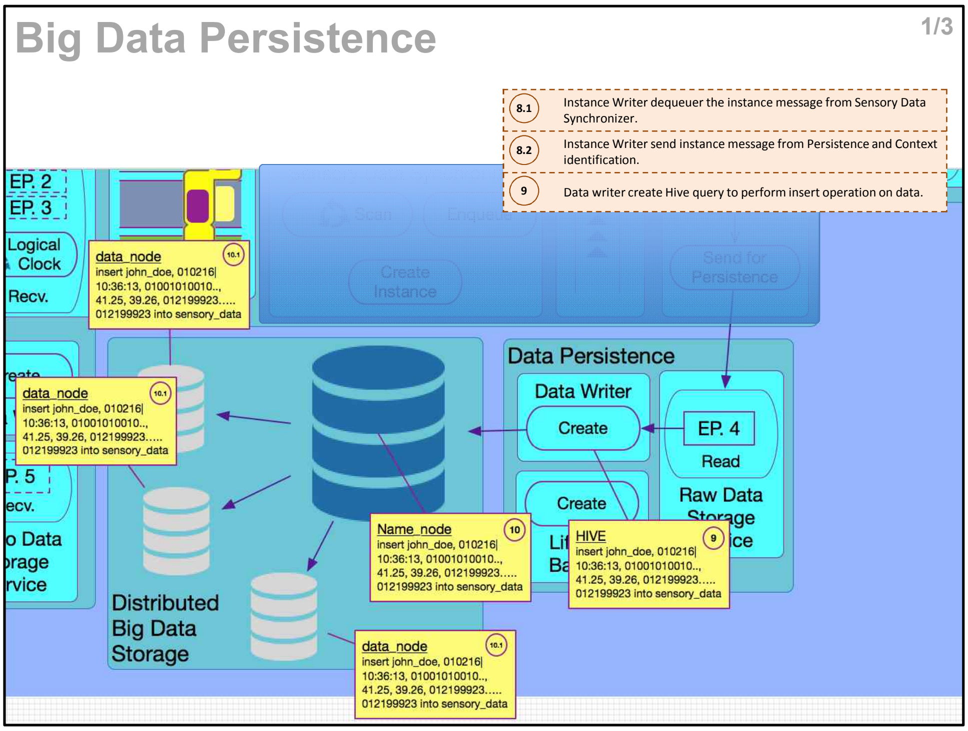


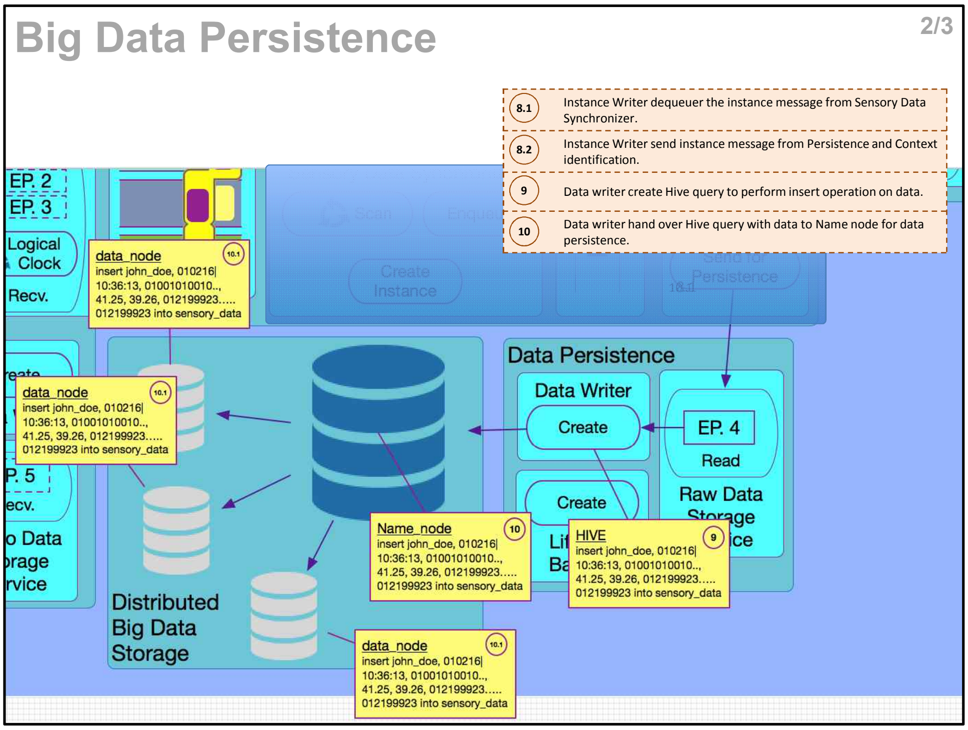
RAW SENSORY BUFFER SYNCHRONIZATION

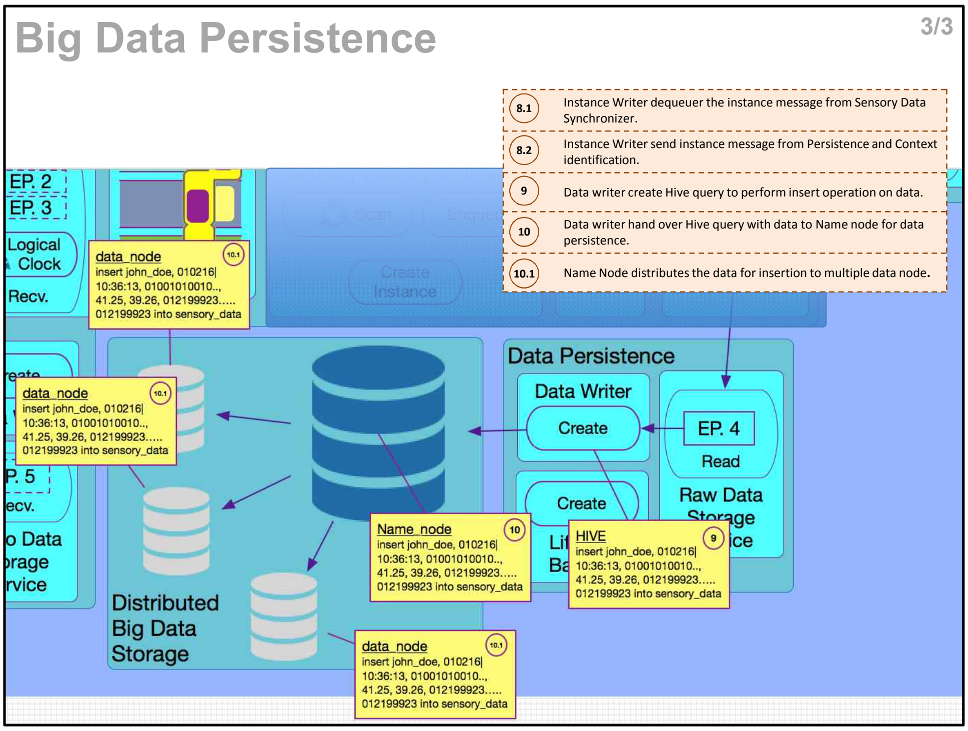
5/5




SEND FOR CONTEXT AND PERSISTENCE
1/1
Information Curation Layer
Curation Layer
sory Data
quisition
Raw Content
Buffers
Data Acquisition and
send{user_id: john_doe
timestamp: 010216|10:36:13
audiostream:01001010010..
lat:41.25, long:39.26
act_smartwatch:012199923.....
act_kinect:012199923}
8.2
Instance Writer
Send for
Context
Dequeue
Send for
Persistence
Context
Reader
EP. 6
Recv.
6 End point places the received messages in respective Raw content buffers.
7.1 Sensory Data Synchronizer scanned messages from the buffer on the basis of user_id and time stamp.
7.2 Sensory Data Synchronizer create instance of scanned messages of Raw Content Buffer.
7.3 Sensory Data Synchronizer en-queues the message instance in Sensory Data Queue.
8.1 Instance Writer dequeuer the instance message from Sensory Data Synchronizer.
8.2 Instance Writer send instance message from Persistence and Context identification.
send{user_id: john_doe
timestamp: 010216|10:36:13
audiostream:01001010010..
lat:41.25, long:39.26
act_smartwatch:012199923.....
act_kinect:012199923}
8.2
Data Persistence
Data Writer
Create
EP. 4
Read
Raw Data
Writer
P. 5

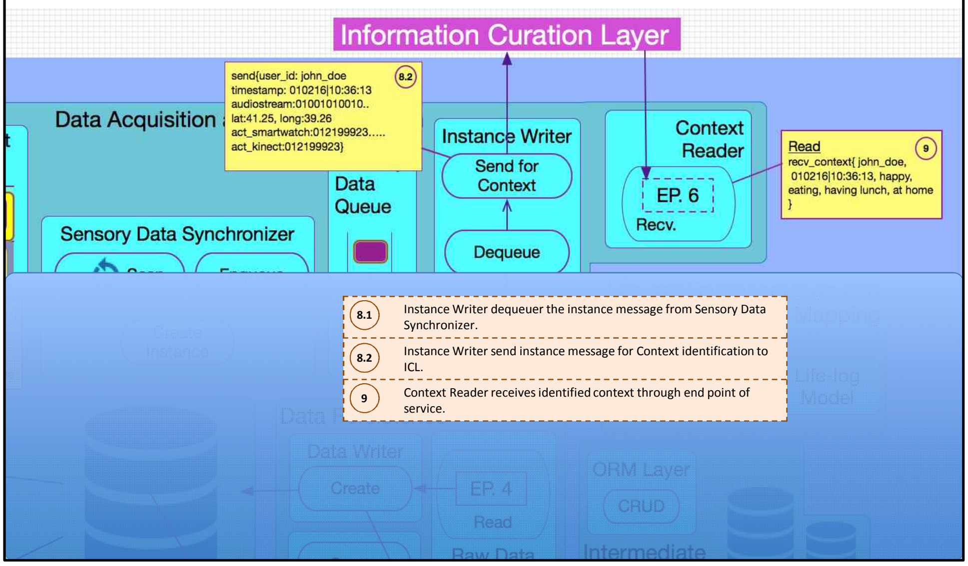

Big Data Persistence
1/3
8.1 Instance Writer dequeuer the instance message from Sensory Data Synchronizer.
8.2 Instance Writer send instance message from Persistence and Context identification.
9 Data writer create Hive query to perform insert operation on data.
EP. 2
EP. 3
Logical
Clock
Recv.
data_node
10.1
insert john_doe, 010216|
10:36:13, 01001010010..,
41.25, 39.26, 012199923.....
012199923 into sensory_data
Create
Instance
Send for
Persistence
Data Persistence
Data Writer
Create
EP. 4
Read
Raw Data
Storage
Service
Create
data_node
10.1
insert john_doe, 010216|
10:36:13, 01001010010..,
41.25, 39.26, 012199923.....
012199923 into sensory_data
P. 5
ecv.
o Data
orage
rvice
Name_node
10
insert john_doe, 010216|
10:36:13, 01001010010..,
41.25, 39.26, 012199923.....
012199923 into sensory_data
HIVE
9
insert john_doe, 010216|
10:36:13, 01001010010..,
41.25, 39.26, 012199923.....
012199923 into sensory_data
Distributed
Big Data
Storage
data_node
10.1
insert john_doe, 010216|
10:36:13, 01001010010..,
41.25, 39.26, 012199923.....
012199923 into sensory_data


Big Data Persistence
2/3
8.1 Instance Writer dequeuer the instance message from Sensory Data Synchronizer.
8.2 Instance Writer send instance message from Persistence and Context identification.
9 Data writer create Hive query to perform insert operation on data.
10 Data writer hand over Hive query with data to Name node for data persistence.
EP. 2
EP. 3
Logical Clock
Recv.
data_node
insert john_doe, 010216| 10:36:13, 01001010010.., 41.25, 39.26, 012199923..... 012199923 into sensory_data
Create Instance
Data Persistence
Data Writer
Create
EP. 4
Read
Raw Data Storage
Name_node
HIVE
Distributed Big Data Storage


Big Data Persistence
3/3
8.1 Instance Writer dequeuer the instance message from Sensory Data Synchronizer.
8.2 Instance Writer send instance message from Persistence and Context identification.
9 Data writer create Hive query to perform insert operation on data.
10 Data writer hand over Hive query with data to Name node for data persistence.
10.1 Name Node distributes the data for insertion to multiple data node.
EP. 2
EP. 3
Logical Clock
Recv.
data_node
insert john_doe, 010216| 10:36:13, 01001010010.., 41.25, 39.26, 012199923..... 012199923 into sensory_data
Create Instance
Data Persistence
Data Writer
Create
EP. 4
Read
Raw Data Storage
Name_node
HIVE
Distributed Big Data Storage

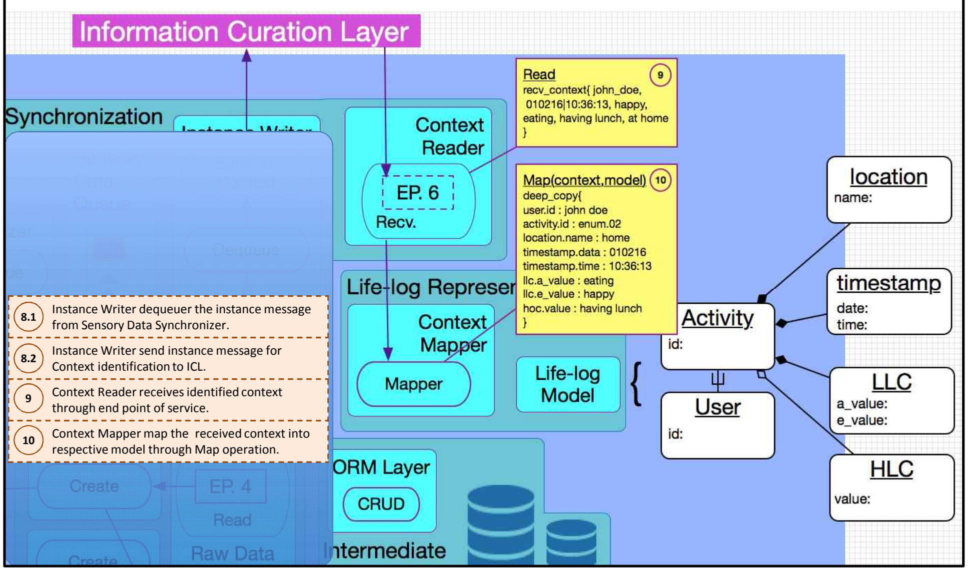
Context Mapping and Persistence

1/8



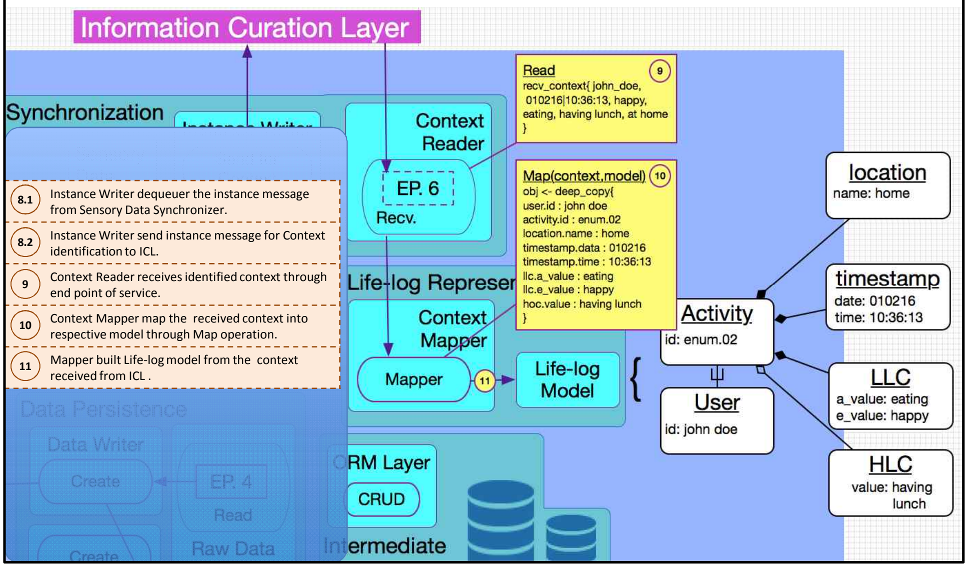
Context Mapping and Persistence

2/8



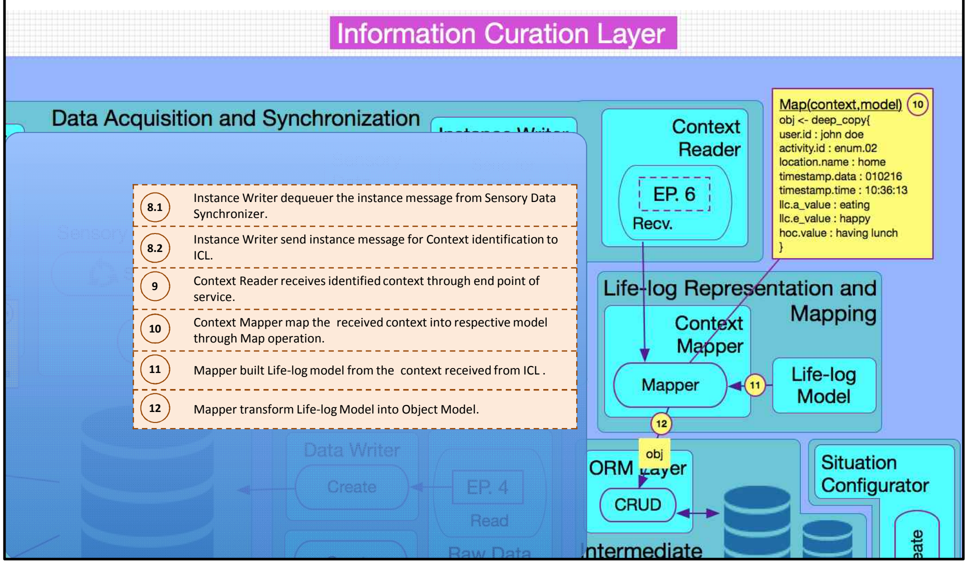
Context Mapping and Persistence

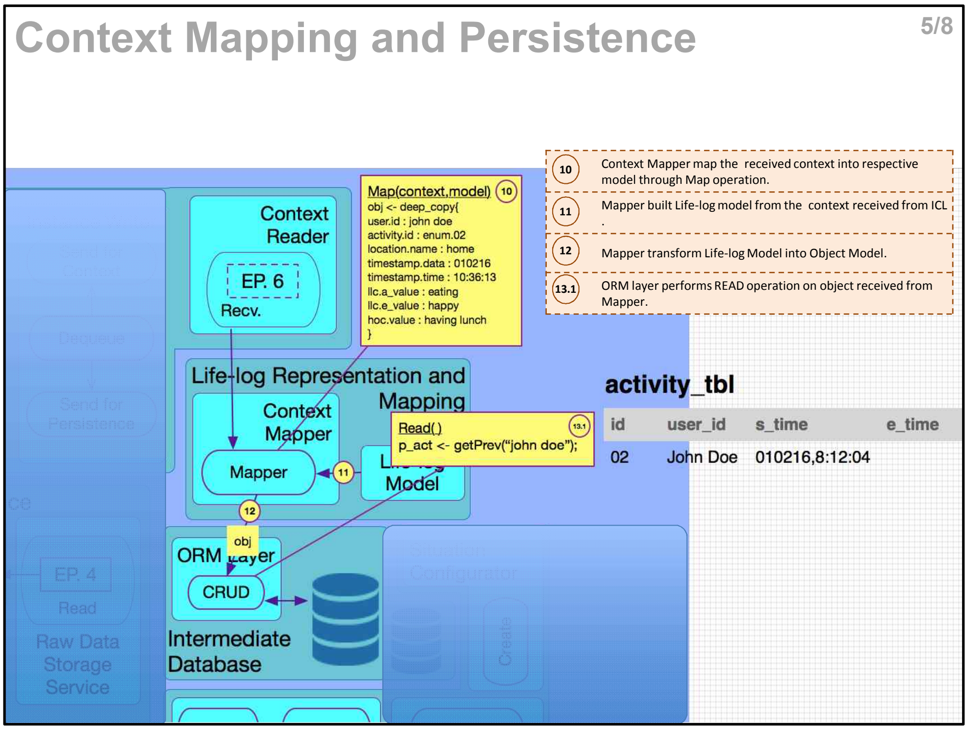
3/8

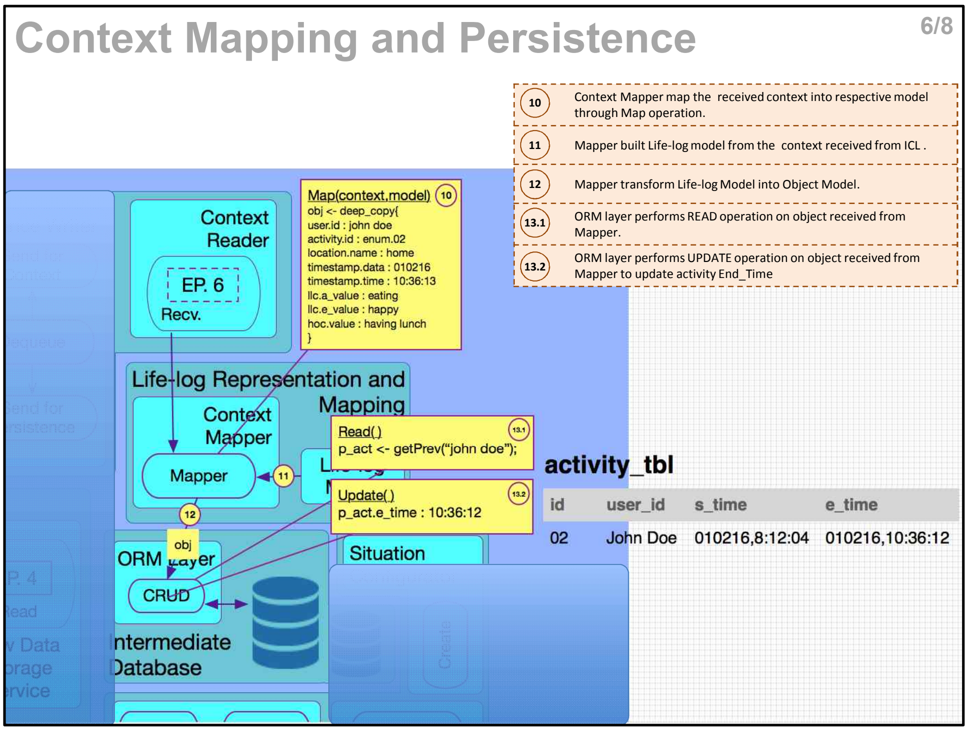


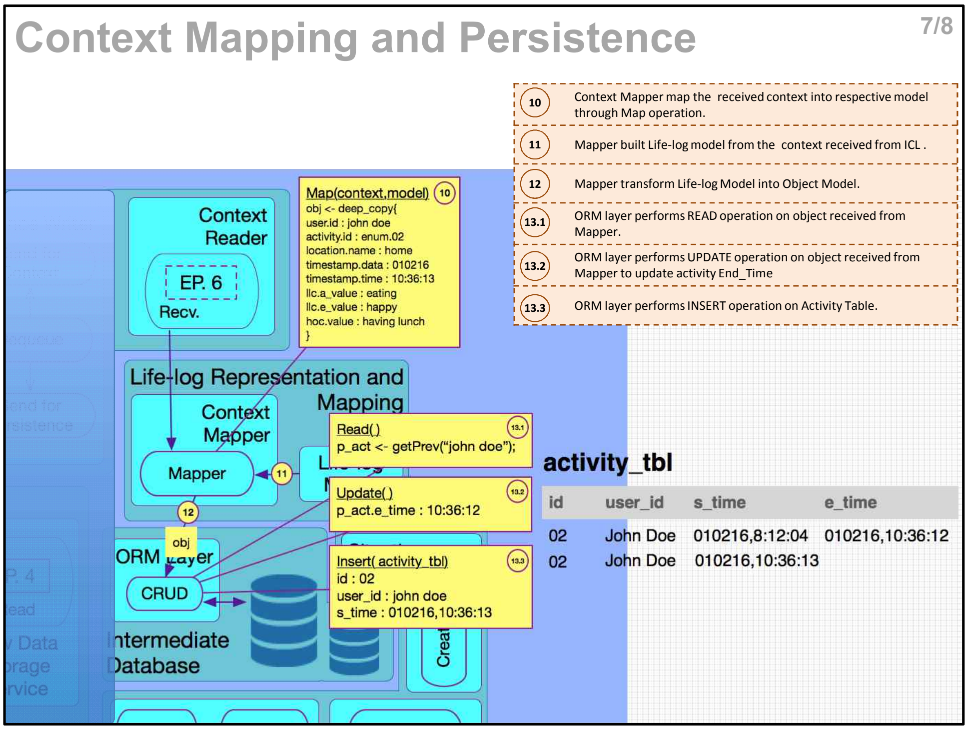
Context Mapping and Persistence

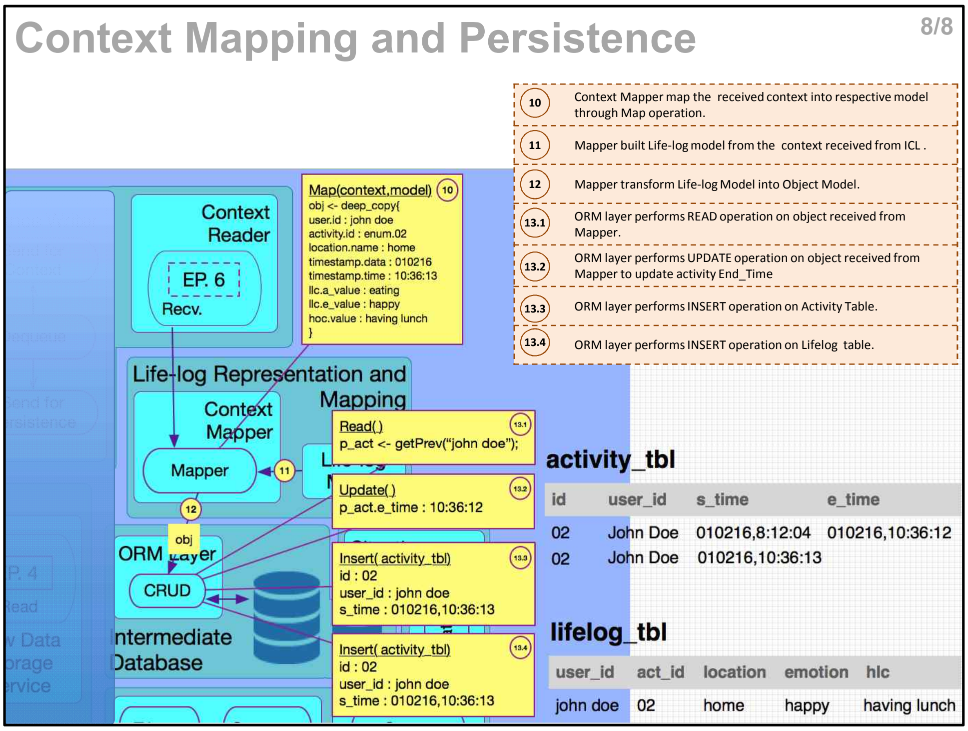
4/8

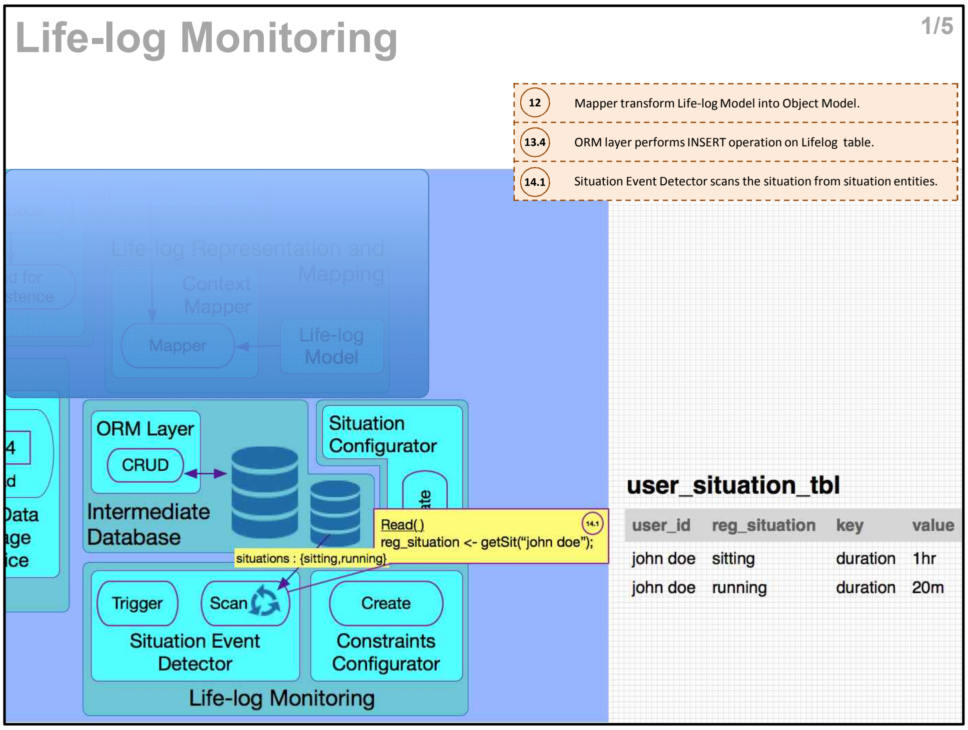


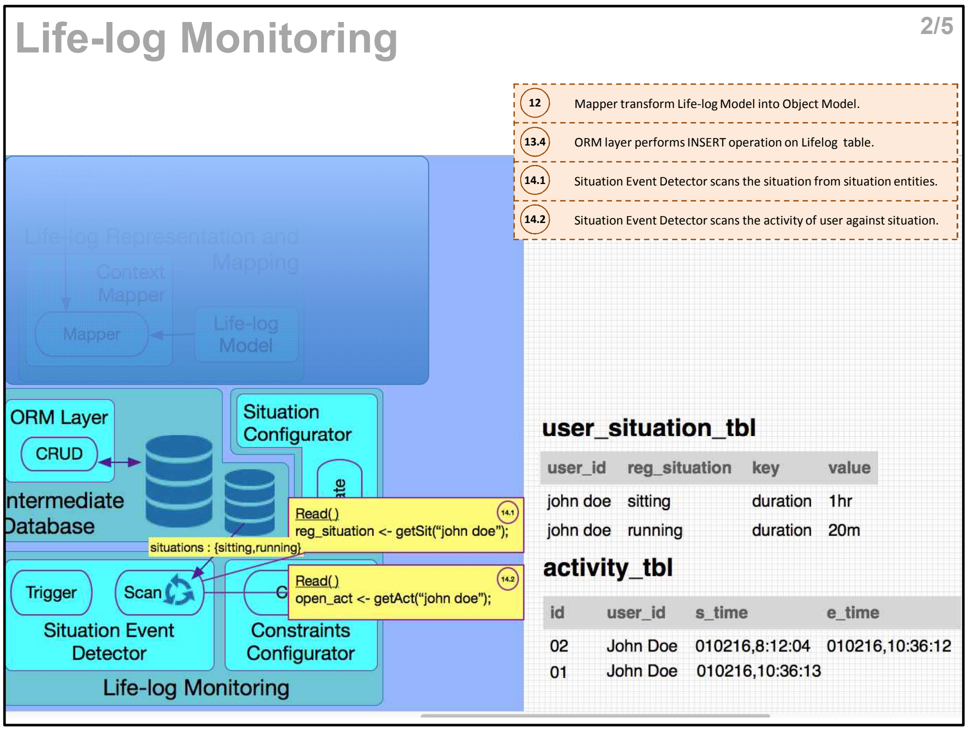

Context Mapping and Persistence
5/8
Context Reader
EP. 6
Recv.
Map(context,model) 10
obj <- deep_copy{
user.id : john doe
activity.id : enum.02
location.name : home
timestamp.data : 010216
timestamp.time : 10:36:13
llc.a_value : eating
llc.e_value : happy
hoc.value : having lunch
}
10 Context Mapper map the received context into respective model through Map operation.
11 Mapper built Life-log model from the context received from ICL .
12 Mapper transform Life-log Model into Object Model.
13.1 ORM layer performs READ operation on object received from Mapper.
Life-log Representation and Mapping
Context Mapper
Mapper
11
Model
Read()
p_act <- getPrev("john doe");
12
obj
ORM Layer
CRUD
Intermediate Database
Raw Data Storage Service
activity_tbl
id | user_id | s_time | e_time
02 | John Doe | 010216,8:12:04 |

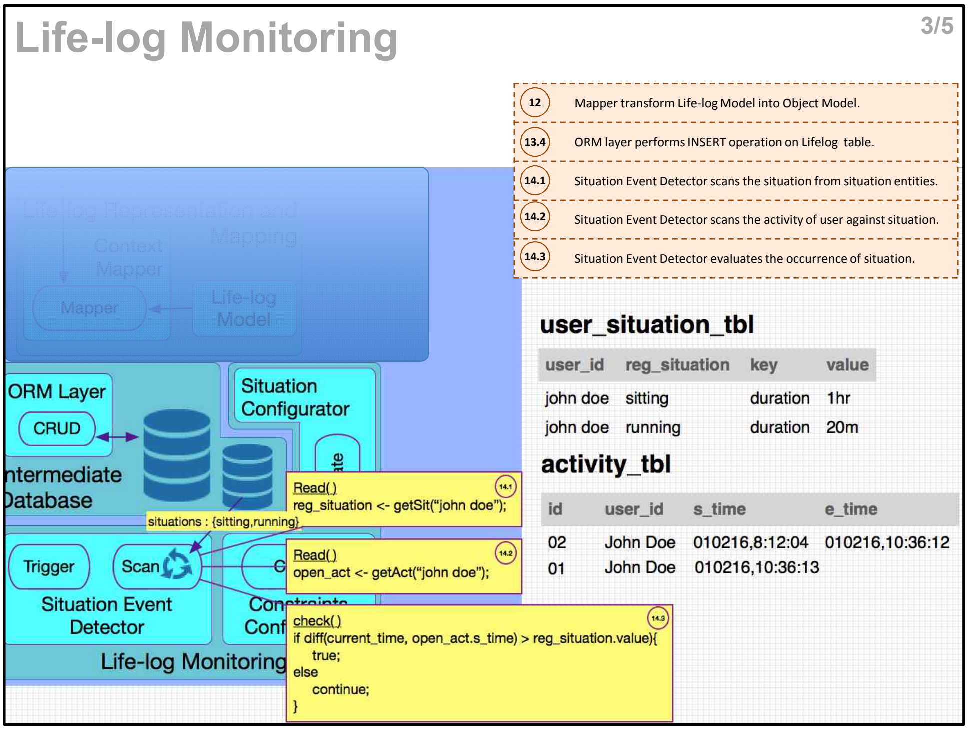

Context Mapping and Persistence
6/8
Context Reader
EP. 6
Recv.
Map(context,model) 10
obj <- deep_copy{
user.id : john doe
activity.id : enum.02
location.name : home
timestamp.data : 010216
timestamp.time : 10:36:13
llc.a_value : eating
llc.e_value : happy
hoc.value : having lunch
}
10 Context Mapper map the received context into respective model through Map operation.
11 Mapper built Life-log model from the context received from ICL .
12 Mapper transform Life-log Model into Object Model.
13.1 ORM layer performs READ operation on object received from Mapper.
13.2 ORM layer performs UPDATE operation on object received from Mapper to update activity End_Time
Life-log Representation and Mapping
Context Mapper
Mapper
11
Read()
p_act <- getPrev("john doe");
Update()
p_act.e_time : 10:36:12
12
obj
ORM Layer
CRUD
Situation
Intermediate Database
activity_tbl
id | user_id | s_time | e_time
02 | John Doe | 010216,8:12:04 | 010216,10:36:12

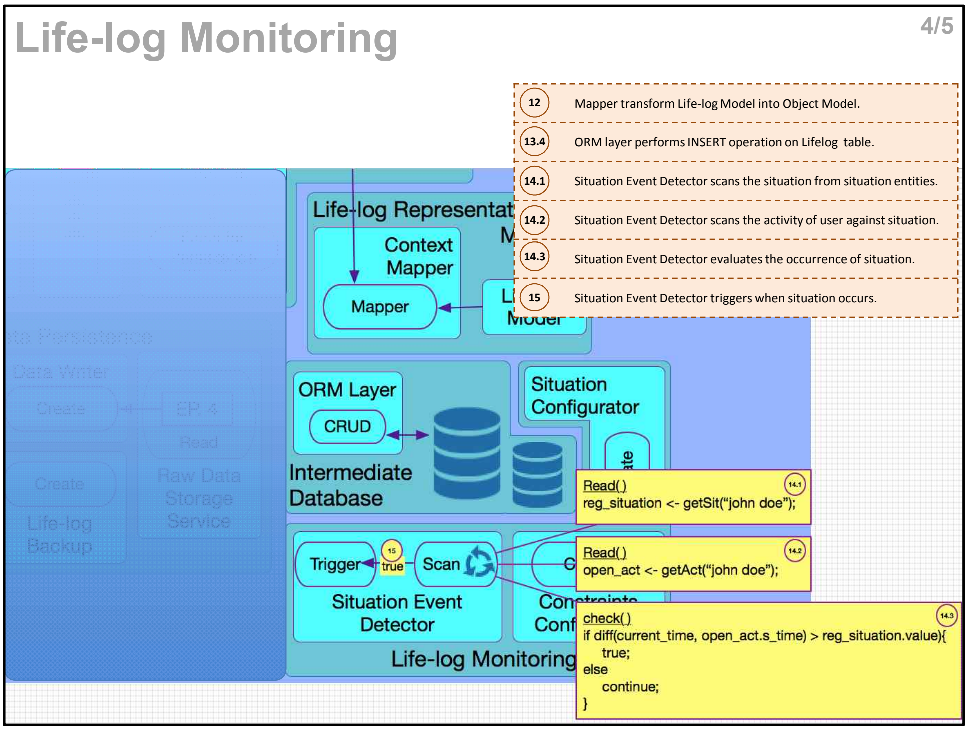

Context Mapping and Persistence
7/8
10 Context Mapper map the received context into respective model through Map operation.
11 Mapper built Life-log model from the context received from ICL .
12 Mapper transform Life-log Model into Object Model.
13.1 ORM layer performs READ operation on object received from Mapper.
13.2 ORM layer performs UPDATE operation on object received from Mapper to update activity End_Time
13.3 ORM layer performs INSERT operation on Activity Table.
Context Reader
EP. 6
Recv.
Map(context,model) 10
obj <- deep_copy{
user.id : john doe
activity.id : enum.02
location.name : home
timestamp.data : 010216
timestamp.time : 10:36:13
llc.a_value : eating
llc.e_value : happy
hoc.value : having lunch
}
Life-log Representation and Mapping
Context Mapper
Mapper
11
12
obj
ORM Layer
CRUD
Intermediate Database
Read()
p_act <- getPrev("john doe");
Update()
p_act.e_time : 10:36:12
Insert(activity_tbl)
id : 02
user_id : john doe
s_time : 010216,10:36:13
activity_tbl
id user_id s_time e_time
02 John Doe 010216,8:12:04 010216,10:36:12
02 John Doe 010216,10:36:13


Context Mapping and Persistence
8/8
10 Context Mapper map the received context into respective model through Map operation.
11 Mapper built Life-log model from the context received from ICL .
12 Mapper transform Life-log Model into Object Model.
13.1 ORM layer performs READ operation on object received from Mapper.
13.2 ORM layer performs UPDATE operation on object received from Mapper to update activity End_Time
13.3 ORM layer performs INSERT operation on Activity Table.
13.4 ORM layer performs INSERT operation on Lifelog table.
Context Reader
EP. 6
Recv.
Map(context,model) 10
obj <- deep_copy{
user.id : john doe
activity.id : enum.02
location.name : home
timestamp.data : 010216
timestamp.time : 10:36:13
llc.a_value : eating
llc.e_value : happy
hoc.value : having lunch
}
Life-log Representation and Mapping
Context Mapper
Mapper
11
12
obj
ORM Layer
CRUD
Intermediate Database
Read()
p_act <- getPrev("john doe");
Update()
p_act.e_time : 10:36:12
Insert(activity_tbl)
id : 02
user_id : john doe
s_time : 010216,10:36:13
Insert(activity_tbl)
id : 02
user_id : john doe
s_time : 010216,10:36:13
activity_tbl
id user_id s_time e_time
02 John Doe 010216,8:12:04 010216,10:36:12
02 John Doe 010216,10:36:13
lifelog_tbl
user_id act_id location emotion hlc
john doe 02 home happy having lunch


Life-log Monitoring
1/5
12 Mapper transform Life-log Model into Object Model.
13.4 ORM layer performs INSERT operation on Lifelog table.
14.1 Situation Event Detector scans the situation from situation entities.
ORM Layer
CRUD
Intermediate Database
Situation Configurator
Read()
reg_situation <- getSit("john doe");
situations : {sitting,running}
Trigger
Scan
Situation Event Detector
Create
Constraints Configurator
Life-log Monitoring
user_situation_tbl
user_id reg_situation key value
john doe sitting duration 1hr
john doe running duration 20m

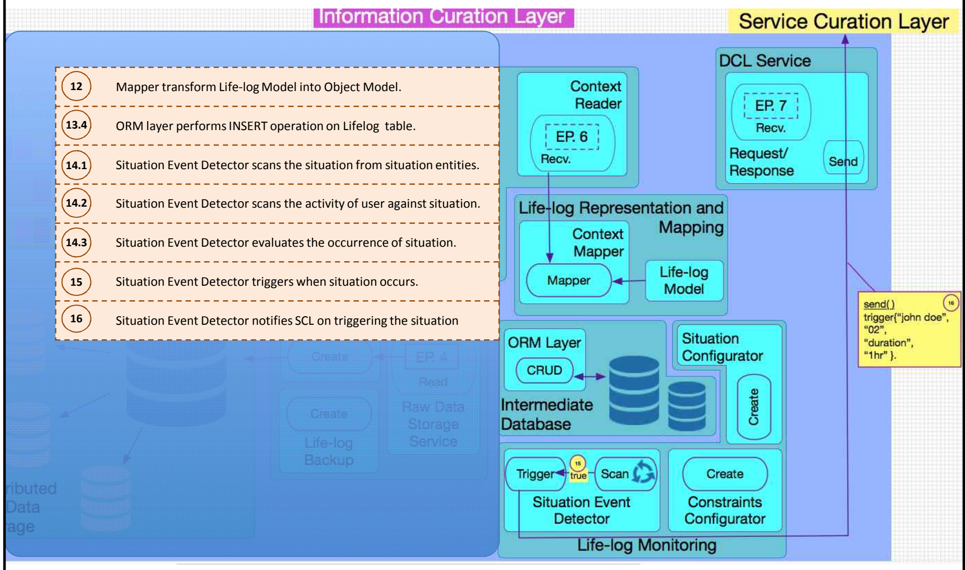

Life-log Monitoring
2/5
12 Mapper transform Life-log Model into Object Model.
13.4 ORM layer performs INSERT operation on Lifelog table.
14.1 Situation Event Detector scans the situation from situation entities.
14.2 Situation Event Detector scans the activity of user against situation.
ORM Layer
CRUD
ntermediate Database
Situation Configurator
Read()
reg_situation <- getSit("john doe");
situations : {sitting,running}
Read()
open_act <- getAct("john doe");
Trigger
Scan
Situation Event Detector
Constraints Configurator
Life-log Monitoring
user_situation_tbl
user_id reg_situation key value
john doe sitting duration 1hr
john doe running duration 20m
activity_tbl
id user_id s_time e_time
02 John Doe 010216,8:12:04 010216,10:36:12
01 John Doe 010216,10:36:13


Life-log Monitoring
3/5
12 Mapper transform Life-log Model into Object Model.
13.4 ORM layer performs INSERT operation on Lifelog table.
14.1 Situation Event Detector scans the situation from situation entities.
14.2 Situation Event Detector scans the activity of user against situation.
14.3 Situation Event Detector evaluates the occurrence of situation.
user_situation_tbl
user_id reg_situation key value
john doe sitting duration 1hr
john doe running duration 20m
activity_tbl
id user_id s_time e_time
02 John Doe 010216,8:12:04 010216,10:36:12
01 John Doe 010216,10:36:13
ORM Layer
CRUD
ntermediate
Database
Situation
Configurator
situations : {sitting,running}
Read()
reg_situation <- getSit("john doe");
Read()
open_act <- getAct("john doe");
Trigger
Scan
Situation Event
Detector
check()
if diff(current_time, open_act.s_time) > reg_situation.value){
true;
else
continue;
}
Life-log Monitoring


Life-log Monitoring
4/5
12 Mapper transform Life-log Model into Object Model.
13.4 ORM layer performs INSERT operation on Lifelog table.
14.1 Situation Event Detector scans the situation from situation entities.
14.2 Situation Event Detector scans the activity of user against situation.
14.3 Situation Event Detector evaluates the occurrence of situation.
15 Situation Event Detector triggers when situation occurs.
Life-log Representat
Context
Mapper
Mapper
ORM Layer
CRUD
Intermediate
Database
Situation
Configurator
Read()
reg_situation <- getSit("john doe");
Read()
open_act <- getAct("john doe");
Trigger
true
Scan
Situation Event
Detector
check()
if diff(current_time, open_act.s_time) > reg_situation.value){
true;
else
continue;
}
Life-log Monitoring

Life-log Monitoring

5/5

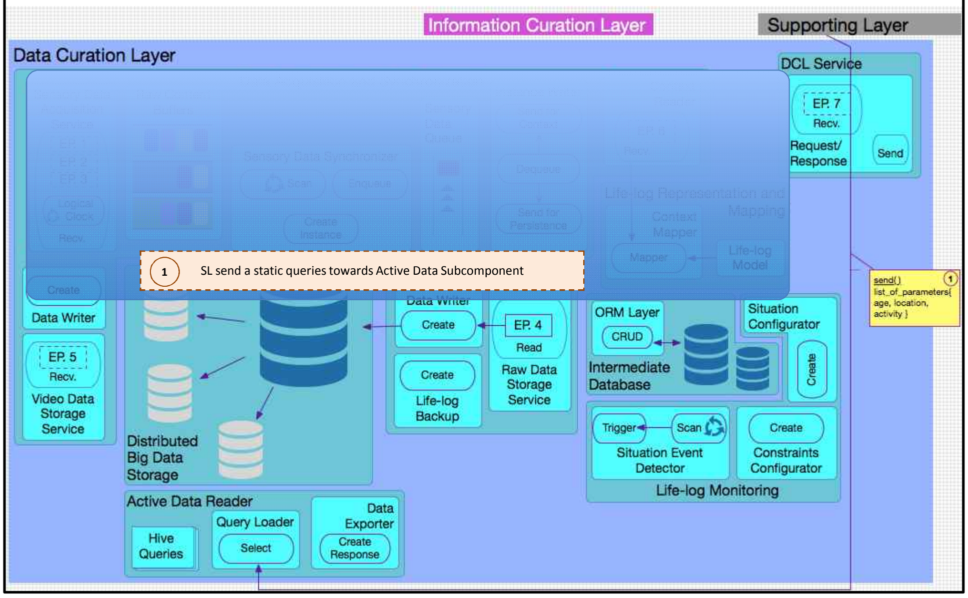


Scenario 2: Real-time

Desc. : Raw-sensory data from big data storage is requested for **visualization** and **analytics**.

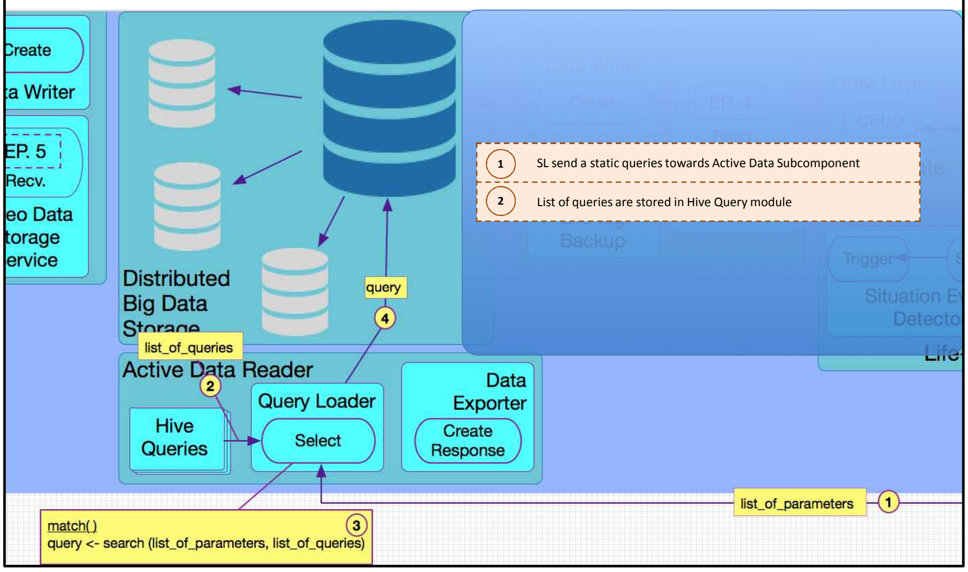
SL Request of Parameters

1/1



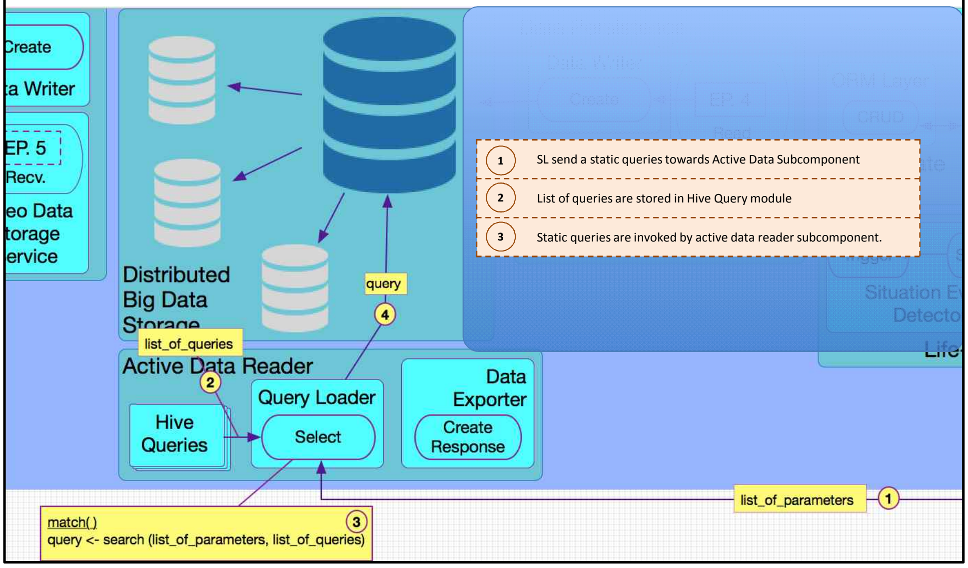
Hive Query Selection

1/3



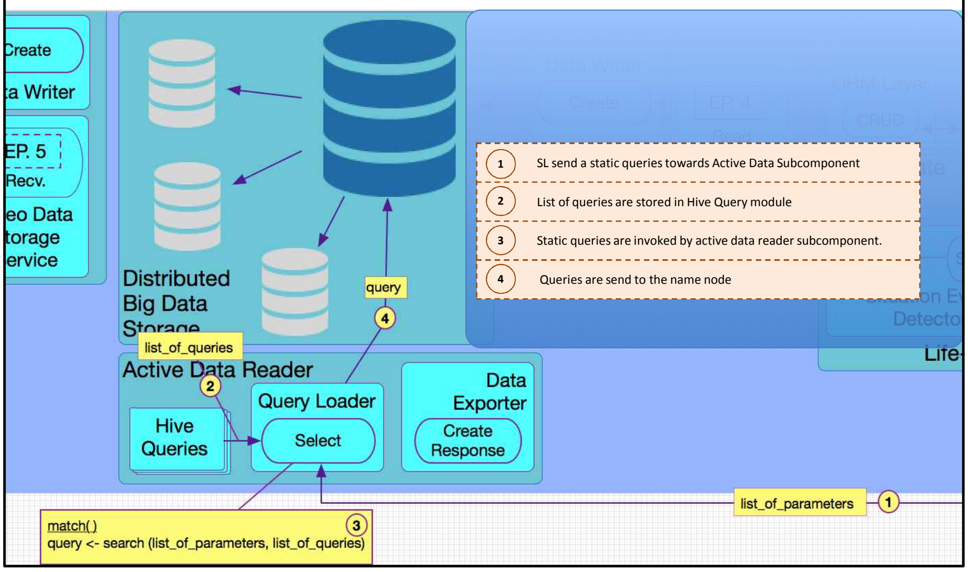
Hive Query Selection

2/3



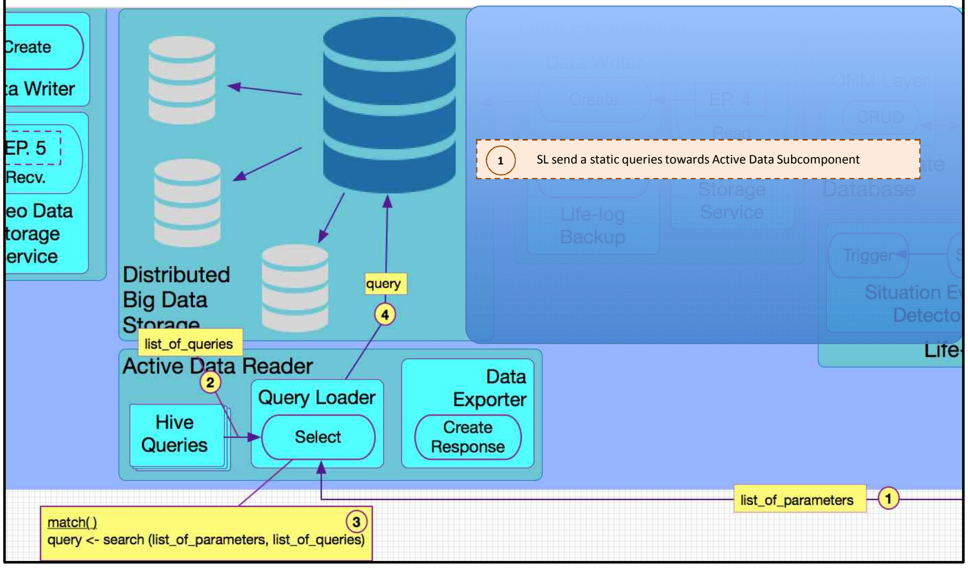
Hive Query Selection

3/3



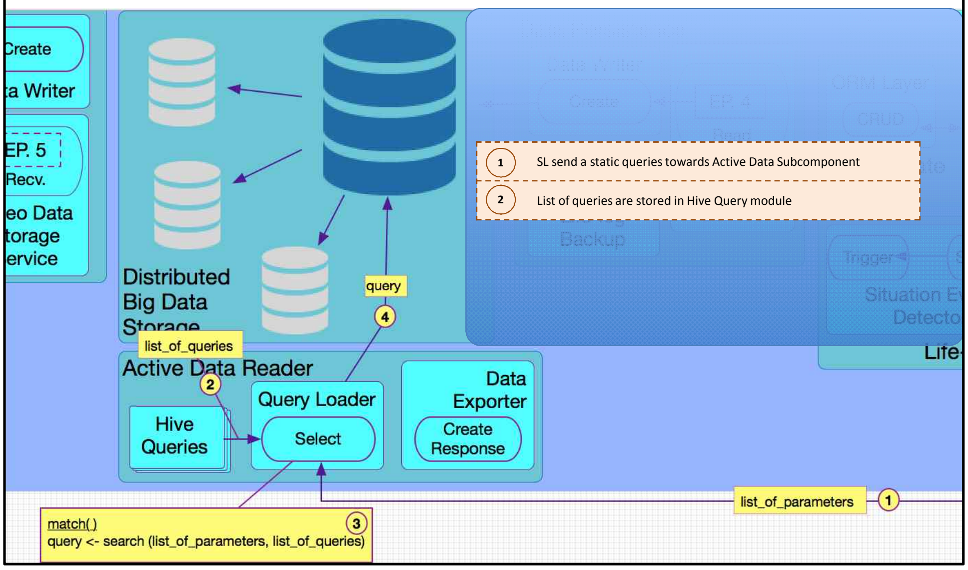
Query Execution and Response Generation

1/8



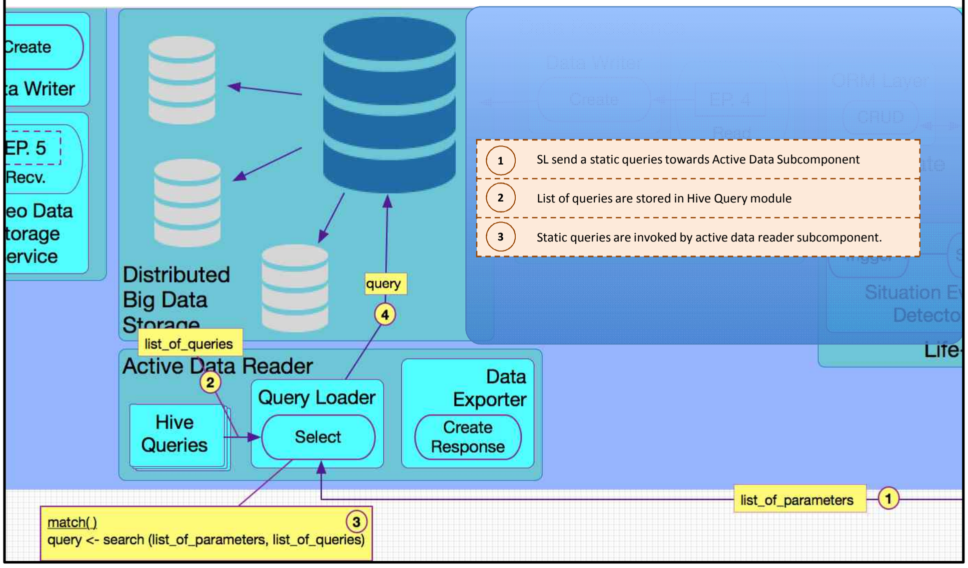
Query Execution and Response Generation

2/8



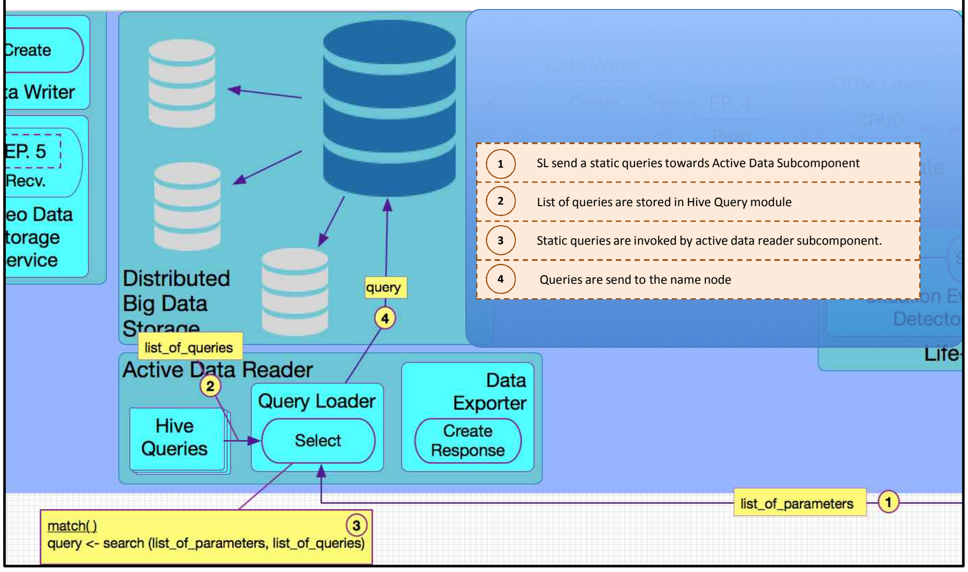
Query Execution and Response Generation

3/8



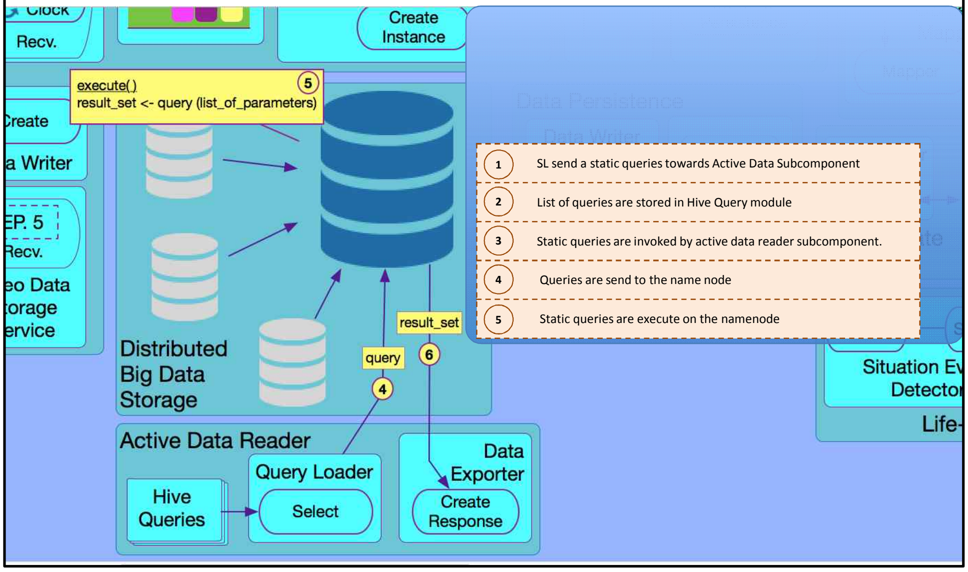
Query Execution and Response Generation

4/8



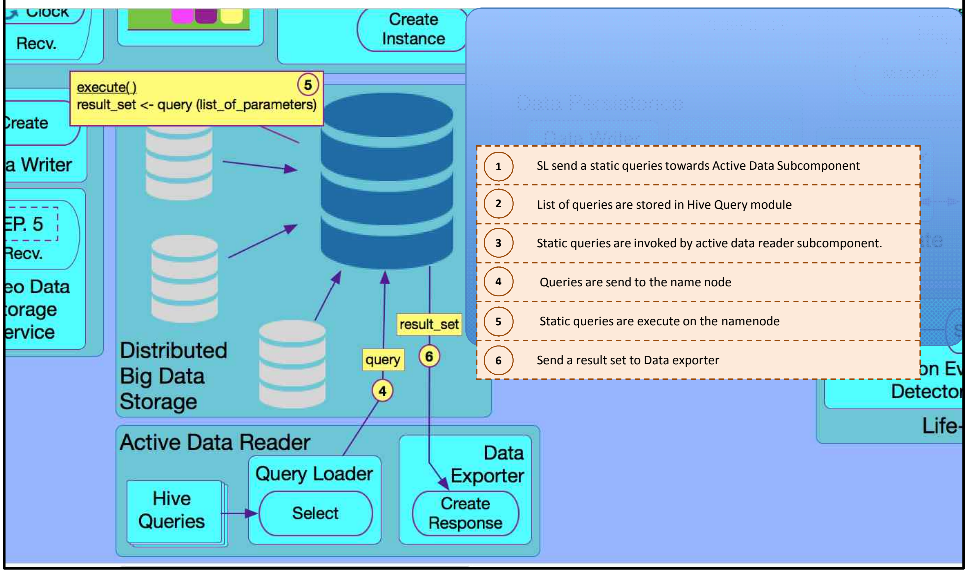
Query Execution and Response Generation

5/8



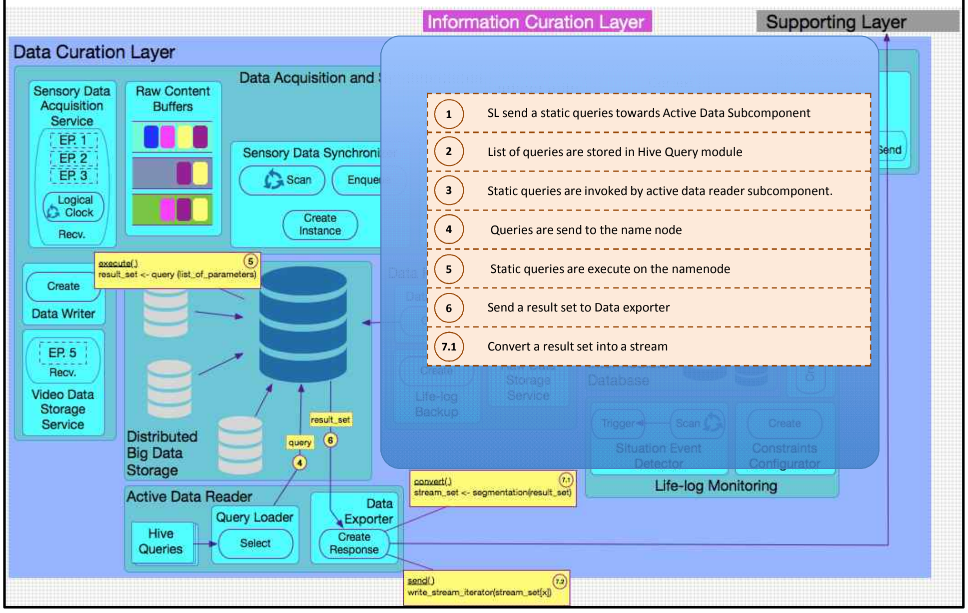
Query Execution and Response Generation

6/8



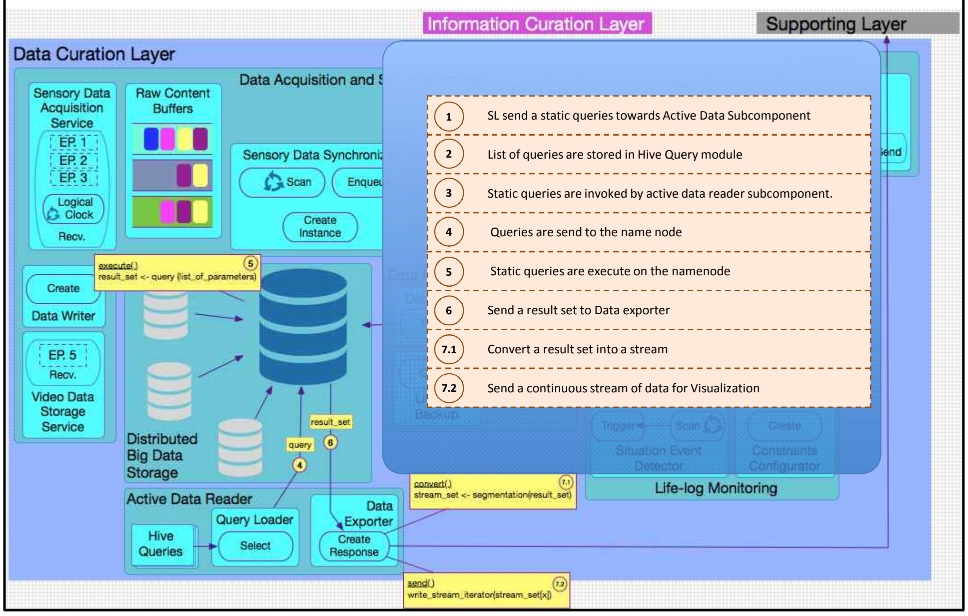
Query Execution and Response Generation

7/8



Query Execution and Response Generation

8/8

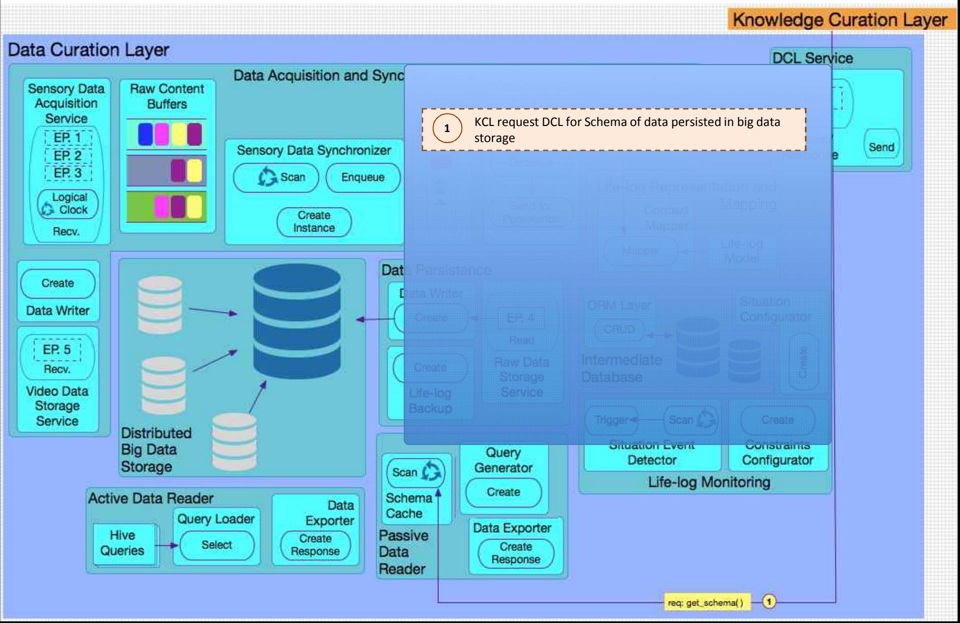


Scenario 3: Off-line (Passive)

Desc. : Raw-sensory data from big data storage is requested for **model training** by KCL.

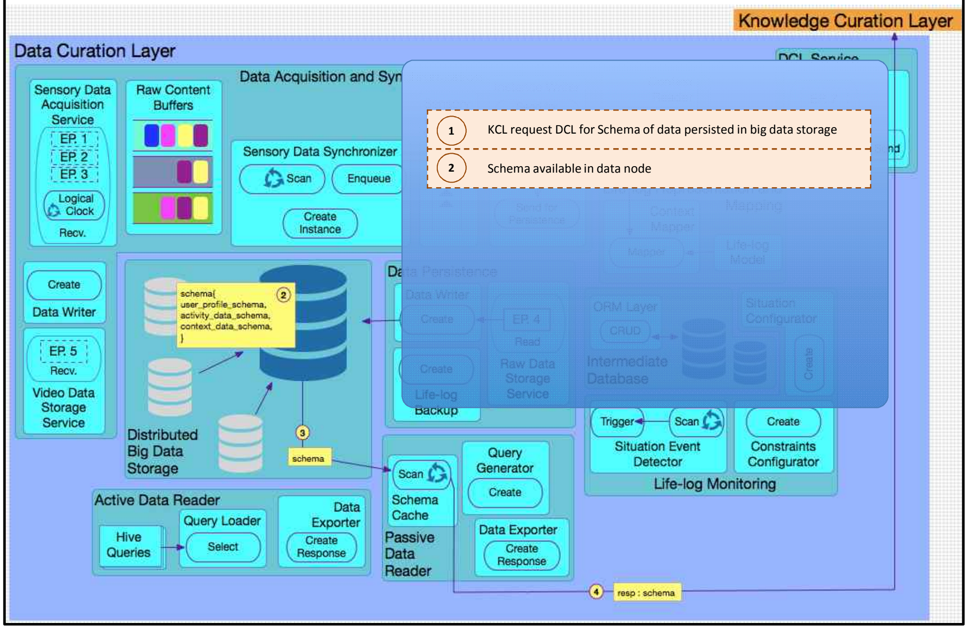
Get Schema

1/4



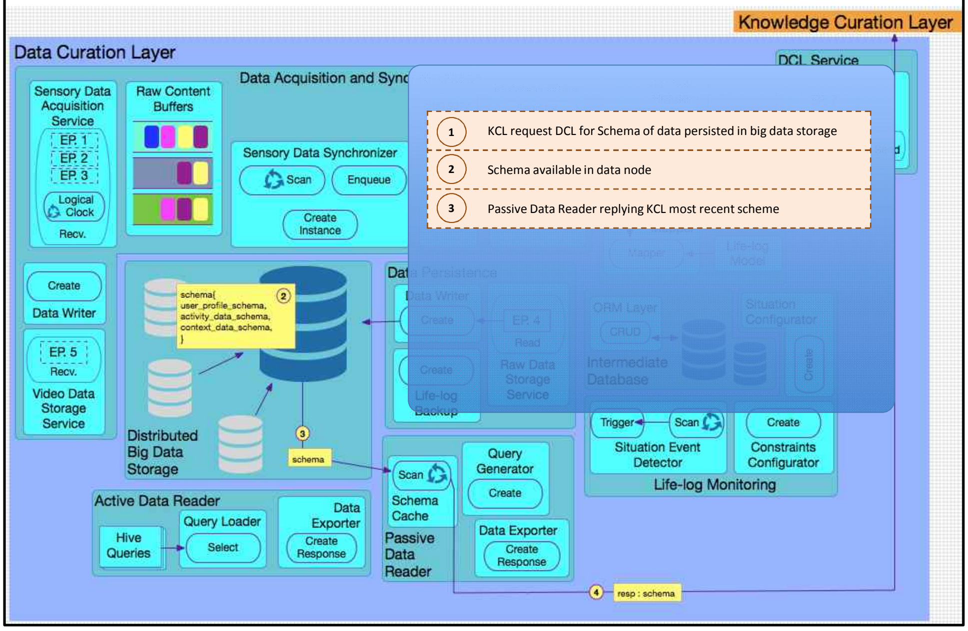
Get Schema

2/4

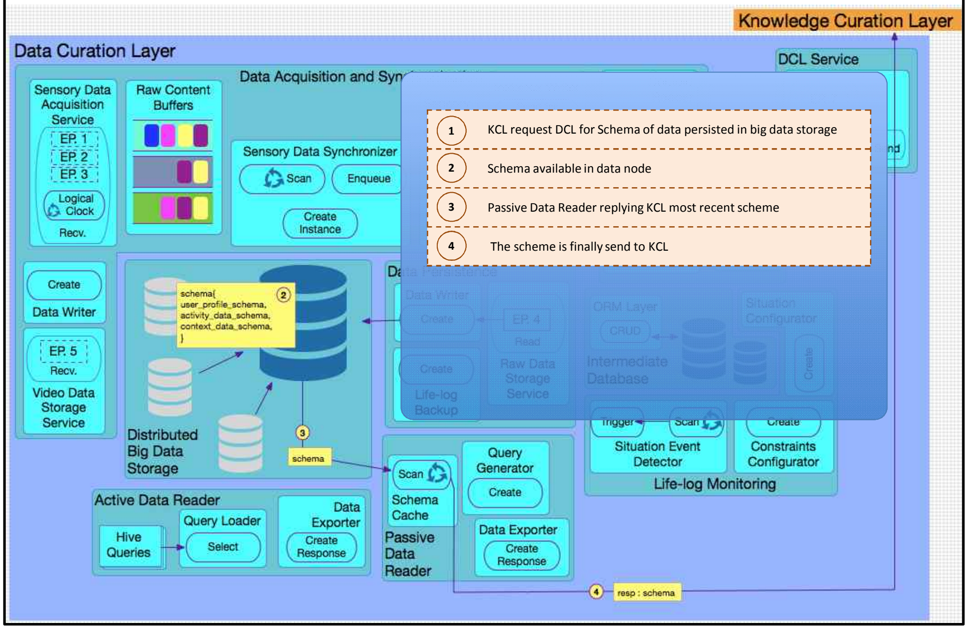


Get Schema

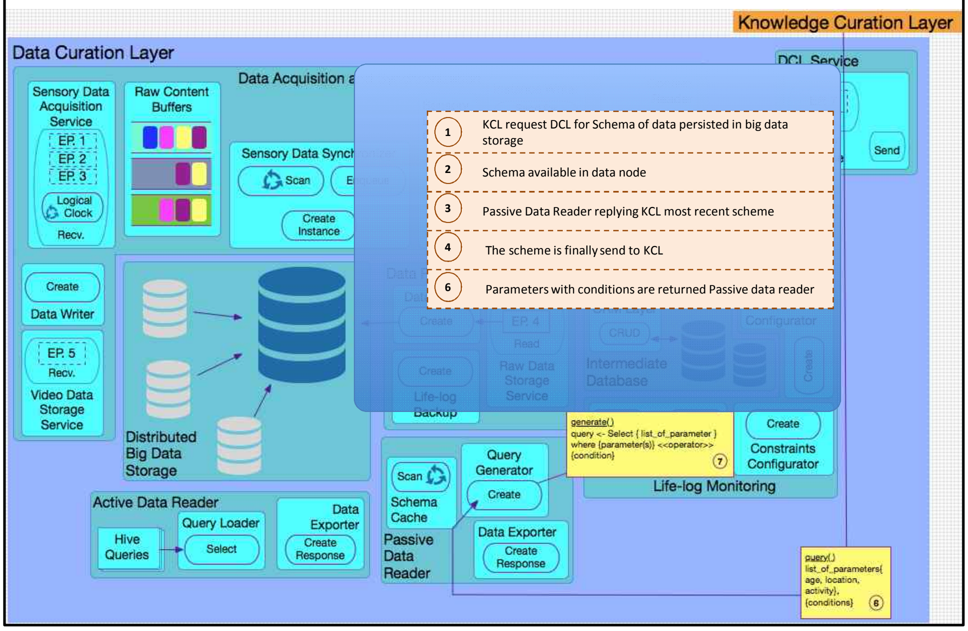
3/4



Get Schema

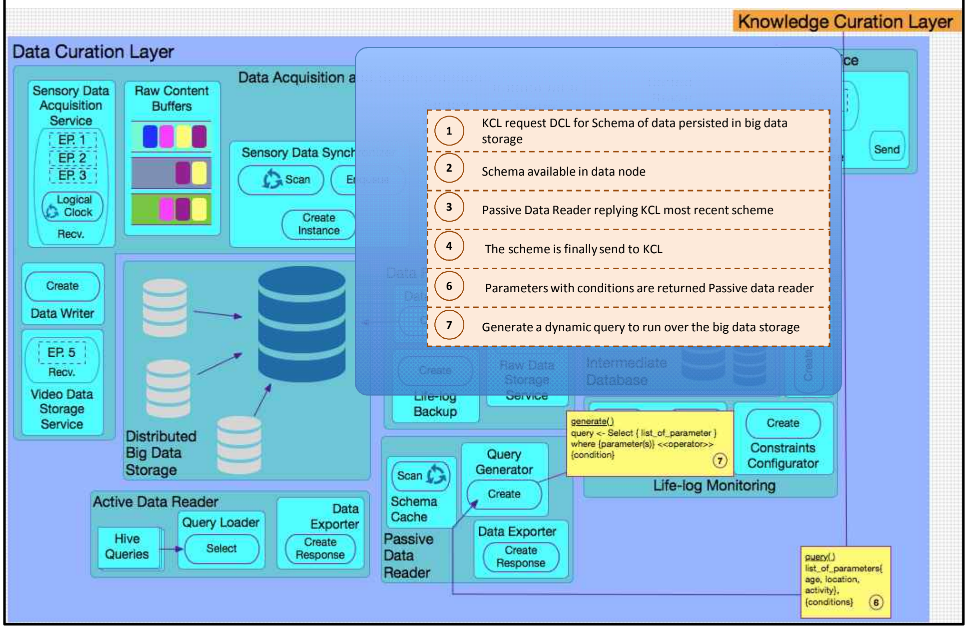


Generate and Execute Query



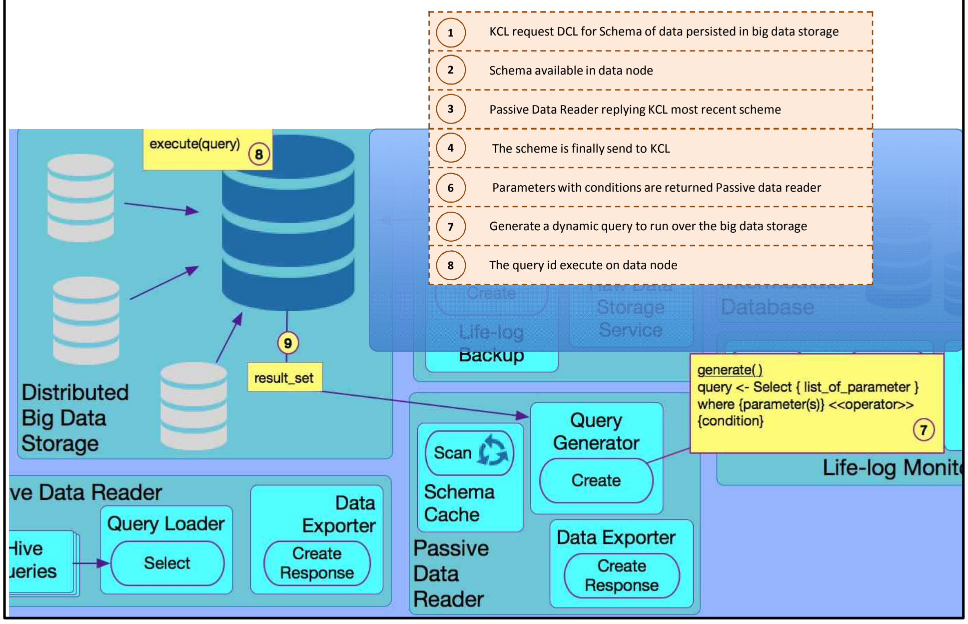
Generate and Execute Query

1/2

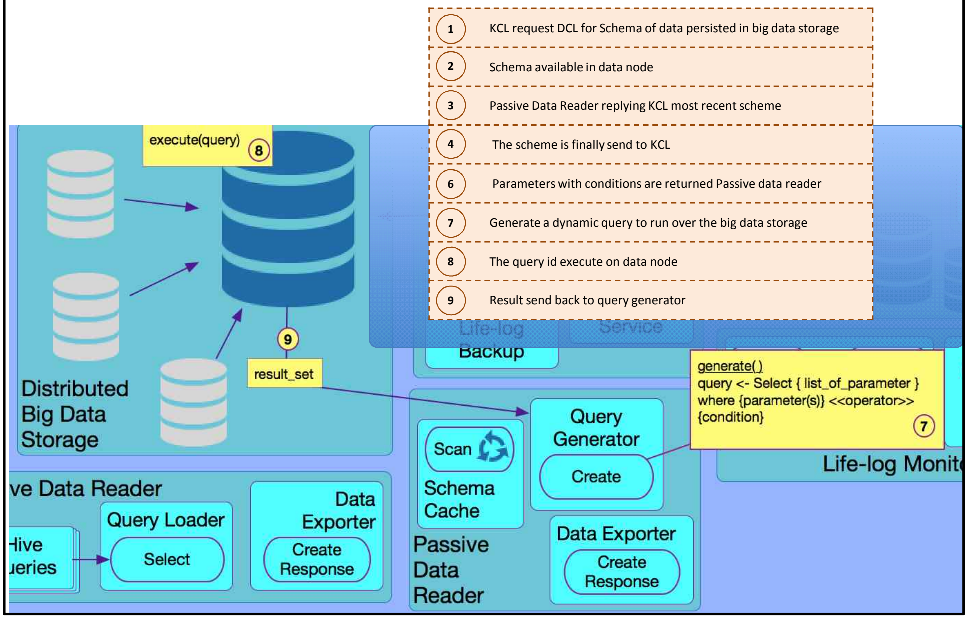


Generate and Execute Query

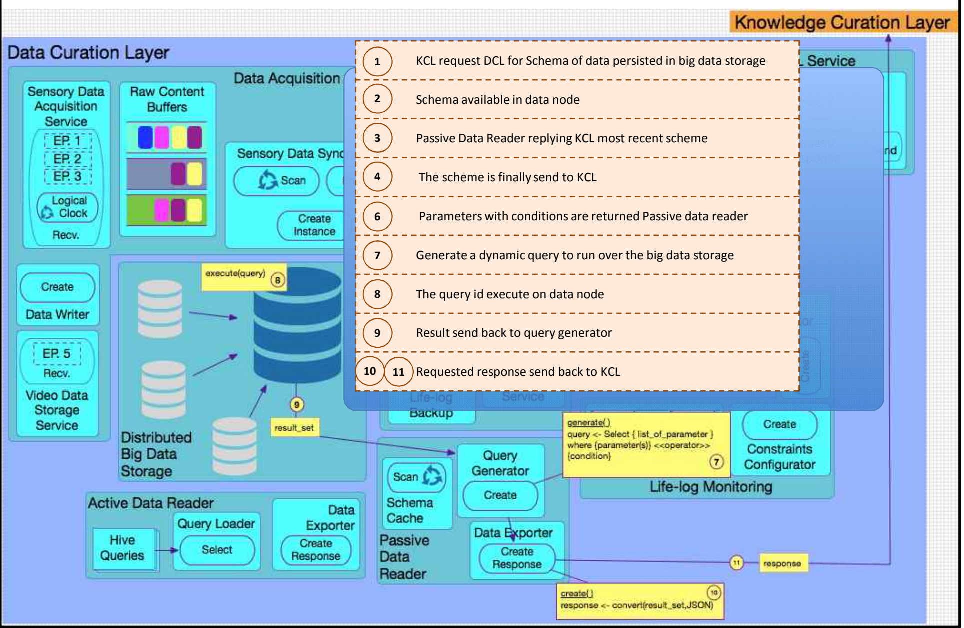
2/2



Generate and Execute Query



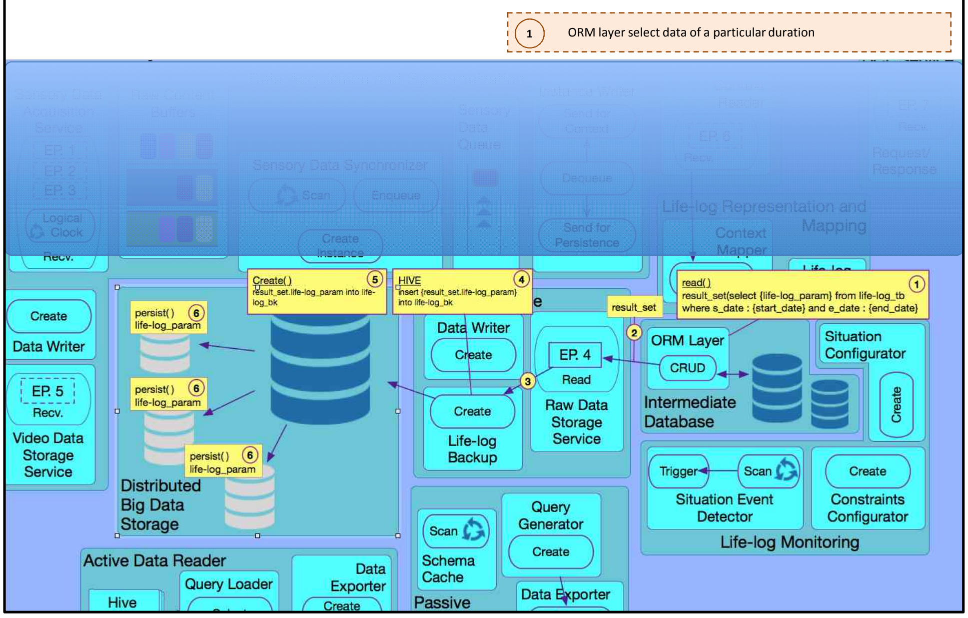
Create and Send Response

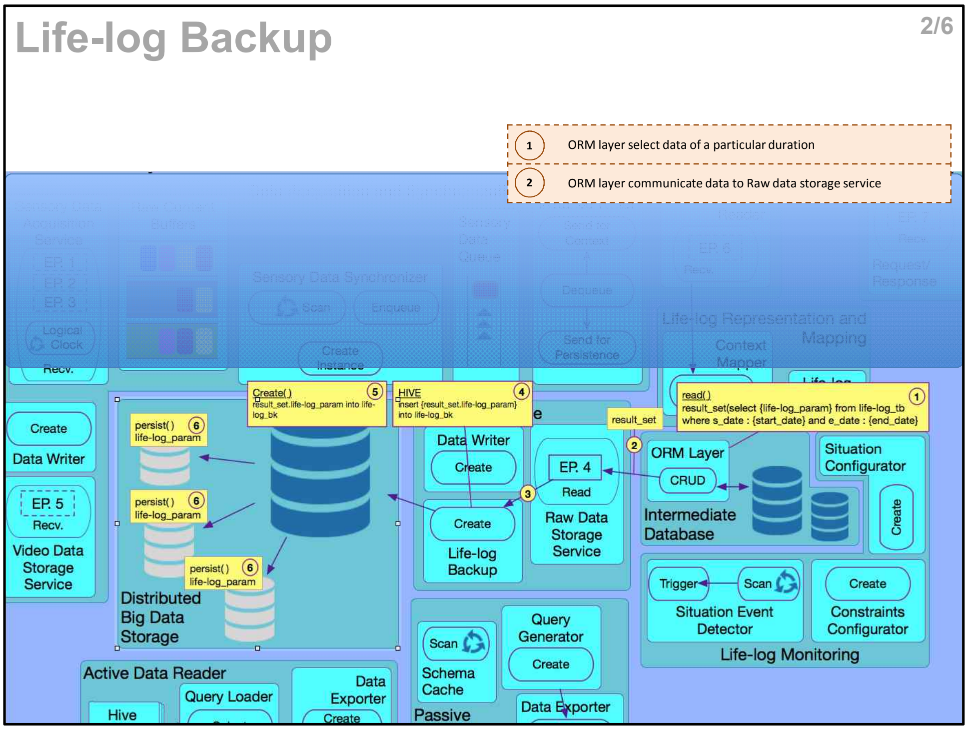


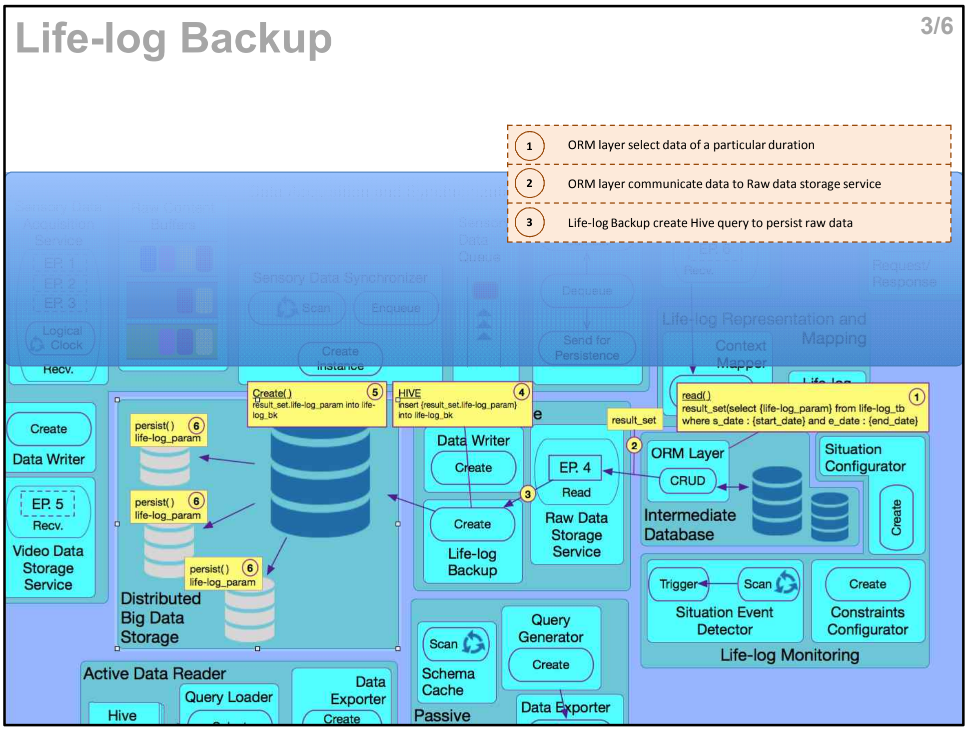
Scenario 4: Periodic

Desc. : **Life-log** data is **backed up** over big data storage
Video data contents from **Kinect source** are **backed Up** in big data storage

Life-log Backup

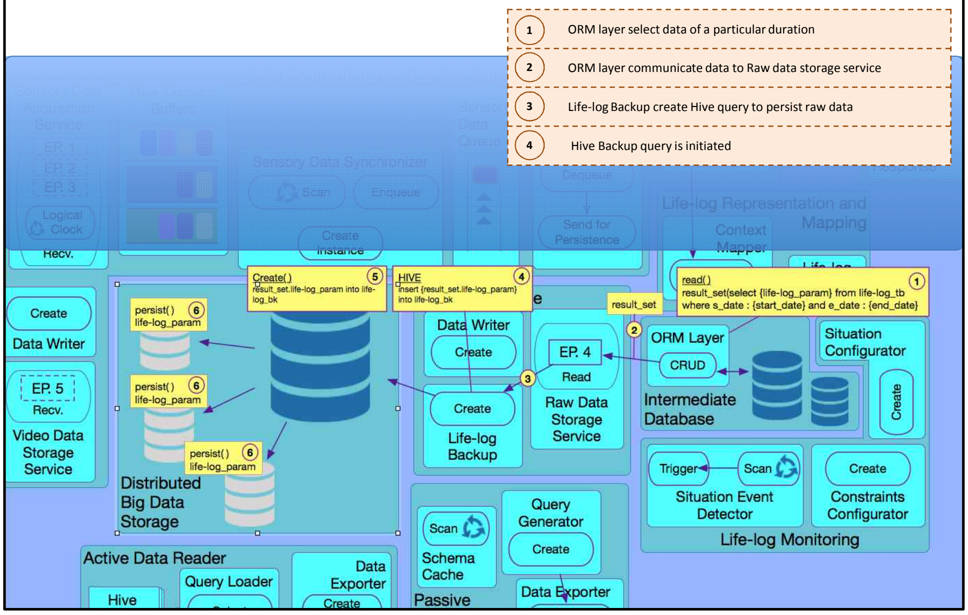



Life-log Backup
2/6
1 ORM layer select data of a particular duration
2 ORM layer communicate data to Raw data storage service
read()
result_set(select {life-log_param} from life-log_tb where s_date : {start_date} and e_date : {end_date}
result_set
Create()
result_set.life-log_param into life-log_bk
HIVE
insert {result_set.life-log_param} into life-log_bk
persist() life-log_param
Create
Data Writer
EP. 5
Recv.
Video Data Storage Service
Distributed Big Data Storage
Data Writer
Create
Life-log Backup
EP. 4
Read
Raw Data Storage Service
ORM Layer
CRUD
Intermediate Database
Situation Configurator
Trigger
Scan
Situation Event Detector
Constraints Configurator
Life-log Monitoring
Query Generator
Schema Cache
Passive
Data Exporter
Active Data Reader
Query Loader
Hive


Life-log Backup
3/6
1 ORM layer select data of a particular duration
2 ORM layer communicate data to Raw data storage service
3 Life-log Backup create Hive query to persist raw data
read()
result_set(select {life-log_param} from life-log_tb where s_date : {start_date} and e_date : {end_date}
result_set
Create()
result_set.life-log_param into life-log_bk
HIVE
insert {result_set.life-log_param} into life-log_bk
persist() life-log_param
Create
Data Writer
EP. 5
Recv.
Video Data Storage Service
Distributed Big Data Storage
Data Writer
Create
Life-log Backup
EP. 4
Read
Raw Data Storage Service
ORM Layer
CRUD
Intermediate Database
Situation Configurator
Trigger
Scan
Situation Event Detector
Constraints Configurator
Life-log Monitoring
Query Generator
Schema Cache
Passive
Data Exporter
Active Data Reader
Query Loader
Hive

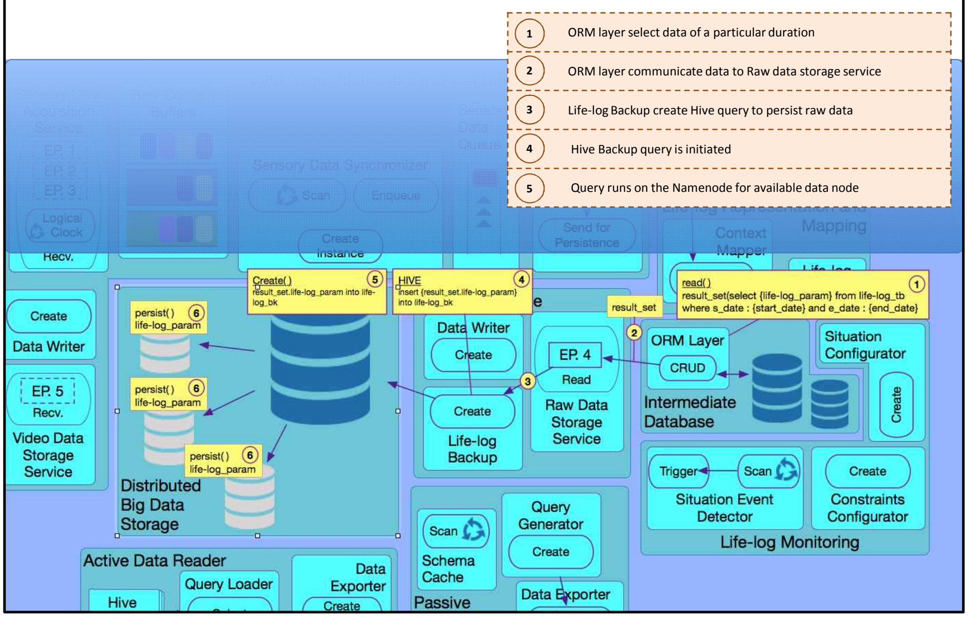
Life-log Backup

4/6

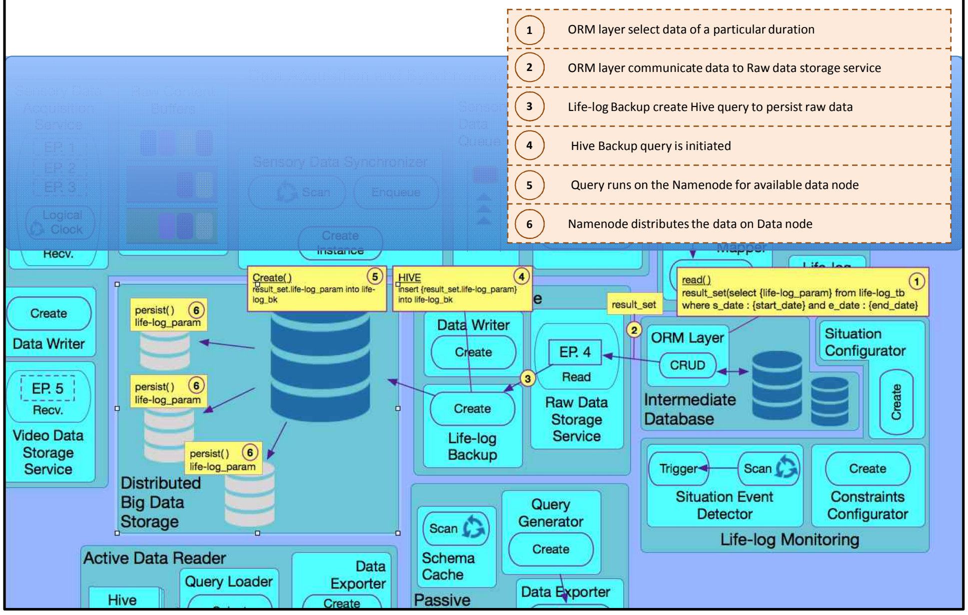


Life-log Backup

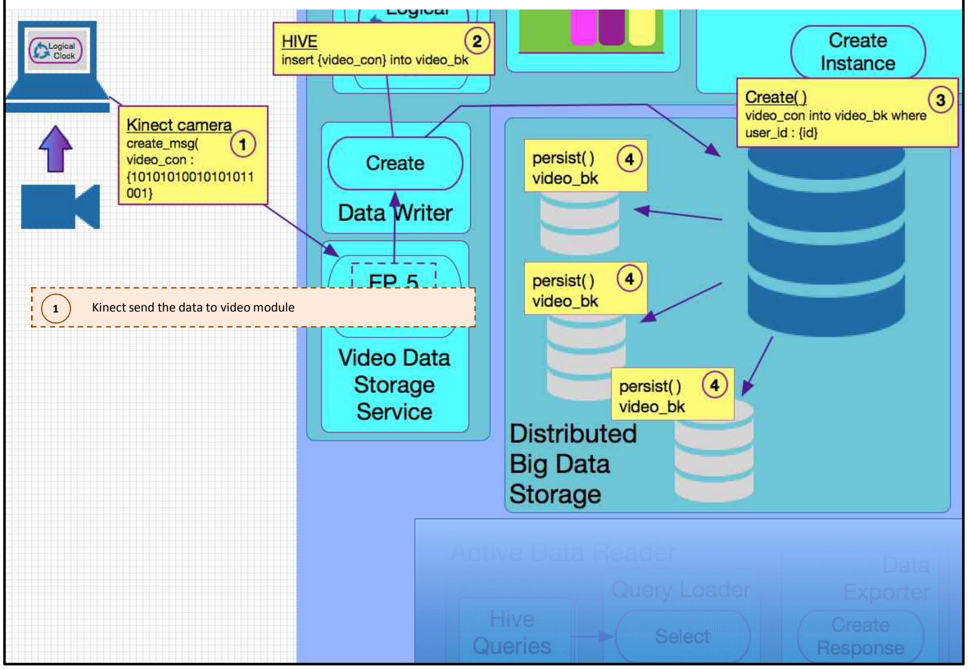
5/6

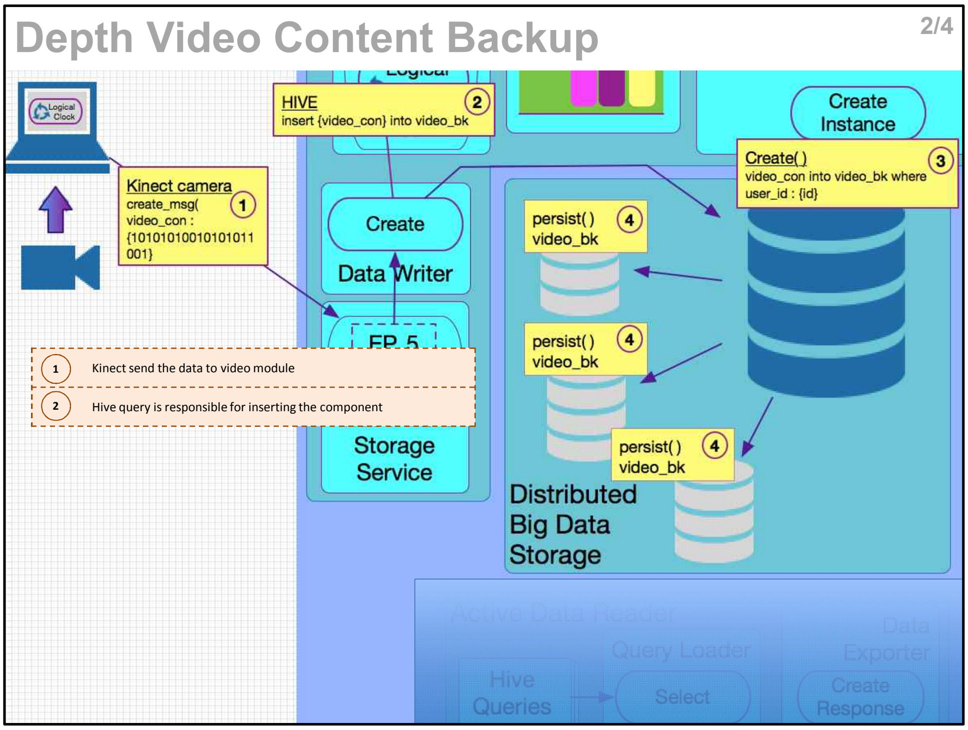


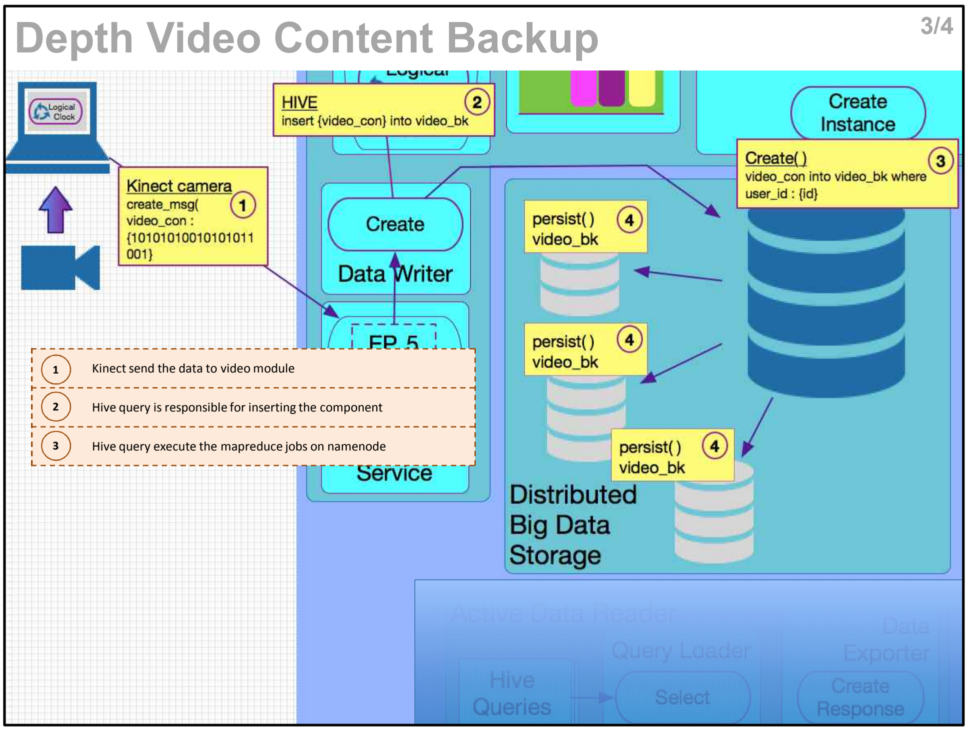
Life-log Backup

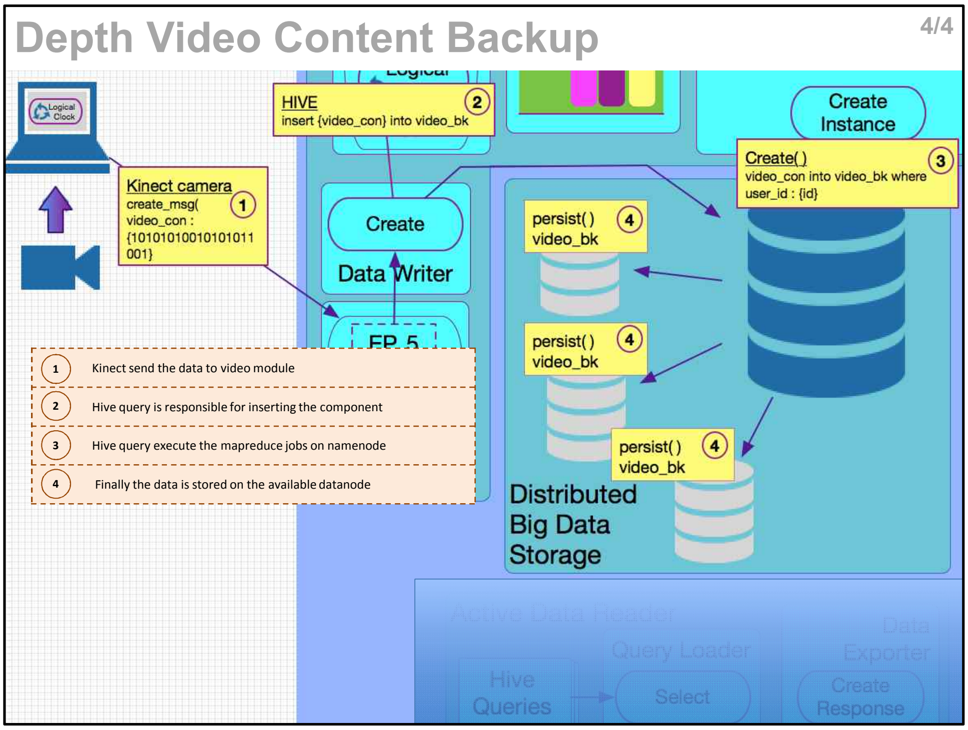


Depth Video Content Backup



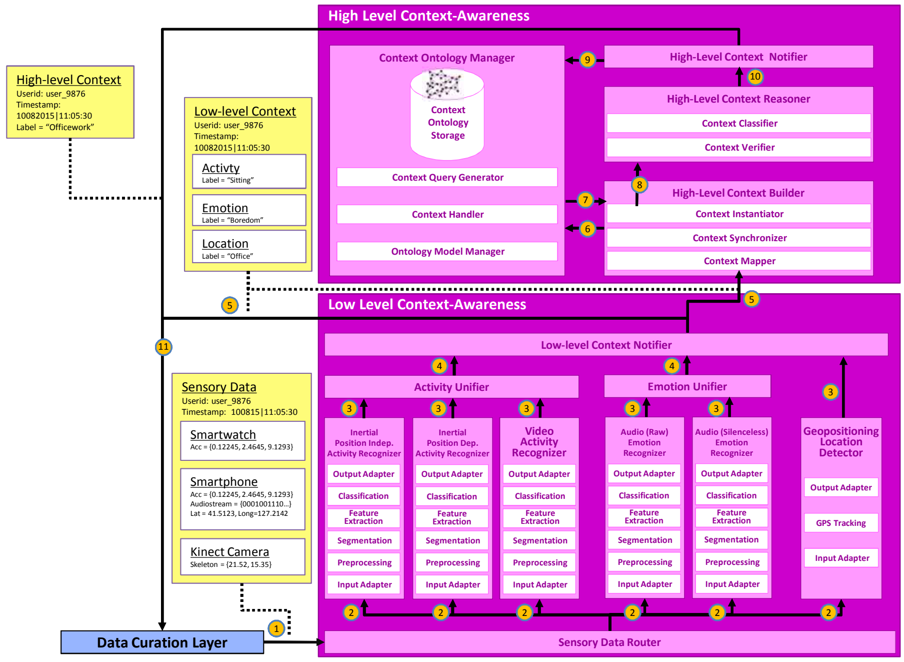

Depth Video Content Backup
2/4
Logical Clock
HIVE
insert {video_con} into video_bk
2
Create Instance
Create()
video_con into video_bk where user_id : {id}
3
Kinect camera
create_msg(
video_con :
{10101010010101011
001}
1
Create
Data Writer
FP 5
persist()
video_bk
4
1 Kinect send the data to video module
2 Hive query is responsible for inserting the component
Storage Service
Distributed Big Data Storage
Active Data Reader
Query Loader
Hive Queries
Select
Data Exporter
Create Response


Depth Video Content Backup
3/4
Logical Clock
HIVE
insert {video_con} into video_bk
2
Create Instance
Create()
video_con into video_bk where user_id : {id}
3
Kinect camera
create_msg(
video_con :
{10101010010101011
001}
1
Create
Data Writer
FP 5
persist()
video_bk
4
1 Kinect send the data to video module
2 Hive query is responsible for inserting the component
3 Hive query execute the mapreduce jobs on namenode
Service
Distributed Big Data Storage
Active Data Reader
Query Loader
Hive Queries
Select
Data Exporter
Create Response



STENCILS for ICL Micro-Animation

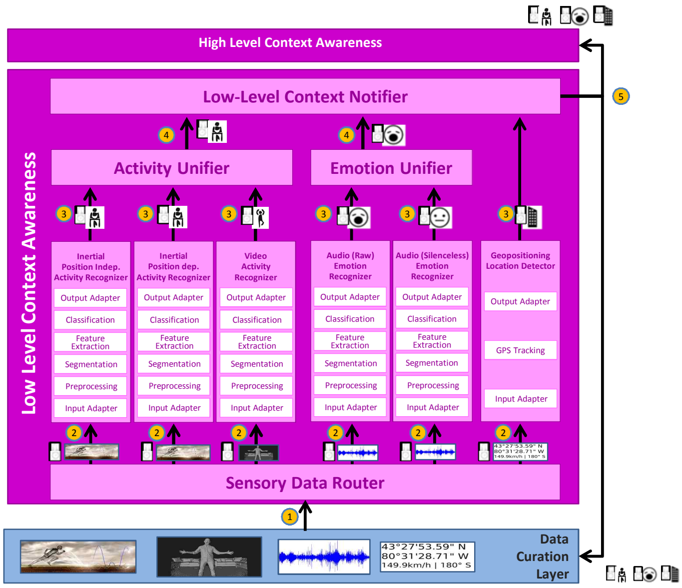
ICL Detailed Architecture



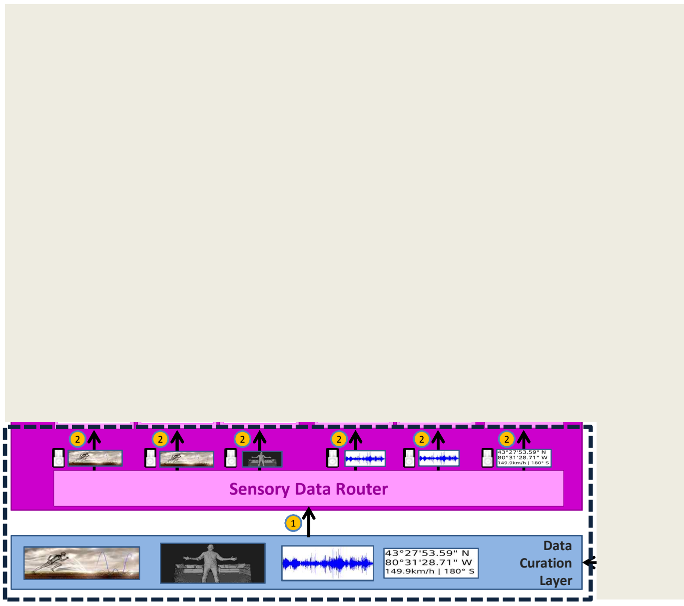
Scenario 1: Low level Context Recognition

Desc. : Low-level Context is **recognized** based on Raw-sensory data from multimodal data sources which is received by DCL.

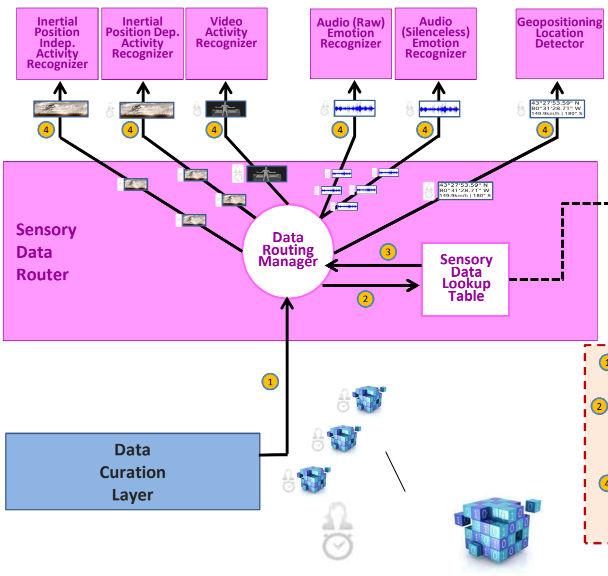
LLCA: Overall



LLCA: Sensory Data Routing



LLCA: Sensory Data Routing



Sensory Data Lookup Table

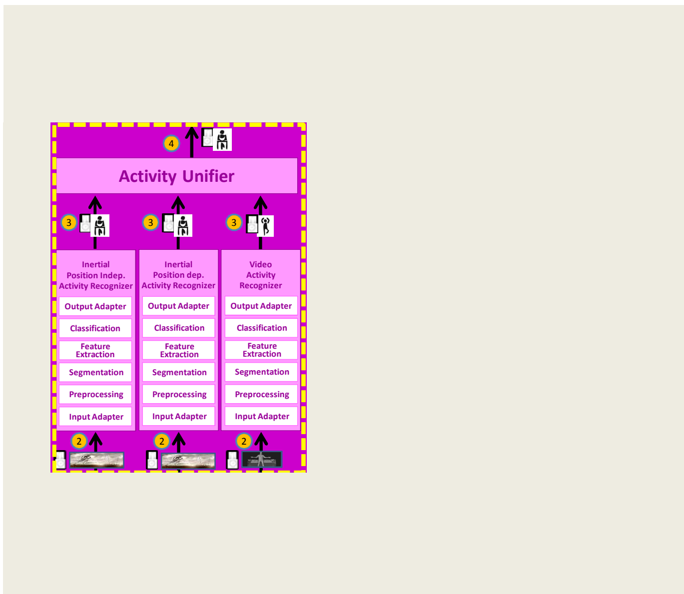
3 Recognizer type	Compatible data
Inertial PI - AR	Acc, Gyr
Inertial PD -AR	Acc, Gyr
Video AR	Pos
Audio ER	Wav
Audio ER	Wav
Geopos. LD	GPS

1 Raw sensory data is received from DCL 2

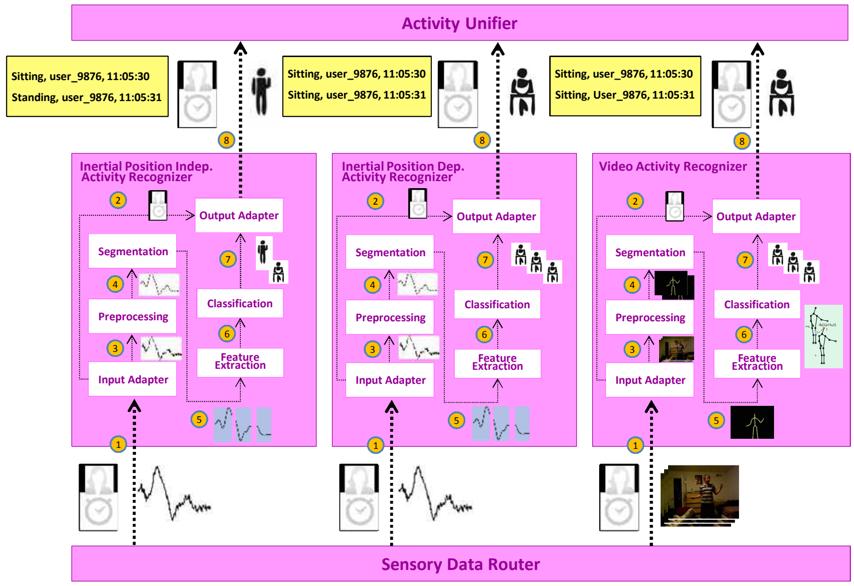
2 3 The routing manager checks the lookup table to determine potential consumers of the data

4 The compatible data is provided to each recognizer

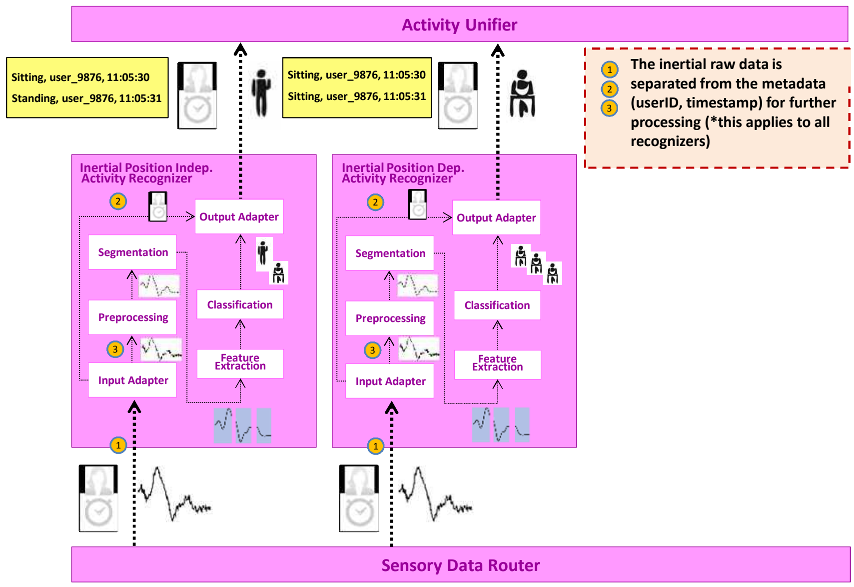
LLCA: Activity Recognition



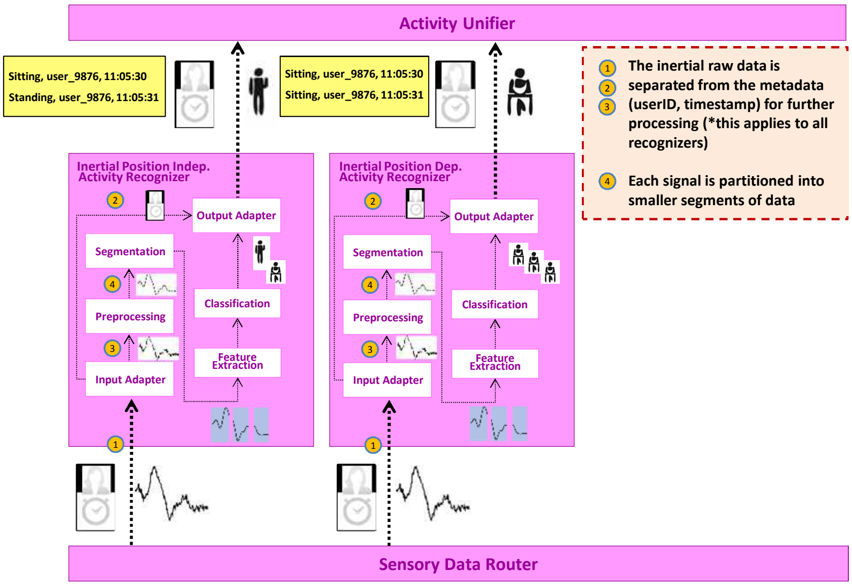
LLCA: Activity Recognition



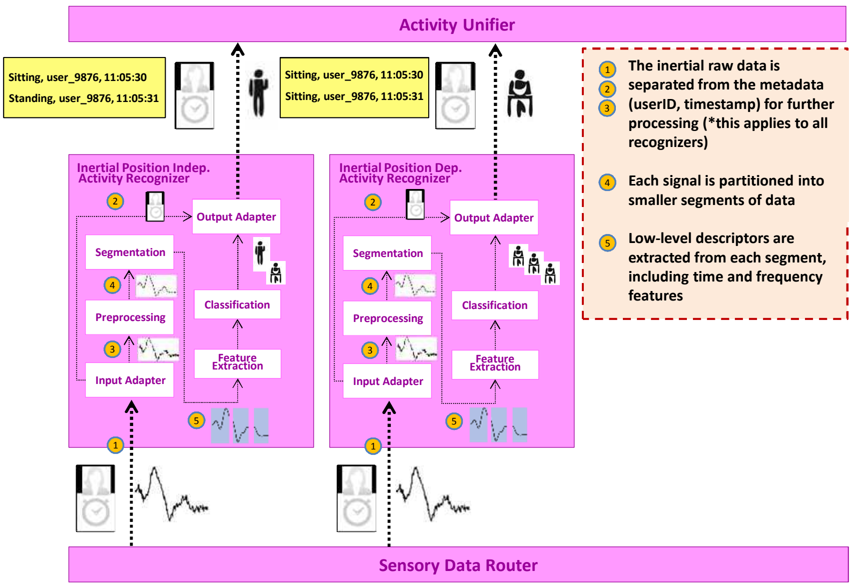
LLCA: Activity Recognition



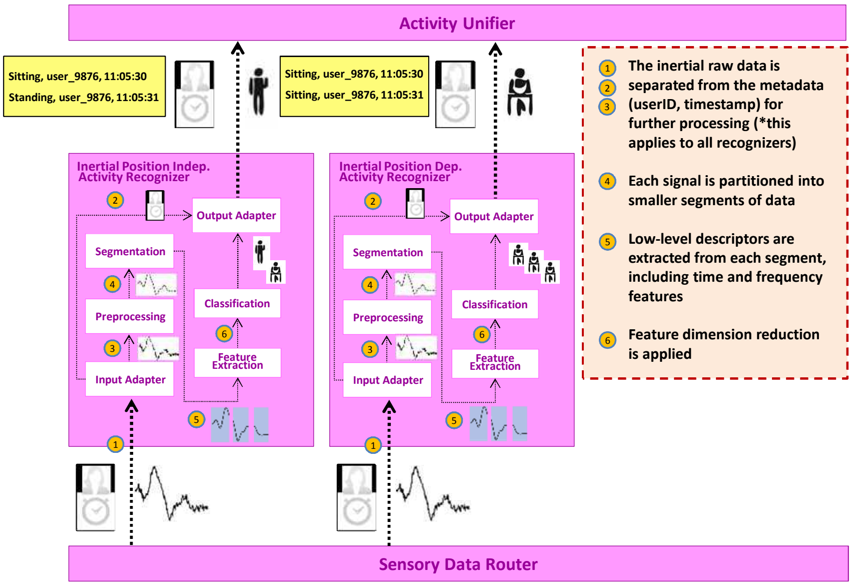
LLCA: Activity Recognition



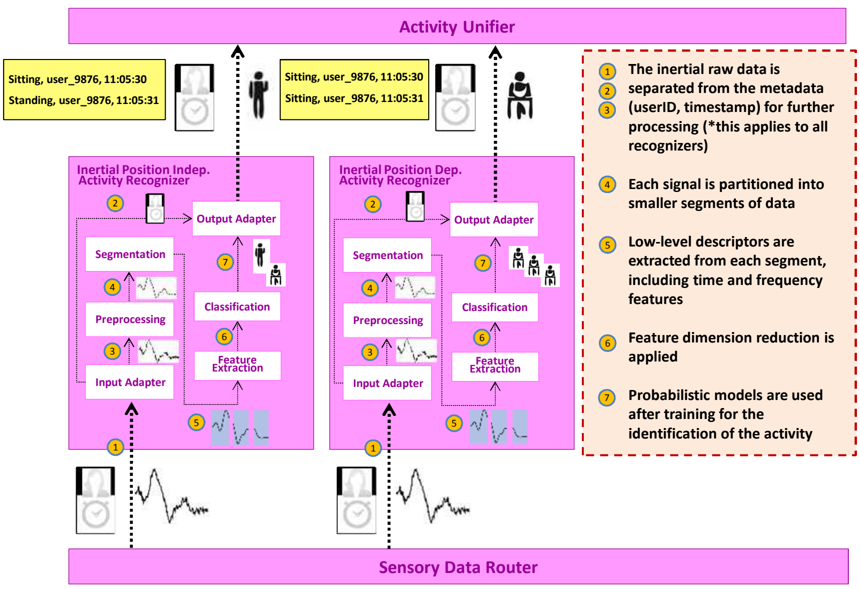
LLCA: Activity Recognition



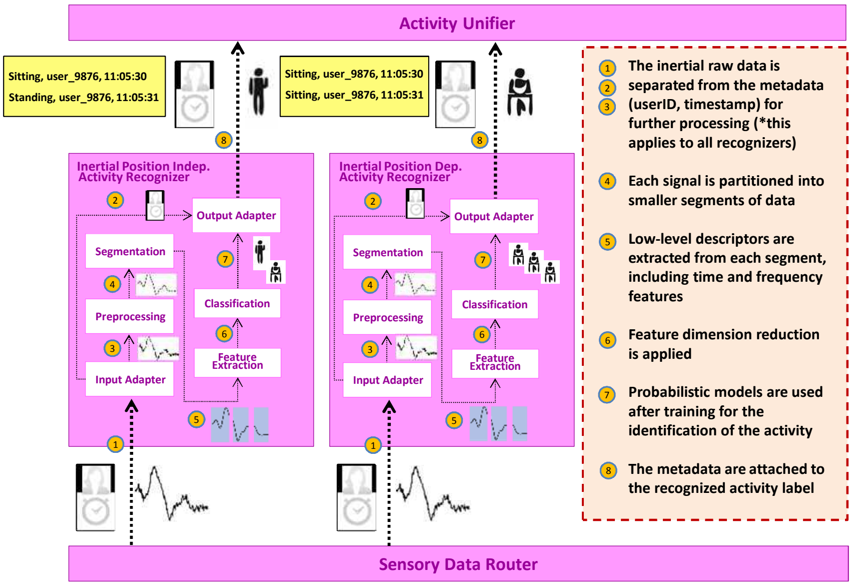
LLCA: Activity Recognition



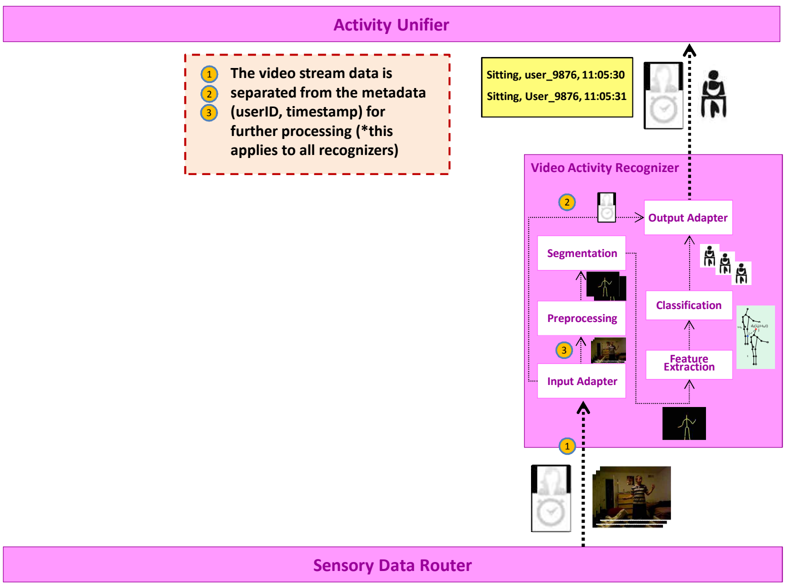
LLCA: Activity Recognition



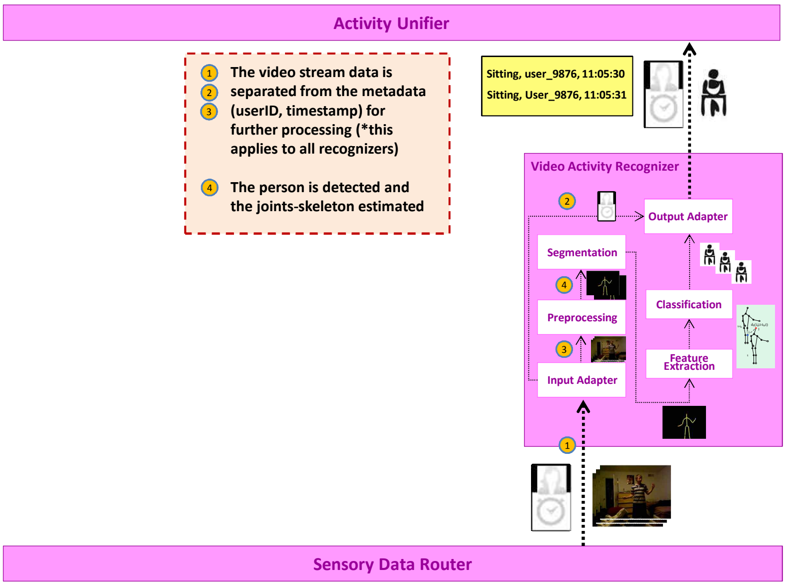
LLCA: Activity Recognition



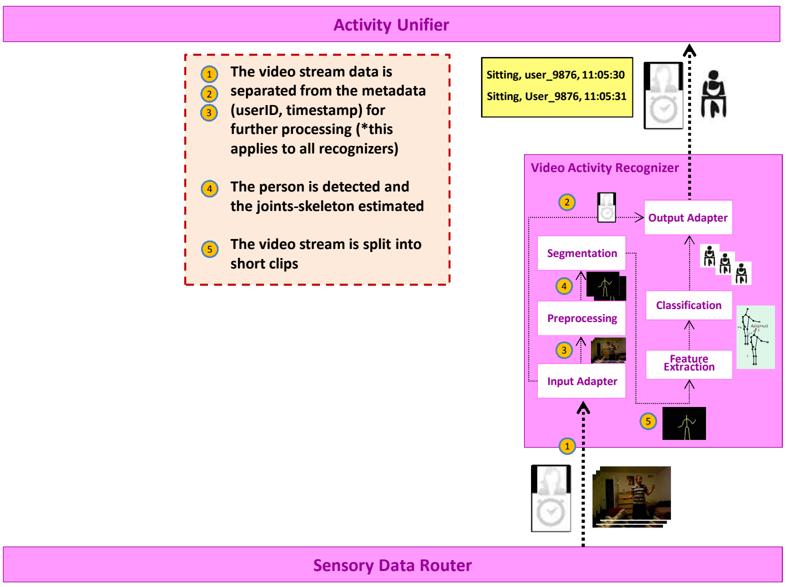
LLCA: Activity Recognition



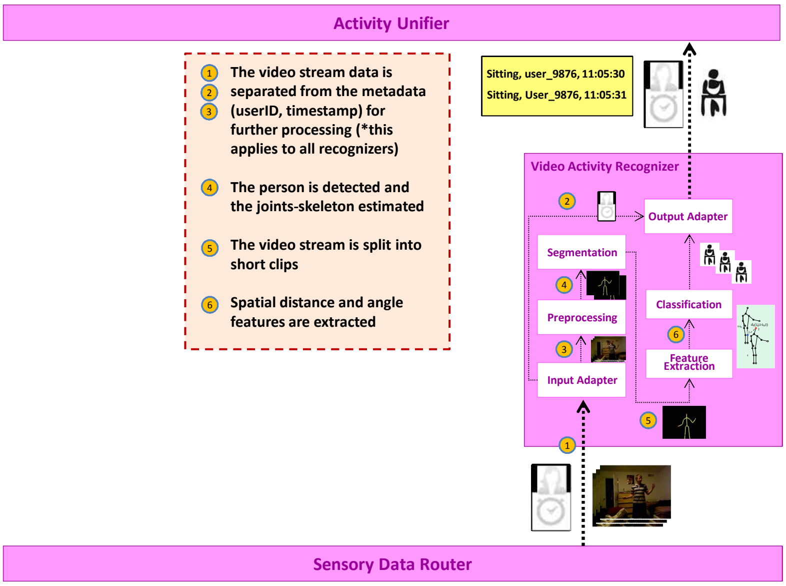
LLCA: Activity Recognition



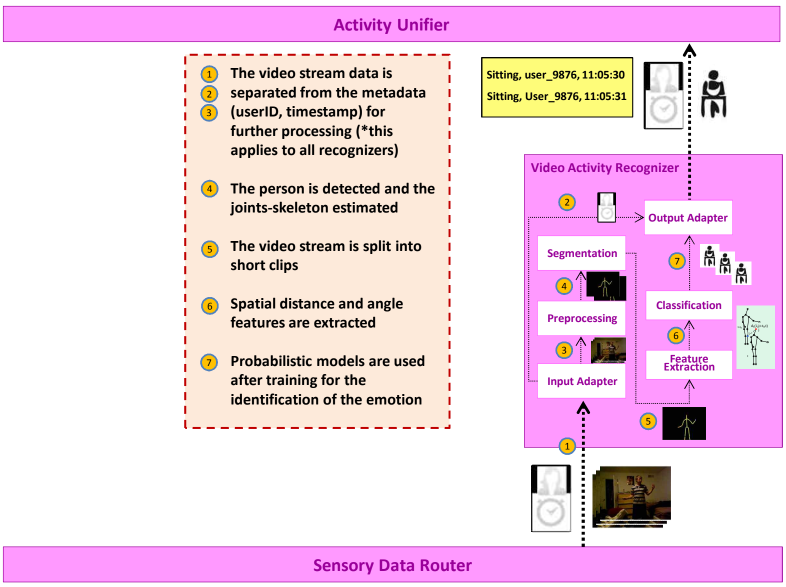
LLCA: Activity Recognition



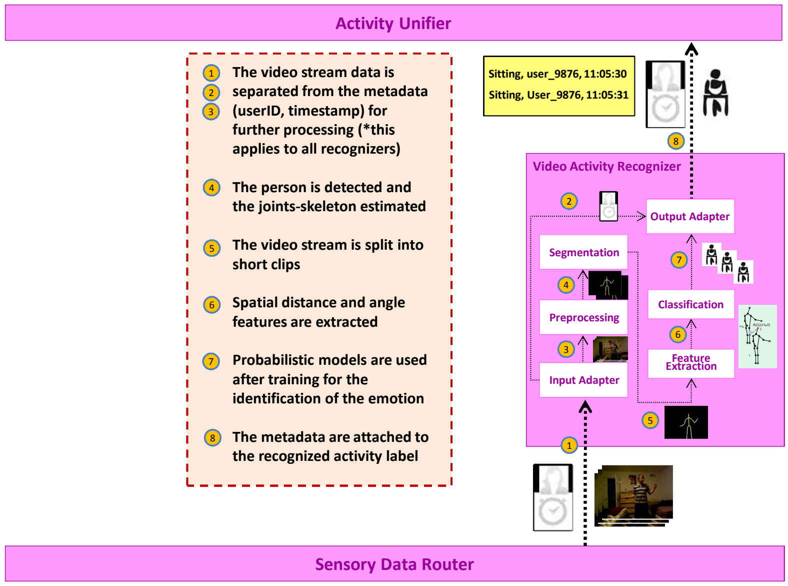
LLCA: Activity Recognition



LLCA: Activity Recognition

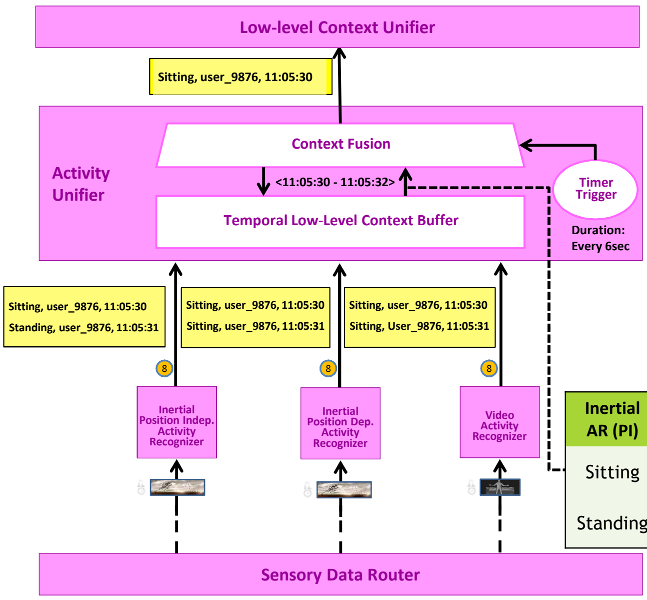


LLCA: Activity Recognition



LLCA: Activity Recognition

8 LLC (here activities) identified by each recognizer are received and buffered in a table

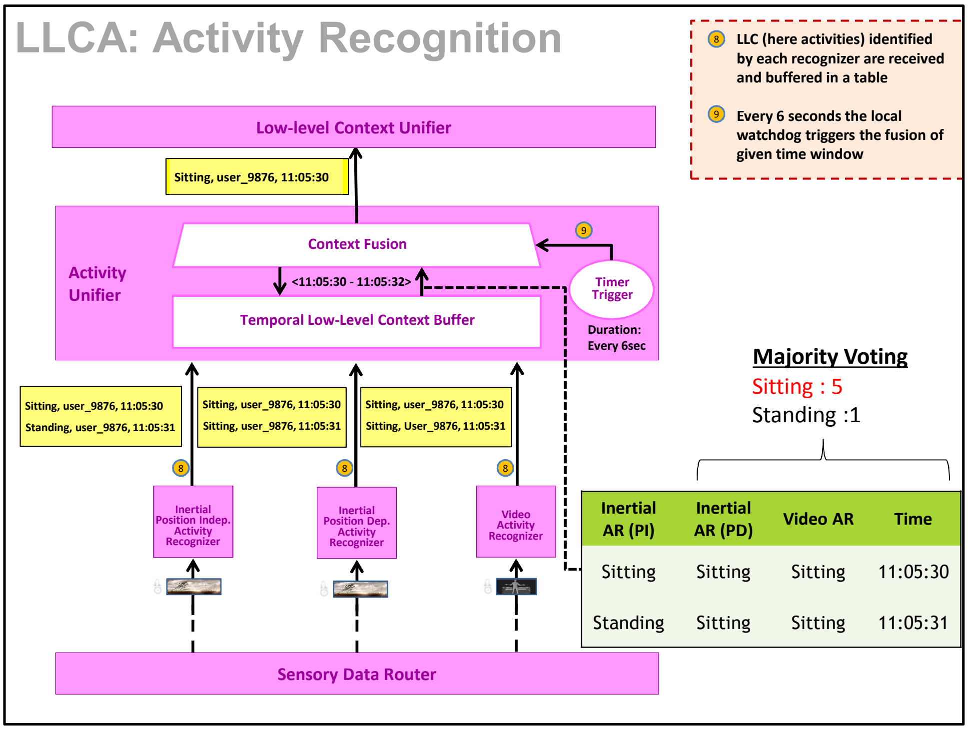


Majority Voting

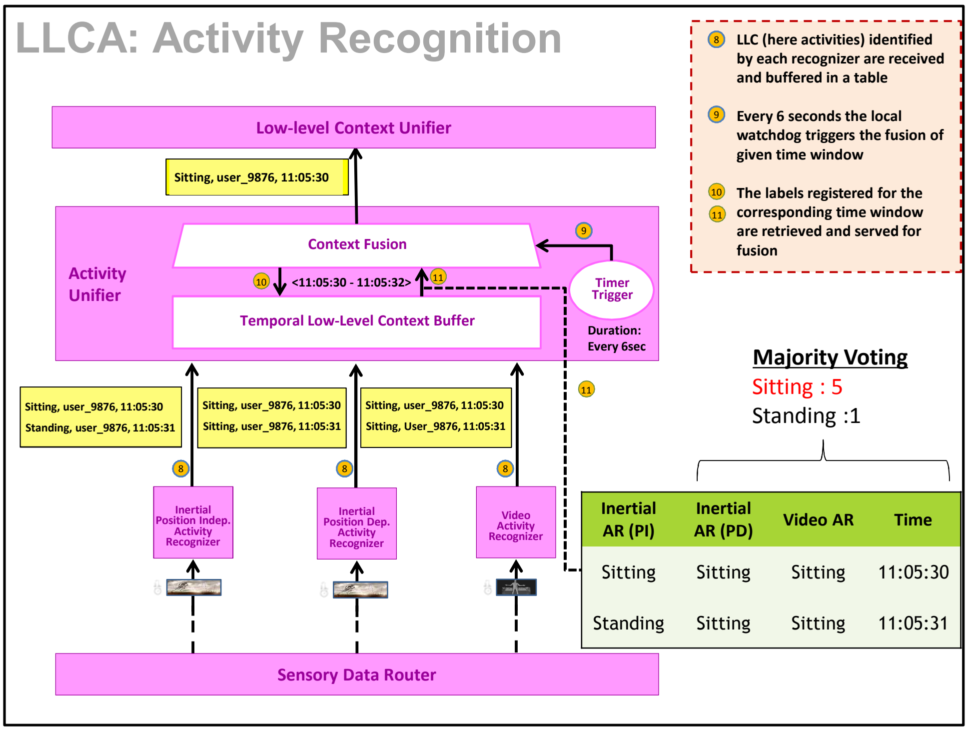
Sitting : 5

Standing :1

Inertial AR (PI)	Inertial AR (PD)	Video AR	Time
Sitting	Sitting	Sitting	11:05:30
Standing	Sitting	Sitting	11:05:31

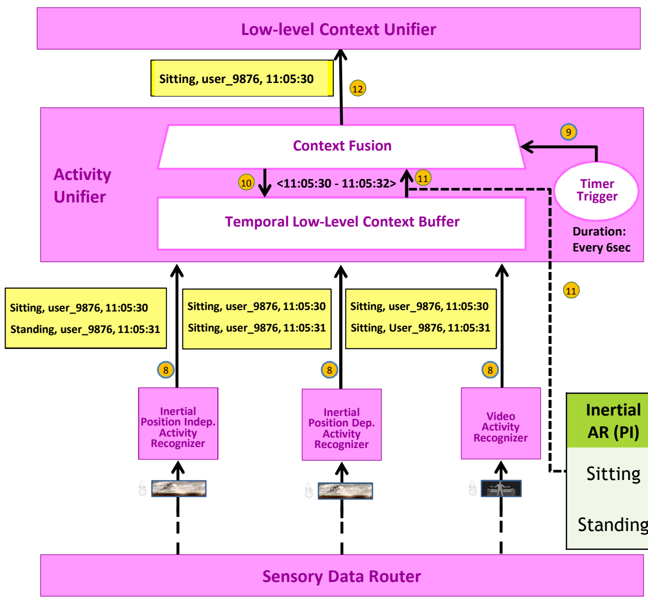


Inertial AR (PI)	Inertial AR (PD)	Video AR	Time
Sitting	Sitting	Sitting	11:05:30
Standing	Sitting	Sitting	11:05:31



Inertial AR (PI)	Inertial AR (PD)	Video AR	Time
Sitting	Sitting	Sitting	11:05:30
Standing	Sitting	Sitting	11:05:31

LLCA: Activity Recognition



8 LLC (here activities) identified by each recognizer are received and buffered in a table

9 Every 6 seconds the local watchdog triggers the fusion of given time window

10 11 The labels registered for the corresponding time window are retrieved and served for fusion

12 The labels are combined into one by using majority voting

Majority Voting

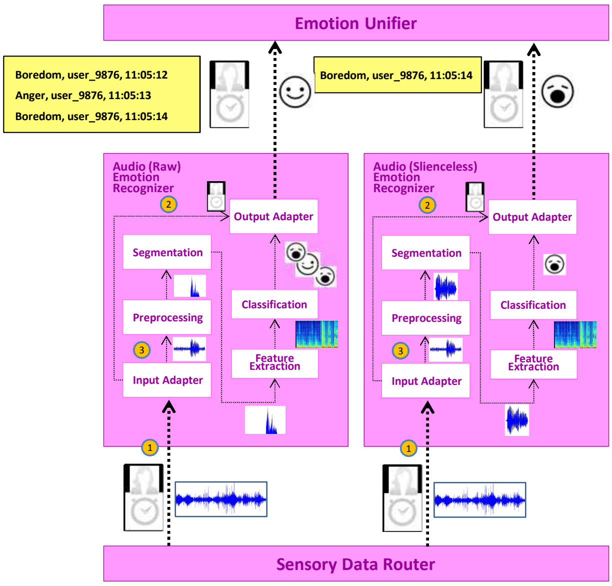
Sitting : 5

Standing :1

Inertial AR (PI)	Inertial AR (PD)	Video AR	Time
Sitting	Sitting	Sitting	11:05:30
Standing	Sitting	Sitting	11:05:31

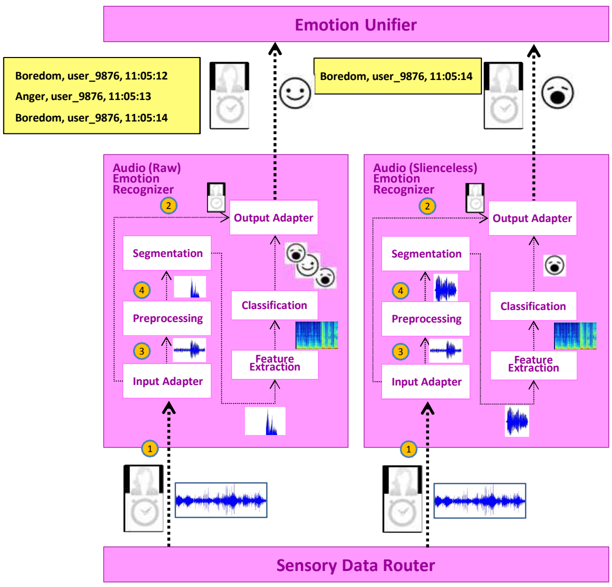
LLCA: Emotion Recognition

LLCA: Emotion Recognition



1 2 3 The raw audio data is separated from the metadata (userID, timestamp) for further processing (*this applies to all recognizers)

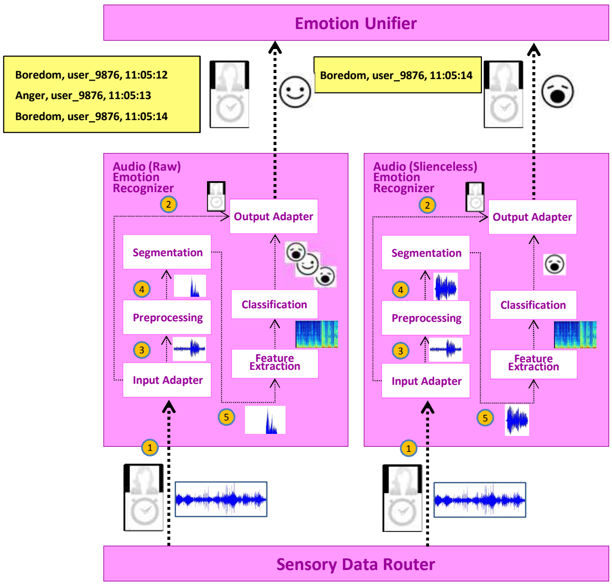
LLCA: Emotion Recognition



1 2 3 The raw audio data is separated from the metadata (userID, timestamp) for further processing (*this applies to all recognizers)

4 The raw audio signal is adapted through reducing the background noise and removing silences from the speech

LLCA: Emotion Recognition

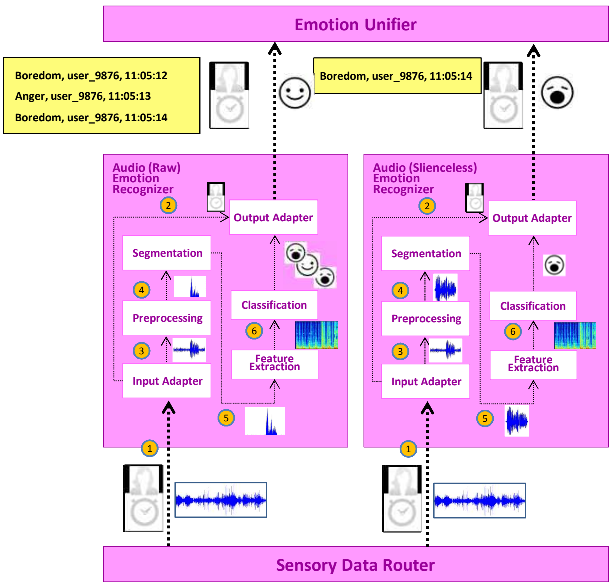


1 2 3 The raw audio data is separated from the metadata (userID, timestamp) for further processing (*this applies to all recognizers)

4 The raw audio signal is adapted through reducing the background noise and removing silences from the speech

5 Each speech segment is partitioned into smaller segments of data

LLCA: Emotion Recognition



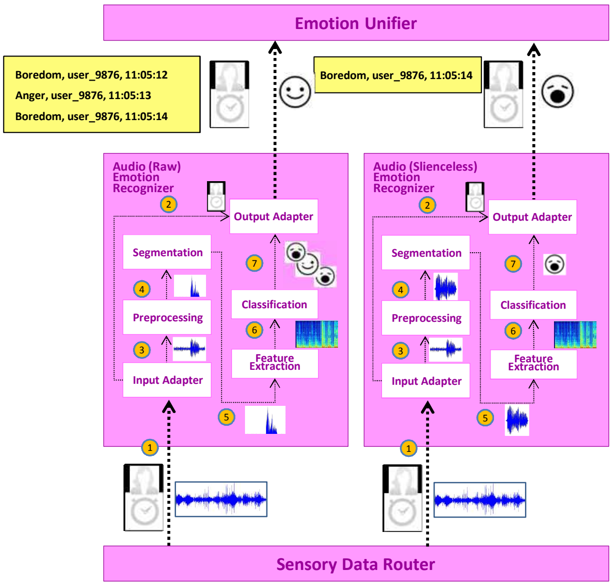
1 2 3 The raw audio data is separated from the metadata (userID, timestamp) for further processing (*this applies to all recognizers)

4 The raw audio signal is adapted through reducing the background noise and removing silences from the speech

5 Each speech segment is partitioned into smaller segments of data

6 Low-level descriptors are extracted from each segment, including cepstral coefficients and statistical values

LLCA: Emotion Recognition



1 2 3 The raw audio data is separated from the metadata (userID, timestamp) for further processing (*this applies to all recognizers)

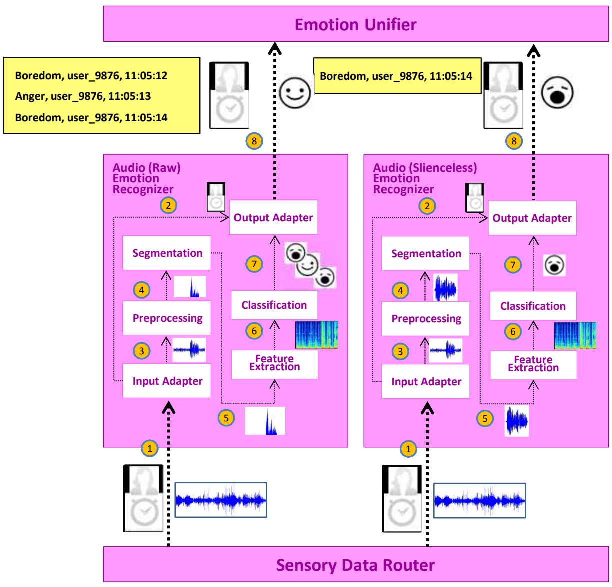
4 The raw audio signal is adapted through reducing the background noise and removing silences from the speech

5 Each speech segment is partitioned into smaller segments of data

6 Low-level descriptors are extracted from each segment, including cepstral coefficients and statistical values

7 Probabilistic models are used after training for the identification of the emotion

LLCA: Emotion Recognition



1 2 3 The raw audio data is separated from the metadata (userID, timestamp) for further processing (*this applies to all recognizers)

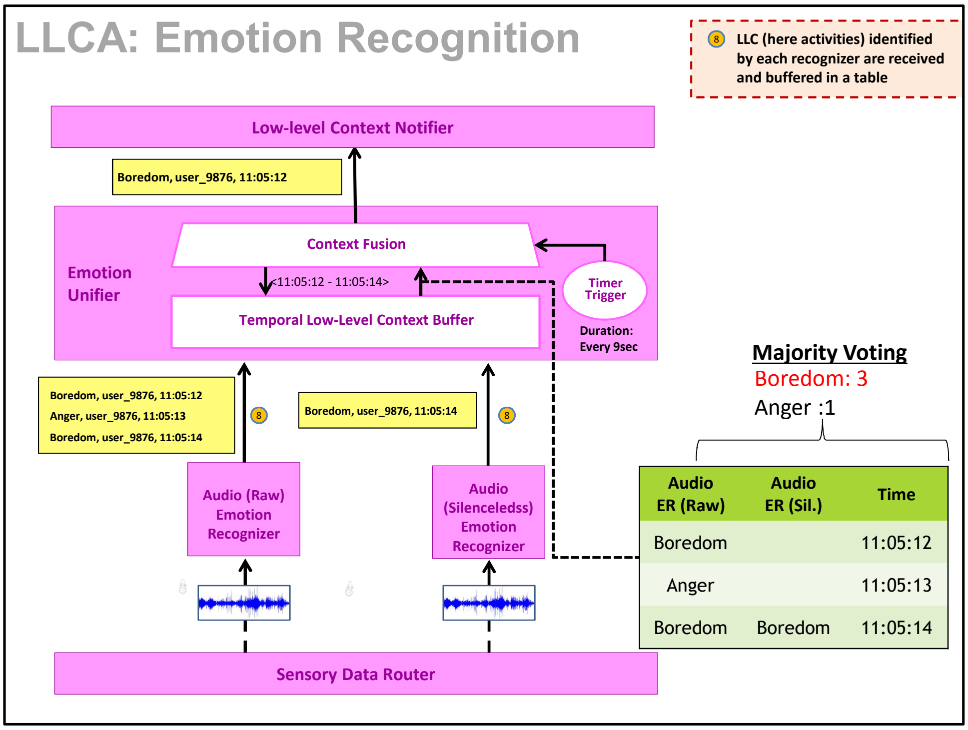
4 The raw audio signal is adapted through reducing the background noise and removing silences from the speech

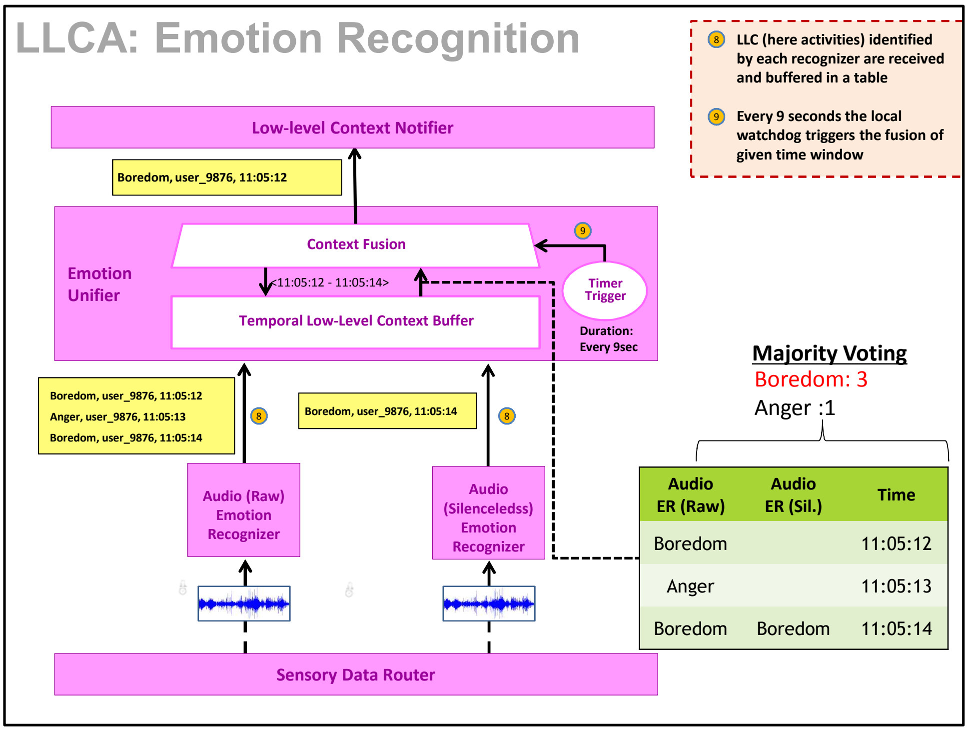
5 Each speech segment is partitioned into smaller segments of data

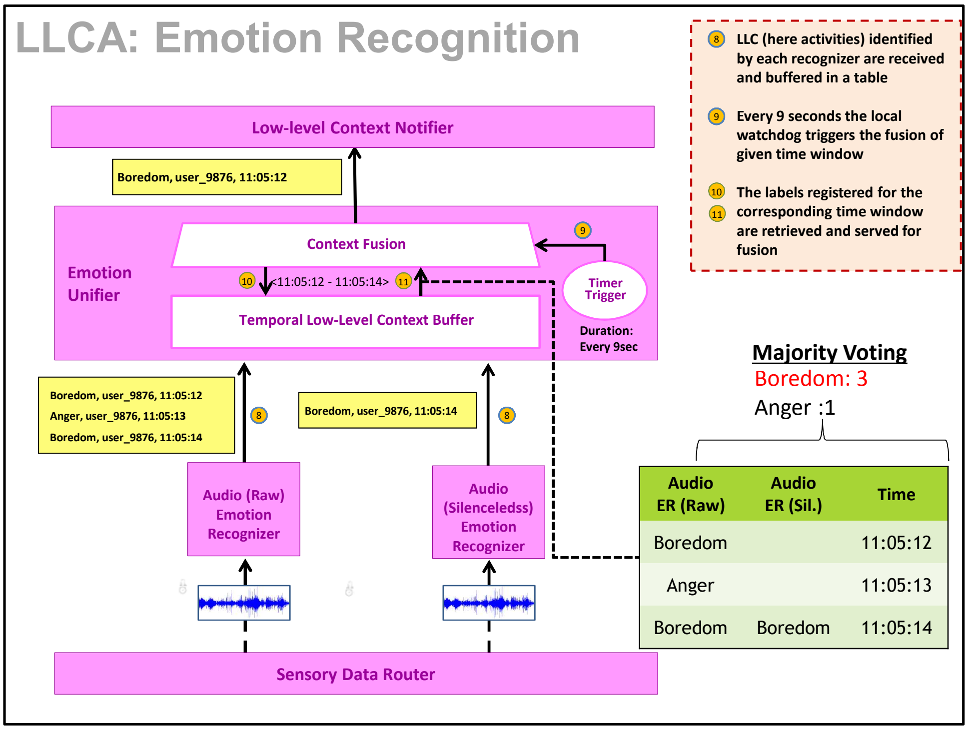
6 Low-level descriptors are extracted from each segment, including cepstral coefficients and statistical values

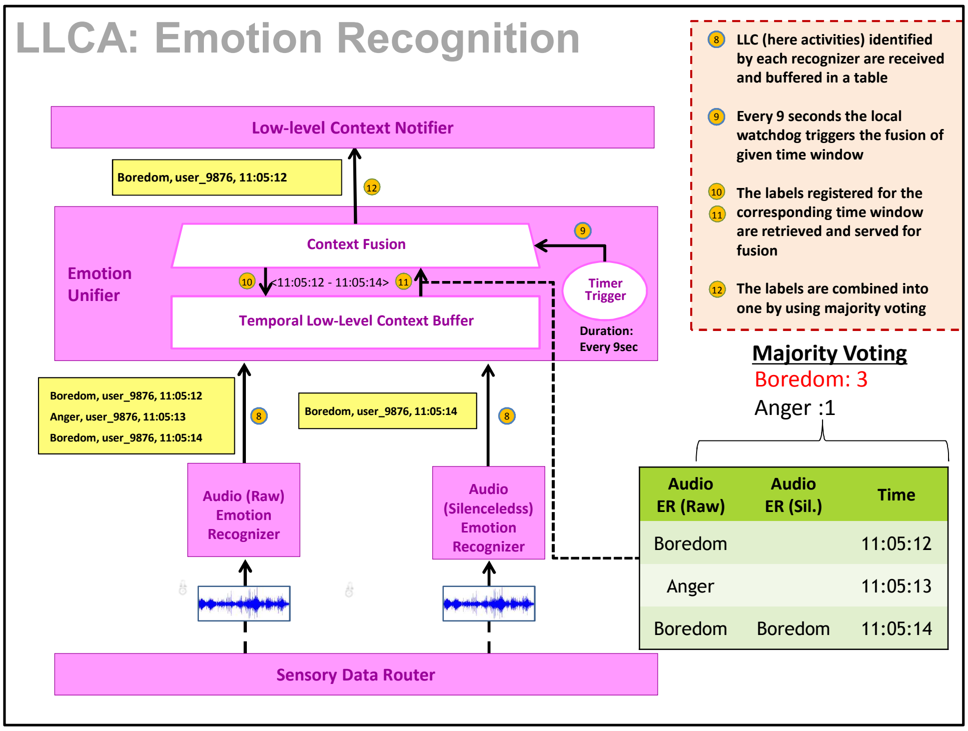
7 Probabilistic models are used after training for the identification of the emotion

8 The metadata are attached to the recognized emotion label

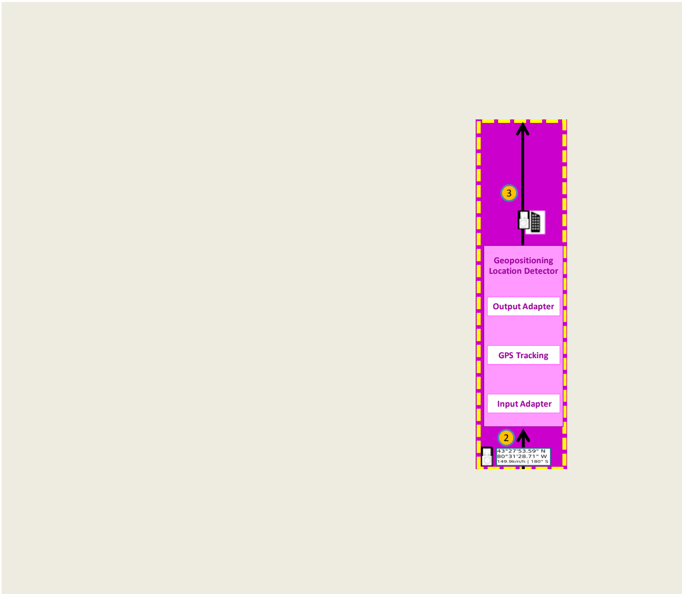

LLCA: Emotion Recognition
8 LLC (here activities) identified by each recognizer are received and buffered in a table
Low-level Context Notifier
Boredom, user_9876, 11:05:12
Emotion Unifier
Context Fusion
<11:05:12 - 11:05:14>
Temporal Low-Level Context Buffer
Timer Trigger
Duration: Every 9sec
Boredom, user_9876, 11:05:12
Anger, user_9876, 11:05:13
Boredom, user_9876, 11:05:14
Boredom, user_9876, 11:05:14
Audio (Raw) Emotion Recognizer
Audio (Silenceledss) Emotion Recognizer
Sensory Data Router
Majority Voting
Boredom: 3
Anger :1
Audio ER (Raw)
Audio ER (Sil.)
Time
Boredom
11:05:12
Anger
11:05:13
Boredom
Boredom
11:05:14


LLCA: Emotion Recognition
8 LLC (here activities) identified by each recognizer are received and buffered in a table
9 Every 9 seconds the local watchdog triggers the fusion of given time window
Low-level Context Notifier
Boredom, user_9876, 11:05:12
Emotion Unifier
Context Fusion
<11:05:12 - 11:05:14>
Temporal Low-Level Context Buffer
Timer Trigger
Duration: Every 9sec
Boredom, user_9876, 11:05:12
Anger, user_9876, 11:05:13
Boredom, user_9876, 11:05:14
Boredom, user_9876, 11:05:14
Audio (Raw) Emotion Recognizer
Audio (Silenceledss) Emotion Recognizer
Sensory Data Router
Majority Voting
Boredom: 3
Anger :1
Audio ER (Raw)
Audio ER (Sil.)
Time
Boredom
11:05:12
Anger
11:05:13
Boredom
Boredom
11:05:14


LLCA: Emotion Recognition
8 LLC (here activities) identified by each recognizer are received and buffered in a table
9 Every 9 seconds the local watchdog triggers the fusion of given time window
10 11 The labels registered for the corresponding time window are retrieved and served for fusion
Low-level Context Notifier
Boredom, user_9876, 11:05:12
Emotion Unifier
Context Fusion
9
10 <11:05:12 - 11:05:14> 11
Timer Trigger
Temporal Low-Level Context Buffer
Duration: Every 9sec
Boredom, user_9876, 11:05:12
Anger, user_9876, 11:05:13
Boredom, user_9876, 11:05:14
8
Boredom, user_9876, 11:05:14
8
Audio (Raw) Emotion Recognizer
Audio (Silenceledss) Emotion Recognizer
Sensory Data Router
Majority Voting
Boredom: 3
Anger :1
Audio ER (Raw) | Audio ER (Sil.) | Time
Boredom | | 11:05:12
Anger | | 11:05:13
Boredom | Boredom | 11:05:14

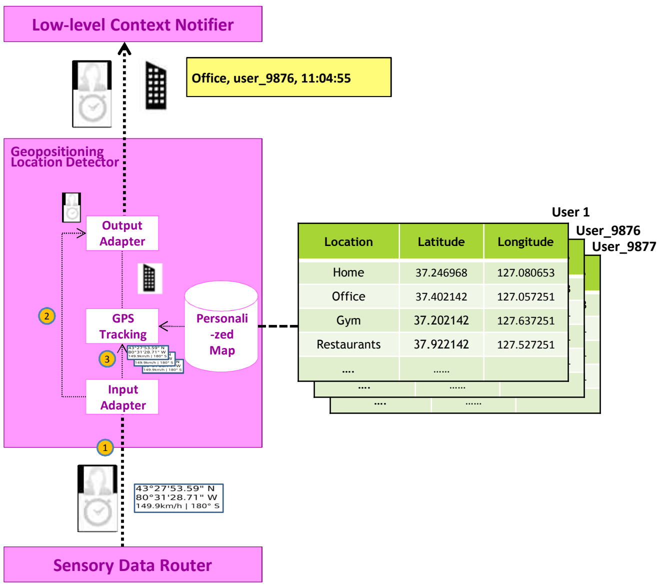

LLCA: Emotion Recognition
8 LLC (here activities) identified by each recognizer are received and buffered in a table
9 Every 9 seconds the local watchdog triggers the fusion of given time window
10 11 The labels registered for the corresponding time window are retrieved and served for fusion
12 The labels are combined into one by using majority voting
Low-level Context Notifier
Boredom, user_9876, 11:05:12
12
Emotion Unifier
Context Fusion
9
10 <11:05:12 - 11:05:14> 11
Timer Trigger
Temporal Low-Level Context Buffer
Duration: Every 9sec
Boredom, user_9876, 11:05:12
Anger, user_9876, 11:05:13
Boredom, user_9876, 11:05:14
8
Boredom, user_9876, 11:05:14
8
Audio (Raw) Emotion Recognizer
Audio (Silenceledss) Emotion Recognizer
Sensory Data Router
Majority Voting
Boredom: 3
Anger :1
Audio ER (Raw) | Audio ER (Sil.) | Time
Boredom | | 11:05:12
Anger | | 11:05:13
Boredom | Boredom | 11:05:14

LLCA: Location Detection



LLCA: Location Detection

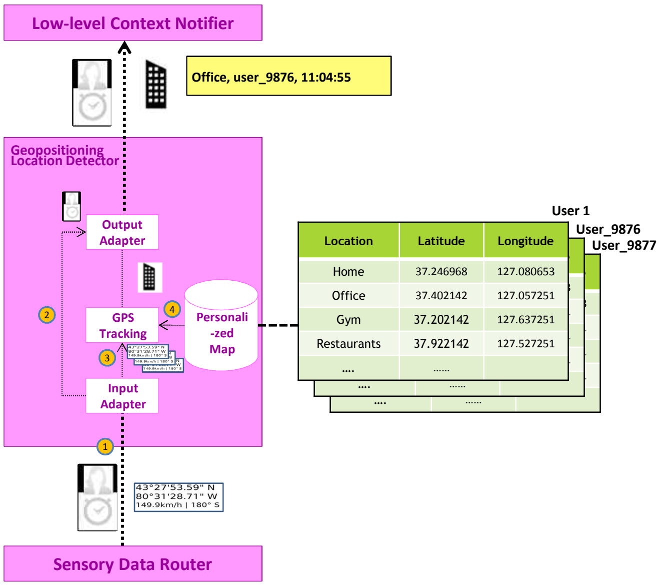
(1) (2) (3) The GPS data is separated from the metadata (userID, timestamp) for further processing (*this applies to all recognizers)



LLCA: Location Detection

1. The GPS data is separated from
2. the metadata (userID,
3. timestamp) for further processing (*this applies to all recognizers)

4. Deliver personalized map information such as latitude, longitude.



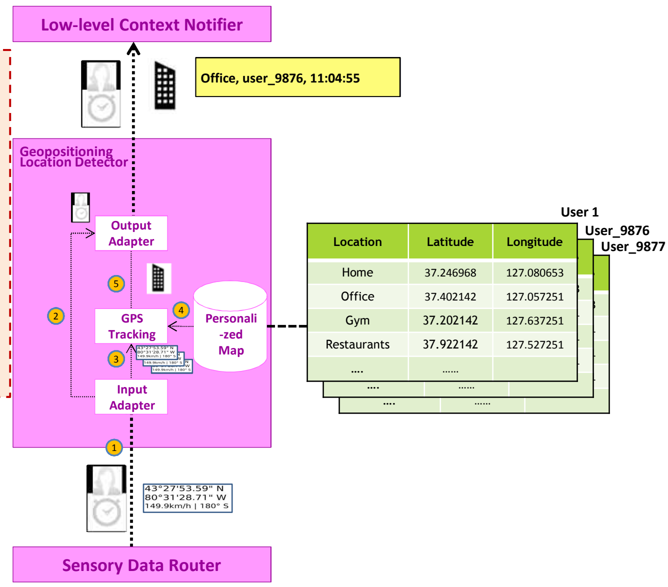
Location	Latitude	Longitude
Home	37.246968	127.080653
Office	37.402142	127.057251
Gym	37.202142	127.637251
Restaurants	37.922142	127.527251
....	......	

LLCA: Location Detection

1. The GPS data is separated from
2. the metadata (userID, timestamp)
3. for further processing (*this applies to all recognizers)

4. Deliver personalized map information such as latitude, longitude.

5. Based on the GPS coordinates (longitude and latitude) an area of confidence is calculated. And calculated coordinates within the person area are used to determine the personal location. To that end a simple lookup table is used



Location	Latitude	Longitude
Home	37.246968	127.080653
Office	37.402142	127.057251
Gym	37.202142	127.637251
Restaurants	37.922142	127.527251
....	......	

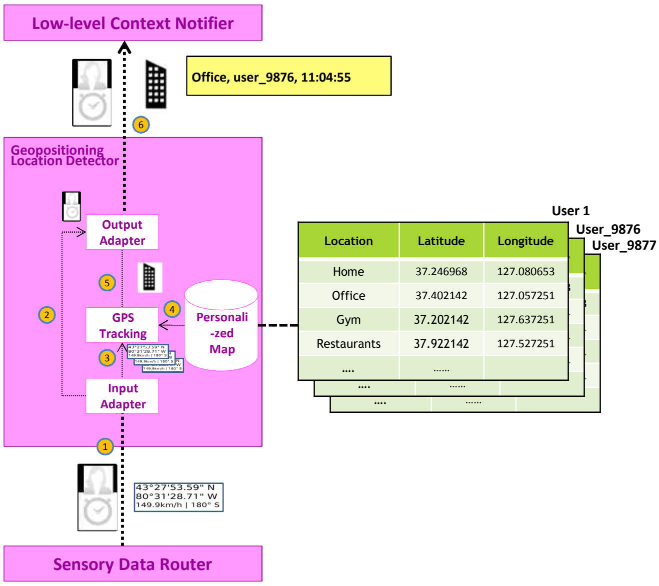
LLCA: Location Detection

1. The GPS data is separated from
2. the metadata (userID, timestamp)
3. for further processing (*this applies to all recognizers)

4. Deliver personalized map information such as latitude, longitude.

5. Based on the GPS coordinates (longitude and latitude) an area of confidence is calculated. And calculated coordinates within the person area are used to determine the personal location. To that end a simple lookup table is used

6. The metadata are attached to the recognized emotion label

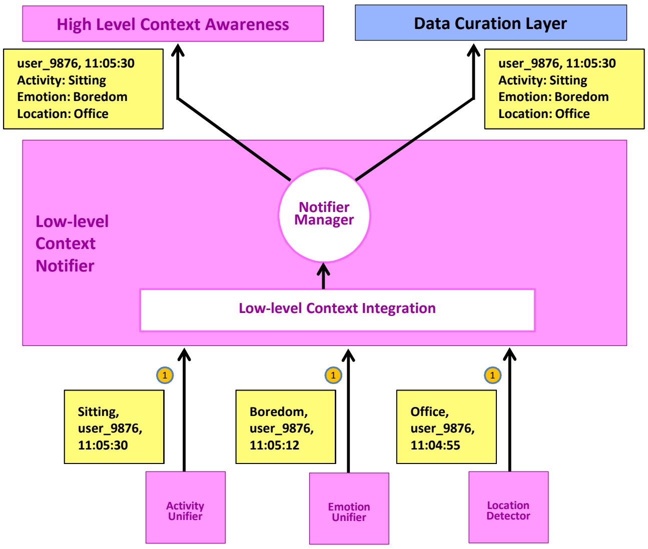


Location	Latitude	Longitude
Home	37.246968	127.080653
Office	37.402142	127.057251
Gym	37.202142	127.637251
Restaurants	37.922142	127.527251
....	......	

Scenario 2: LLC Notification to DCL & HLCA

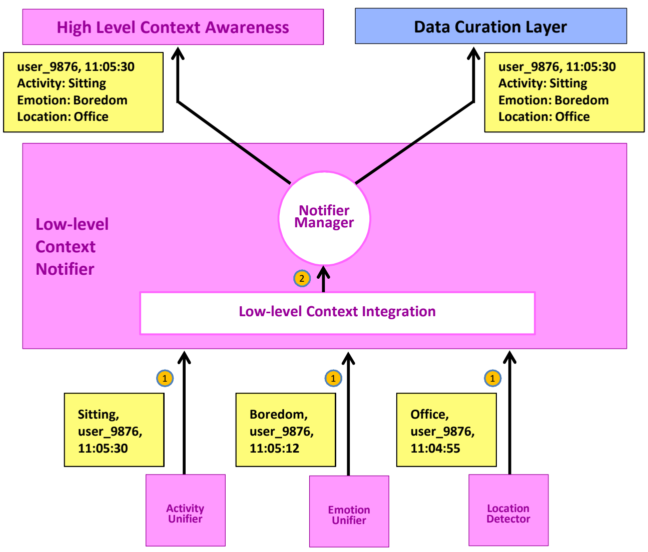
Desc. : Low-level Context is **notified** to High Level Context Awareness and **persisted** at Data Curation Layer.

LLCA: Low-level Context Notifier



1 The Low-level Context data is gathered from each recognizer with metadata (user Id, timestamp)

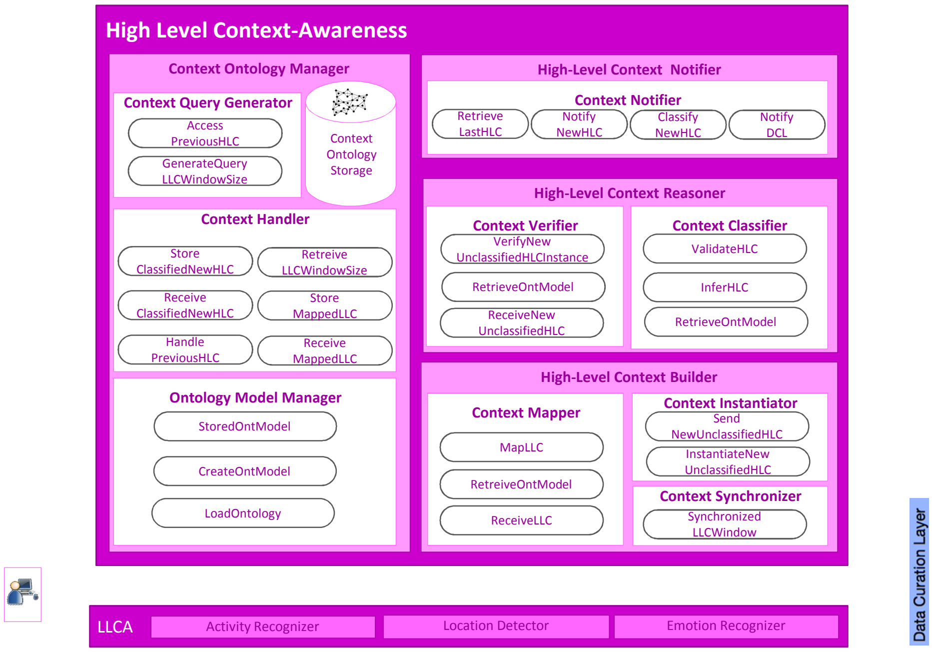
LLCA: Low-level Context Notifier



1 The Low-level Context data is gathered from each recognizer with metadata (user Id, timestamp)

2 3 Communicate the recognized Low-level Context to Data Curation Layer and High Level Context Awareness Module

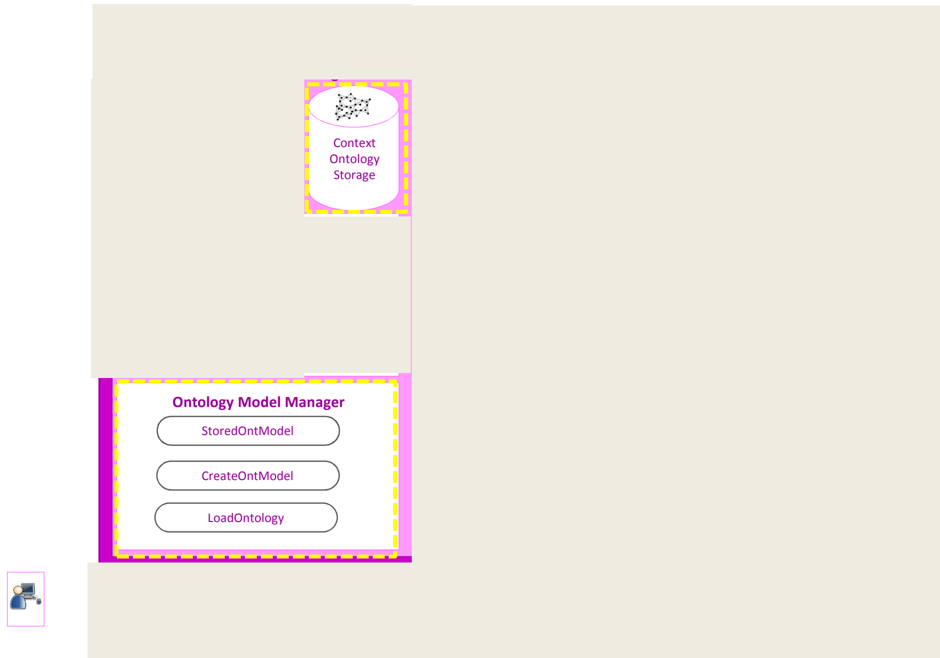
Overall HLCA



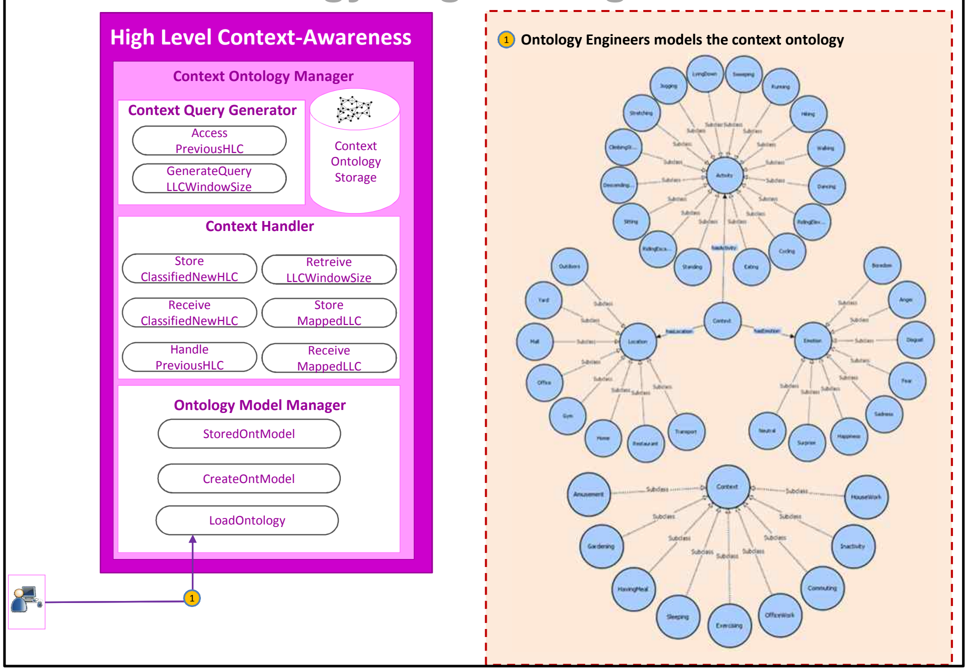
Scenario 3: Unclassified HLC from LLC

Desc. : Low level activities and High level activities are **modeled** in ontology and **persisted** in triple storage. Low level activities are **mapped**, then **synchronized** in a window size for HLC reasoning instantiation.

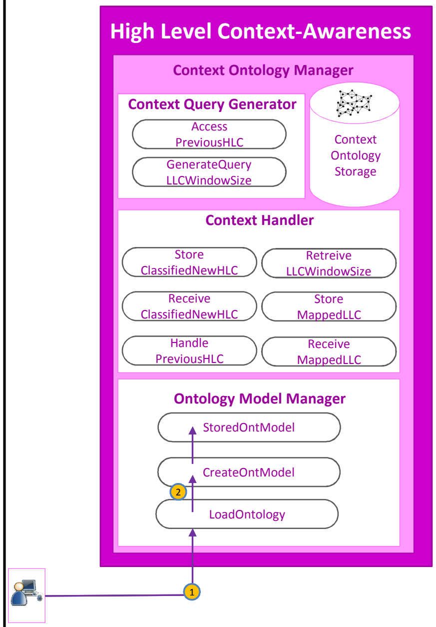
Context Ontology Engineering & Persistence



Context Ontology Engineering & Persistence

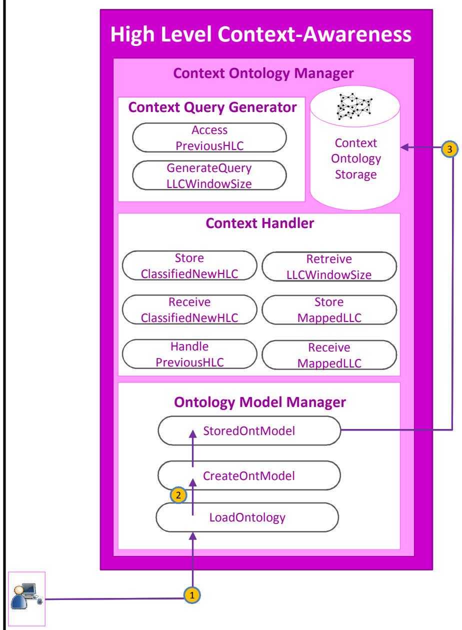


Context Ontology Engineering & Persistence



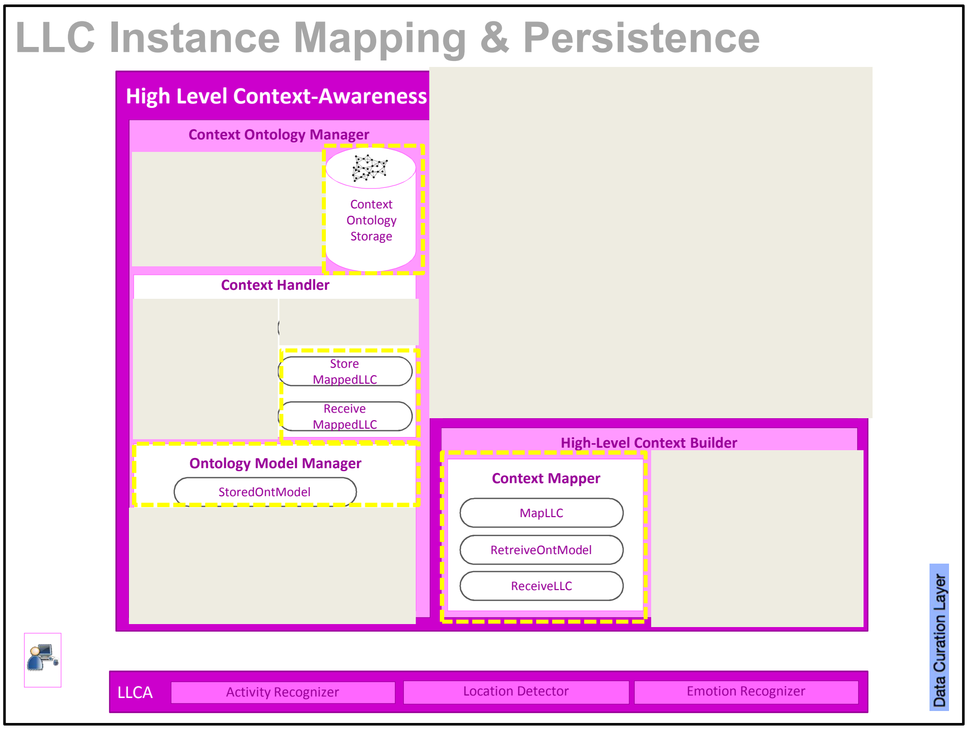
1. Ontology Engineers models the context ontology
2. Ontology is loaded and Ont Model is created for storage as triple storage

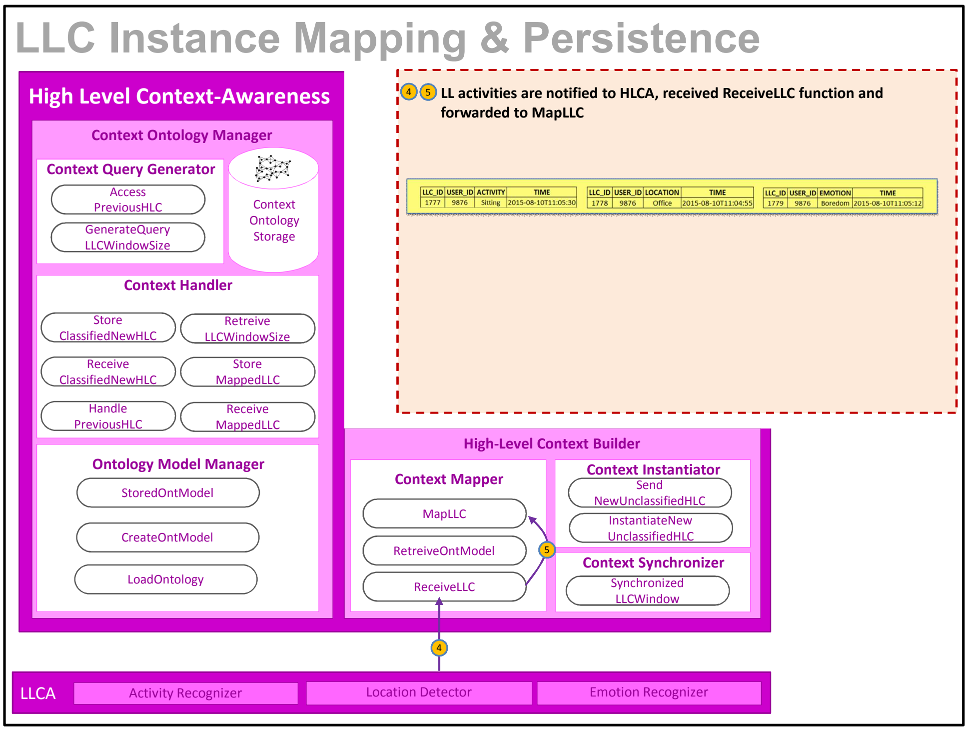
Context Ontology Engineering & Persistence

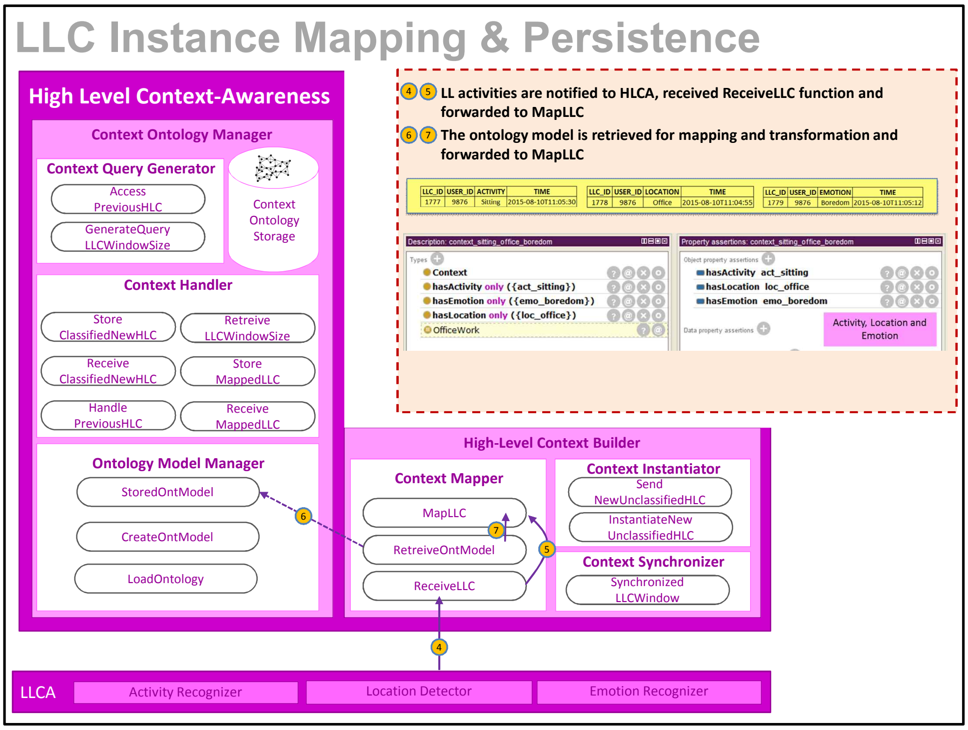


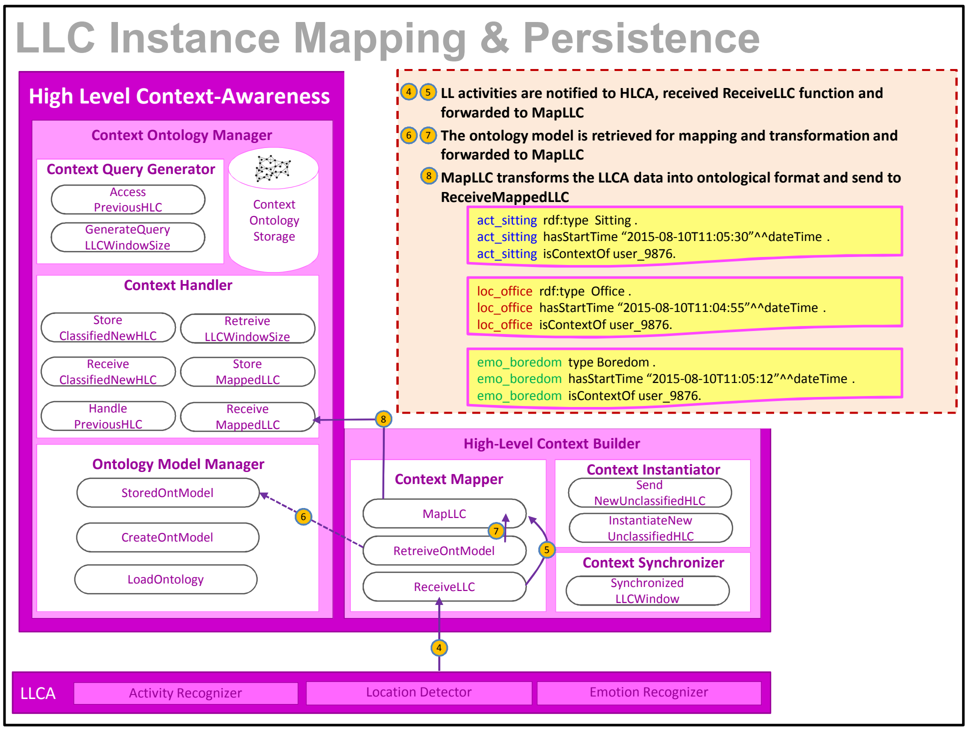
1. Ontology Engineers models the context ontology
2. Ontology is loaded and Ont Model is created for storage as triple storage
3. Store Context Ontology in Jena TDB

Subject	Predicate	Object
high-level-context	subclass of	context.
low-level-context	subclass of	context.
anger	subclass of	emotion.
happiness	subclass of	emotion.
neutral	subclass of	emotion.
sadness	subclass of	emotion.
unidentified-emotion	subclass of	emotion.
gym	subclass of	location.
home	subclass of	location.
eating	subclass of	activity.
lying-down	subclass of	activity.
running	subclass of	activity.
sitting	subclass of	activity.
standing	subclass of	activity.
stretching	subclass of	activity.
sweeping	subclass of	activity.
unidentified-activity	subclass of	activity.
walking	subclass of	activity.

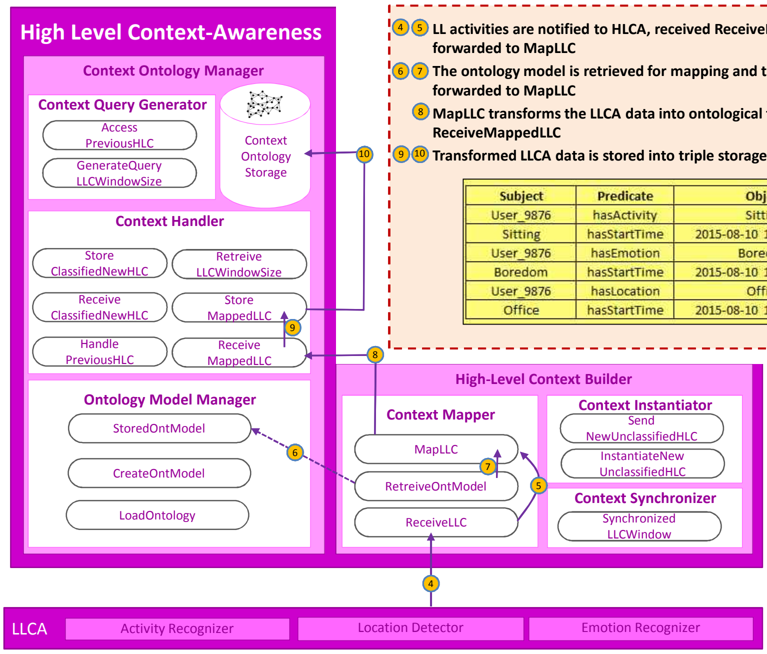

LLC Instance Mapping & Persistence
High Level Context-Awareness
Context Ontology Manager
Context Ontology Storage
Context Handler
Store MappedLLC
Receive MappedLLC
Ontology Model Manager
StoredOntModel
High-Level Context Builder
Context Mapper
MapLLC
RetreiveOntModel
ReceiveLLC
LLCA
Activity Recognizer
Location Detector
Emotion Recognizer
Data Curation Layer


LLC Instance Mapping & Persistence
High Level Context-Awareness
Context Ontology Manager
Context Query Generator
Access PreviousHLC
GenerateQuery LLCWindowSize
Context Ontology Storage
Context Handler
Store ClassifiedNewHLC
Retreive LLCWindowSize
Receive ClassifiedNewHLC
Store MappedLLC
Handle PreviousHLC
Receive MappedLLC
Ontology Model Manager
StoredOntModel
CreateOntModel
LoadOntology
4 5 LL activities are notified to HLCA, received ReceiveLLC function and forwarded to MapLLC
LLC_ID | USER_ID | ACTIVITY | TIME
1777 | 9876 | Sitting | 2015-08-10T11:05:30
LLC_ID | USER_ID | LOCATION | TIME
1778 | 9876 | Office | 2015-08-10T11:04:55
LLC_ID | USER_ID | EMOTION | TIME
1779 | 9876 | Boredom | 2015-08-10T11:05:12
High-Level Context Builder
Context Mapper
MapLLC
RetreiveOntModel
ReceiveLLC
Context Instantiator
Send NewUnclassifiedHLC
InstantiateNew UnclassifiedHLC
Context Synchronizer
Synchronized LLCWindow
5
4
LLCA
Activity Recognizer
Location Detector
Emotion Recognizer


LLC Instance Mapping & Persistence
4 5 LL activities are notified to HLCA, received ReceiveLLC function and forwarded to MapLLC
6 7 The ontology model is retrieved for mapping and transformation and forwarded to MapLLC
LLC_ID USER_ID ACTIVITY TIME
1777 9876 Sitting 2015-08-10T11:05:30
LLC_ID USER_ID LOCATION TIME
1778 9876 Office 2015-08-10T11:04:55
LLC_ID USER_ID EMOTION TIME
1779 9876 Boredom 2015-08-10T11:05:12
Description: context_sitting_office_boredom
Types
Context
hasActivity only ({act_sitting})
hasEmotion only ({emo_boredom})
hasLocation only ({loc_office})
OfficeWork
Property assertions: context_sitting_office_boredom
Object property assertions
hasActivity act_sitting
hasLocation loc_office
hasEmotion emo_boredom
Data property assertions
Activity, Location and Emotion
High Level Context-Awareness
Context Ontology Manager
Context Query Generator
Access PreviousHLC
GenerateQuery LLCWindowSize
Context Ontology Storage
Context Handler
Store ClassifiedNewHLC
Retreive LLCWindowSize
Receive ClassifiedNewHLC
Store MappedLLC
Handle PreviousHLC
Receive MappedLLC
Ontology Model Manager
StoredOntModel
CreateOntModel
LoadOntology
High-Level Context Builder
Context Mapper
MapLLC
RetreiveOntModel
ReceiveLLC
Context Instantiator
Send NewUnclassifiedHLC
InstantiateNew UnclassifiedHLC
Context Synchronizer
Synchronized LLCWindow
LLCA
Activity Recognizer
Location Detector
Emotion Recognizer


LLC Instance Mapping & Persistence
4 5 LL activities are notified to HLCA, received ReceiveLLC function and forwarded to MapLLC
6 7 The ontology model is retrieved for mapping and transformation and forwarded to MapLLC
8 MapLLC transforms the LLCA data into ontological format and send to ReceiveMappedLLC
act_sitting rdf:type Sitting .
act_sitting hasStartTime "2015-08-10T11:05:30"^^dateTime .
act_sitting isContextOf user_9876.
loc_office rdf:type Office .
loc_office hasStartTime "2015-08-10T11:04:55"^^dateTime .
loc_office isContextOf user_9876.
emo_boredom type Boredom .
emo_boredom hasStartTime "2015-08-10T11:05:12"^^dateTime .
emo_boredom isContextOf user_9876.
High Level Context-Awareness
Context Ontology Manager
Context Query Generator
Access PreviousHLC
GenerateQuery LLCWindowSize
Context Ontology Storage
Context Handler
Store ClassifiedNewHLC
Retreive LLCWindowSize
Receive ClassifiedNewHLC
Store MappedLLC
Handle PreviousHLC
Receive MappedLLC
Ontology Model Manager
StoredOntModel
CreateOntModel
LoadOntology
High-Level Context Builder
Context Mapper
MapLLC
RetreiveOntModel
ReceiveLLC
Context Instantiator
Send NewUnclassifiedHLC
InstantiateNew UnclassifiedHLC
Context Synchronizer
Synchronized LLCWindow
LLCA
Activity Recognizer
Location Detector
Emotion Recognizer

LLC Instance Mapping & Persistence



4 5 LL activities are notified to HLCA, received ReceiveLLC function and forwarded to MapLLC

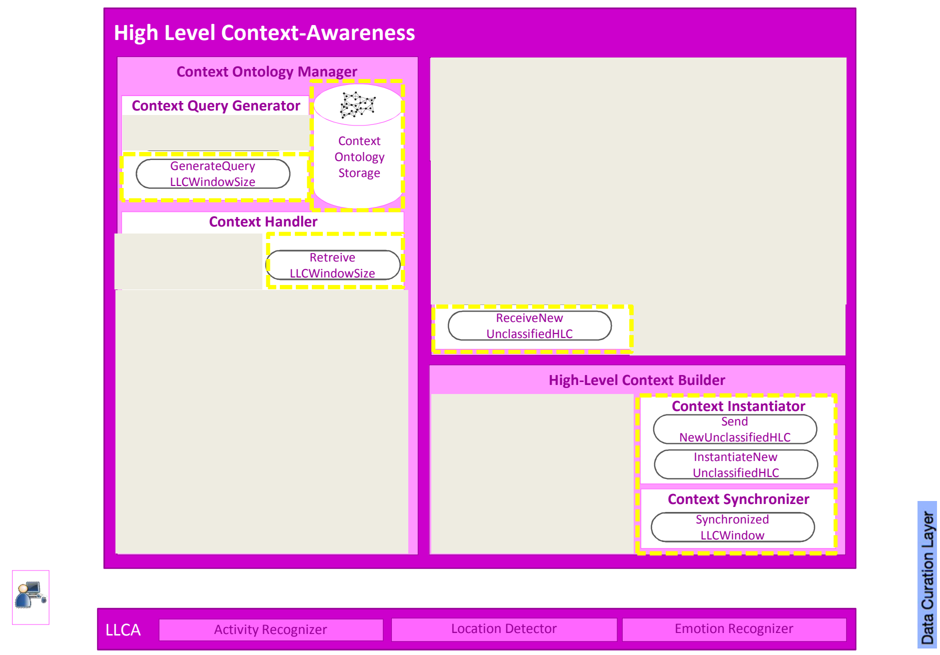
6 7 The ontology model is retrieved for mapping and transformation and forwarded to MapLLC

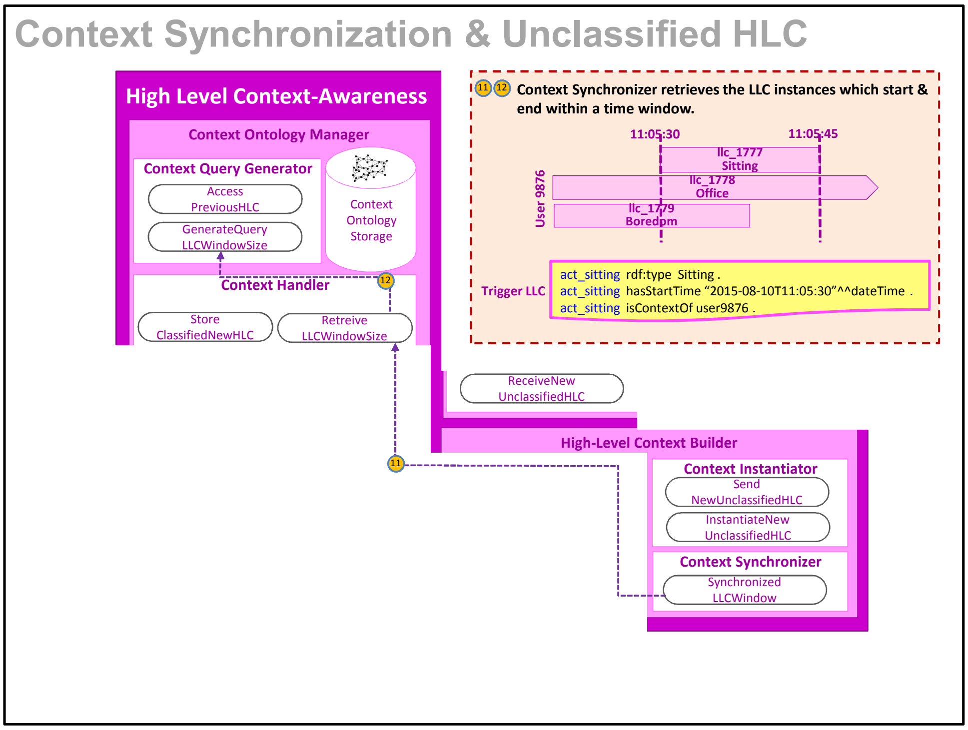
8 MapLLC transforms the LLCA data into ontological format and send to ReceiveMappedLLC

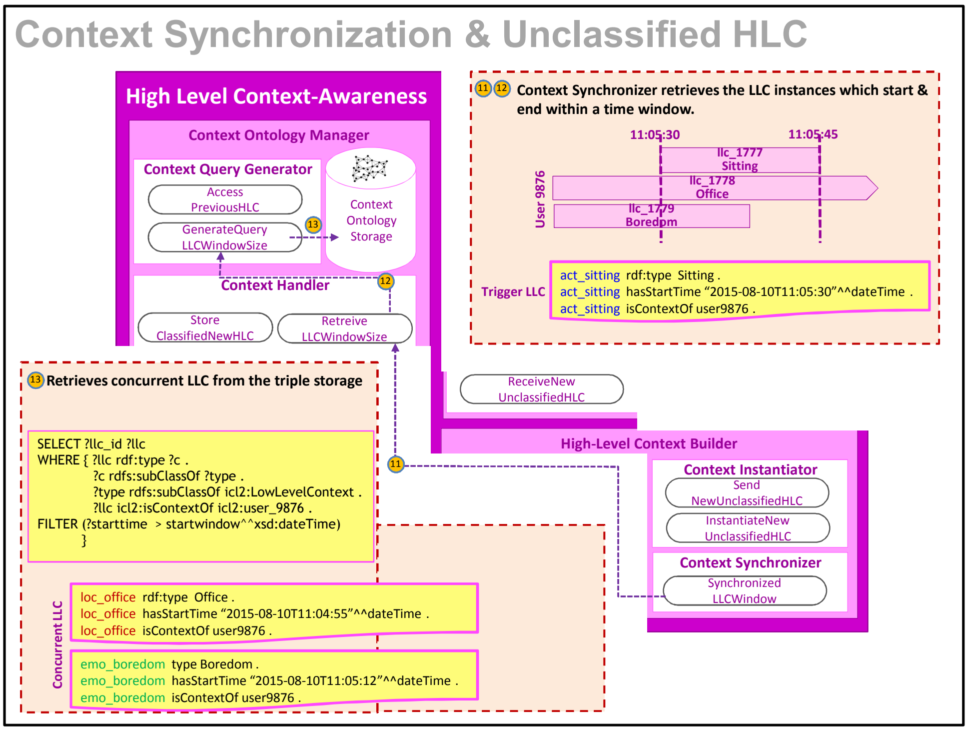
9 10 Transformed LLCA data is stored into triple storage

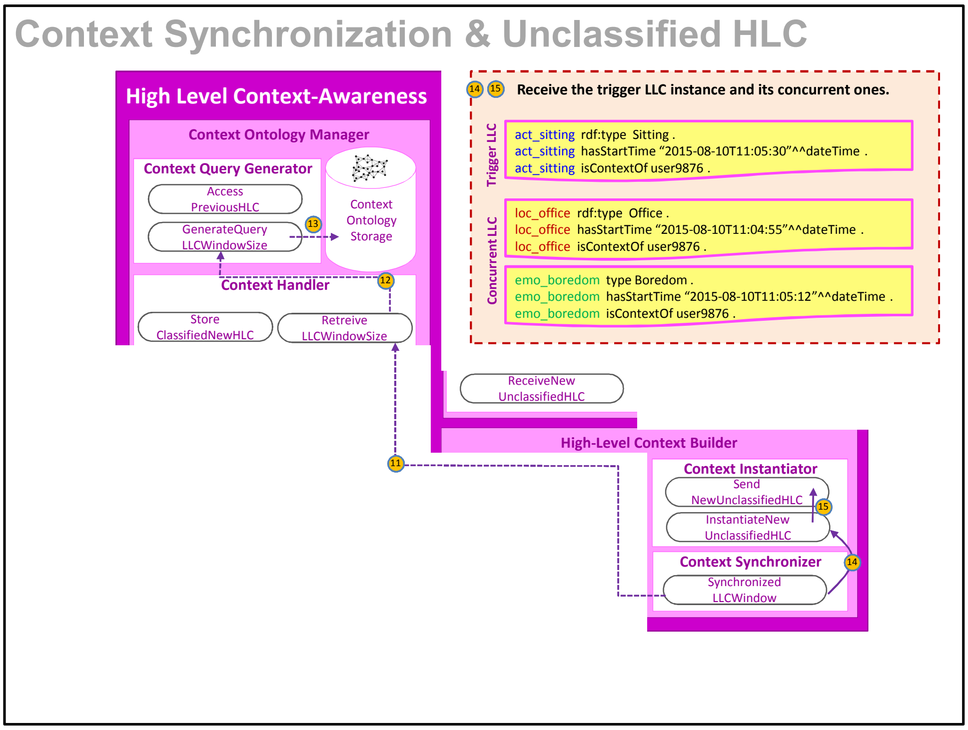
Subject	Predicate	Object
User_9876	hasActivity	Sitting
Sitting	hasStartTime	2015-08-10 11:05:30 AM
User_9876	hasEmotion	Boredom
Boredom	hasStartTime	2015-08-10 11:05:12 AM
User_9876	hasLocation	Office
Office	hasStartTime	2015-08-10 11:04:55 AM

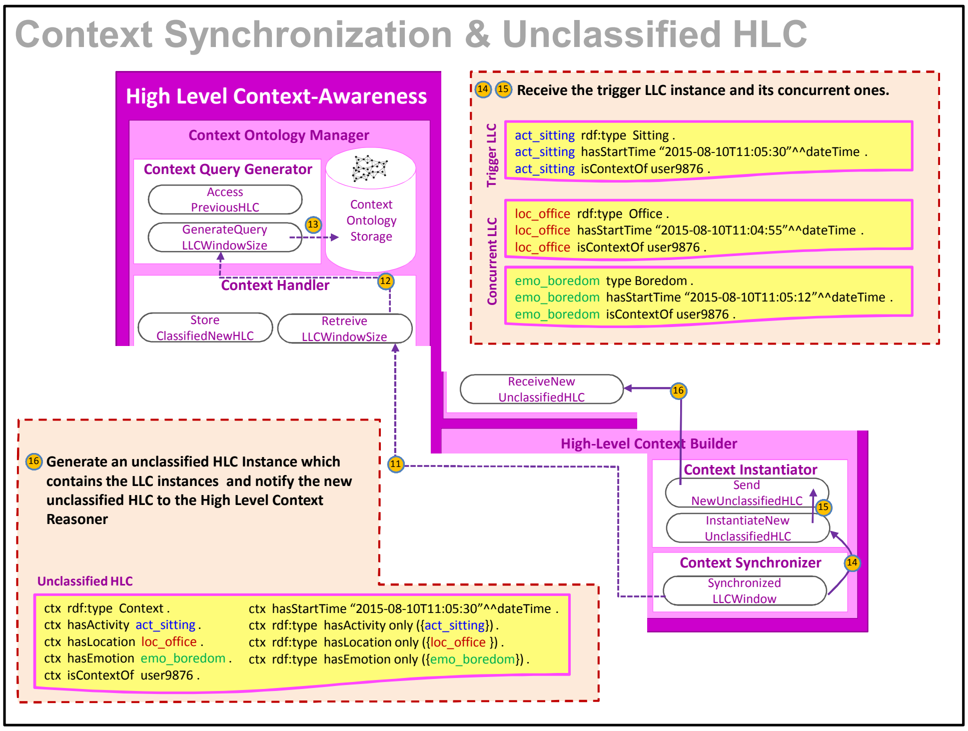
LLC Instance Mapping & Persistence




Context Synchronization & Unclassified HLC
High Level Context-Awareness
Context Ontology Manager
Context Query Generator
Access PreviousHLC
GenerateQuery LLCWindowSize
Context Ontology Storage
Context Handler
Store ClassifiedNewHLC
Retreive LLCWindowSize
11 12 Context Synchronizer retrieves the LLC instances which start & end within a time window.
11:05:30
11:05:45
llc_1777 Sitting
llc_1778 Office
llc_1779 Boredom
User 9876
Trigger LLC
act_sitting rdf:type Sitting .
act_sitting hasStartTime "2015-08-10T11:05:30"^^dateTime .
act_sitting isContextOf user9876 .
ReceiveNew UnclassifiedHLC
High-Level Context Builder
Context Instantiator
Send NewUnclassifiedHLC
InstantiateNew UnclassifiedHLC
Context Synchronizer
Synchronized LLCWindow


Context Synchronization & Unclassified HLC
High Level Context-Awareness
Context Ontology Manager
Context Query Generator
Access PreviousHLC
GenerateQuery LLCWindowSize
Context Ontology Storage
Context Handler
Store ClassifiedNewHLC
Retreive LLCWindowSize
11 12 Context Synchronizer retrieves the LLC instances which start & end within a time window.
11:05:30
11:05:45
llc_1777 Sitting
llc_1778 Office
llc_1779 Boredom
User 9876
Trigger LLC
act_sitting rdf:type Sitting .
act_sitting hasStartTime "2015-08-10T11:05:30"^^dateTime .
act_sitting isContextOf user9876 .
13 Retrieves concurrent LLC from the triple storage
SELECT ?llc_id ?llc
WHERE { ?llc rdf:type ?c .
?c rdfs:subClassOf ?type .
?type rdfs:subClassOf icl2:LowLevelContext .
?llc icl2:isContextOf icl2:user_9876 .
FILTER (?starttime > startwindow^^xsd:dateTime)
}
Concurrent LLC
loc_office rdf:type Office .
loc_office hasStartTime "2015-08-10T11:04:55"^^dateTime .
loc_office isContextOf user9876 .
emo_boredom type Boredom .
emo_boredom hasStartTime "2015-08-10T11:05:12"^^dateTime .
emo_boredom isContextOf user9876 .
ReceiveNew UnclassifiedHLC
High-Level Context Builder
Context Instantiator
Send NewUnclassifiedHLC
InstantiateNew UnclassifiedHLC
Context Synchronizer
Synchronized LLCWindow

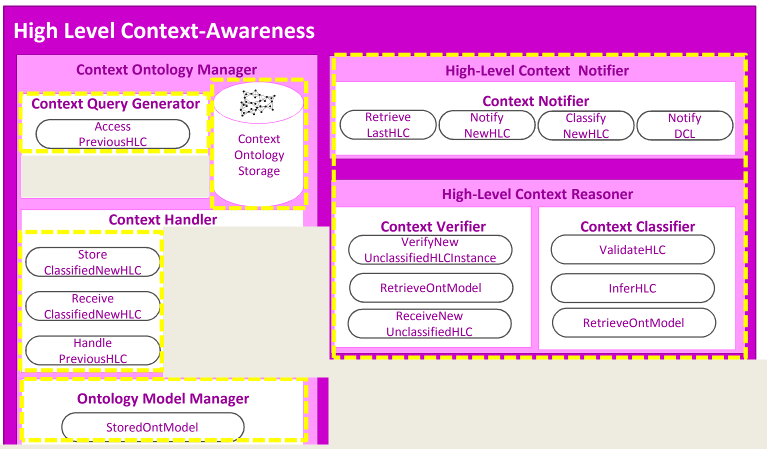

Context Synchronization & Unclassified HLC
High Level Context-Awareness
Context Ontology Manager
Context Query Generator
Access PreviousHLC
GenerateQuery LLCWindowSize
Context Ontology Storage
Context Handler
Store ClassifiedNewHLC
Retreive LLCWindowSize
ReceiveNew UnclassifiedHLC
High-Level Context Builder
Context Instantiator
Send NewUnclassifiedHLC
InstantiateNew UnclassifiedHLC
Context Synchronizer
Synchronized LLCWindow
Receive the trigger LLC instance and its concurrent ones.
Trigger LLC
act_sitting rdf:type Sitting .
act_sitting hasStartTime "2015-08-10T11:05:30"^^dateTime .
act_sitting isContextOf user9876 .
Concurrent LLC
loc_office rdf:type Office .
loc_office hasStartTime "2015-08-10T11:04:55"^^dateTime .
loc_office isContextOf user9876 .
emo_boredom type Boredom .
emo_boredom hasStartTime "2015-08-10T11:05:12"^^dateTime .
emo_boredom isContextOf user9876 .

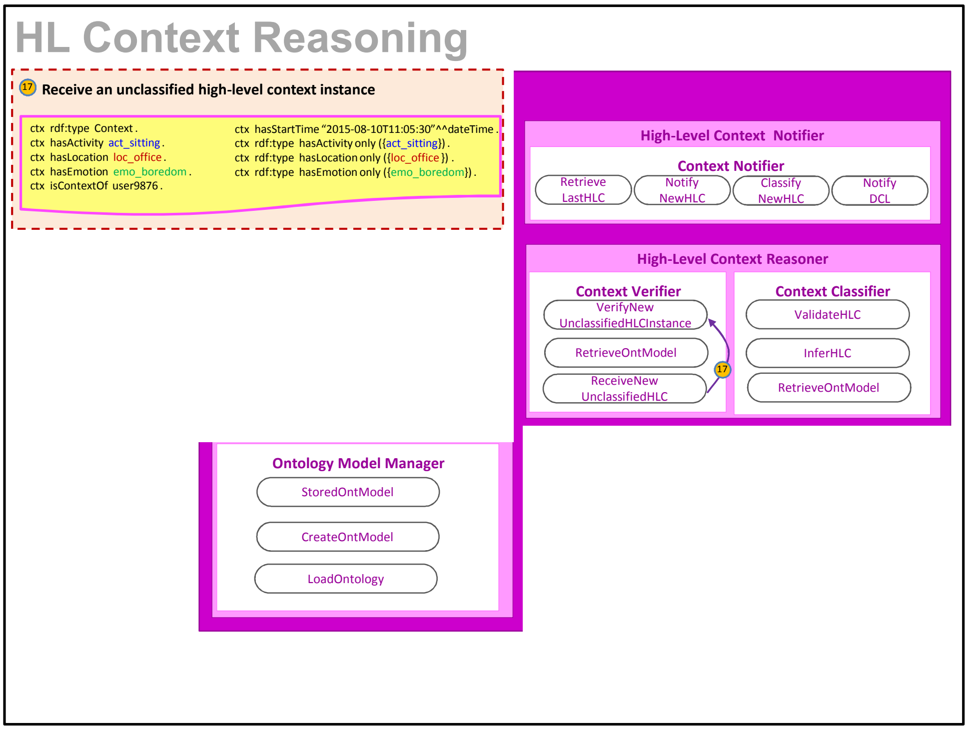

Context Synchronization & Unclassified HLC
High Level Context-Awareness
Context Ontology Manager
Context Query Generator
Access PreviousHLC
GenerateQuery LLCWindowSize
Context Ontology Storage
Context Handler
Store ClassifiedNewHLC
Retreive LLCWindowSize
ReceiveNew UnclassifiedHLC
High-Level Context Builder
Context Instantiator
Send NewUnclassifiedHLC
InstantiateNew UnclassifiedHLC
Context Synchronizer
Synchronized LLCWindow
Receive the trigger LLC instance and its concurrent ones.
Trigger LLC
act_sitting rdf:type Sitting .
act_sitting hasStartTime "2015-08-10T11:05:30"^^dateTime .
act_sitting isContextOf user9876 .
Concurrent LLC
loc_office rdf:type Office .
loc_office hasStartTime "2015-08-10T11:04:55"^^dateTime .
loc_office isContextOf user9876 .
emo_boredom type Boredom .
emo_boredom hasStartTime "2015-08-10T11:05:12"^^dateTime .
emo_boredom isContextOf user9876 .
Generate an unclassified HLC Instance which contains the LLC instances and notify the new unclassified HLC to the High Level Context Reasoner
Unclassified HLC
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .

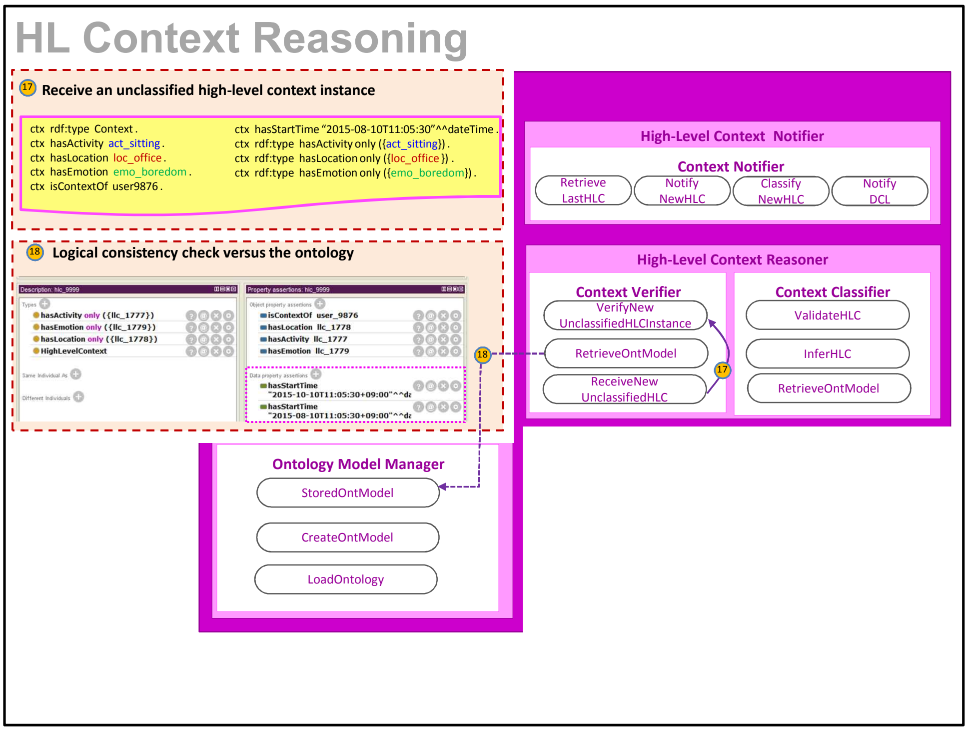
Scenario 4: Classified HLC to DCL

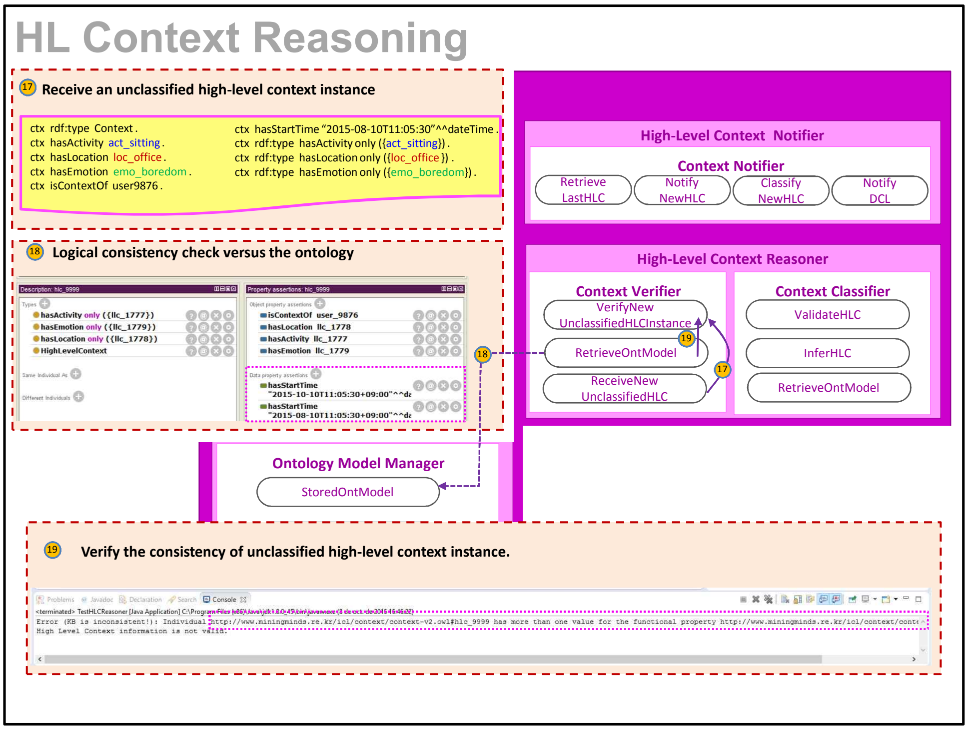
Desc. : LLC instances based **reasoning** for HLC identification and **persistence** in DCL and context ontology storage.

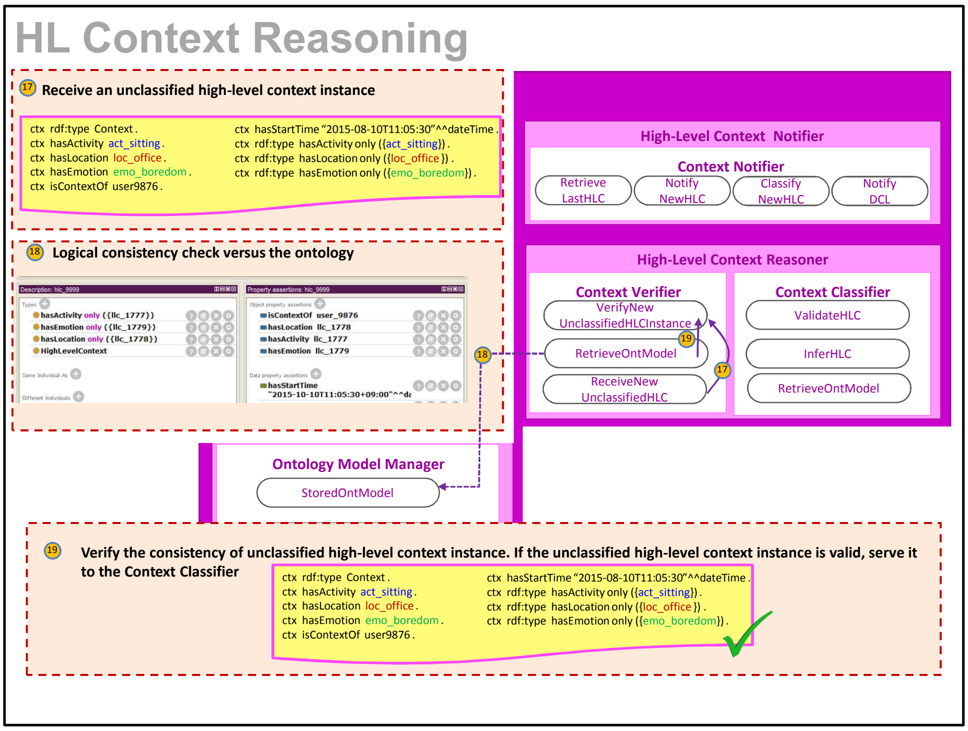
HL Context Reasoning

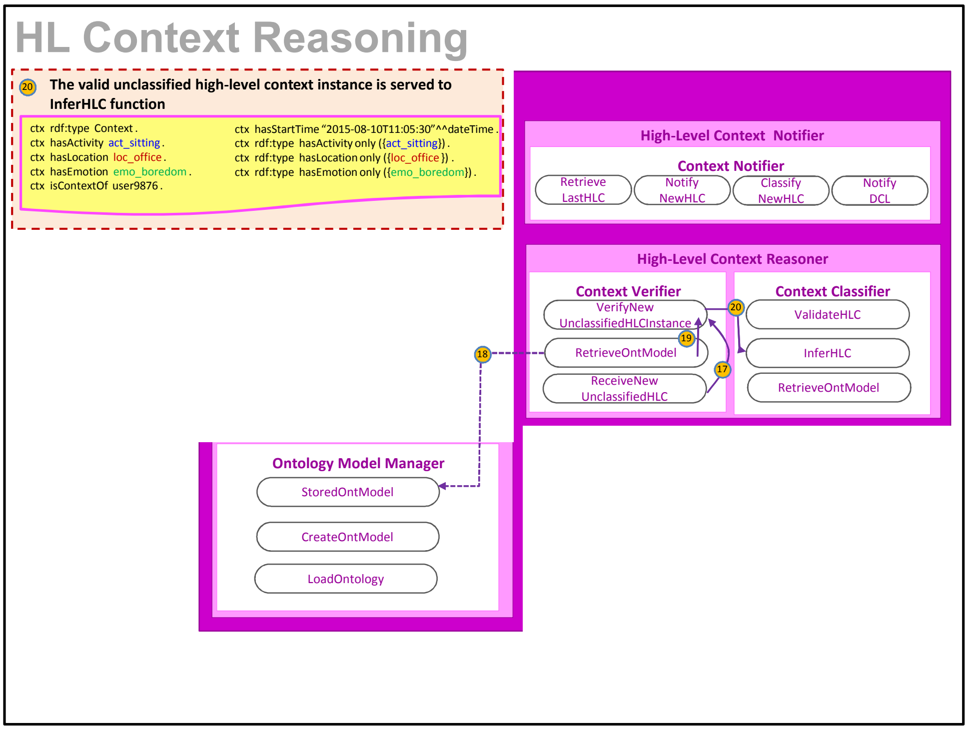


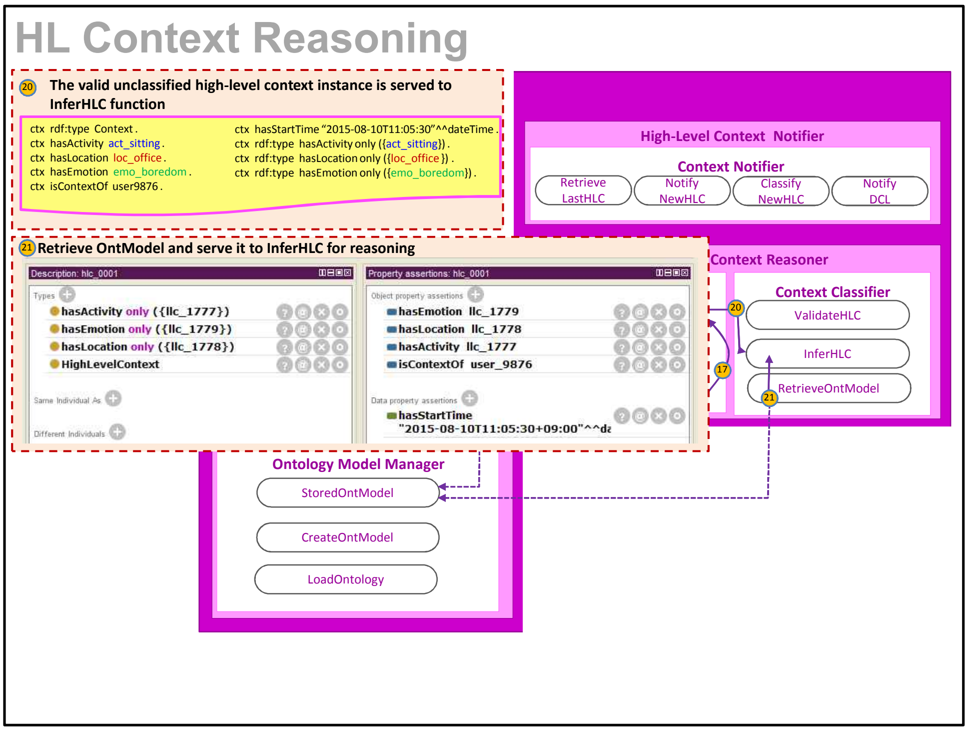

HL Context Reasoning
17 Receive an unclassified high-level context instance
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office}) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
High-Level Context Reasoner
Context Verifier
VerifyNew UnclassifiedHLCInstance
RetrieveOntModel
ReceiveNew UnclassifiedHLC
17
Context Classifier
ValidateHLC
InferHLC
RetrieveOntModel
Ontology Model Manager
StoredOntModel
CreateOntModel
LoadOntology

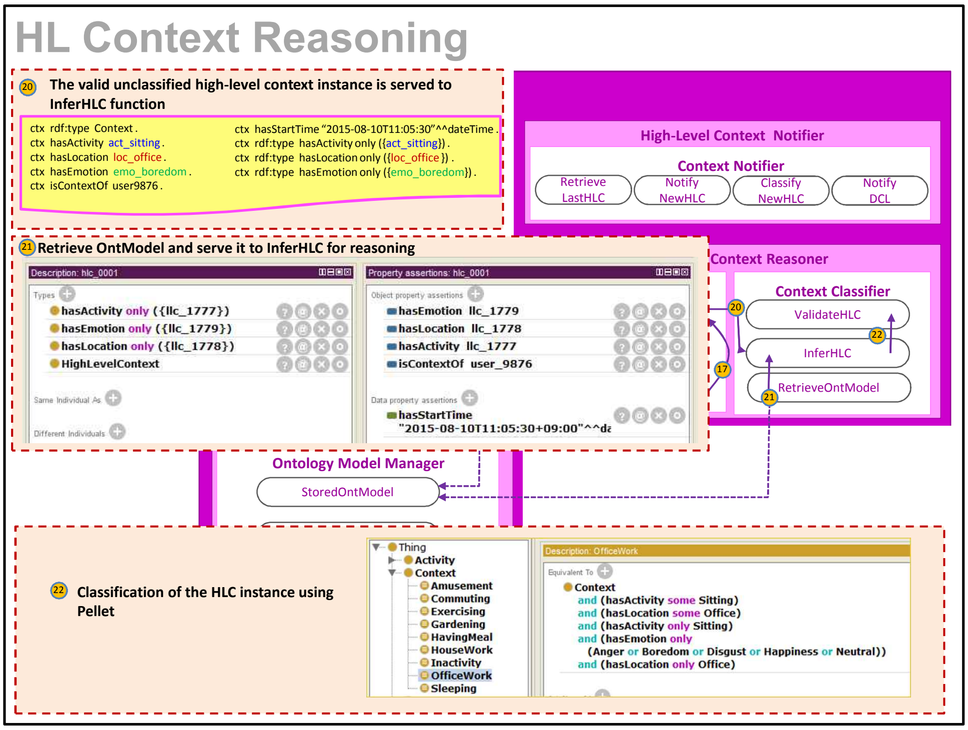

HL Context Reasoning
17 Receive an unclassified high-level context instance
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office}) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
18 Logical consistency check versus the ontology
hasActivity only ({llc_1777})
hasEmotion only ({llc_1779})
hasLocation only ({llc_1778})
HighLevelContext
isContextOf user_9876
hasLocation llc_1778
hasActivity llc_1777
hasEmotion llc_1779
hasStartTime
hasStartTime
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
High-Level Context Reasoner
Context Verifier
VerifyNew UnclassifiedHLCInstance
RetrieveOntModel
ReceiveNew UnclassifiedHLC
17
18
Context Classifier
ValidateHLC
InferHLC
RetrieveOntModel
Ontology Model Manager
StoredOntModel
CreateOntModel
LoadOntology

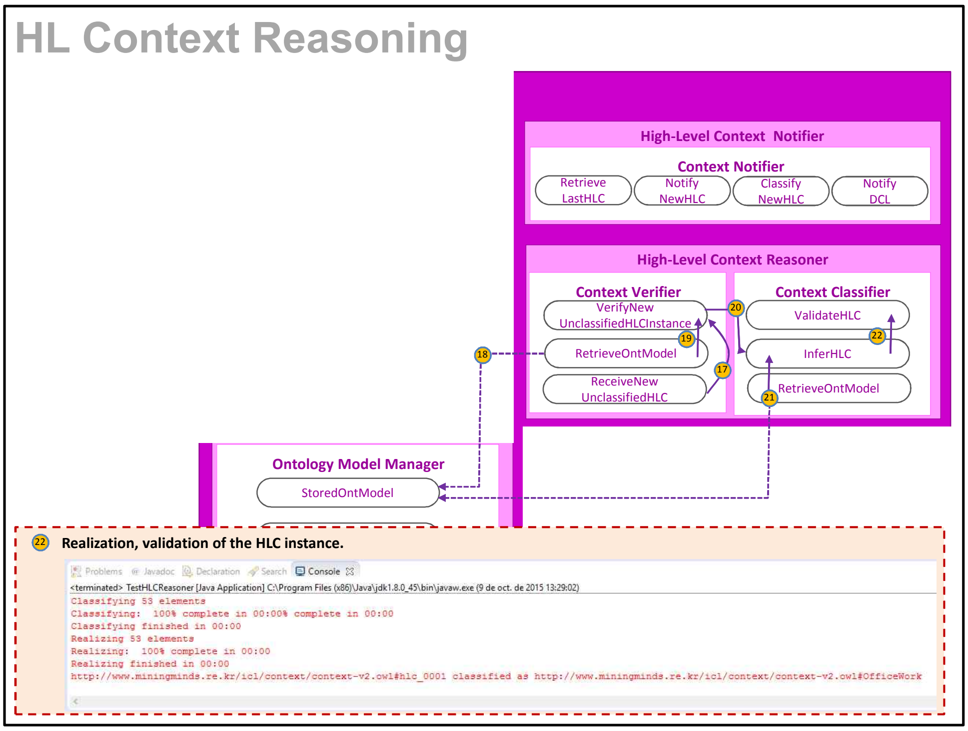

HL Context Reasoning
17 Receive an unclassified high-level context instance
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office}) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
18 Logical consistency check versus the ontology
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
High-Level Context Reasoner
Context Verifier
VerifyNew UnclassifiedHLCInstance
RetrieveOntModel
ReceiveNew UnclassifiedHLC
Context Classifier
ValidateHLC
InferHLC
RetrieveOntModel
Ontology Model Manager
StoredOntModel
19 Verify the consistency of unclassified high-level context instance.

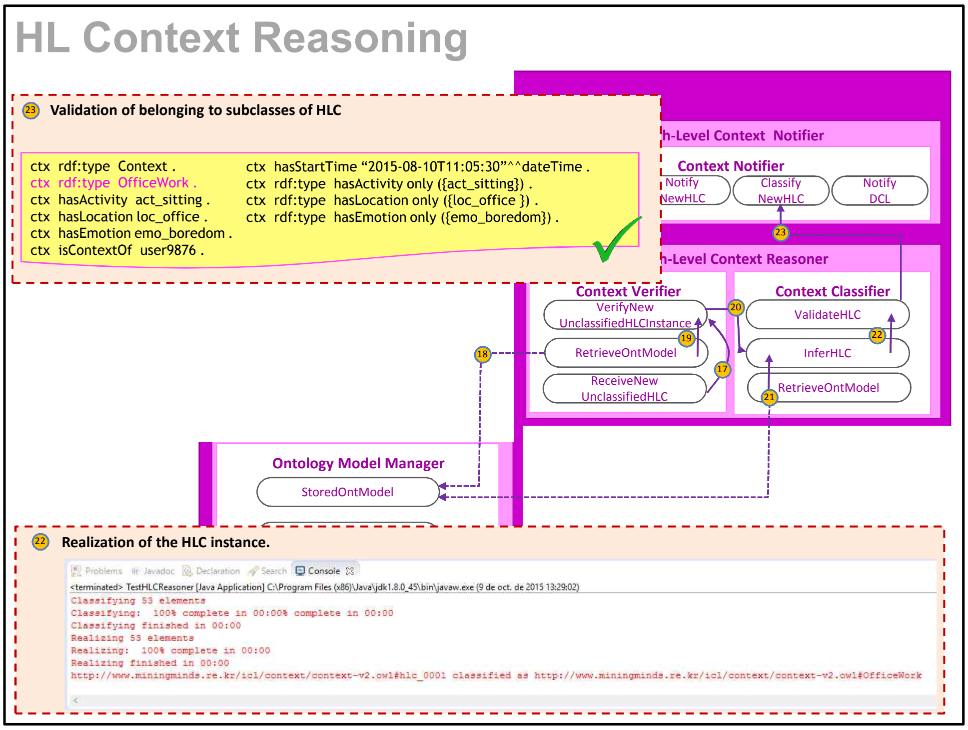

HL Context Reasoning
17 Receive an unclassified high-level context instance
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office}) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
18 Logical consistency check versus the ontology
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
High-Level Context Reasoner
Context Verifier
VerifyNew UnclassifiedHLCInstance
RetrieveOntModel
ReceiveNew UnclassifiedHLC
Context Classifier
ValidateHLC
InferHLC
RetrieveOntModel
Ontology Model Manager
StoredOntModel
19 Verify the consistency of unclassified high-level context instance. If the unclassified high-level context instance is valid, serve it to the Context Classifier
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office}) .
ctx rdf:type hasEmotion only ({emo_boredom}) .

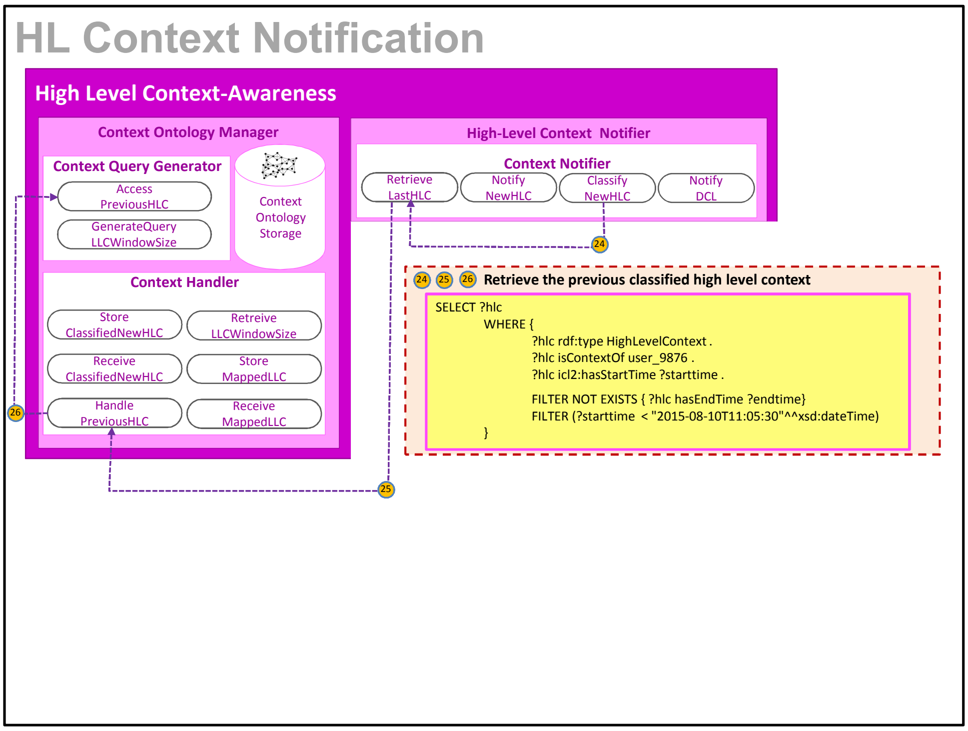

HL Context Reasoning
20 The valid unclassified high-level context instance is served to InferHLC function
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office}) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
High-Level Context Reasoner
Context Verifier
VerifyNew UnclassifiedHLCInstance
RetrieveOntModel
ReceiveNew UnclassifiedHLC
Context Classifier
ValidateHLC
InferHLC
RetrieveOntModel
Ontology Model Manager
StoredOntModel
CreateOntModel
LoadOntology


HL Context Reasoning
20 The valid unclassified high-level context instance is served to InferHLC function
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office}) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
21 Retrieve OntModel and serve it to InferHLC for reasoning
Description: hlc_0001
Types
hasActivity only ({llc_1777})
hasEmotion only ({llc_1779})
hasLocation only ({llc_1778})
HighLevelContext
Same Individual As
Different Individuals
Property assertions: hlc_0001
Object property assertions
hasEmotion llc_1779
hasLocation llc_1778
hasActivity llc_1777
isContextOf user_9876
Data property assertions
hasStartTime "2015-08-10T11:05:30+09:00"^^da
Context Reasoner
Context Classifier
ValidateHLC
InferHLC
RetrieveOntModel
Ontology Model Manager
StoredOntModel
CreateOntModel
LoadOntology

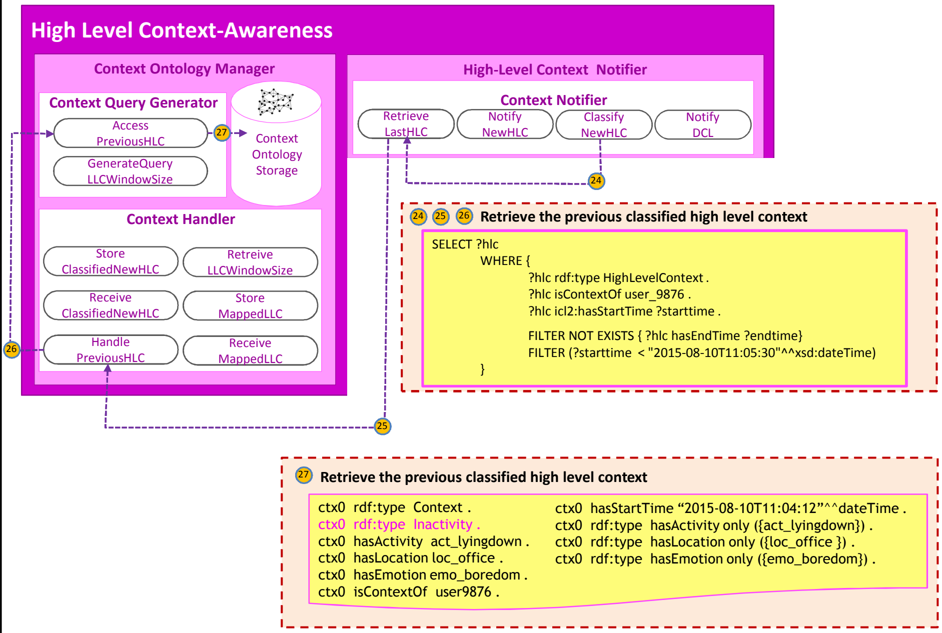

HL Context Reasoning
20 The valid unclassified high-level context instance is served to InferHLC function
ctx rdf:type Context .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office}) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
21 Retrieve OntModel and serve it to InferHLC for reasoning
Description: hlc_0001
Types
hasActivity only ({llc_1777})
hasEmotion only ({llc_1779})
hasLocation only ({llc_1778})
HighLevelContext
Same Individual As
Different Individuals
Property assertions: hlc_0001
Object property assertions
hasEmotion llc_1779
hasLocation llc_1778
hasActivity llc_1777
isContextOf user_9876
Data property assertions
hasStartTime "2015-08-10T11:05:30+09:00"^^da
Context Reasoner
Context Classifier
ValidateHLC
InferHLC
RetrieveOntModel
Ontology Model Manager
StoredOntModel
22 Classification of the HLC instance using Pellet
Thing
Activity
Context
Amusement
Commuting
Exercising
Gardening
HavingMeal
HouseWork
Inactivity
OfficeWork
Sleeping
Description: OfficeWork
Equivalent To
Context
and (hasActivity some Sitting)
and (hasLocation some Office)
and (hasActivity only Sitting)
and (hasEmotion only
(Anger or Boredom or Disgust or Happiness or Neutral))
and (hasLocation only Office)


HL Context Reasoning
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
High-Level Context Reasoner
Context Verifier
VerifyNew UnclassifiedHLCInstance
RetrieveOntModel
ReceiveNew UnclassifiedHLC
Context Classifier
ValidateHLC
InferHLC
RetrieveOntModel
Ontology Model Manager
StoredOntModel
22 Realization, validation of the HLC instance.
Problems Javadoc Declaration Search Console
<terminated> TestHLCReasoner [Java Application] C:\Program Files (x86)\Java\jdk1.8.0_45\bin\javaw.exe (9 de oct. de 2015 13:29:02)
Classifying 53 elements
Classifying: 100% complete in 00:00% complete in 00:00
Classifying finished in 00:00
Realizing 53 elements
Realizing: 100% complete in 00:00
Realizing finished in 00:00
http://www.miningminds.re.kr/icl/context/context-v2.owl#hlc_0001 classified as http://www.miningminds.re.kr/icl/context/context-v2.owl#OfficeWork

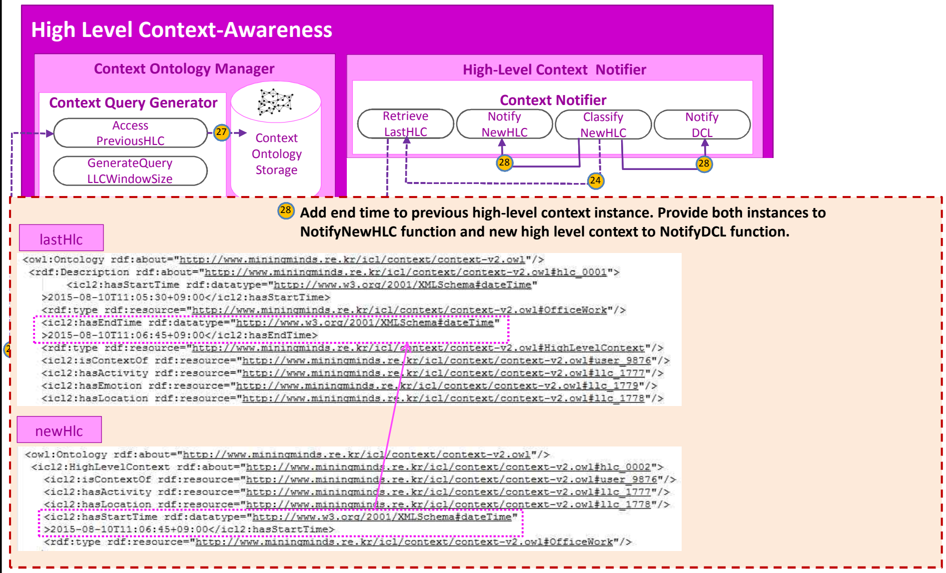

HL Context Reasoning
23 Validation of belonging to subclasses of HLC
ctx rdf:type Context .
ctx rdf:type OfficeWork .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime “2015-08-10T11:05:30”^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .
h-Level Context Notifier
Context Notifier
Notify NewHLC
Classify NewHLC
Notify DCL
h-Level Context Reasoner
Context Verifier
VerifyNew UnclassifiedHLCInstance
RetrieveOntModel
ReceiveNew UnclassifiedHLC
Context Classifier
ValidateHLC
InferHLC
RetrieveOntModel
Ontology Model Manager
StoredOntModel
22 Realization of the HLC instance.
Problems Javadoc Declaration Search Console
<terminated> TestHLCReasoner [Java Application] C:\Program Files (x86)\Java\jdk1.8.0_45\bin\javaw.exe (9 de oct. de 2015 13:29:02)
Classifying 53 elements
Classifying: 100% complete in 00:00% complete in 00:00
Classifying finished in 00:00
Realizing 53 elements
Realizing: 100% complete in 00:00
Realizing finished in 00:00
http://www.miningminds.re.kr/icl/context/context-v2.owl#hlc_0001 classified as http://www.miningminds.re.kr/icl/context/context-v2.owl#OfficeWork

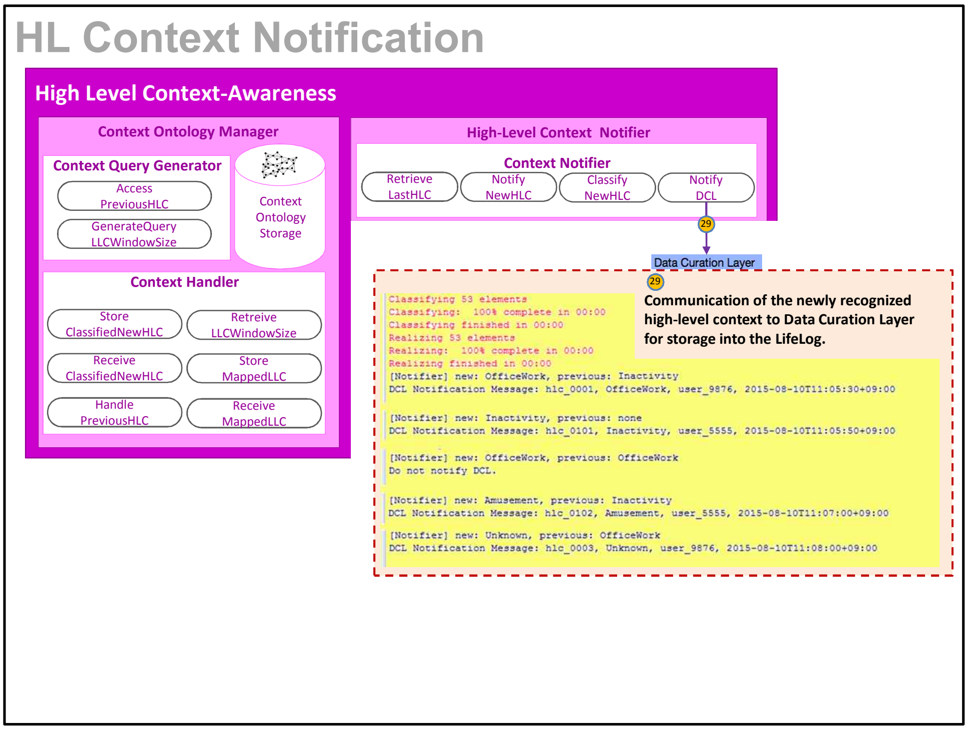

HL Context Notification
High Level Context-Awareness
Context Ontology Manager
Context Query Generator
Access PreviousHLC
GenerateQuery LLCWindowSize
Context Ontology Storage
Context Handler
Store ClassifiedNewHLC
Retreive LLCWindowSize
Receive ClassifiedNewHLC
Store MappedLLC
Handle PreviousHLC
Receive MappedLLC
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
24 25 26 Retrieve the previous classified high level context
SELECT ?hlc
WHERE {
?hlc rdf:type HighLevelContext .
?hlc isContextOf user_9876 .
?hlc icl2:hasStartTime ?starttime .
FILTER NOT EXISTS { ?hlc hasEndTime ?endtime}
FILTER (?starttime < "2015-08-10T11:05:30"^^xsd:dateTime)
}

HL Context Notification

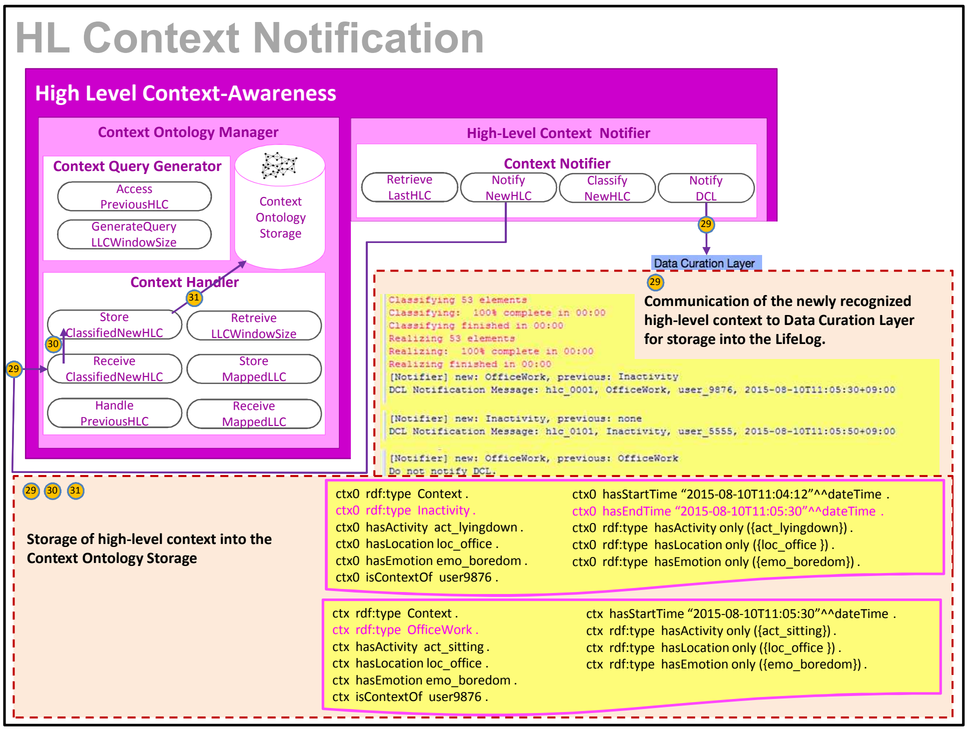


HL Context Notification





HL Context Notification
High Level Context-Awareness
Context Ontology Manager
Context Query Generator
Access PreviousHLC
GenerateQuery LLCWindowSize
Context Ontology Storage
Context Handler
Store ClassifiedNewHLC
Retreive LLCWindowSize
Receive ClassifiedNewHLC
Store MappedLLC
Handle PreviousHLC
Receive MappedLLC
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
29
Data Curation Layer
29
Communication of the newly recognized high-level context to Data Curation Layer for storage into the LifeLog.
Classifying 53 elements
Classifying: 100% complete in 00:00
Classifying finished in 00:00
Realizing 53 elements
Realizing: 100% complete in 00:00
Realizing finished in 00:00
[Notifier] new: OfficeWork, previous: Inactivity
DCL Notification Message: hlc_0001, OfficeWork, user_9876, 2015-08-10T11:05:30+09:00
[Notifier] new: Inactivity, previous: none
DCL Notification Message: hlc_0101, Inactivity, user_5555, 2015-08-10T11:05:50+09:00
[Notifier] new: OfficeWork, previous: OfficeWork
Do not notify DCL.
[Notifier] new: Amusement, previous: Inactivity
DCL Notification Message: hlc_0102, Amusement, user_5555, 2015-08-10T11:07:00+09:00
[Notifier] new: Unknown, previous: OfficeWork
DCL Notification Message: hlc_0003, Unknown, user_9876, 2015-08-10T11:08:00+09:00

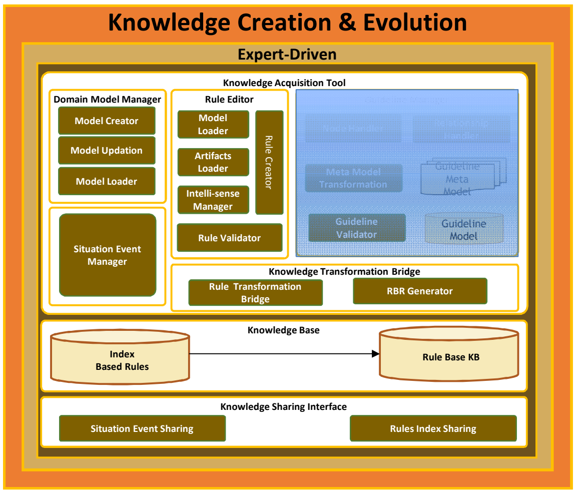


HL Context Notification
High Level Context-Awareness
Context Ontology Manager
Context Query Generator
Access PreviousHLC
GenerateQuery LLCWindowSize
Context Ontology Storage
Context Handler
31
Store ClassifiedNewHLC
30
Retreive LLCWindowSize
29
Receive ClassifiedNewHLC
Store MappedLLC
Handle PreviousHLC
Receive MappedLLC
High-Level Context Notifier
Context Notifier
Retrieve LastHLC
Notify NewHLC
Classify NewHLC
Notify DCL
29
Data Curation Layer
29
Communication of the newly recognized high-level context to Data Curation Layer for storage into the LifeLog.
Classifying 53 elements
Classifying: 100% complete in 00:00
Classifying finished in 00:00
Realizing 53 elements
Realizing: 100% complete in 00:00
Realizing finished in 00:00
[Notifier] new: OfficeWork, previous: Inactivity
DCL Notification Message: hlc_0001, OfficeWork, user_9876, 2015-08-10T11:05:30+09:00
[Notifier] new: Inactivity, previous: none
DCL Notification Message: hlc_0101, Inactivity, user_5555, 2015-08-10T11:05:50+09:00
[Notifier] new: OfficeWork, previous: OfficeWork
Do not notify DCL.
29 30 31
Storage of high-level context into the Context Ontology Storage
ctx0 rdf:type Context .
ctx0 rdf:type Inactivity .
ctx0 hasActivity act_lyingdown .
ctx0 hasLocation loc_office .
ctx0 hasEmotion emo_boredom .
ctx0 isContextOf user9876 .
ctx0 hasStartTime "2015-08-10T11:04:12"^^dateTime .
ctx0 hasEndTime "2015-08-10T11:05:30"^^dateTime .
ctx0 rdf:type hasActivity only ({act_lyingdown}) .
ctx0 rdf:type hasLocation only ({loc_office }) .
ctx0 rdf:type hasEmotion only ({emo_boredom}) .
ctx rdf:type Context .
ctx rdf:type OfficeWork .
ctx hasActivity act_sitting .
ctx hasLocation loc_office .
ctx hasEmotion emo_boredom .
ctx isContextOf user9876 .
ctx hasStartTime "2015-08-10T11:05:30"^^dateTime .
ctx rdf:type hasActivity only ({act_sitting}) .
ctx rdf:type hasLocation only ({loc_office }) .
ctx rdf:type hasEmotion only ({emo_boredom}) .

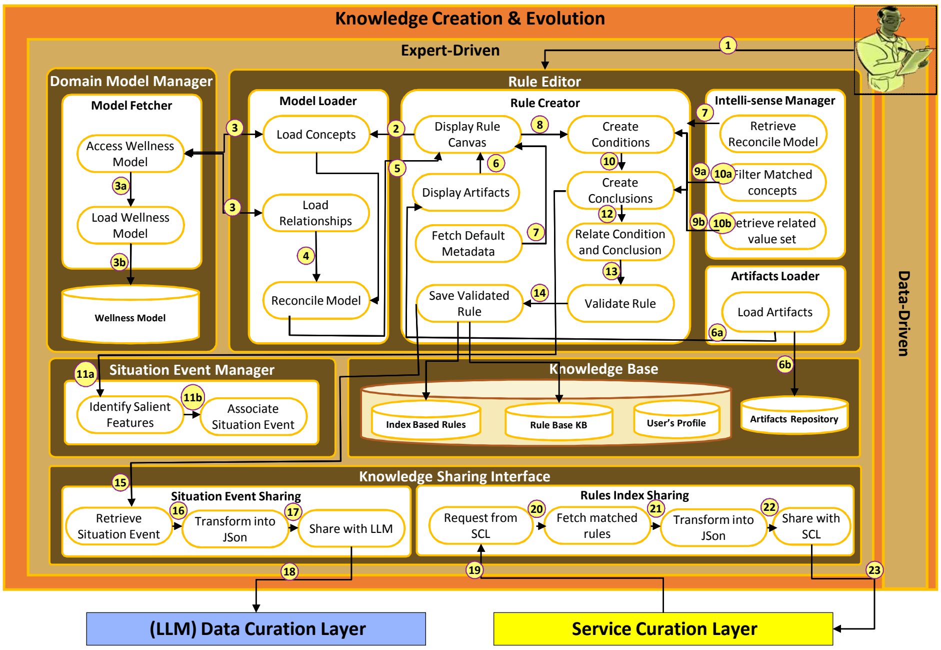
STENCILS for KCL Micro-Animation

STENCILS for Expert-Driven Micro-Animation

KCL Expert Driven Detailed Architecture



KCL Expert Driven Detailed Architecture



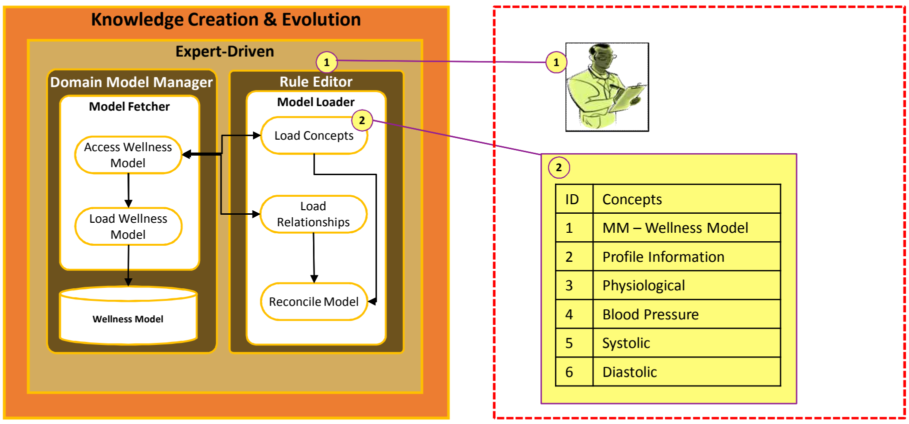
Scenario 1: Rule Creation

Desc. : I-KAT **rule creation** scenario for physical activity through **wellness** model and **Intelli-sense** support.

Major Steps:

1. Wellness Model Loading
2. Rule Creation
3. Intelli-sense based concept filtration

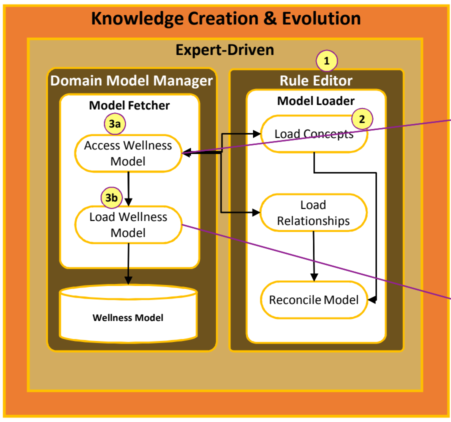
1. Wellness Model Loading



1 Domain expert starts creating rule using Rule Editor

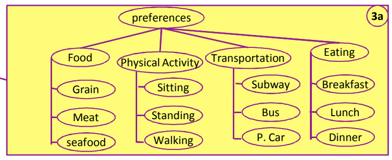
2 Load Concepts fetches all the concepts from wellness model

1. Wellness Model Loading



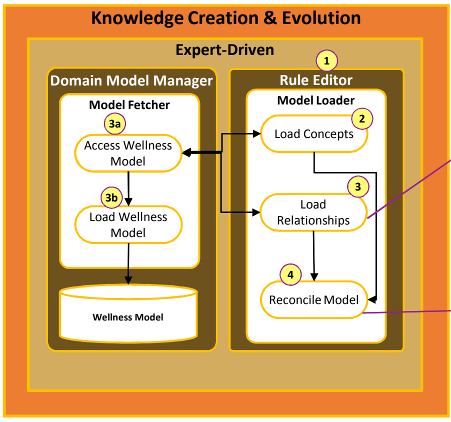
3b

```
<beans:bean id="dataSource1"
class="org.apache.commons.dbcp.BasicDataSource" destroy-
method="close">
    <beans:property name="driverClassName"
value="com.microsoft.sqlserver.jdbc.SQLServerDriver" />
    <beans:property name="url"
value="jdbc:sqlserver://163.180.116.194:1433;instance=SQLEXPRESS;dat
abaseName="DBName" />
    <beans:property name="username" value="sa" />
    <beans:property name="password" value="abcdefg" />
</beans:bean>
```



1	Domain expert starts creating rule using Rule Editor
2	Load Concepts fetches all the concepts from wellness model
3a	Access Wellness Model function get the connection to the wellness model in database
3b	Load Wellness Model loads the complete wellness model to the editor

1. Wellness Model Loading



3

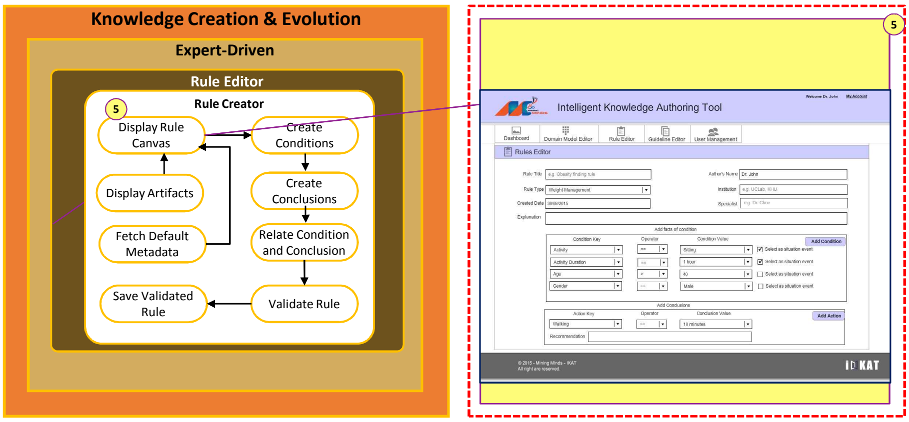
C1ID	RelationID	C2ID
41	116680003	38
42	116680003	38
43	116680003	38
44	116680003	38
45	116680003	38

4

C1	R	C2
Pain	Is-a	Symptom
Allergies	Is-a	Symptom
Sleep Disorder	Is-a	Symptom
Fatigue	Is-a	Symptom
Digestion Disorder	Is-a	Symptom

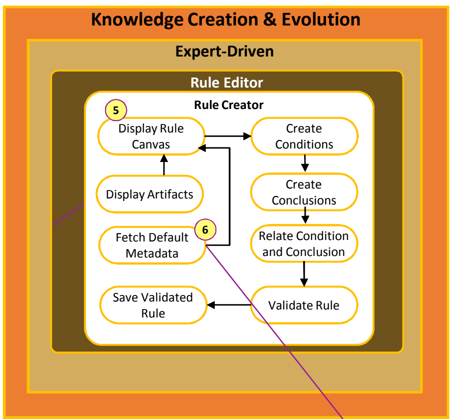
3a	Access Wellness Model function get the connection to the wellness model in database
3b	Load Wellness Model loads the complete wellness model to the editor
3	Load Relationships loads all the relationships among the concepts
4	Reconcile Model merge the concepts and relationships and transforms id's to labels of concepts.

2. Rule Creation



5 The graphical user interface (GUI) of the Rule Editor is rendered by Display Rule Canvas

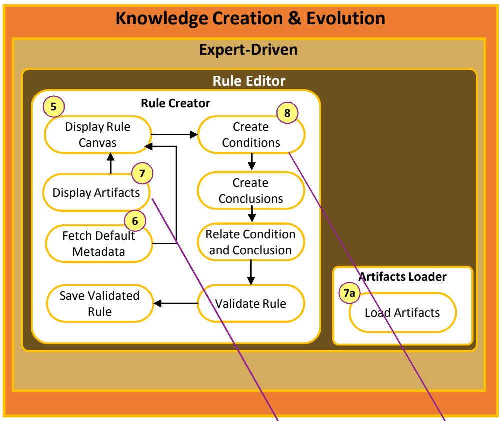
2. Rule Creation



5 The graphical user interface (GUI) of the Rule Editor is rendered by Display Rule Canvas

6 The default information (Metadata)about the rules and some guidelines are fetched to Rule Editor by Fetch Default Metadata.

2. Rule Creation

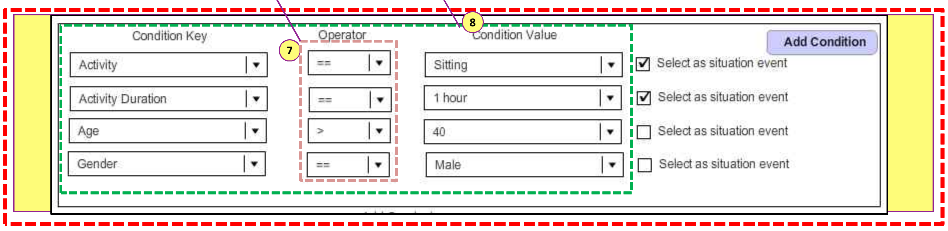


5 The graphical user interface (GUI) of the Rule Editor is rendered by Display Rule Canvas

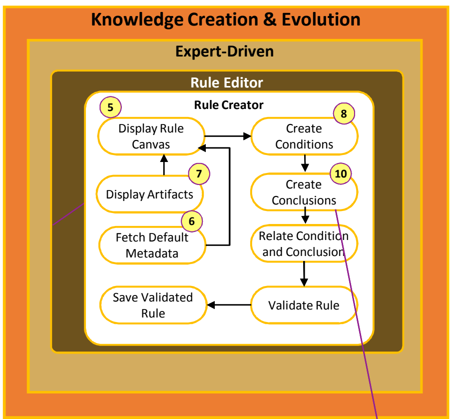
6 The default information (Metadata)about the rules and some guidelines are fetched to Rule Editor by Fetch Default Metadata.

7 Required operators are loaded to Rule Editor by Display Artifacts with the help of Load Artifacts

8 Create Conditions facilitates the expert to create desired facts that includes in rule creation with the help of Condition Keys, Operators, and Condition Values



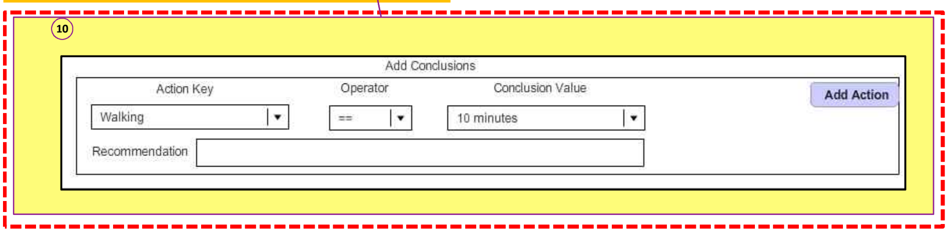
2. Rule Creation



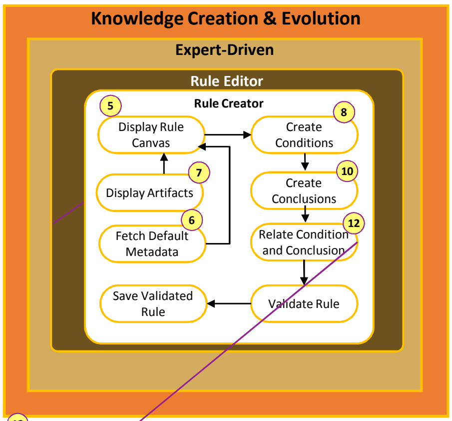
7 Required operators are loaded to Rule Editor by Display Artifacts with the help of Load Artifacts

8 Create Conditions facilitates the expert to create desired facts that includes in rule creation with the help of Condition Keys, Operators, and Condition Values

10 The Create Conclusions facilitates the domain expert to create multiple Conclusions with some Recommendations text for the explanation purpose



2. Rule Creation



8 Create Conditions facilitates the expert to create desired facts that includes in rule creation with the help of Condition Keys, Operators, and Condition Values

10 The Create Conclusions facilitates the domain expert to create multiple Conclusions with some Recommendations text for the explanation purpose

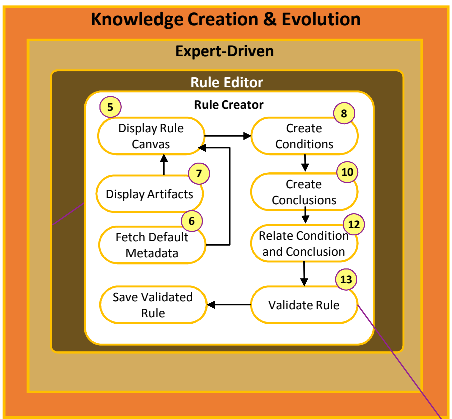
12 The Condition Key is associate with corresponding rule and that rule id is associate with corresponding conclusions.

12

RuleConID	RuleID	ConditionID
20550	20092	1
20551	20093	1
20552	20092	3
20553	20093	3
20554	20092	5

Rule ID	Key	Operator	Value
20092	Current Activity	=	Walking
20092	Activity Duration	=	10 min

2. Rule Creation



10 The Create Conclusions facilitates the domain expert to create multiple Conclusions with some Recommendations text for the explanation purpose

12 The Condition Key is associate with corresponding rule and that rule id is associate with corresponding conclusions.

13 Rule is tested with some auto generated cases to check the duplication and conflict of rule. It enhances the expert satisfaction level.

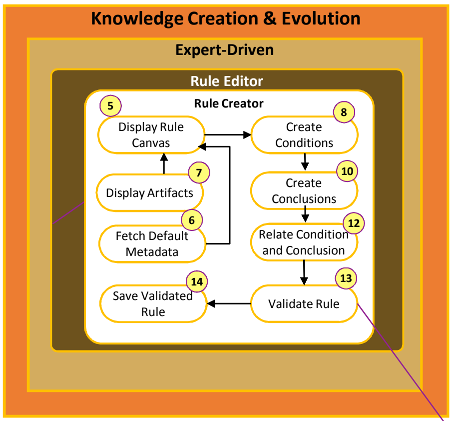
13 **Auto-generated Test Cases**

Gender	Age	Health Status	Disability	Activity	Activity Duration	Select
Male	42	normal	None	Sitting	1 Hour	☑
Male	45	normal	None	Sitting	1 Hour	☐
Male	40	normal	None	Sitting	1 Hour	☐
Male	42	normal	None	Sitting	1 Hour	☐

Add new test case | Execute selected case

The Created Rule is successfully executed for the selected case. (No duplication and No Conflict)

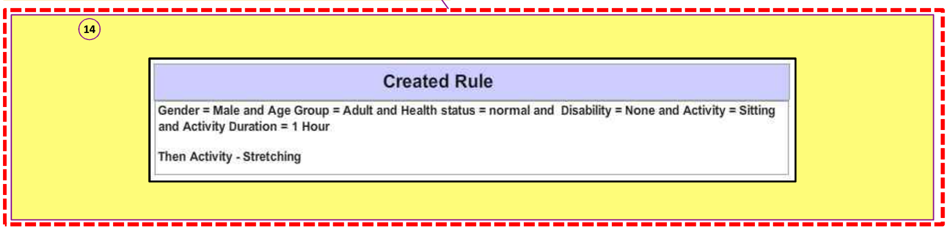
2. Rule Creation



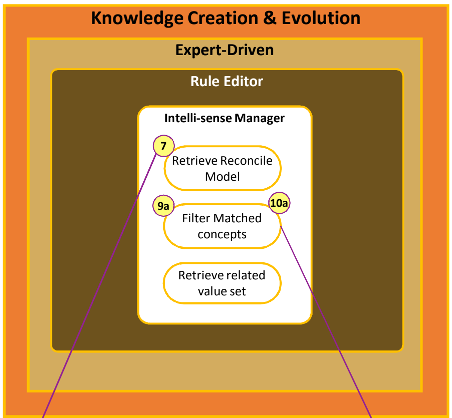
12 The Condition Key is associate with corresponding rule and that rule id is associate with corresponding conclusions.

13 Rule is tested with some auto generated cases to check the duplication and conflict of rule. It enhances the expert satisfaction level.

14 Created rule is saved into the knowledge base by Save Validated Rule



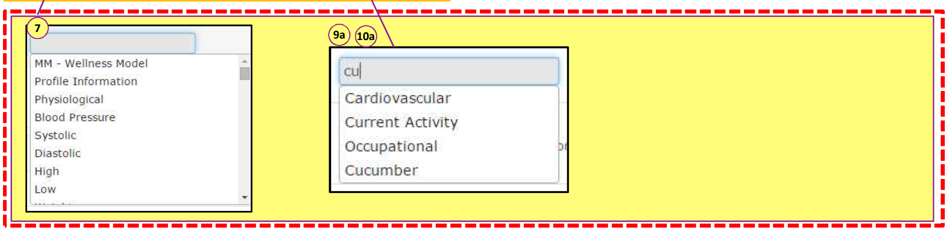
3. Intelli-sense based concepts filtration



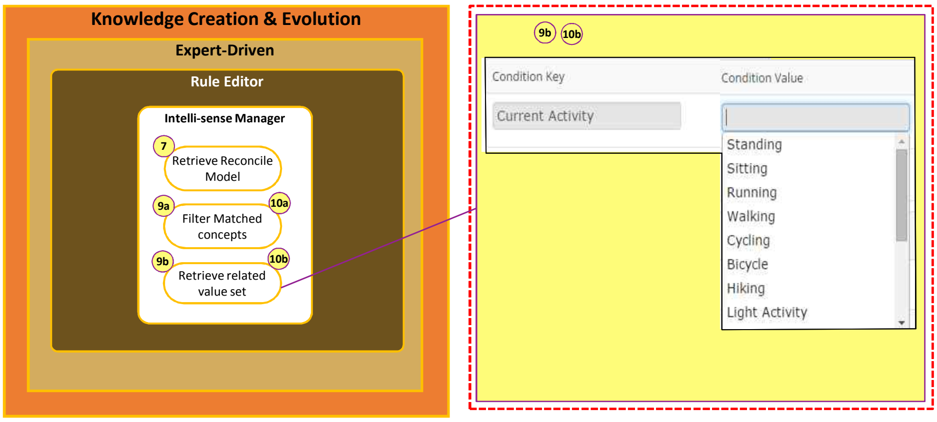
7 Intelli-sense functionality first fetches all the reconcile model in form of list

9a During write up the reconcile model is filtered out according to written characters in condition creation

10a During write up the reconcile model is filtered out according to written characters in conclusion creation



3. Intelli-sense based concepts filtration



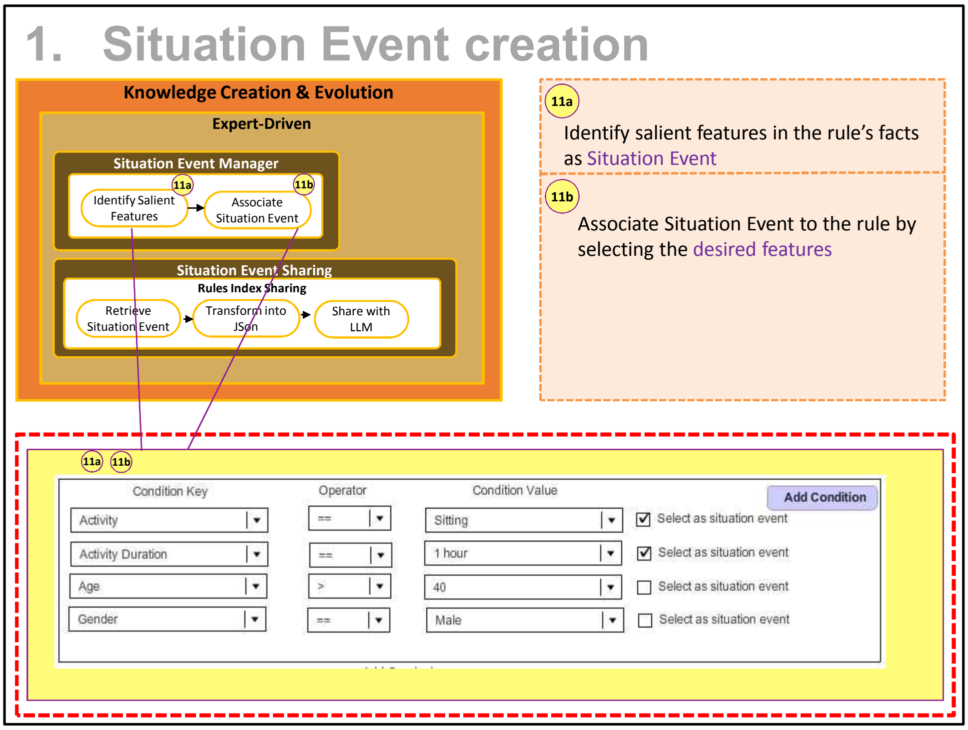
- 9a During write up the reconcile model is filtered out according to written characters in condition
- 10a During write up the reconcile model is filtered out according to written characters in conclusion
- 9b Retrieve all possible values set of selected concept in condition
- 10b Retrieve all possible values set of selected concept in conclusion

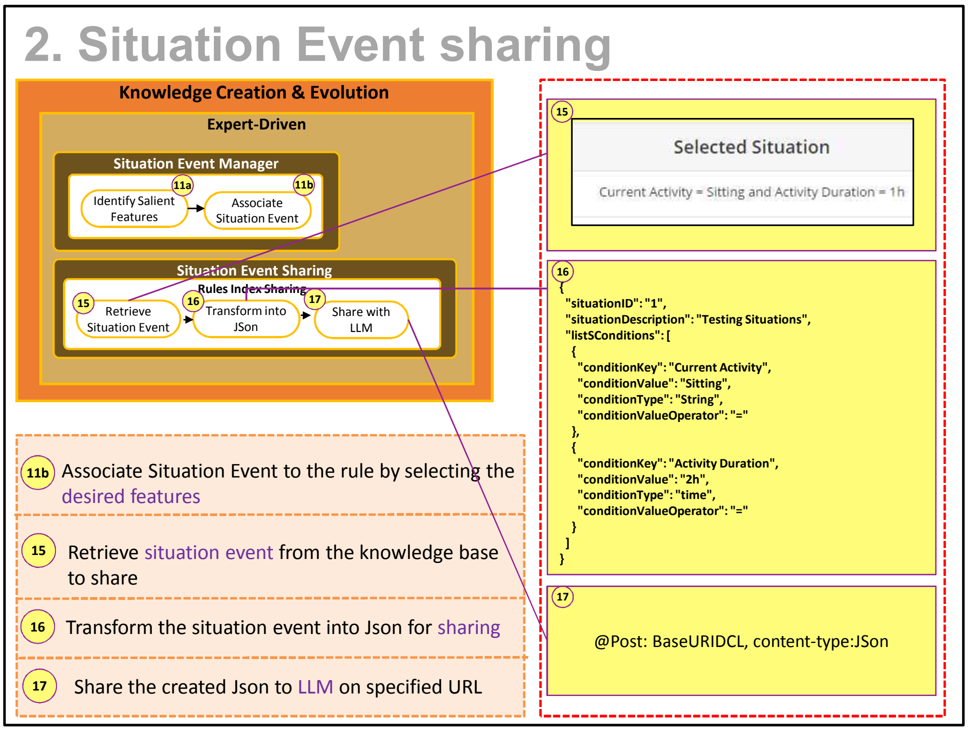
Scenario 2: Situation Event creation and sharing

Desc. : I-KAT **situation event** association with rule and **sharing** with LLM (DCL).

Major Steps:

1. Situation Event Creation
2. Situation Event Sharing


1. Situation Event creation
Knowledge Creation & Evolution
Expert-Driven
Situation Event Manager
11a
11b
Identify Salient Features
Associate Situation Event
Situation Event Sharing
Rules Index Sharing
Retrieve Situation Event
Transform into JSon
Share with LLM
11a
Identify salient features in the rule's facts as Situation Event
11b
Associate Situation Event to the rule by selecting the desired features
11a 11b
Condition Key
Operator
Condition Value
Add Condition
Activity
==
Sitting
Select as situation event
Activity Duration
==
1 hour
Select as situation event
Age
>
40
Select as situation event
Gender
==
Male
Select as situation event


2. Situation Event sharing
Knowledge Creation & Evolution
Expert-Driven
Situation Event Manager
11a
11b
Identify Salient Features
Associate Situation Event
Situation Event Sharing
Rules Index Sharing
15
Retrieve Situation Event
16
Transform into JSon
17
Share with LLM
11b
Associate Situation Event to the rule by selecting the desired features
15
Retrieve situation event from the knowledge base to share
16
Transform the situation event into Json for sharing
17
Share the created Json to LLM on specified URL
15
Selected Situation
Current Activity = Sitting and Activity Duration = 1h
16
{
"situationID": "1",
"situationDescription": "Testing Situations",
"listSConditions": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionType": "String",
"conditionValueOperator": "="
},
{
"conditionKey": "Activity Duration",
"conditionValue": "2h",
"conditionType": "time",
"conditionValueOperator": "="
}
]
}
17
@Post: BaseURIDCL, content-type:JSon

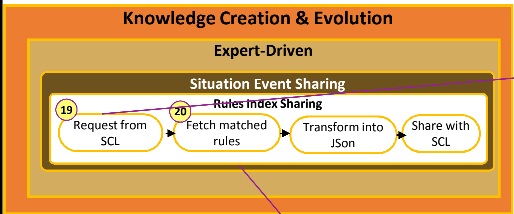
Scenario 3: Rule sharing for recommendation

Desc. : I-KAT **rule sharing** with SCL for **recommendation** generation based on received **situation event**

Major Steps:

1. Situation Event received from SCL
2. Retrieved matched rules (Situation Reasoning)
3. Share matched rules with SCL

1. Situation Event received from SCL
2. Situation Reasoner



19 Request from Service Curation Layer to get all matched rules for situation occurring

20 The situation reasoner fetches all matched rules from knowledge base

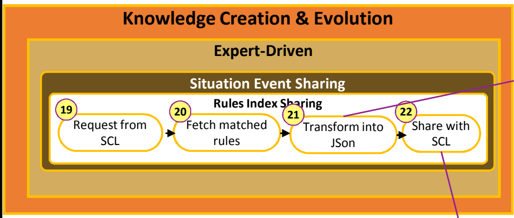
19


```
{
 "situationID": "1",
 "situationDescription": "Testing Situations",
 "listSConditions": [
  {
   "conditionKey": "Current Activity",
   "conditionValue": "Sitting",
   "conditionType": "String",
   "conditionValueOperator": "="
  },
  {
   "conditionKey": "Activity Duration",
   "conditionValue": "2h",
   "conditionType": "time",
   "conditionValueOperator": "="
  }
 ]
}
```


20


```
**************************************************
INFO : - Rule-ID: 20103
INFO : - Rule-ID: 20102
INFO : - Rules loaded successfully:[2016/02/04 17:41:46]: 2
INFO : - Time of Execution: 24 millisecond
INFO : -
**************************************************
```

3. Shares matched rules



19 Request from Service Curation Layer to get all matched rules for situation occurring

20 The situation reasoner fetches all matched rules from knowledge base

21 Transform the matched rules into Json for sharing

22 Share the created Json to SCL on specified URL

21

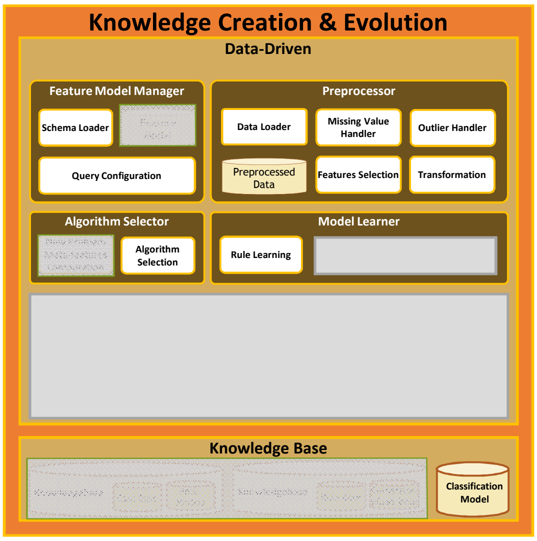

```
[
 {
  "ruleConclusion": "Take a five-minute walk.",
  "conclusionList": [
   {
    "conclusionKey": "Current Activity",
    "conclusionValue": "Walking",
    "conclusionOperator": "=",
    "ruleID": 20103,
    "conclusionID": 10149
   }
  ],
  "conditionList": [
   {
    "conditionKey": "Current Activity",
    "conditionValue": "Sitting",
    "conditionType": "String",
    "conditionValueOperator": "=",
    "isItSituation": false,
    "conditionID": 1
   },
   {
    "conditionKey": "Activity Duration",
    "conditionValue": "2h",
    "conditionType": null,
    "conditionValueOperator": "=",
    "isItSituation": false,
    "conditionID": 20163
   }
  ],
  "ruleID": 20103
 }
]
```


22

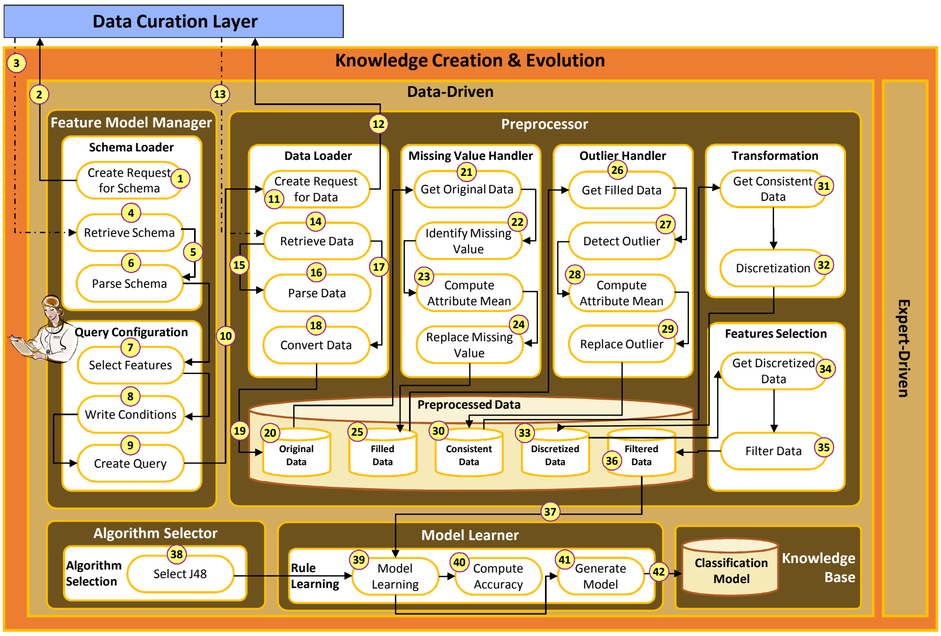
@Post: BaseURISCL, content-type:JSon

STENCILS for Data-Driven Micro-Animation

KCL Data Driven Detailed Architecture



KCL Data Driven Detailed Architecture



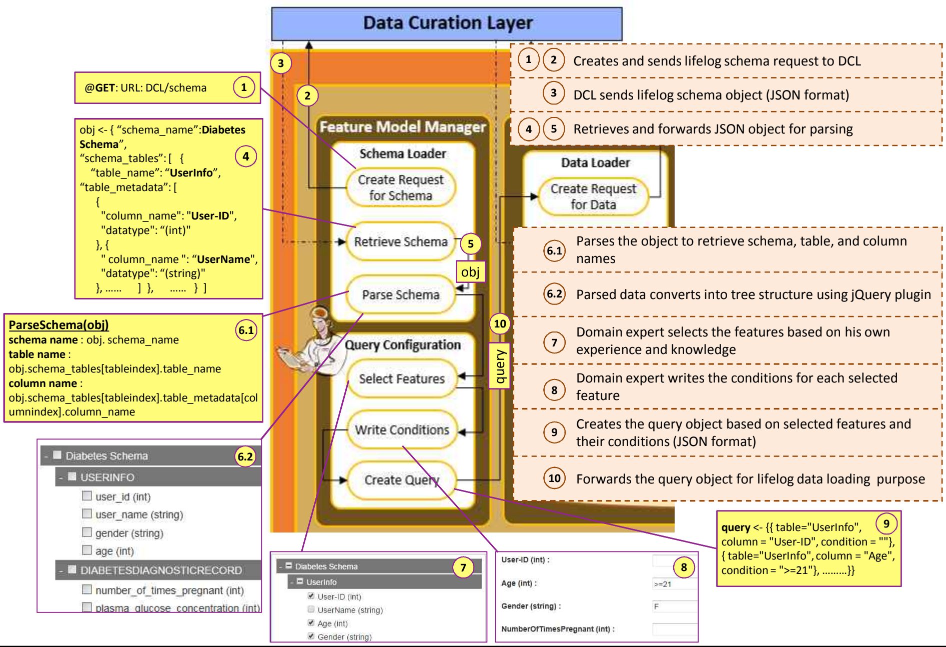
Scenario: Classification Model Creation

Desc. : Lifelog **schema** and **Data** is **acquired** from DCL. **Preprocessed** the data to generate the **classification model** using **J48** algorithm.

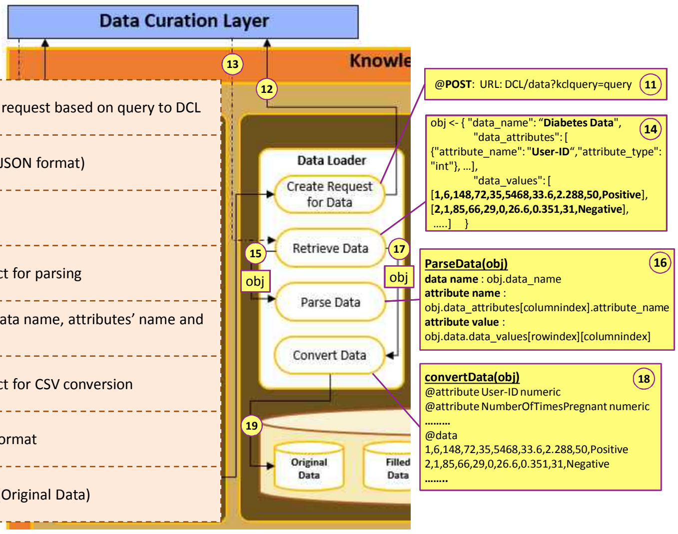
Major Steps:

1. Data Selection and Retrieval Process
2. Data Preprocessing
3. Model Learning Process

1 - Data Selection and Retrieval Process (1/2)

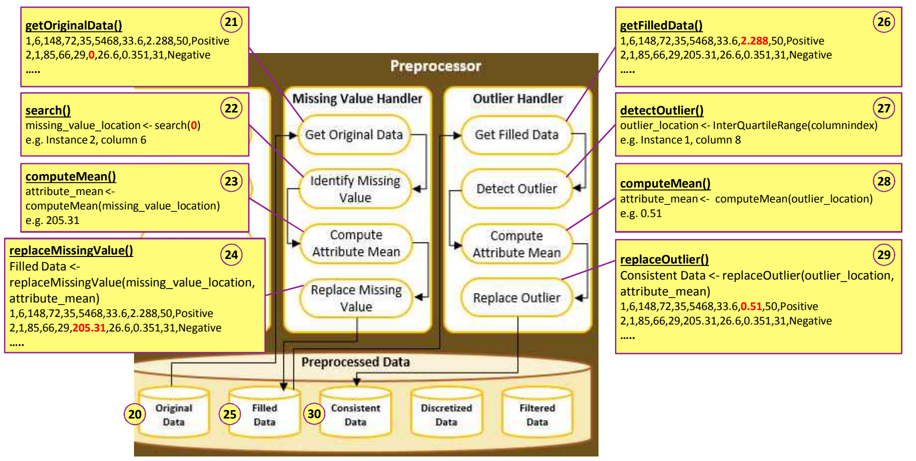


1 - Data Selection and Retrieval Process (2/2)



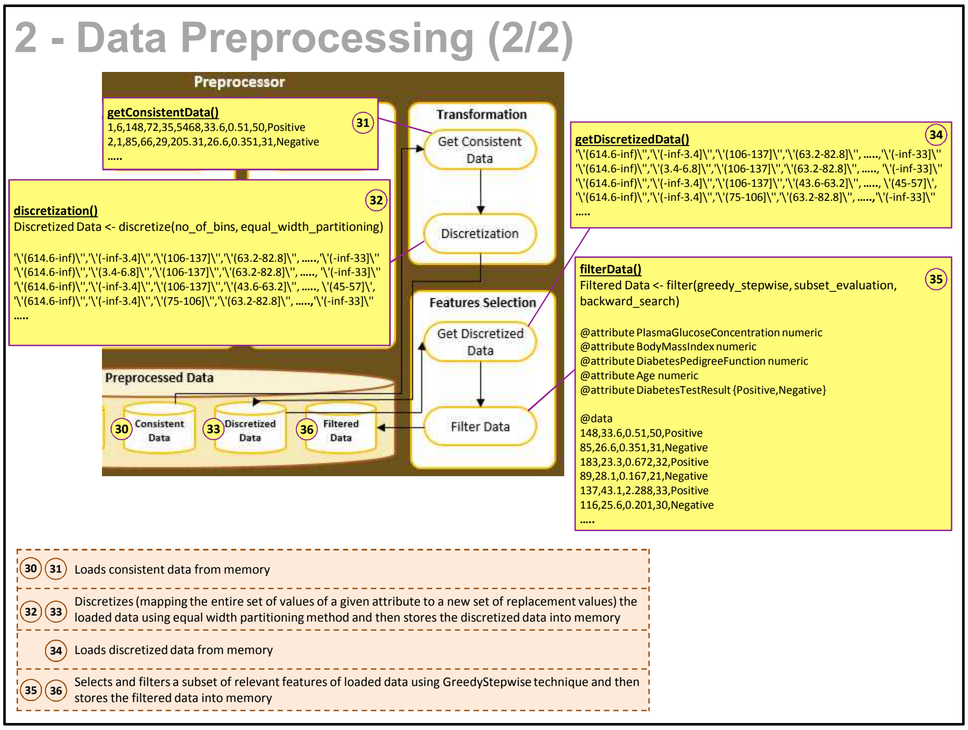
11 12	Creates and sends lifelog data request based on query to DCL
13	DCL sends lifelog data object (JSON format)
14	Retrieves JSON object
15	Forwards retrieved JSON object for parsing
16	Parses the object to retrieve data name, attributes' name and value
17	Forwards retrieved JSON object for CSV conversion
18	Converts the object into CSV format
19	Persists the data into CSV file (Original Data)

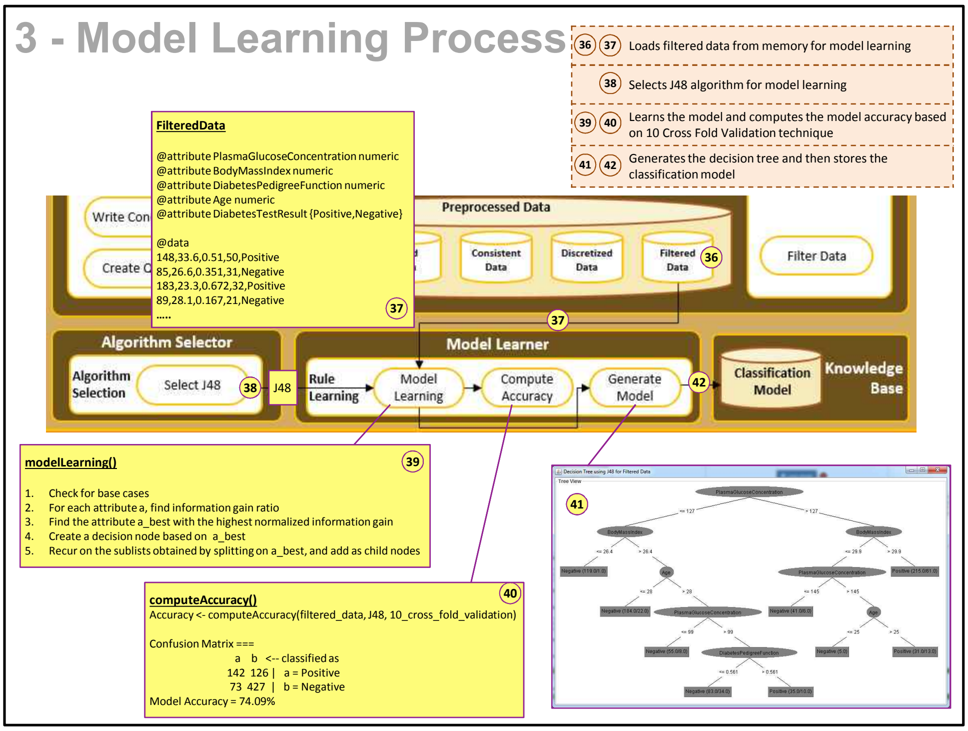
2 - Data Preprocessing (1/2)



20 21	Loads original data from CSV file
22	Identifies and records the locations of missing values (i.e. 0) from original data
23	Computes the mean of all attributes having missing values
24 25	Replaces each missing value with its corresponding attribute mean and then stores the filled data into memory

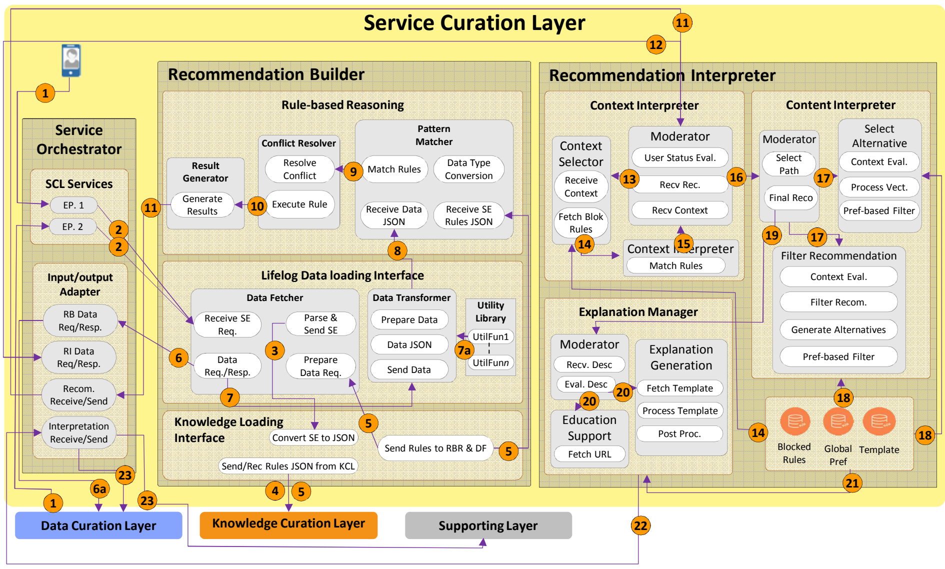
26	Loads filled data from memory
27	Detects the outliers using InterQuartileRange technique and records the locations of outliers (an observation which deviates so much from the other observations) from filled data
28	Computes the mean of all attributes having outliers
29 30	Replaces each outlier with its corresponding attribute mean and then stores the consistent data into memory


2 - Data Preprocessing (2/2)
Preprocessor
getConsistentData()
1,6,148,72,35,5468,33.6,0.51,50,Positive
2,1,85,66,29,205.31,26.6,0.351,31,Negative
.....
31
Transformation
Get Consistent Data
Discretization
getDiscretizedData()
34
'\'(614.6-inf)\'','\'(-inf-3.4]\'','\'(106-137]\'','\'(63.2-82.8]\'', ..., '\'(-inf-33]\''
'\'(614.6-inf)\'','\'(3.4-6.8]\'','\'(106-137]\'','\'(63.2-82.8]\'', ..., '\'(-inf-33]\''
'\'(614.6-inf)\'','\'(-inf-3.4]\'','\'(106-137]\'','\'(43.6-63.2]\'', ..., \'(45-57]\',
'\'(614.6-inf)\'','\'(-inf-3.4]\'','\'(75-106]\'','\'(63.2-82.8]\'', ...,'\'(-inf-33]\''
.....
discretization()
32
Discretized Data <- discretize(no_of_bins, equal_width_partitioning)
'\'(614.6-inf)\'','\'(-inf-3.4]\'','\'(106-137]\'','\'(63.2-82.8]\'', ..., '\'(-inf-33]\''
'\'(614.6-inf)\'','\'(3.4-6.8]\'','\'(106-137]\'','\'(63.2-82.8]\'', ..., '\'(-inf-33]\''
'\'(614.6-inf)\'','\'(-inf-3.4]\'','\'(106-137]\'','\'(43.6-63.2]\'', ..., \'(45-57]\',
'\'(614.6-inf)\'','\'(-inf-3.4]\'','\'(75-106]\'','\'(63.2-82.8]\'', ...,'\'(-inf-33]\''
.....
Features Selection
Get Discretized Data
Filter Data
filterData()
35
Filtered Data <- filter(greedy_stepwise, subset_evaluation, backward_search)
@attribute PlasmaGlucoseConcentration numeric
@attribute BodyMassIndex numeric
@attribute DiabetesPedigreeFunction numeric
@attribute Age numeric
@attribute DiabetesTestResult {Positive,Negative}
@data
148,33.6,0.51,50,Positive
85,26.6,0.351,31,Negative
183,23.3,0.672,32,Positive
89,28.1,0.167,21,Negative
137,43.1,2.288,33,Positive
116,25.6,0.201,30,Negative
.....
Preprocessed Data
30 Consistent Data
33 Discretized Data
36 Filtered Data
30 31 Loads consistent data from memory
32 33 Discretizes (mapping the entire set of values of a given attribute to a new set of replacement values) the loaded data using equal width partitioning method and then stores the discretized data into memory
34 Loads discretized data from memory
35 36 Selects and filters a subset of relevant features of loaded data using GreedyStepwise technique and then stores the filtered data into memory


3 - Model Learning Process
36 37 Loads filtered data from memory for model learning
38 Selects J48 algorithm for model learning
39 40 Learns the model and computes the model accuracy based on 10 Cross Fold Validation technique
41 42 Generates the decision tree and then stores the classification model
FilteredData
@attribute PlasmaGlucoseConcentration numeric
@attribute BodyMassIndex numeric
@attribute DiabetesPedigreeFunction numeric
@attribute Age numeric
@attribute DiabetesTestResult {Positive,Negative}
@data
148,33.6,0.51,50,Positive
85,26.6,0.351,31,Negative
183,23.3,0.672,32,Positive
89,28.1,0.167,21,Negative
.....
37
Preprocessed Data
Consistent Data
Discretized Data
Filtered Data
36
Filter Data
Algorithm Selector
Algorithm Selection
Select J48
38
J48
Model Learner
Rule Learning
Model Learning
Compute Accuracy
Generate Model
42
Classification Model
Knowledge Base
modelLearning()
39
1. Check for base cases
2. For each attribute a, find information gain ratio
3. Find the attribute a_best with the highest normalized information gain
4. Create a decision node based on a_best
5. Recur on the sublists obtained by splitting on a_best, and add as child nodes
computeAccuracy()
40
Accuracy <- computeAccuracy(filtered_data, J48, 10_cross_fold_validation)
Confusion Matrix ===
a b <-- classified as
142 126 | a = Positive
73 427 | b = Negative
Model Accuracy = 74.09%
41

STENCILS for SCL MMV2.0 Micro-Animation

SCL Detailed Architecture



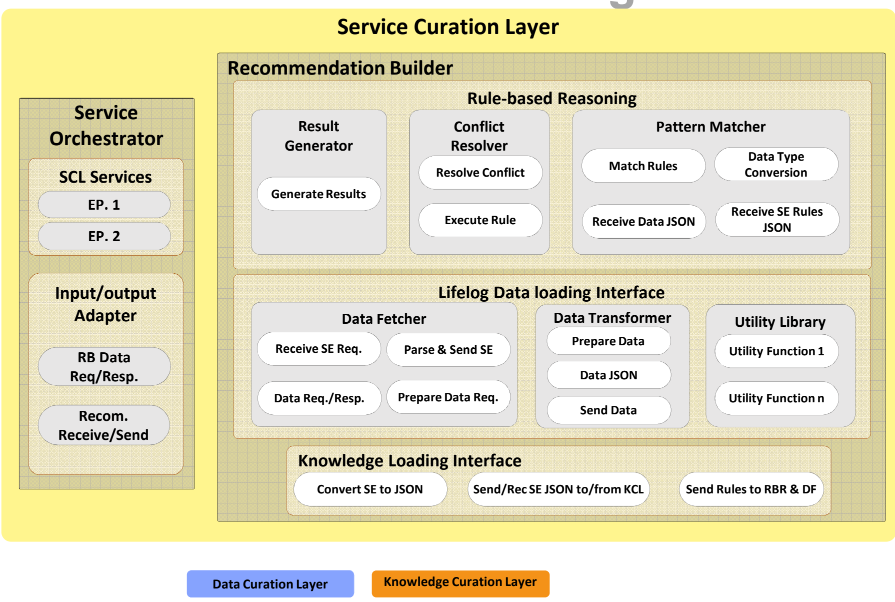
Service Curation Layer - Scenarios

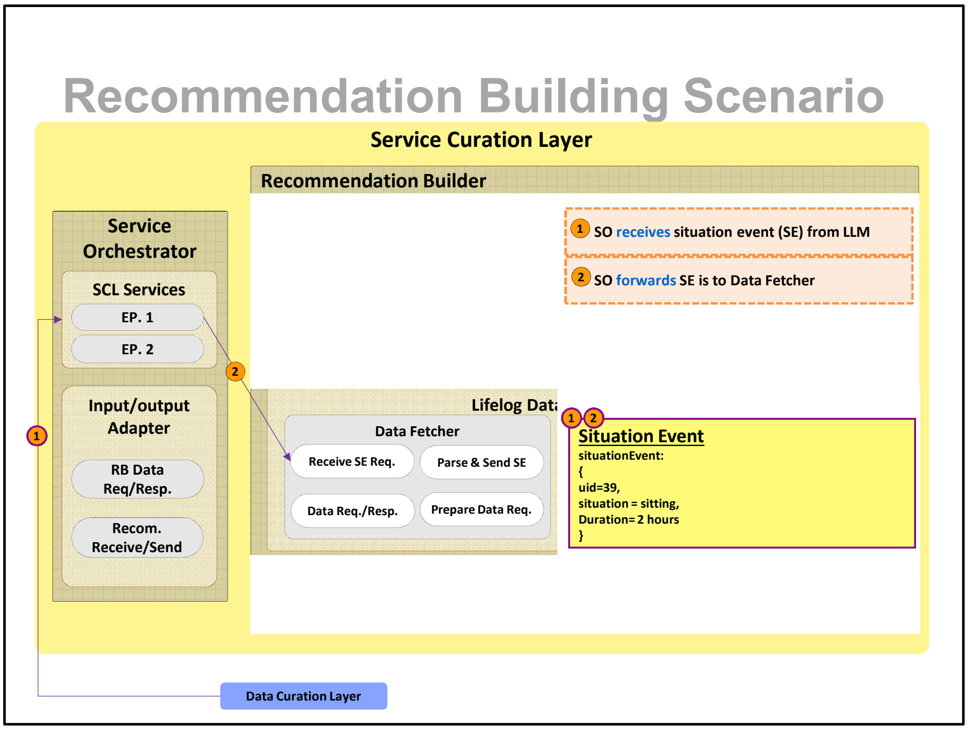
Scenario 1: Recommendation Building

Desc : Generation of Recommendations for the Situations, triggered by LLM.

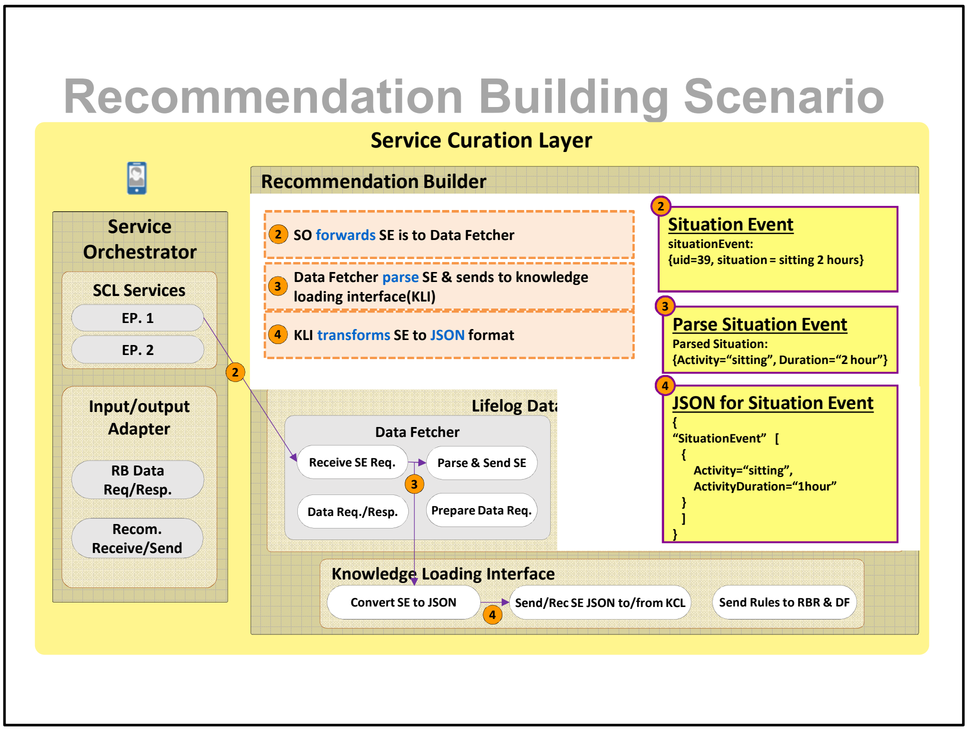
1. Situation Event (SE) received from LLM to Service Orchestrator (SO).
2. SO forwards the situation to Recommendation Builder(RB).
3. RB builds the recommendations Based on rules retrieved from KCL and life-log/user profile data from DCL.
4. RB sends the recommendation back to SO.
5. SO forwards the recommendation to Recommendation Interpreter.

Recommendation Building Scenario

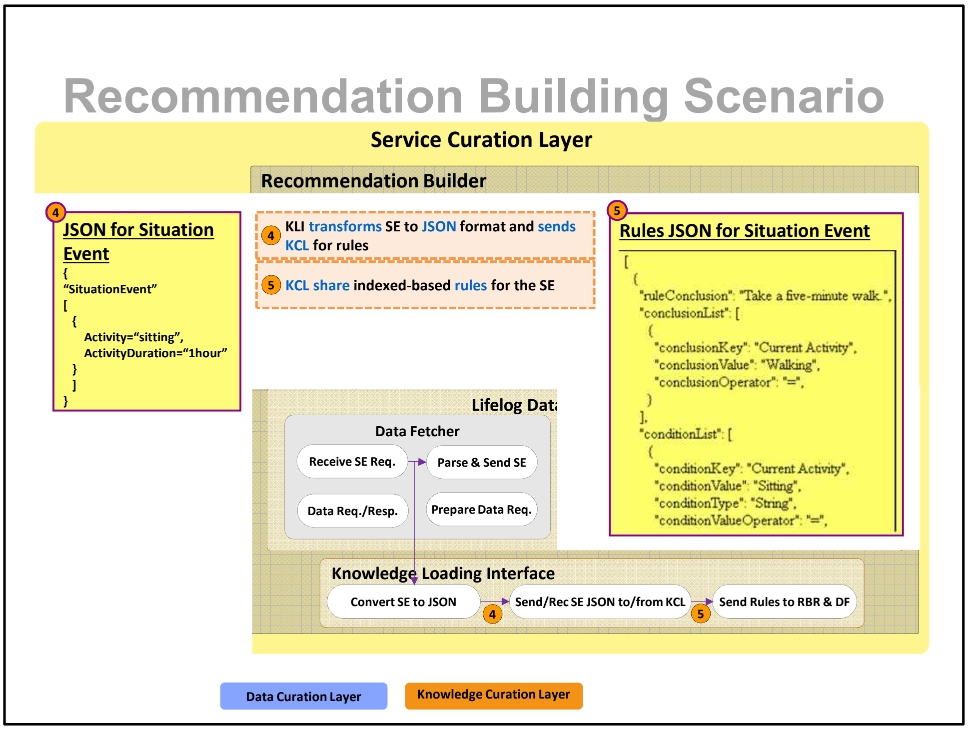


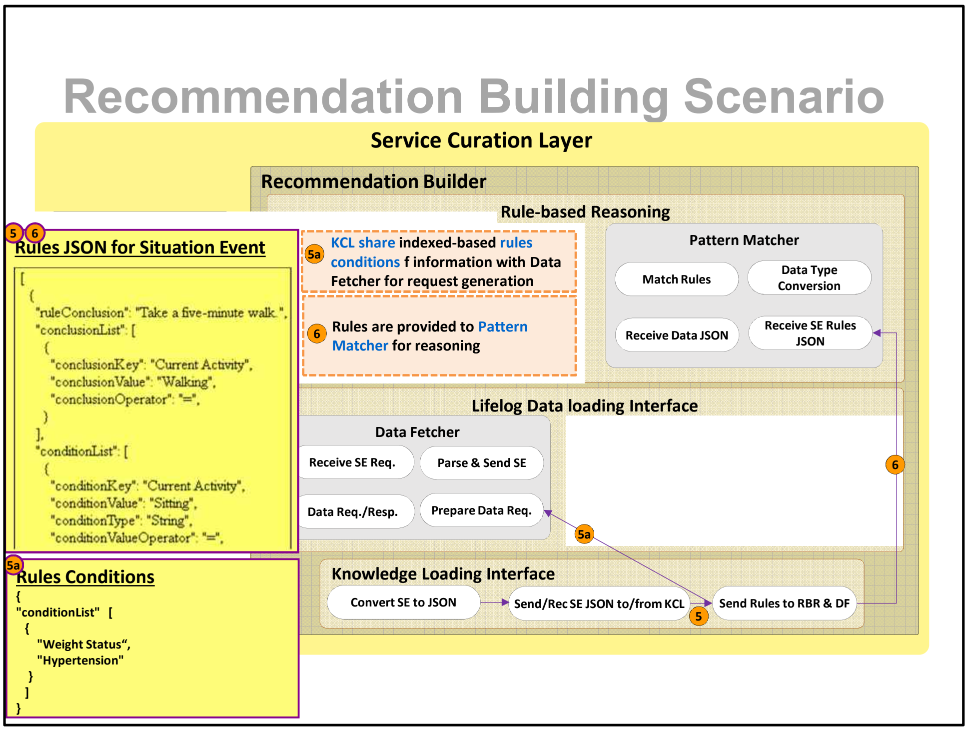


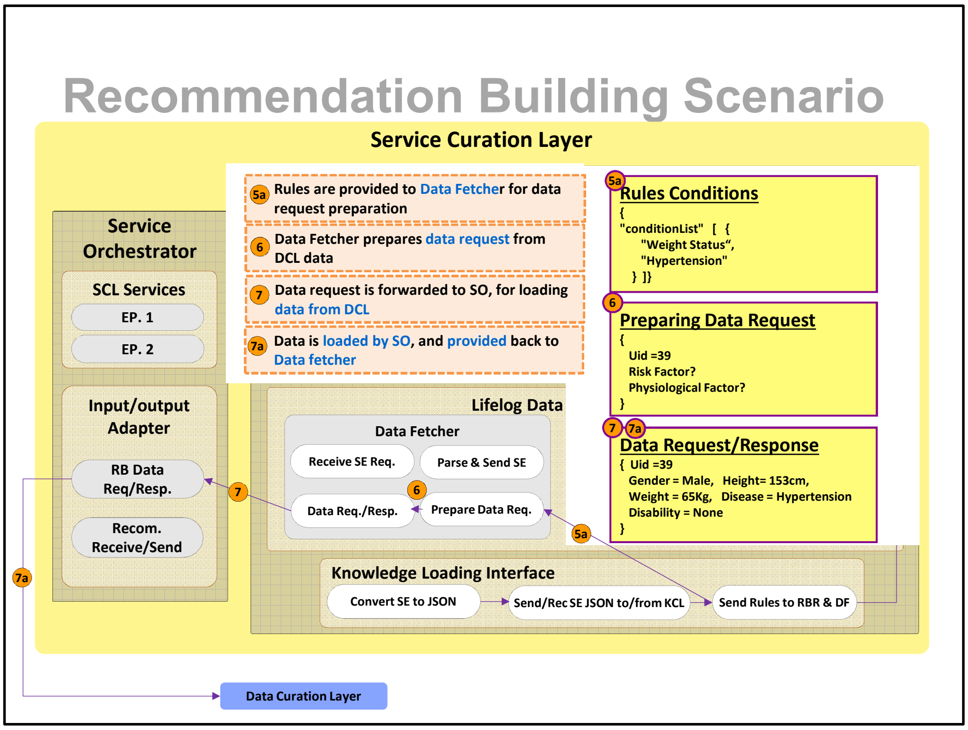
Recommendation Building Scenario
Service Curation Layer
Recommendation Builder
1 SO receives situation event (SE) from LLM
2 SO forwards SE is to Data Fetcher
Service Orchestrator
SCL Services
EP. 1
EP. 2
Input/output Adapter
RB Data Req/Resp.
Recom. Receive/Send
Lifelog Data
Data Fetcher
Receive SE Req.
Parse & Send SE
Data Req./Resp.
Prepare Data Req.
Situation Event
situationEvent:
{
uid=39,
situation = sitting,
Duration= 2 hours
}
Data Curation Layer

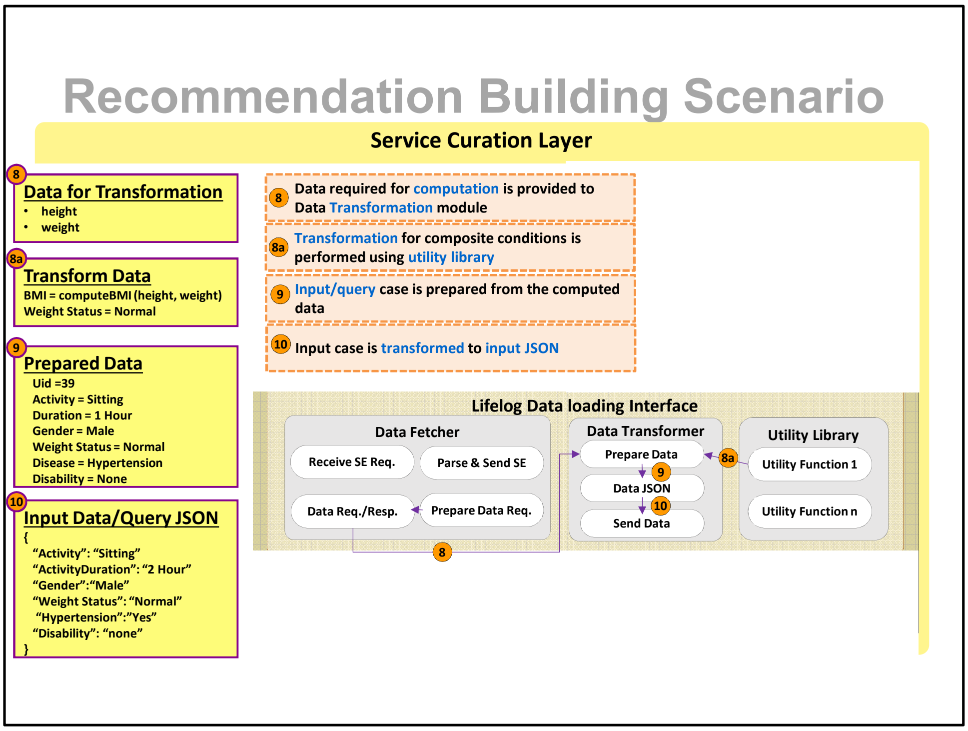


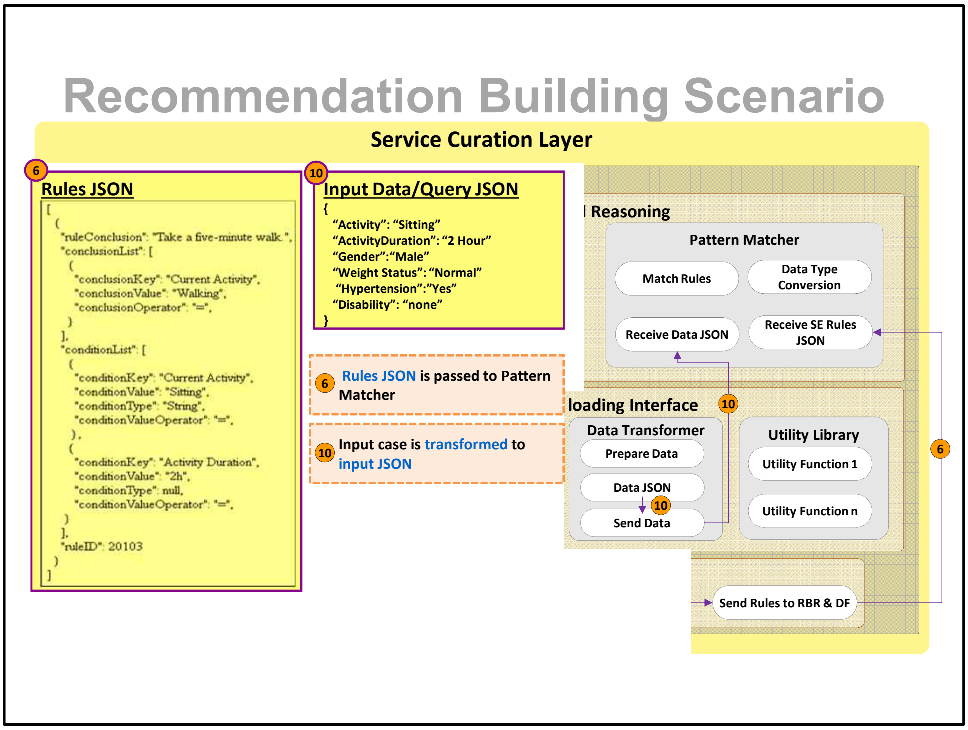
Recommendation Building Scenario
Service Curation Layer
Recommendation Builder
2 SO forwards SE is to Data Fetcher
3 Data Fetcher parse SE & sends to knowledge loading interface(KLI)
4 KLI transforms SE to JSON format
Situation Event
situationEvent:
{uid=39, situation = sitting 2 hours}
Parse Situation Event
Parsed Situation:
{Activity="sitting", Duration="2 hour"}
JSON for Situation Event
{
"SituationEvent" [
{
Activity="sitting",
ActivityDuration="1hour"
}
]
}
Service Orchestrator
SCL Services
EP. 1
EP. 2
Input/output Adapter
RB Data Req/Resp.
Recom. Receive/Send
Lifelog Data
Data Fetcher
Receive SE Req.
Parse & Send SE
Data Req./Resp.
Prepare Data Req.
Knowledge Loading Interface
Convert SE to JSON
Send/Rec SE JSON to/from KCL
Send Rules to RBR & DF


Recommendation Building Scenario
Service Curation Layer
Recommendation Builder
4 JSON for Situation Event
{
"SituationEvent"
[
{
Activity="sitting",
ActivityDuration="1hour"
}
]
}
4 KLI transforms SE to JSON format and sends KCL for rules
5 KCL share indexed-based rules for the SE
5 Rules JSON for Situation Event
[
{
"ruleConclusion": "Take a five-minute walk.",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking",
"conclusionOperator": "=",
}
],
"conditionList": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionType": "String",
"conditionValueOperator": "=",
Lifelog Data
Data Fetcher
Receive SE Req.
Parse & Send SE
Data Req./Resp.
Prepare Data Req.
Knowledge Loading Interface
Convert SE to JSON
4
Send/Rec SE JSON to/from KCL
5
Send Rules to RBR & DF
Data Curation Layer
Knowledge Curation Layer

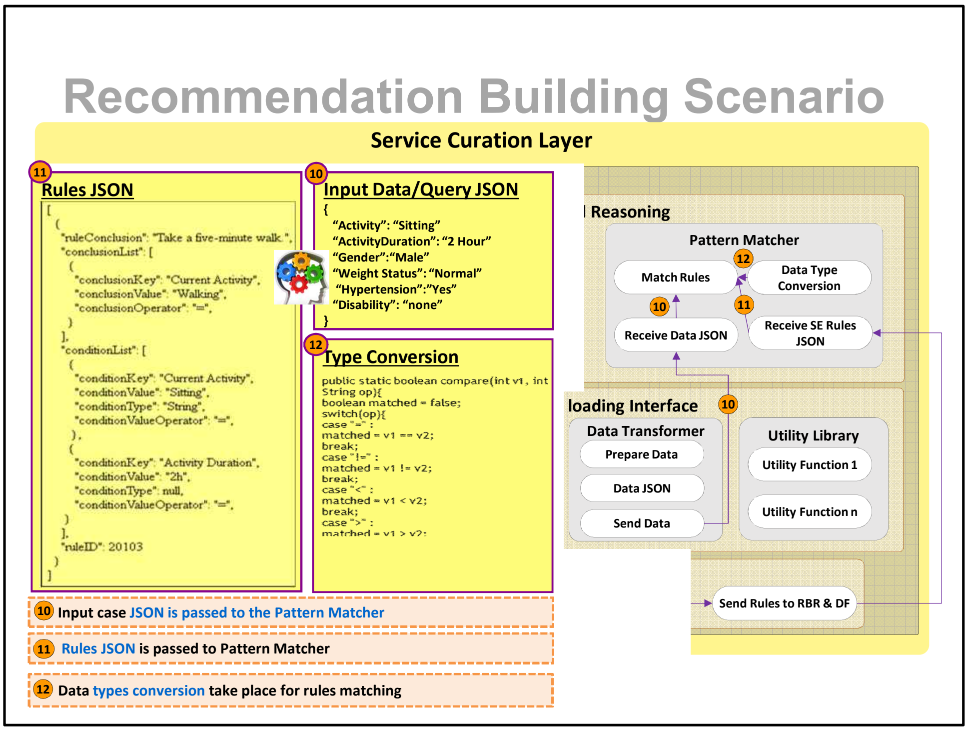

Recommendation Building Scenario
Service Curation Layer
Recommendation Builder
Rule-based Reasoning
5 6 Rules JSON for Situation Event
[
{
"ruleConclusion": "Take a five-minute walk.",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking",
"conclusionOperator": "=",
}
],
"conditionList": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionType": "String",
"conditionValueOperator": "=",
5a Rules Conditions
{
"conditionList" [
{
"Weight Status",
"Hypertension"
}
]
}
5a KCL share indexed-based rules conditions f information with Data Fetcher for request generation
6 Rules are provided to Pattern Matcher for reasoning
Pattern Matcher
Match Rules
Data Type Conversion
Receive Data JSON
Receive SE Rules JSON
Lifelog Data loading Interface
Data Fetcher
Receive SE Req.
Parse & Send SE
Data Req./Resp.
Prepare Data Req.
5a
6
Knowledge Loading Interface
Convert SE to JSON
Send/Rec SE JSON to/from KCL
5
Send Rules to RBR & DF


Recommendation Building Scenario
Service Curation Layer
Service Orchestrator
SCL Services
EP. 1
EP. 2
Input/output Adapter
RB Data Req/Resp.
Recom. Receive/Send
5a Rules are provided to Data Fetcher for data request preparation
6 Data Fetcher prepares data request from DCL data
7 Data request is forwarded to SO, for loading data from DCL
7a Data is loaded by SO, and provided back to Data fetcher
Lifelog Data
Data Fetcher
Receive SE Req.
Parse & Send SE
Data Req./Resp.
Prepare Data Req.
Knowledge Loading Interface
Convert SE to JSON
Send/Rec SE JSON to/from KCL
Send Rules to RBR & DF
Rules Conditions
{
"conditionList" [{
"Weight Status",
"Hypertension"
}]}
Preparing Data Request
{
Uid =39
Risk Factor?
Physiological Factor?
}
Data Request/Response
{ Uid =39
Gender = Male, Height= 153cm,
Weight = 65Kg, Disease = Hypertension
Disability = None
}
Data Curation Layer

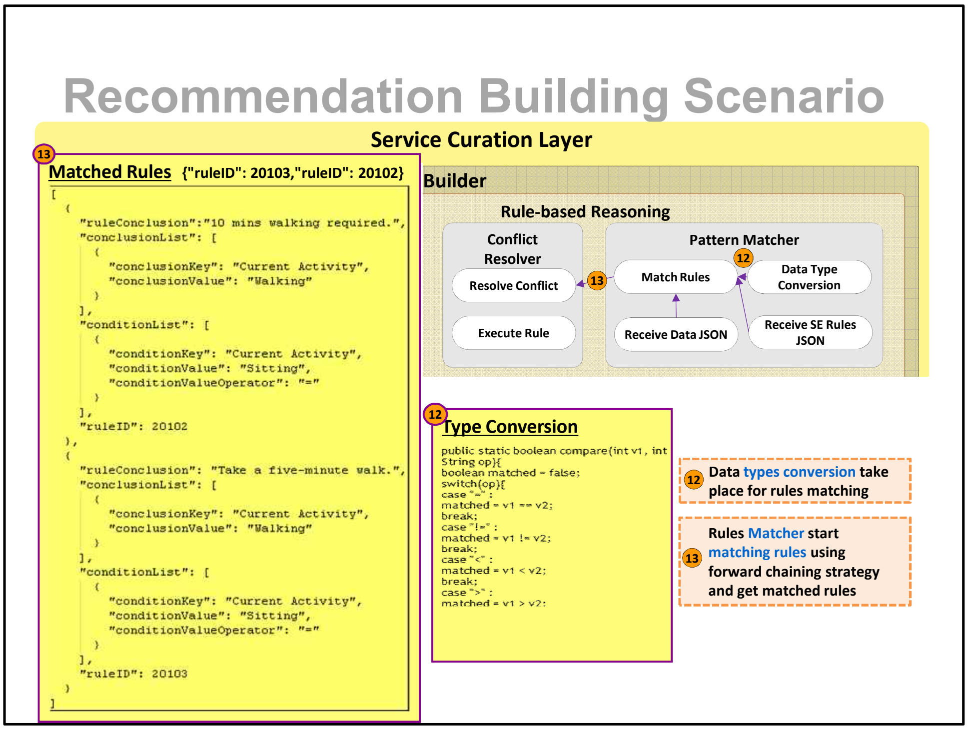

Recommendation Building Scenario
Service Curation Layer
Data for Transformation
height
weight
Transform Data
BMI = computeBMI (height, weight)
Weight Status = Normal
Prepared Data
Uid =39
Activity = Sitting
Duration = 1 Hour
Gender = Male
Weight Status = Normal
Disease = Hypertension
Disability = None
Input Data/Query JSON
{
"Activity": "Sitting"
"ActivityDuration": "2 Hour"
"Gender":"Male"
"Weight Status": "Normal"
"Hypertension":"Yes"
"Disability": "none"
}
8 Data required for computation is provided to Data Transformation module
8a Transformation for composite conditions is performed using utility library
9 Input/query case is prepared from the computed data
10 Input case is transformed to input JSON
Lifelog Data loading Interface
Data Fetcher
Receive SE Req.
Parse & Send SE
Data Req./Resp.
Prepare Data Req.
Data Transformer
Prepare Data
Data JSON
Send Data
Utility Library
Utility Function 1
Utility Function n

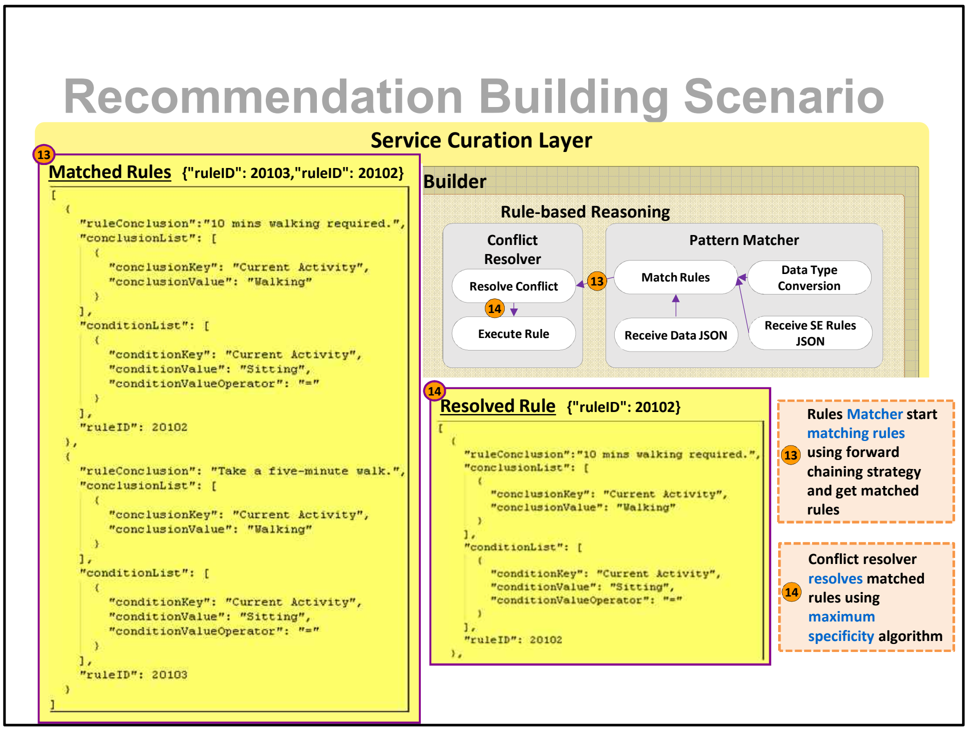


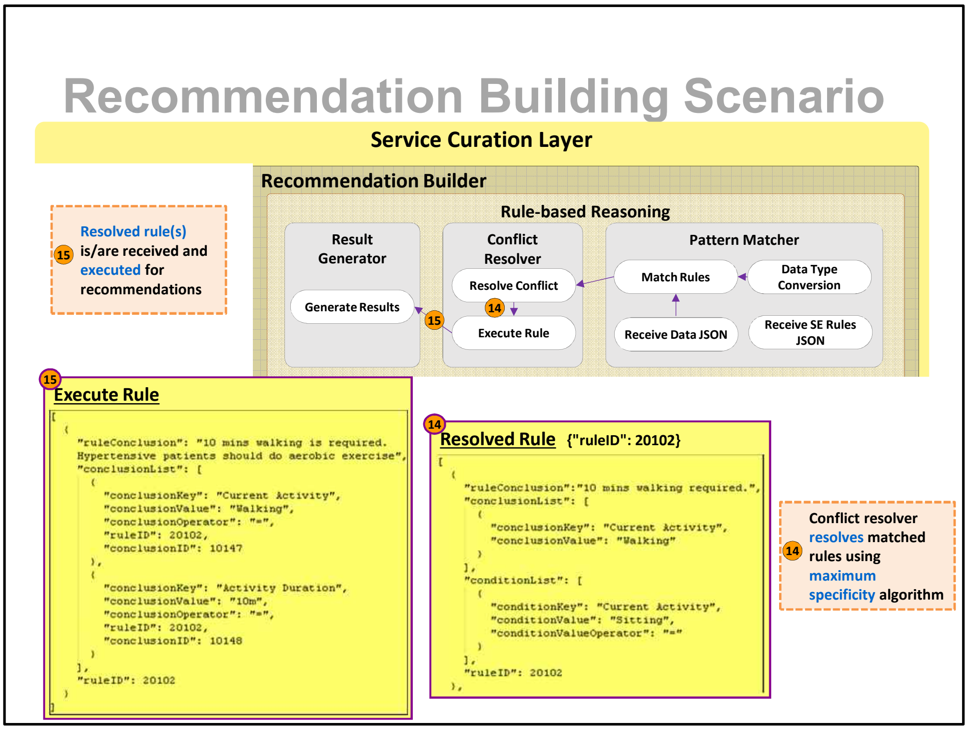
Recommendation Building Scenario
Service Curation Layer
6
Rules JSON
[
{
"ruleConclusion": "Take a five-minute walk.",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking",
"conclusionOperator": "=",
}
],
"conditionList": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionType": "String",
"conditionValueOperator": "=",
},
{
"conditionKey": "Activity Duration",
"conditionValue": "2h",
"conditionType": null,
"conditionValueOperator": "=",
}
],
"ruleID": 20103
}
]
10
Input Data/Query JSON
{
"Activity": "Sitting"
"ActivityDuration": "2 Hour"
"Gender":"Male"
"Weight Status": "Normal"
"Hypertension":"Yes"
"Disability": "none"
}
6 Rules JSON is passed to Pattern Matcher
10 Input case is transformed to input JSON
Reasoning
Pattern Matcher
Match Rules
Data Type Conversion
Receive Data JSON
Receive SE Rules JSON
loading Interface
Data Transformer
Prepare Data
Data JSON
10
Send Data
Utility Library
Utility Function 1
Utility Function n
Send Rules to RBR & DF



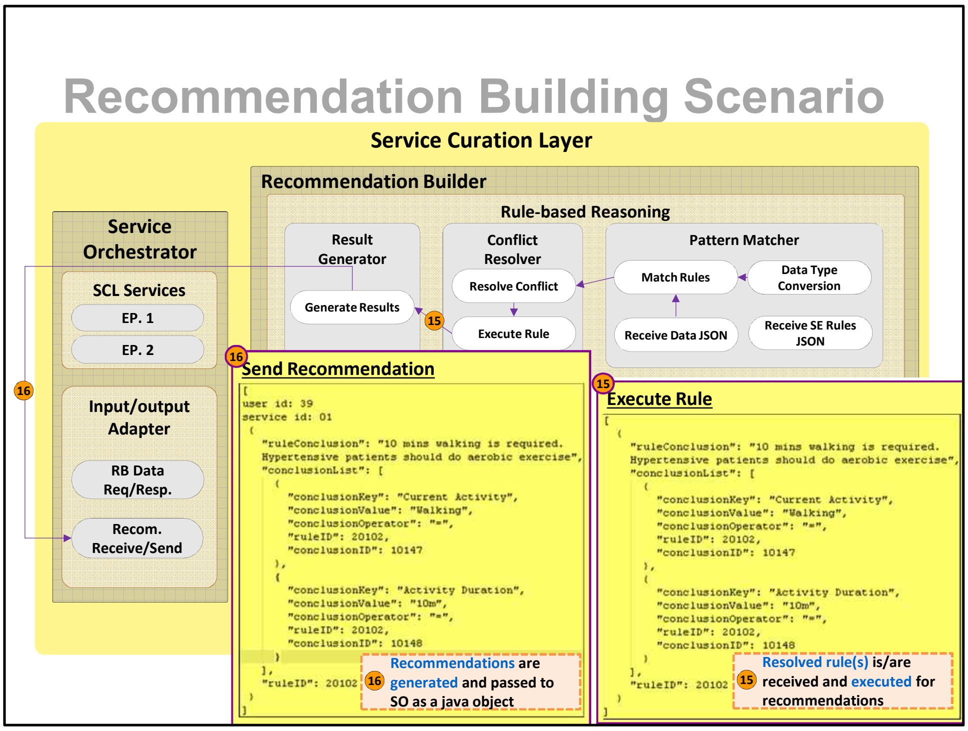
Recommendation Building Scenario
Service Curation Layer
11
Rules JSON
[
{
"ruleConclusion": "Take a five-minute walk.",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking",
"conclusionOperator": "=",
}
],
"conditionList": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionType": "String",
"conditionValueOperator": "=",
},
{
"conditionKey": "Activity Duration",
"conditionValue": "2h",
"conditionType": null,
"conditionValueOperator": "=",
}
],
"ruleID": 20103
}
]
10
Input Data/Query JSON
{
"Activity": "Sitting"
"ActivityDuration": "2 Hour"
"Gender":"Male"
"Weight Status": "Normal"
"Hypertension":"Yes"
"Disability": "none"
}
12
Type Conversion
public static boolean compare(int v1 , int String op){
boolean matched = false;
switch(op){
case "=" :
matched = v1 == v2;
break;
case "!=" :
matched = v1 != v2;
break;
case "<" :
matched = v1 < v2;
break;
case ">" :
matched = v1 > v2;
Reasoning
Pattern Matcher
Match Rules
Data Type Conversion
Receive Data JSON
Receive SE Rules JSON
loading Interface
Data Transformer
Prepare Data
Data JSON
Send Data
Utility Library
Utility Function 1
Utility Function n
Send Rules to RBR & DF
10 Input case JSON is passed to the Pattern Matcher
11 Rules JSON is passed to Pattern Matcher
12 Data types conversion take place for rules matching


Recommendation Building Scenario
Service Curation Layer
13
Matched Rules {"ruleID": 20103,"ruleID": 20102}
[
{
"ruleConclusion":"10 mins walking required.",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking"
}
],
"conditionList": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionValueOperator": "="
}
],
"ruleID": 20102
},
{
"ruleConclusion": "Take a five-minute walk.",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking"
}
],
"conditionList": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionValueOperator": "="
}
],
"ruleID": 20103
}
]
Builder
Rule-based Reasoning
Conflict Resolver
Resolve Conflict
Execute Rule
Pattern Matcher
Match Rules
Data Type Conversion
Receive Data JSON
Receive SE Rules JSON
12
Type Conversion
public static boolean compare(int v1, int String op){
boolean matched = false;
switch(op){
case "=" :
matched = v1 == v2;
break;
case "!=" :
matched = v1 != v2;
break;
case "<" :
matched = v1 < v2;
break;
case ">" :
matched = v1 > v2;
12 Data types conversion take place for rules matching
13 Rules Matcher start matching rules using forward chaining strategy and get matched rules


Recommendation Building Scenario
Service Curation Layer
13
Matched Rules {"ruleID": 20103,"ruleID": 20102}
[
{
"ruleConclusion":"10 mins walking required.",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking"
}
],
"conditionList": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionValueOperator": "="
}
],
"ruleID": 20102
},
{
"ruleConclusion": "Take a five-minute walk.",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking"
}
],
"conditionList": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionValueOperator": "="
}
],
"ruleID": 20103
}
]
Builder
Rule-based Reasoning
Conflict Resolver
Resolve Conflict
14
Execute Rule
Pattern Matcher
13
Match Rules
Data Type Conversion
Receive Data JSON
Receive SE Rules JSON
14
Resolved Rule {"ruleID": 20102}
[
{
"ruleConclusion":"10 mins walking required.",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking"
}
],
"conditionList": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionValueOperator": "="
}
],
"ruleID": 20102
},
13 Rules Matcher start matching rules using forward chaining strategy and get matched rules
14 Conflict resolver resolves matched rules using maximum specificity algorithm



Recommendation Building Scenario
Service Curation Layer
Recommendation Builder
Rule-based Reasoning
Result Generator
Conflict Resolver
Pattern Matcher
Generate Results
Resolve Conflict
Execute Rule
Match Rules
Data Type Conversion
Receive Data JSON
Receive SE Rules JSON
14
15
15 Resolved rule(s) is/are received and executed for recommendations
15 Execute Rule
[
{
"ruleConclusion": "10 mins walking is required. Hypertensive patients should do aerobic exercise",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking",
"conclusionOperator": "=",
"ruleID": 20102,
"conclusionID": 10147
},
{
"conclusionKey": "Activity Duration",
"conclusionValue": "10m",
"conclusionOperator": "=",
"ruleID": 20102,
"conclusionID": 10148
}
],
"ruleID": 20102
}
]
14 Resolved Rule {"ruleID": 20102}
[
{
"ruleConclusion":"10 mins walking required.",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking"
}
],
"conditionList": [
{
"conditionKey": "Current Activity",
"conditionValue": "Sitting",
"conditionValueOperator": "="
}
],
"ruleID": 20102
},
14 Conflict resolver resolves matched rules using maximum specificity algorithm



Recommendation Building Scenario
Service Curation Layer
Service Orchestrator
SCL Services
EP. 1
EP. 2
Input/output Adapter
RB Data Req/Resp.
Recom. Receive/Send
16
Recommendation Builder
Rule-based Reasoning
Result Generator
Conflict Resolver
Pattern Matcher
Generate Results
Resolve Conflict
Execute Rule
Match Rules
Data Type Conversion
Receive Data JSON
Receive SE Rules JSON
15
16 Send Recommendation
[
user id: 39
service id: 01
{
"ruleConclusion": "10 mins walking is required. Hypertensive patients should do aerobic exercise",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking",
"conclusionOperator": "=",
"ruleID": 20102,
"conclusionID": 10147
},
{
"conclusionKey": "Activity Duration",
"conclusionValue": "10m",
"conclusionOperator": "=",
"ruleID": 20102,
"conclusionID": 10148
}
],
"ruleID": 20102
}
]
16 Recommendations are generated and passed to SO as a java object
15 Execute Rule
[
{
"ruleConclusion": "10 mins walking is required. Hypertensive patients should do aerobic exercise",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking",
"conclusionOperator": "=",
"ruleID": 20102,
"conclusionID": 10147
},
{
"conclusionKey": "Activity Duration",
"conclusionValue": "10m",
"conclusionOperator": "=",
"ruleID": 20102,
"conclusionID": 10148
}
],
"ruleID": 20102
}
]
15 Resolved rule(s) is/are received and executed for recommendations

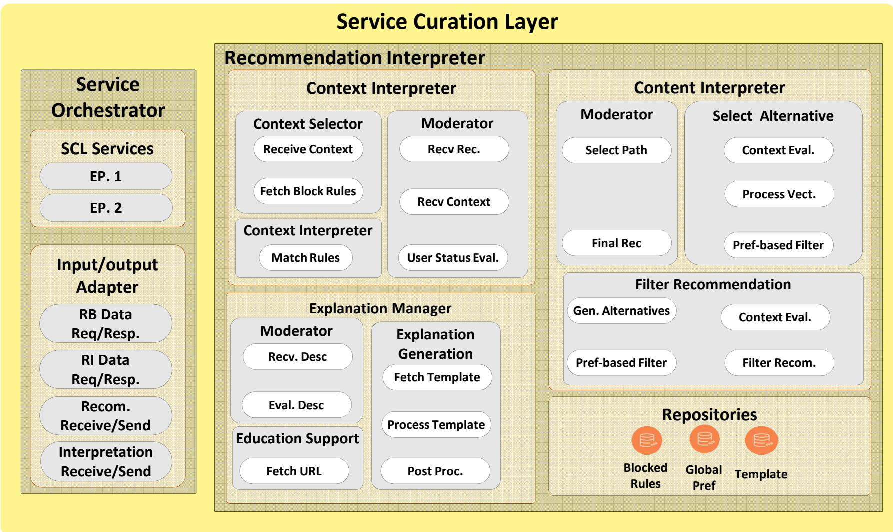
Recommendation Interpreter Scenario

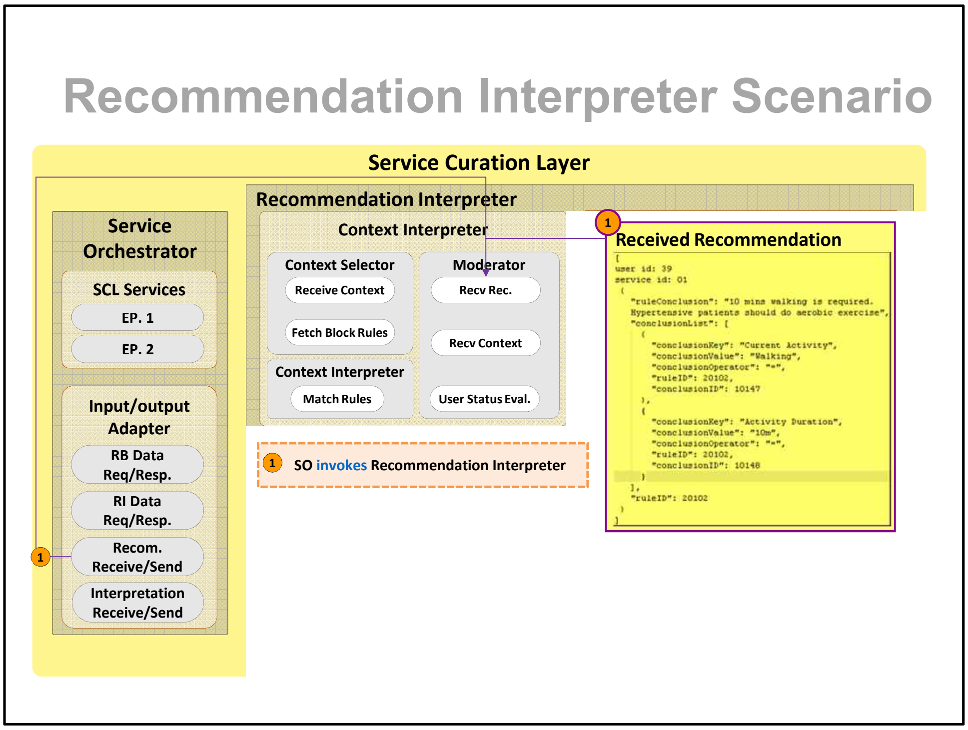
Scenario 2: Recommendation Interpreting

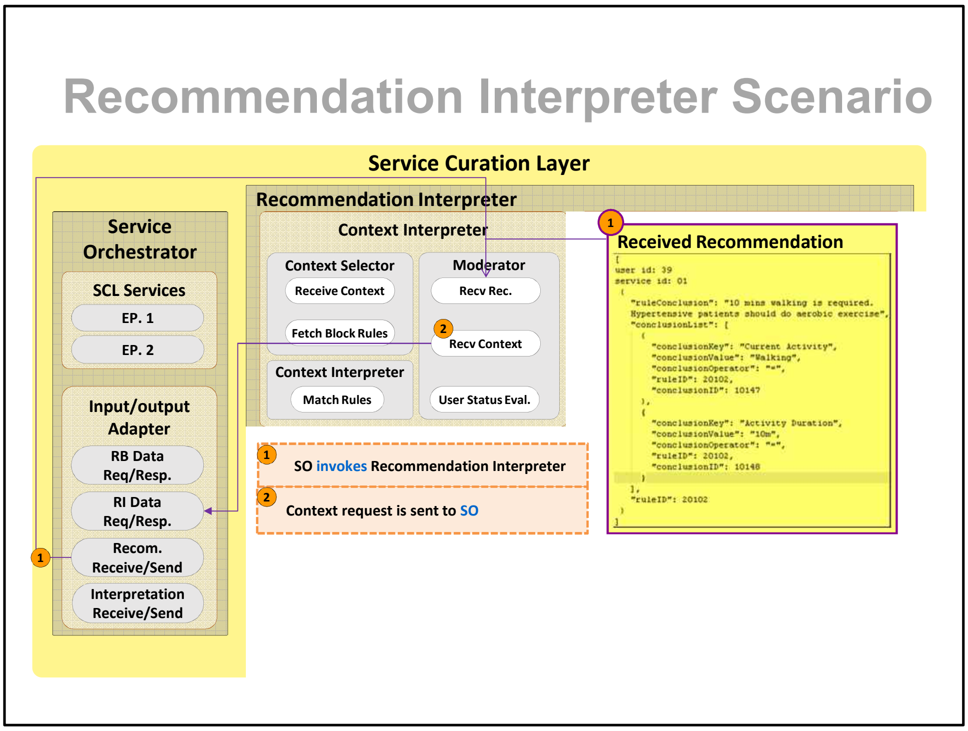
Desc : Interpretation of Recommendations for Personalization.

1. Recommendation received from Service Orchestrator (SO) to Recommendation Interpreter (RI).
2. RI interprets the recommendations Based on the location, high level, and weather context retrieved from DCL.
3. RI sends the personalized recommendation back to SO.
4. SO forwards the personalized recommendation to DCL for persistence and SL for presentation.

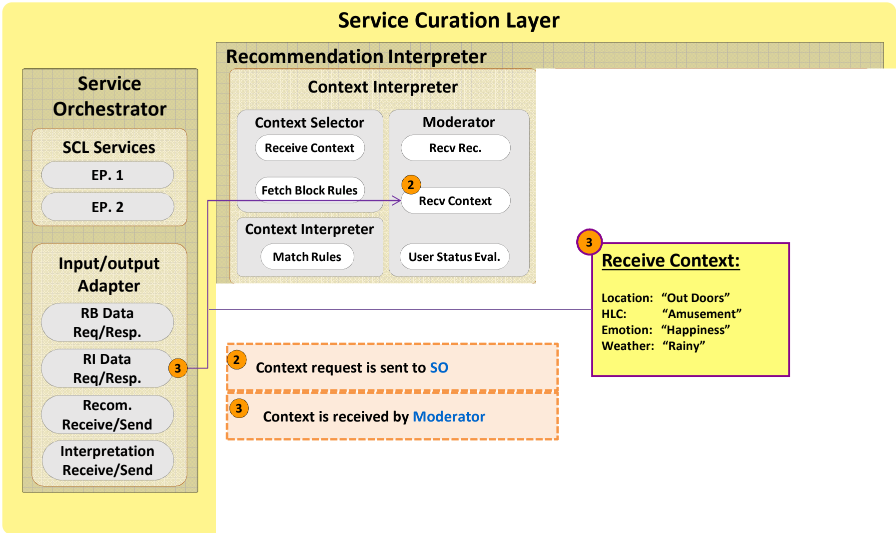
Recommendation Interpreter Scenario



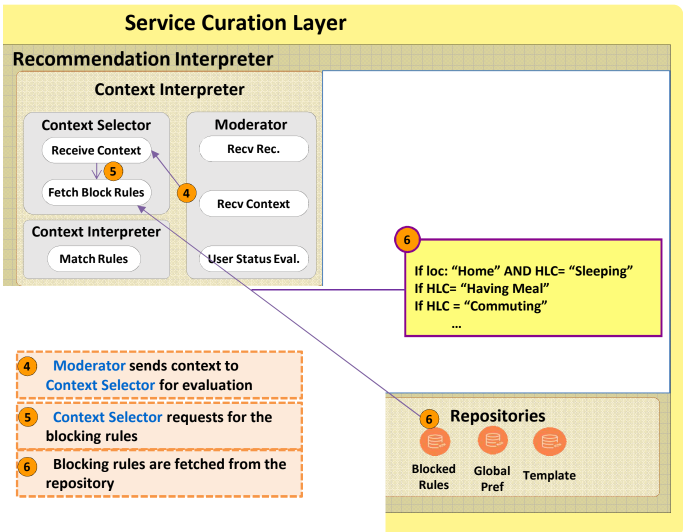

Recommendation Interpreter Scenario
Service Curation Layer
Recommendation Interpreter
Context Interpreter
Service Orchestrator
SCL Services
EP. 1
EP. 2
Input/output Adapter
RB Data Req/Resp.
RI Data Req/Resp.
Recom. Receive/Send
Interpretation Receive/Send
Context Selector
Receive Context
Fetch Block Rules
Context Interpreter
Match Rules
Moderator
Recv Rec.
Recv Context
User Status Eval.
1 SO invokes Recommendation Interpreter
Received Recommendation
[
user id: 39
service id: 01
{
"ruleConclusion": "10 mins walking is required. Hypertensive patients should do aerobic exercise",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking",
"conclusionOperator": "=",
"ruleID": 20102,
"conclusionID": 10147
},
{
"conclusionKey": "Activity Duration",
"conclusionValue": "10m",
"conclusionOperator": "=",
"ruleID": 20102,
"conclusionID": 10148
}
],
"ruleID": 20102
}
]


Recommendation Interpreter Scenario
Service Curation Layer
Recommendation Interpreter
Context Interpreter
Service Orchestrator
SCL Services
EP. 1
EP. 2
Input/output Adapter
RB Data Req/Resp.
RI Data Req/Resp.
Recom. Receive/Send
Interpretation Receive/Send
Context Selector
Receive Context
Fetch Block Rules
Context Interpreter
Match Rules
Moderator
Recv Rec.
Recv Context
User Status Eval.
1 SO invokes Recommendation Interpreter
2 Context request is sent to SO
Received Recommendation
[
user id: 39
service id: 01
{
"ruleConclusion": "10 mins walking is required. Hypertensive patients should do aerobic exercise",
"conclusionList": [
{
"conclusionKey": "Current Activity",
"conclusionValue": "Walking",
"conclusionOperator": "=",
"ruleID": 20102,
"conclusionID": 10147
},
{
"conclusionKey": "Activity Duration",
"conclusionValue": "10m",
"conclusionOperator": "=",
"ruleID": 20102,
"conclusionID": 10148
}
],
"ruleID": 20102
}
]

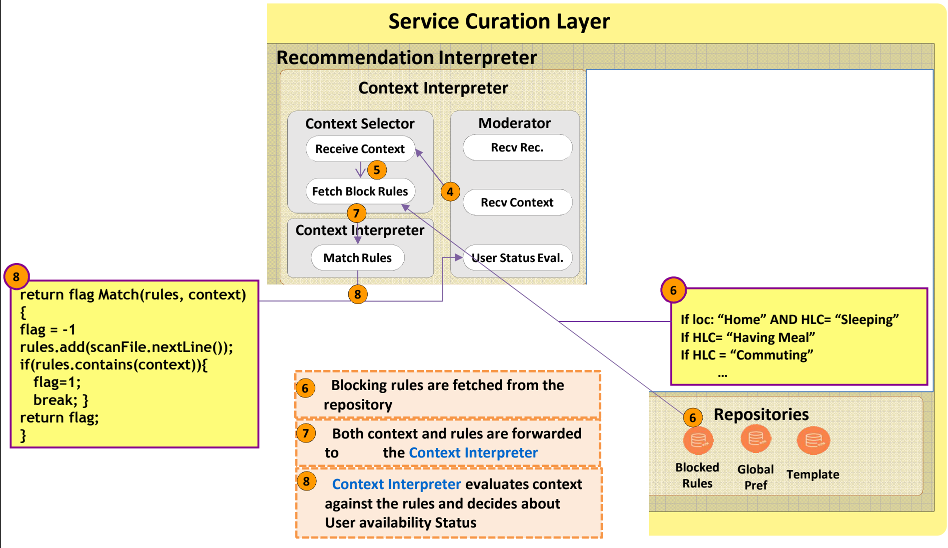
Recommendation Interpreter Scenario



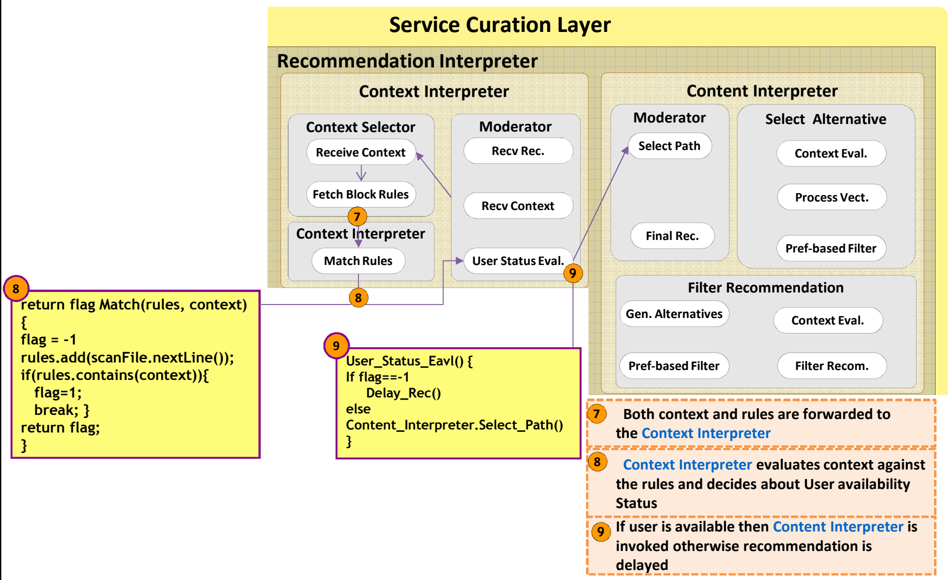
Recommendation Interpreter Scenario



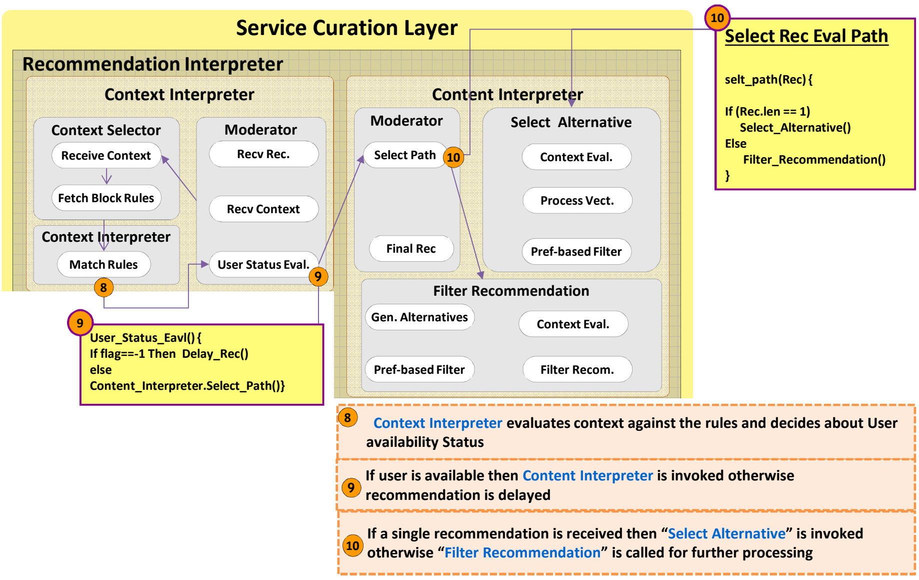
Recommendation Interpreter Scenario



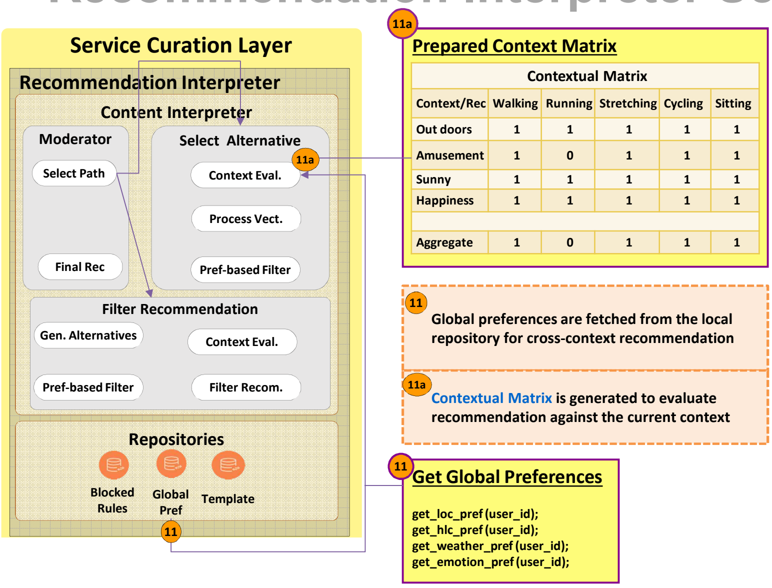
Recommendation Interpreter Scenario



Recommendation Interpreter Scenario



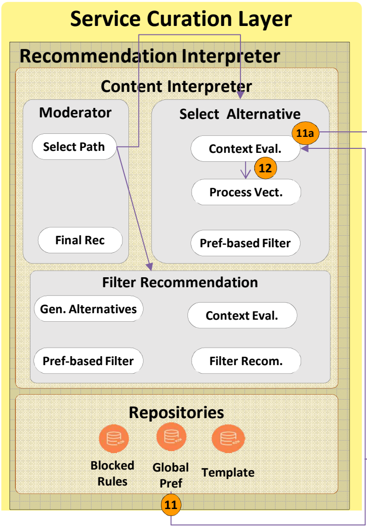
Recommendation Interpreter Scenario



Contextual Matrix

Context/Rec	Walking	Running	Stretching	Cycling	Sitting
Out doors	1	1	1	1	1
Amusement	1	0	1	1	1
Sunny	1	1	1	1	1
Happiness	1	1	1	1	1
Aggregate	1	0	1	1	1

Recommendation Interpreter Scenario



11a **Prepared Context Matrix**

Contextual Matrix					
Context/Rec	Walking	Running	Stretching	Cycling	Sitting
Out doors	1	1	1	1	1
Amusement	1	0	1	0	1
Sunny	1	1	1	1	1
Happiness	1	1	1	1	1
Aggregate	1	0	1	1	1

Running
5m

11 Global preferences are fetched from the local repository for cross-context recommendation

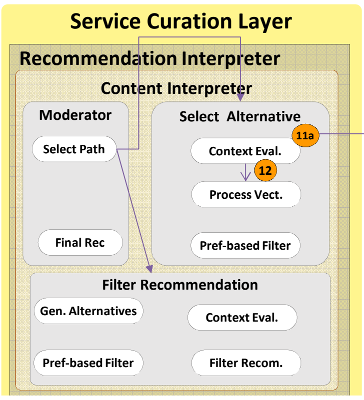
11a Contextual Matrix is generated to evaluate recommendation against the current context

12 Recommendation (running) is unsuitable in the current context

11 **Get Global Preferences**

```
get_loc_pref(user_id);
get_hlc_pref(user_id);
get_weather_pref(user_id);
get_emotion_pref(user_id);
```

Recommendation Interpreter Scenario



11a **Prepared Context Matrix**

Contextual Matrix					
Context/Rec	Walking	Running	Stretching	Cycling	Sitting
Out doors	1	1	1	1	1
Amusement	1	0	1	0	1
Sunny	1	1	1	1	1
Happiness	1	1	1	1	1
Aggregate	1	0	1	1	1

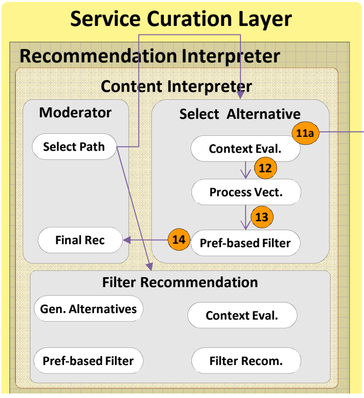
Walking
10m

Stretching
15m

11a Contextual Matrix is generated to evaluate recommendation against the current context

12 Recommendation (running) is unsuitable in the current context
In case of unsuitability, Alternative Recommendations are searched in the Contextual Matrix

Recommendation Interpreter Scenario



14 **Prepared Context Matrix**

Contextual Matrix

Context/Rec	Walking	Running	Stretching	Cycling	Sitting
Out doors	1	1	1	1	1
Amusement	1	0	1	0	1
Sunny	1	1	1	1	1
Happiness	1	1	1	1	1
Aggregate	1	0	1	1	1

Walking 10m
Stretching 15m

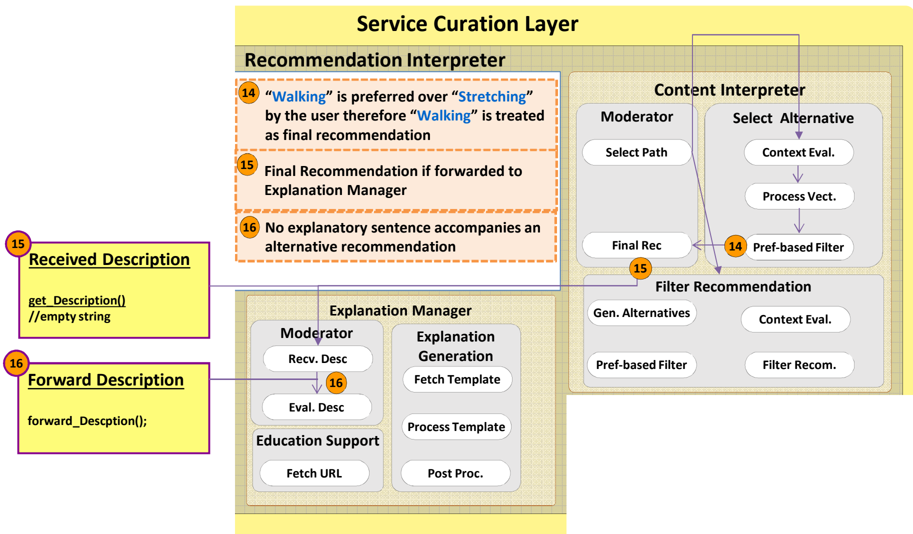
get_pref (user_id);

12 Recommendation (running) is unsuitable in the current context
In case of unsuitability, Alternative Recommendations are searched in the Contextual Matrix

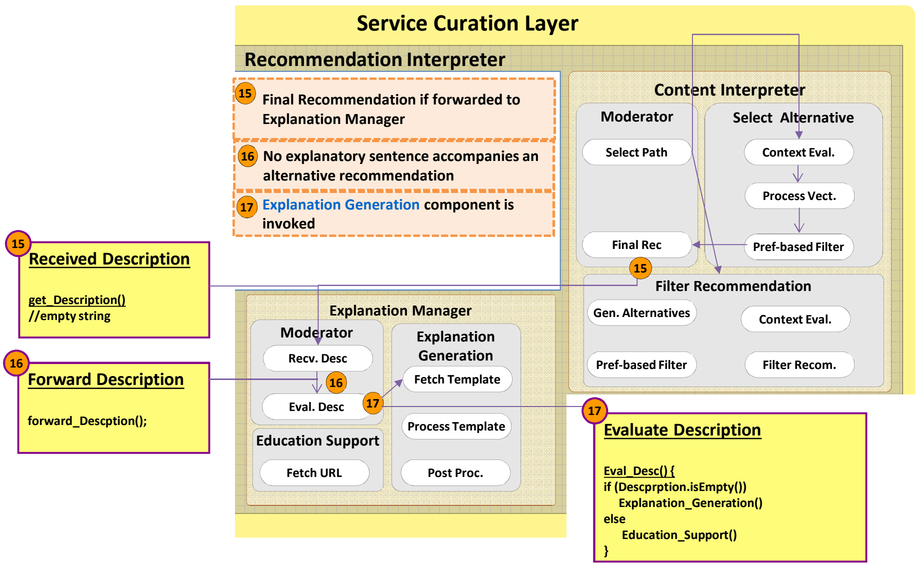
13 For multiple alternative recommendations, User's Preferences are weighed in

14 "Walking" is preferred over "Stretching" by the user therefore "Walking" is treated as final recommendation

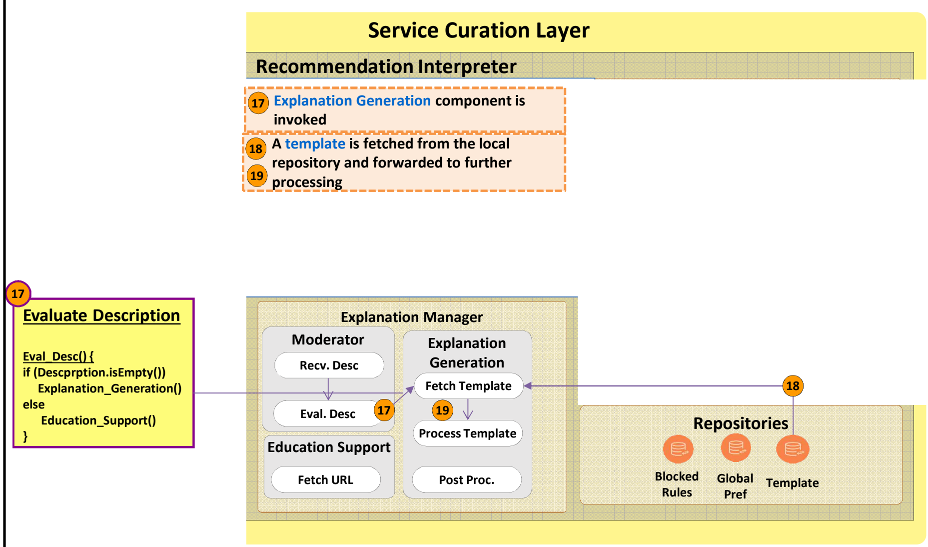
Recommendation Interpreter Scenario

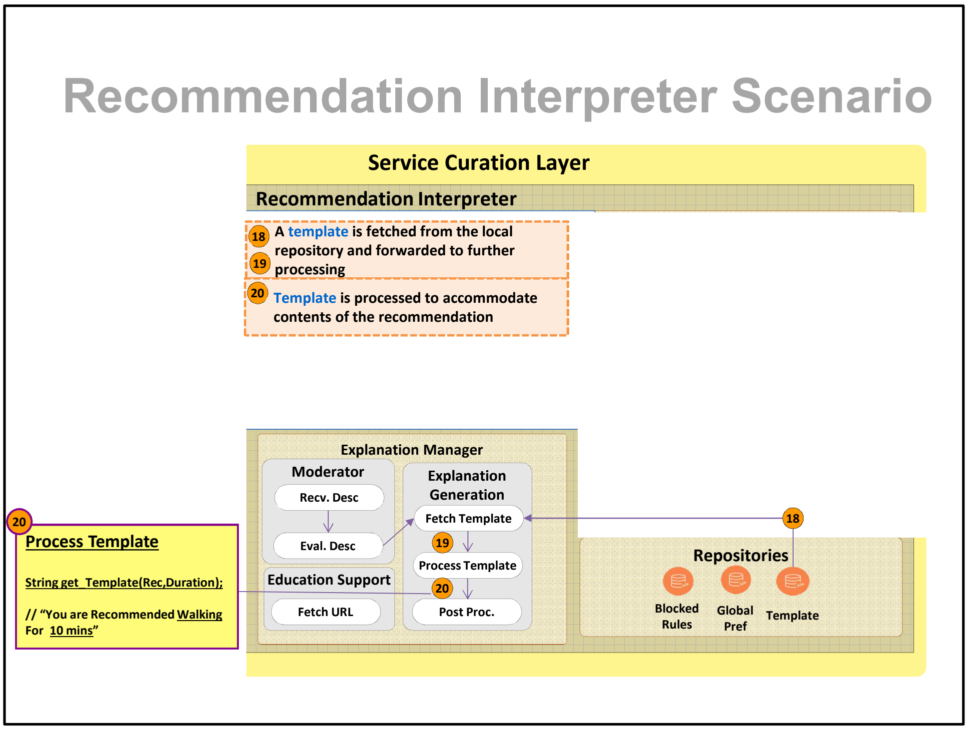


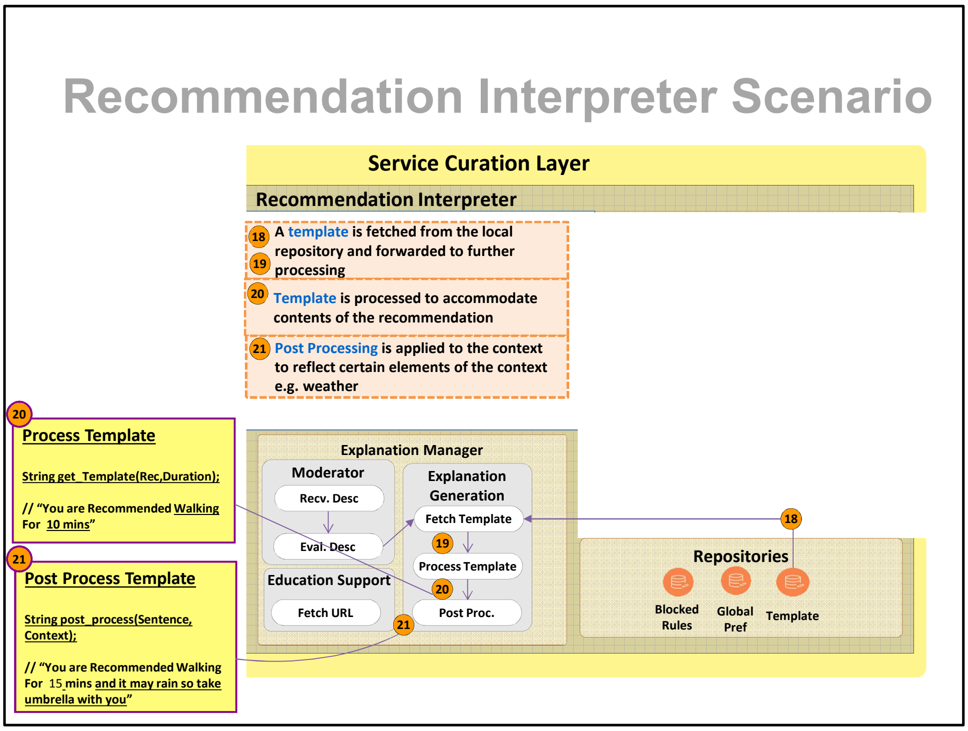
Recommendation Interpreter Scenario

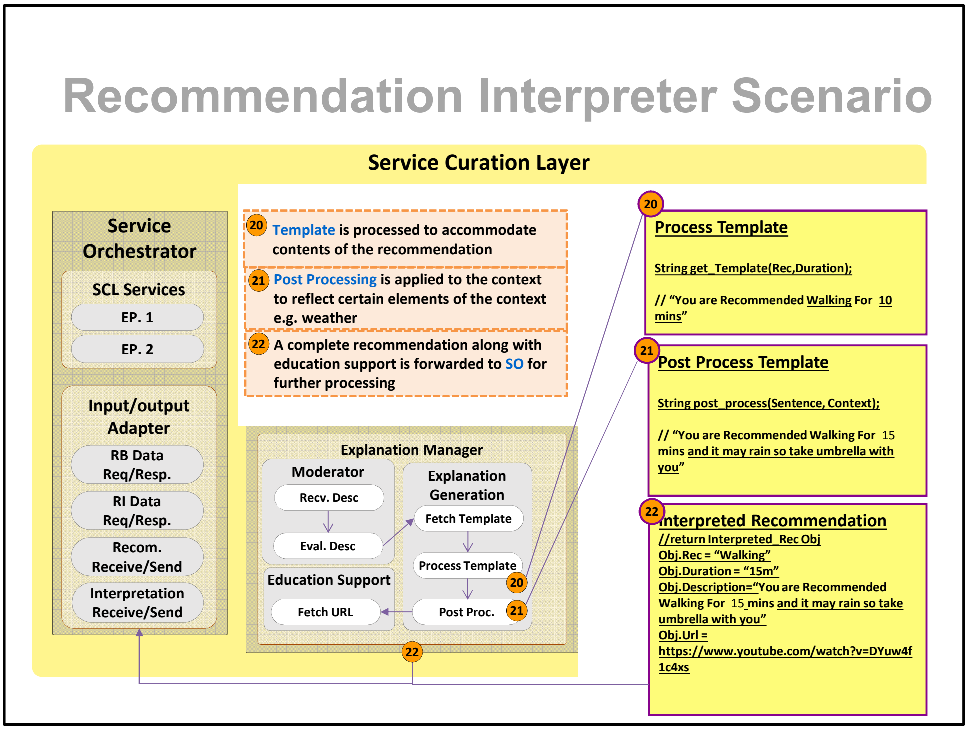


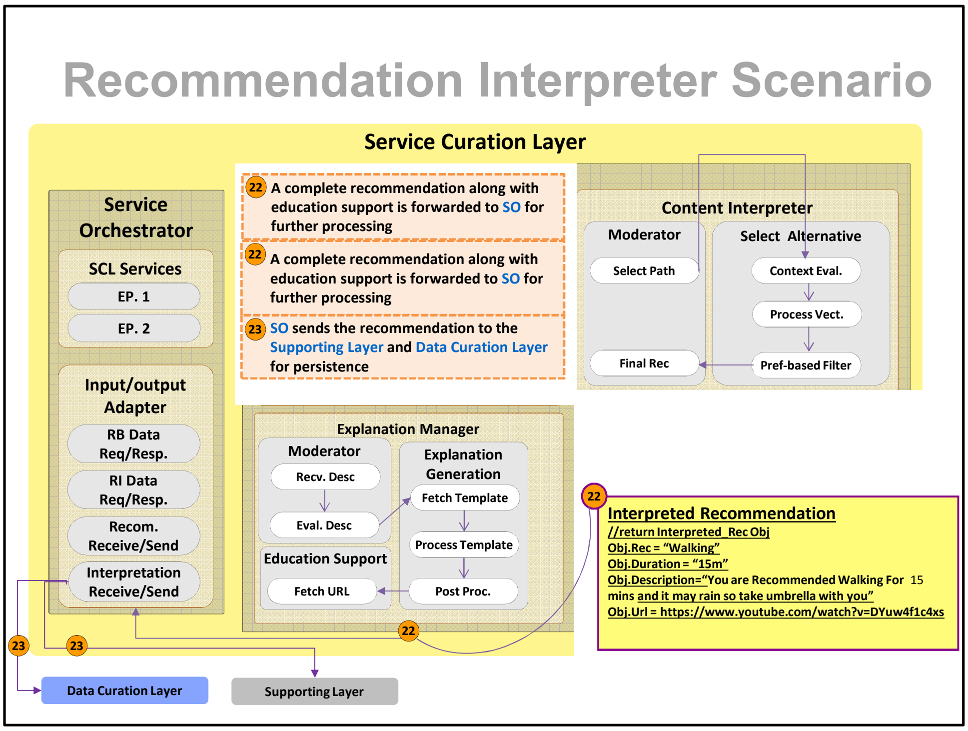
Recommendation Interpreter Scenario




Recommendation Interpreter Scenario
Service Curation Layer
Recommendation Interpreter
18 19 A template is fetched from the local repository and forwarded to further processing
20 Template is processed to accommodate contents of the recommendation
Explanation Manager
Moderator
Recv. Desc
Eval. Desc
Explanation Generation
Fetch Template
19
Process Template
20
Post Proc.
Education Support
Fetch URL
18
Repositories
Blocked Rules
Global Pref
Template
20
Process Template
String get_Template(Rec,Duration);
// "You are Recommended Walking For 10 mins"


Recommendation Interpreter Scenario
Service Curation Layer
Recommendation Interpreter
18 19 A template is fetched from the local repository and forwarded to further processing
20 Template is processed to accommodate contents of the recommendation
21 Post Processing is applied to the context to reflect certain elements of the context e.g. weather
Explanation Manager
Moderator
Recv. Desc
Eval. Desc
Explanation Generation
Fetch Template
19
Process Template
20
Post Proc.
21
Education Support
Fetch URL
18
Repositories
Blocked Rules
Global Pref
Template
20
Process Template
String get_Template(Rec,Duration);
// "You are Recommended Walking For 10 mins"
21
Post Process Template
String post_process(Sentence, Context);
// "You are Recommended Walking For 15 mins and it may rain so take umbrella with you"

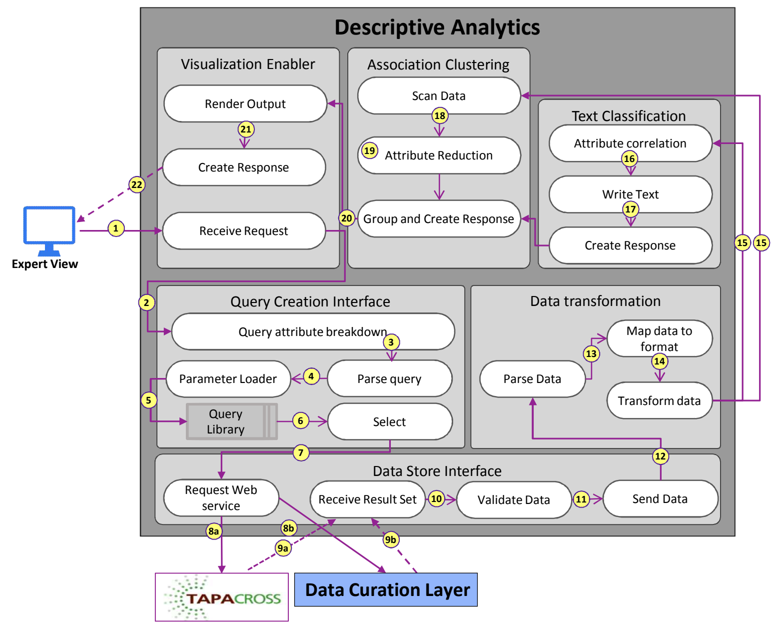

Recommendation Interpreter Scenario
Service Curation Layer
Service Orchestrator
SCL Services
EP. 1
EP. 2
Input/output Adapter
RB Data Req/Resp.
RI Data Req/Resp.
Recom. Receive/Send
Interpretation Receive/Send
20 Template is processed to accommodate contents of the recommendation
21 Post Processing is applied to the context to reflect certain elements of the context e.g. weather
22 A complete recommendation along with education support is forwarded to SO for further processing
Explanation Manager
Moderator
Recv. Desc
Eval. Desc
Explanation Generation
Fetch Template
Process Template
Post Proc.
Education Support
Fetch URL
20 Process Template
String get_Template(Rec,Duration);
// "You are Recommended Walking For 10 mins"
21 Post Process Template
String post_process(Sentence, Context);
// "You are Recommended Walking For 15 mins and it may rain so take umbrella with you"
22 Interpreted Recommendation
//return Interpreted_Rec Obj
Obj.Rec = "Walking"
Obj.Duration = "15m"
Obj.Description="You are Recommended Walking For 15 mins and it may rain so take umbrella with you"
Obj.Url = https://www.youtube.com/watch?v=DYuw4f1c4xs


Recommendation Interpreter Scenario
Service Curation Layer
Service Orchestrator
SCL Services
EP. 1
EP. 2
Input/output Adapter
RB Data Req/Resp.
RI Data Req/Resp.
Recom. Receive/Send
Interpretation Receive/Send
22 A complete recommendation along with education support is forwarded to SO for further processing
22 A complete recommendation along with education support is forwarded to SO for further processing
23 SO sends the recommendation to the Supporting Layer and Data Curation Layer for persistence
Content Interpreter
Moderator
Select Path
Final Rec
Select Alternative
Context Eval.
Process Vect.
Pref-based Filter
Explanation Manager
Moderator
Recv. Desc
Eval. Desc
Explanation Generation
Fetch Template
Process Template
Post Proc.
Education Support
Fetch URL
22 Interpreted Recommendation
//return Interpreted_Rec Obj
Obj.Rec = "Walking"
Obj.Duration = "15m"
Obj.Description="You are Recommended Walking For 15 mins and it may rain so take umbrella with you"
Obj.Url = https://www.youtube.com/watch?v=DYuw4f1c4xs
23
23
Data Curation Layer
Supporting Layer

STENCILS for SL Micro-Animation

Descriptive Analytics

DESCRIPTIVE ANALYTICS DETAILED ARCHITECTURE

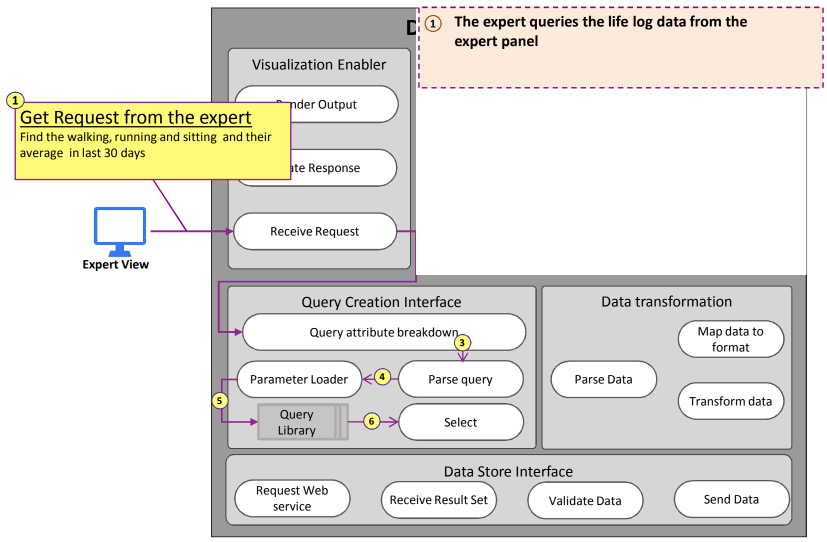


Scenario 1: Visualization and Analytics from Lifelog data and SNS

1. The request is **parsed** from the expert.
2. The request is converted into a **queries** from the library.
3. Data is integrated and **transformed** into a predefined format
4. Trend analysis is done through **calculating facts** and **grouping** data
5. The data is **visualized** in graphs and data facts are presented.

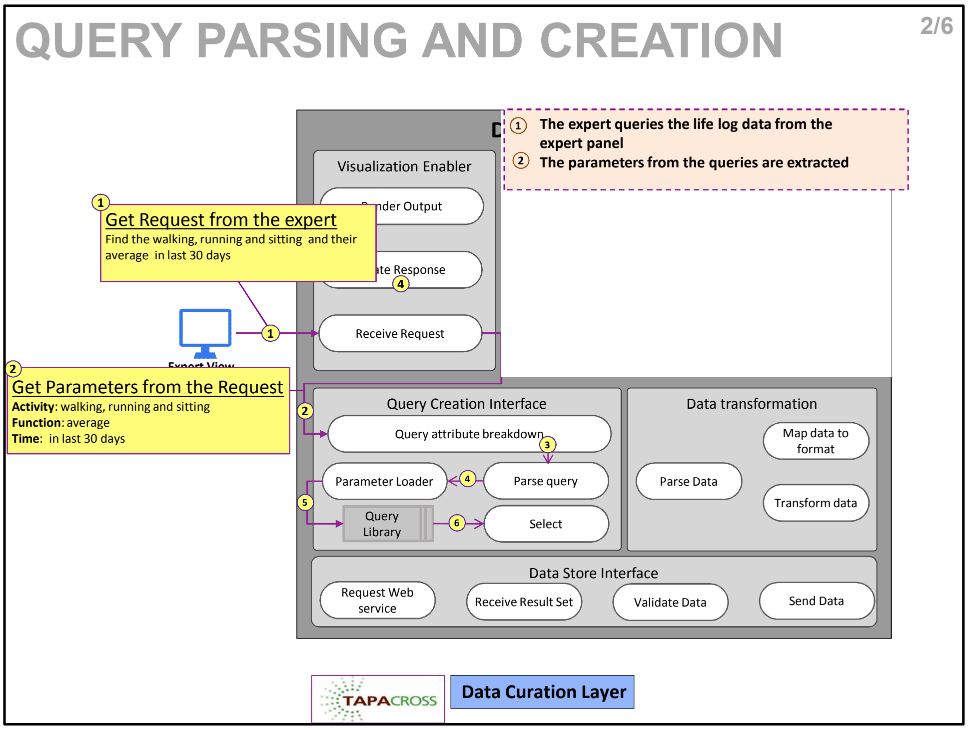
QUERY PARSING AND CREATION

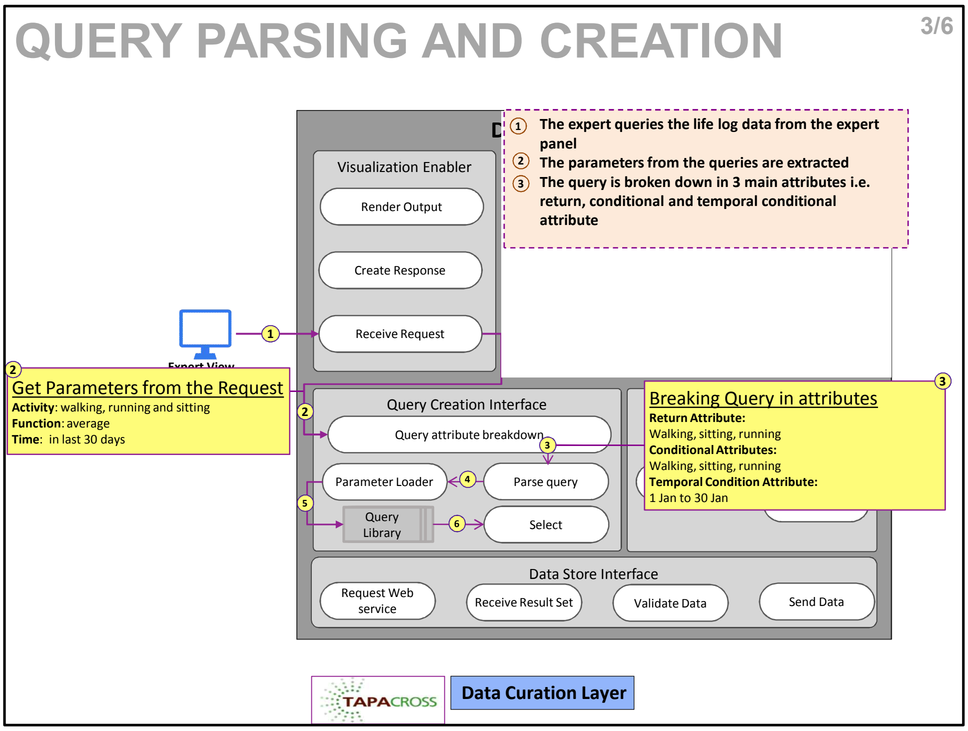
1/6



TAPACROSS

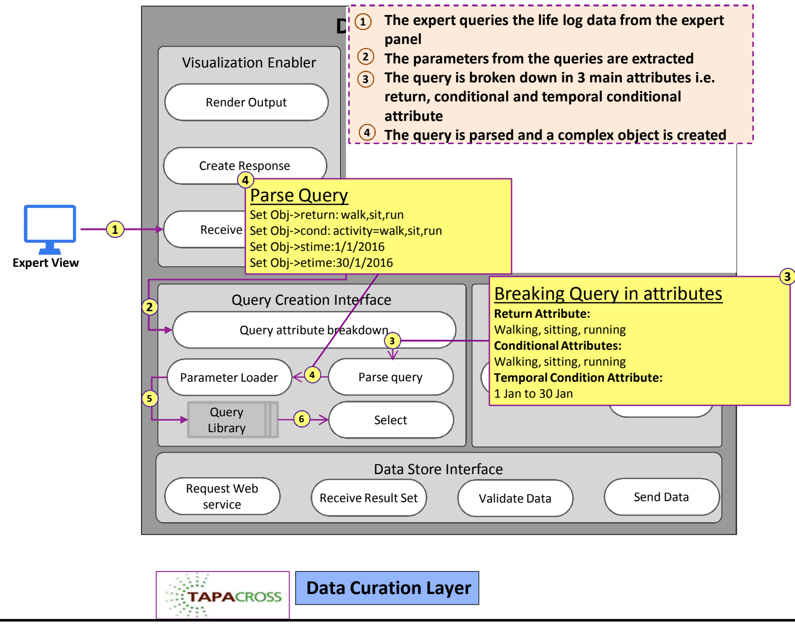
Data Curation Layer


QUERY PARSING AND CREATION
2/6
The expert queries the life log data from the expert panel
The parameters from the queries are extracted
Get Request from the expert
Find the walking, running and sitting and their average in last 30 days
Visualization Enabler
Render Output
Create Response
Receive Request
Expert View
Get Parameters from the Request
Activity: walking, running and sitting
Function: average
Time: in last 30 days
Query Creation Interface
Query attribute breakdown
Parameter Loader
Parse query
Query Library
Select
Data transformation
Map data to format
Parse Data
Transform data
Data Store Interface
Request Web service
Receive Result Set
Validate Data
Send Data
TAPACROSS
Data Curation Layer


QUERY PARSING AND CREATION
3/6
The expert queries the life log data from the expert panel
The parameters from the queries are extracted
The query is broken down in 3 main attributes i.e. return, conditional and temporal conditional attribute
Visualization Enabler
Render Output
Create Response
Receive Request
Expert View
Get Parameters from the Request
Activity: walking, running and sitting
Function: average
Time: in last 30 days
Query Creation Interface
Query attribute breakdown
Parameter Loader
Parse query
Query Library
Select
Breaking Query in attributes
Return Attribute:
Walking, sitting, running
Conditional Attributes:
Walking, sitting, running
Temporal Condition Attribute:
1 Jan to 30 Jan
Data Store Interface
Request Web service
Receive Result Set
Validate Data
Send Data
TAPACROSS
Data Curation Layer

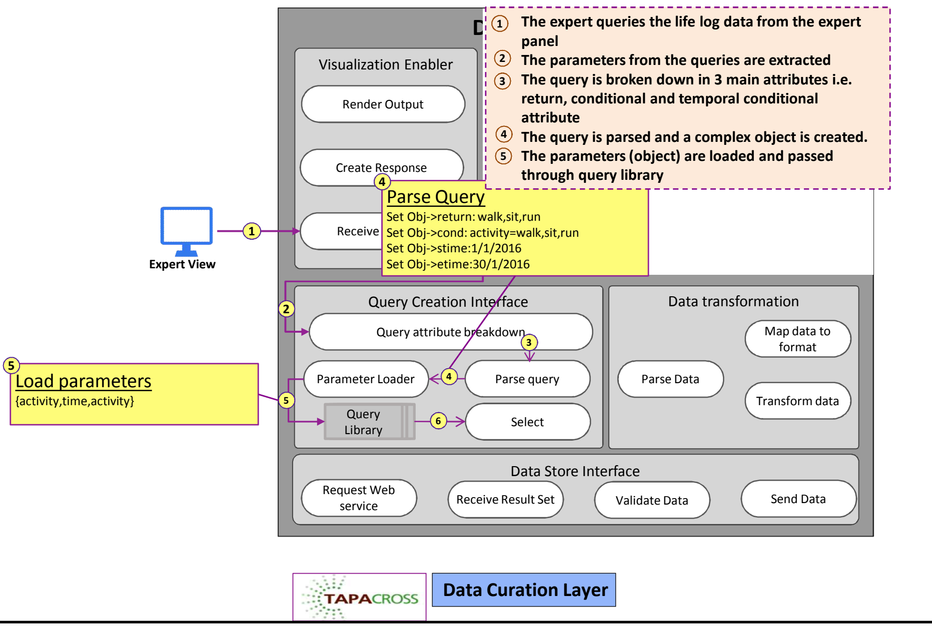
QUERY PARSING AND CREATION

4/6

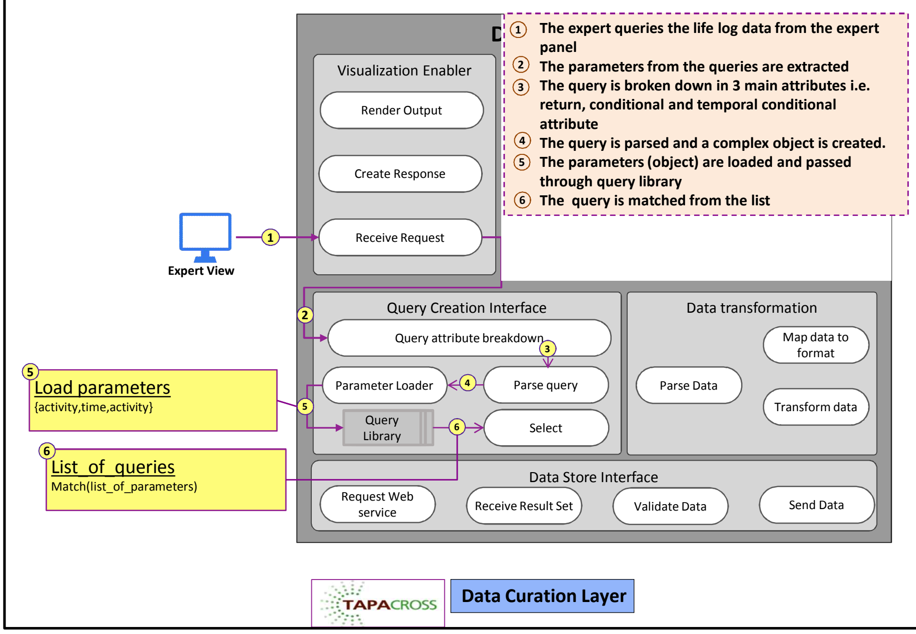


QUERY PARSING AND CREATION

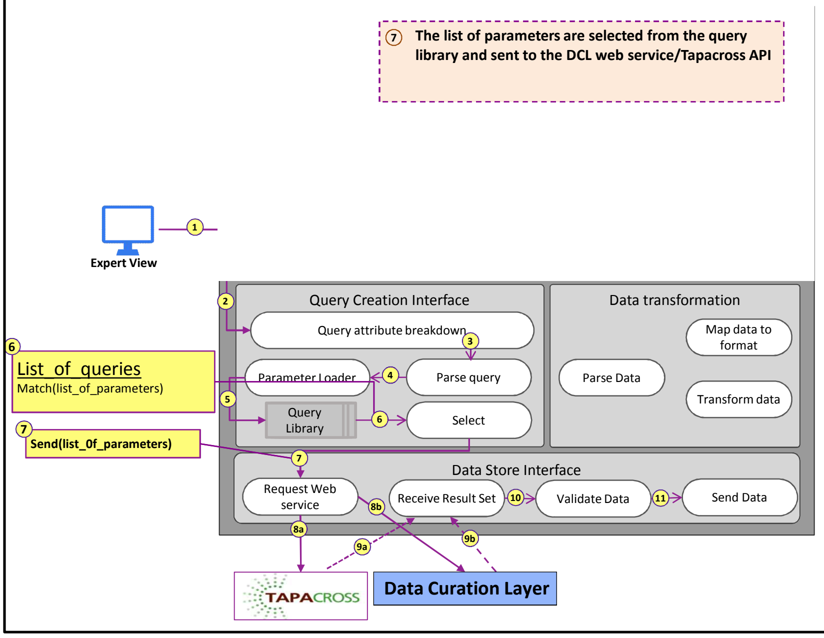
5/6

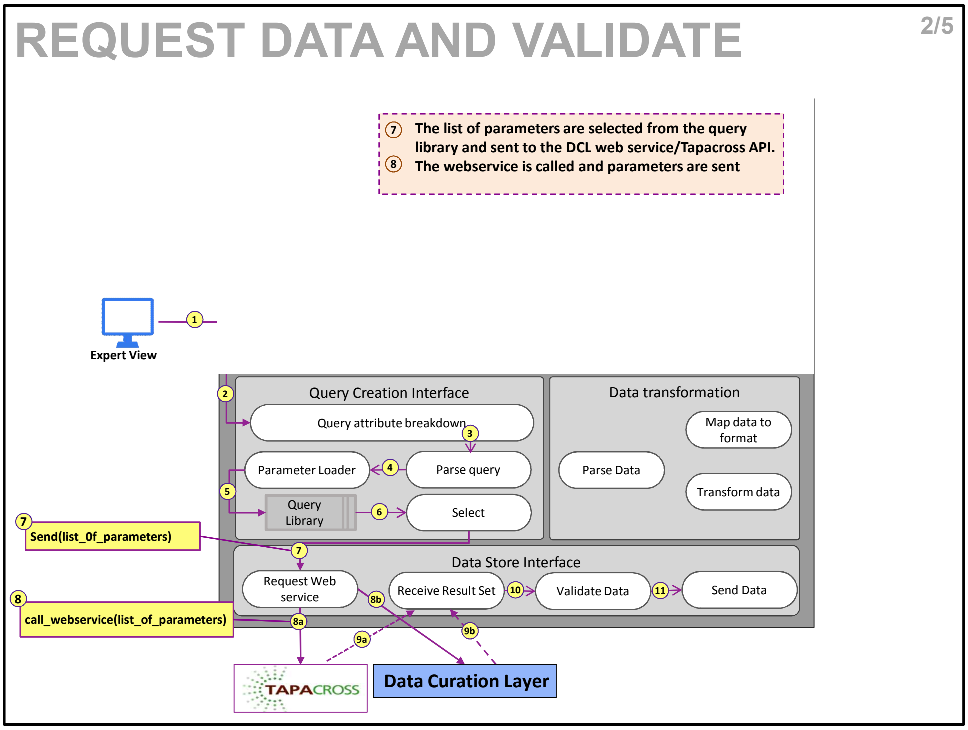


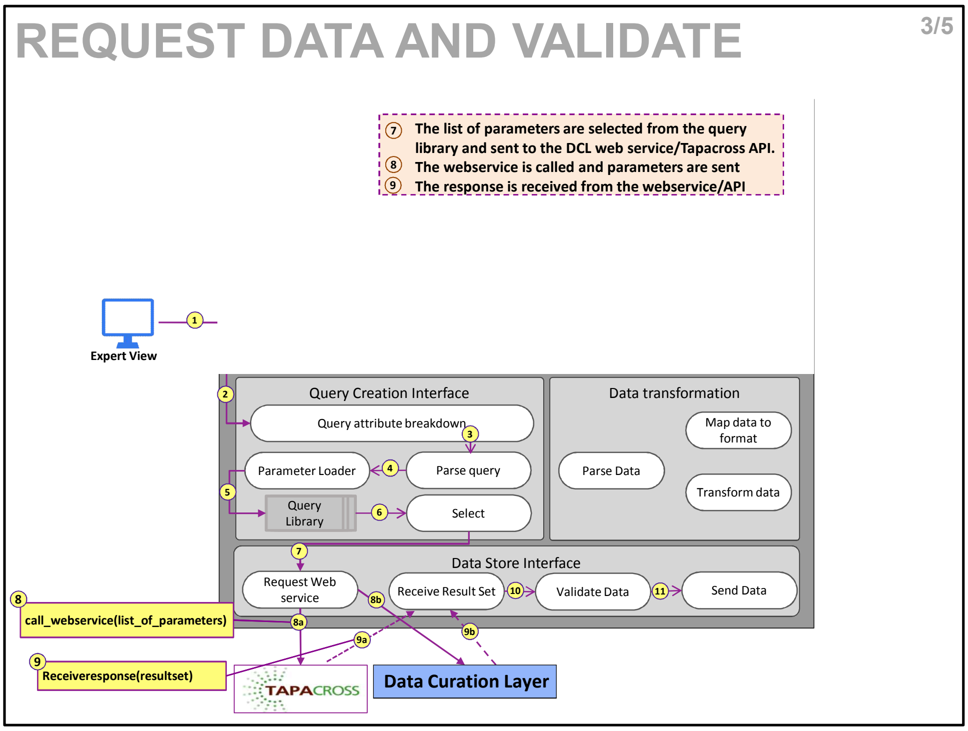
QUERY PARSING AND CREATION



REQUEST DATA AND VALIDATE

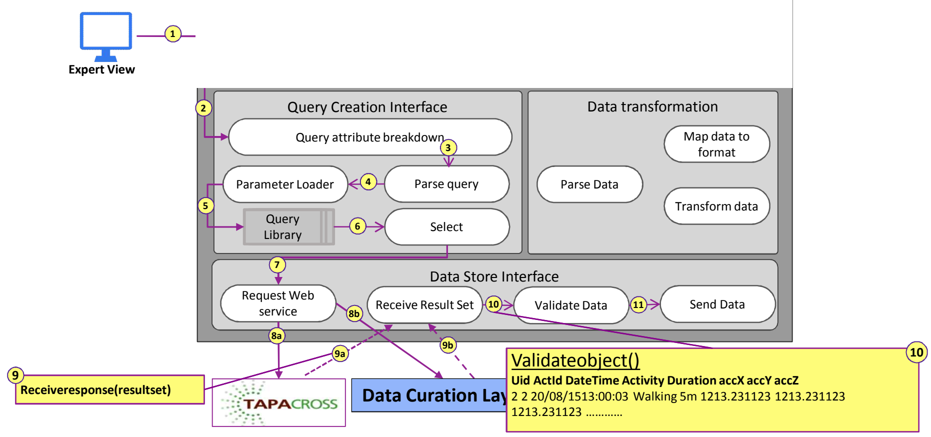



REQUEST DATA AND VALIDATE
2/5
7 The list of parameters are selected from the query library and sent to the DCL web service/Tapacross API.
8 The webservice is called and parameters are sent
Expert View
Query Creation Interface
Query attribute breakdown
Parameter Loader
Parse query
Query Library
Select
Data transformation
Map data to format
Parse Data
Transform data
Send(list_0f_parameters)
call_webservice(list_of_parameters)
Data Store Interface
Request Web service
Receive Result Set
Validate Data
Send Data
TAPACROSS
Data Curation Layer


REQUEST DATA AND VALIDATE
3/5
7 The list of parameters are selected from the query library and sent to the DCL web service/Tapacross API.
8 The webservice is called and parameters are sent
9 The response is received from the webservice/API
Expert View
Query Creation Interface
Query attribute breakdown
Parameter Loader
Parse query
Query Library
Select
Data transformation
Map data to format
Parse Data
Transform data
call_webservice(list_of_parameters)
Receiveresponse(resultset)
Data Store Interface
Request Web service
Receive Result Set
Validate Data
Send Data
TAPACROSS
Data Curation Layer

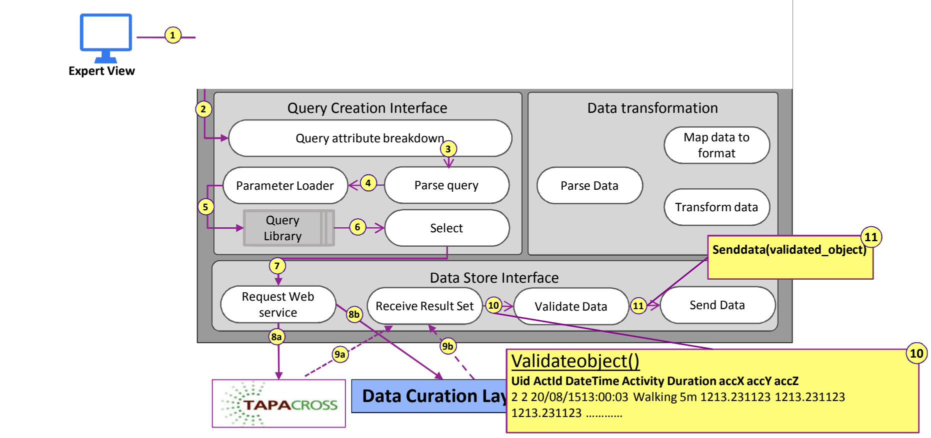
REQUEST DATA AND VALIDATE

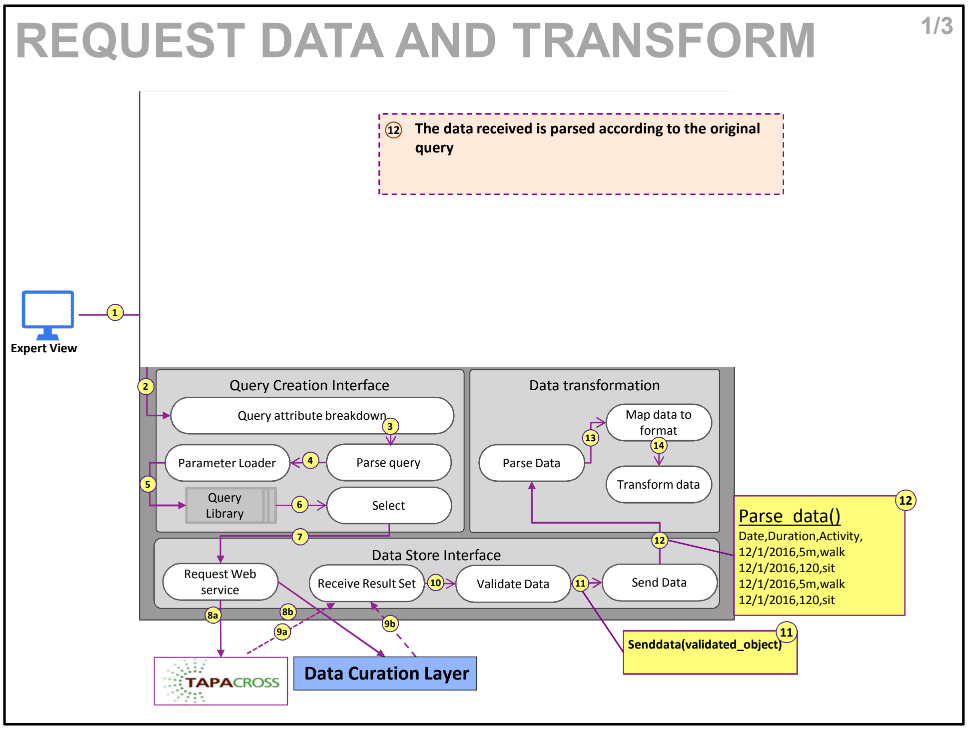
7 The list of parameters are selected from the query library and sent to the DCL web service/Tapacross API.
8 The webservice is called and parameters are sent
9 The response is received from the webservice/API
10 The object is validated against the parameters sent from the webservice.

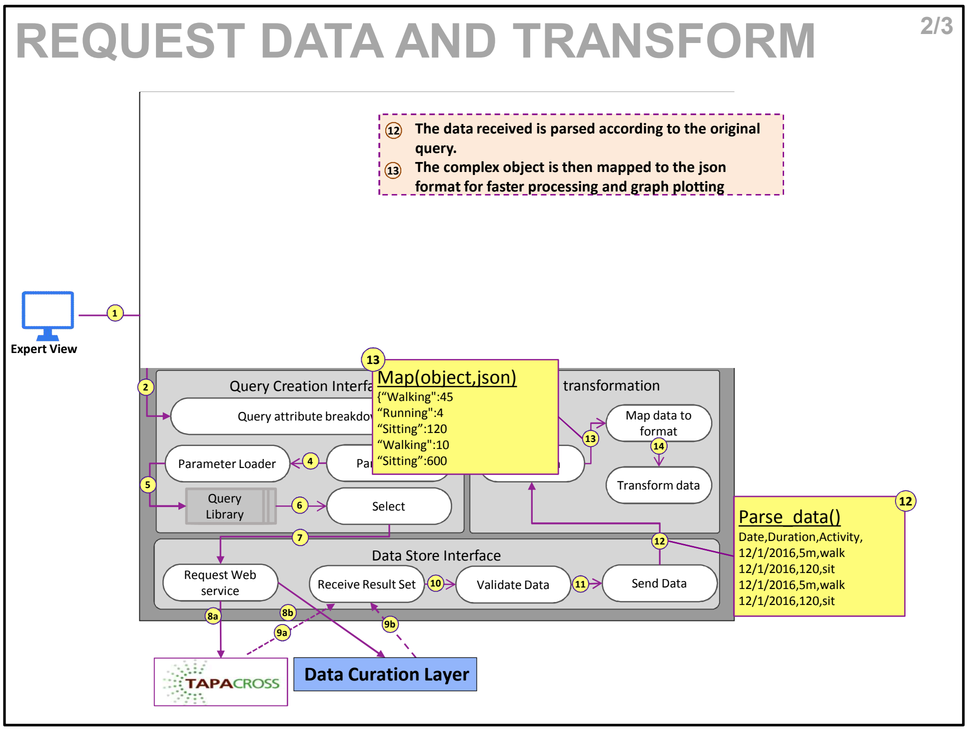


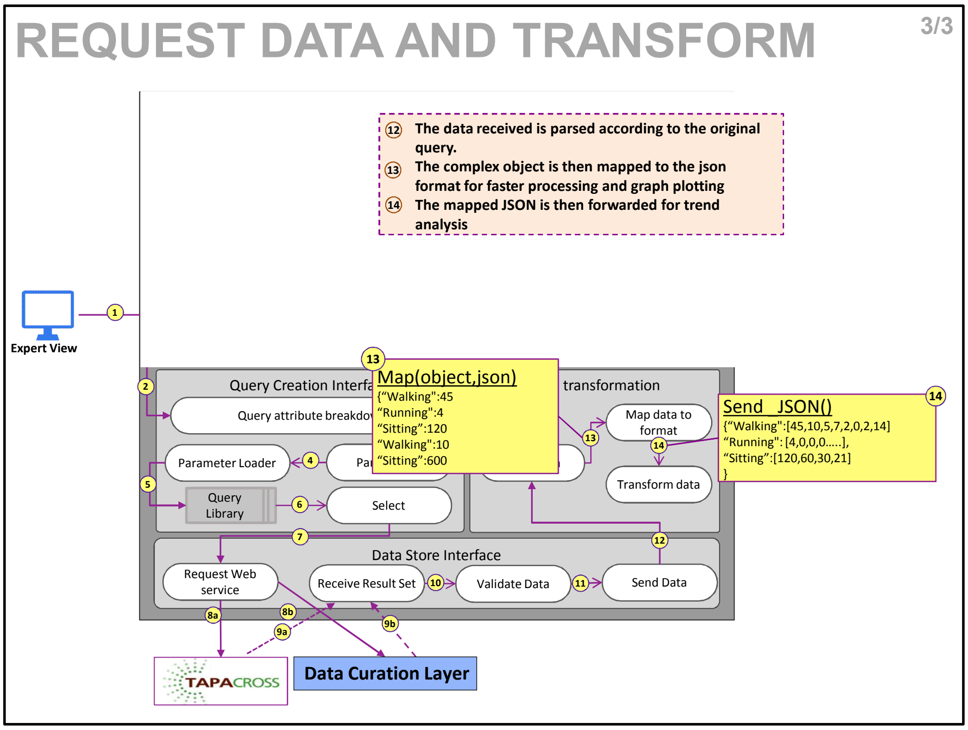
REQUEST DATA AND VALIDATE

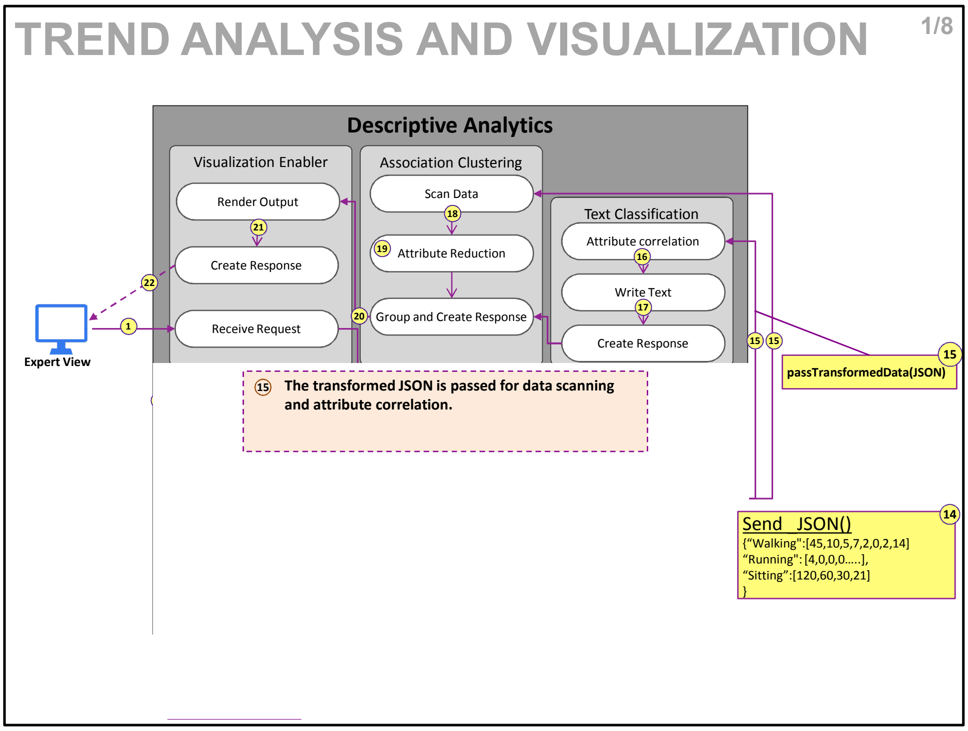
7 The list of parameters are selected from the query library and sent to the DCL web service/Tapacross API.
8 The webservice is called and parameters are sent
9 The response is received from the webservice/API
10 The object is validated against the parameters sent from the webservice.
11 The validated object is sent forward for transformation

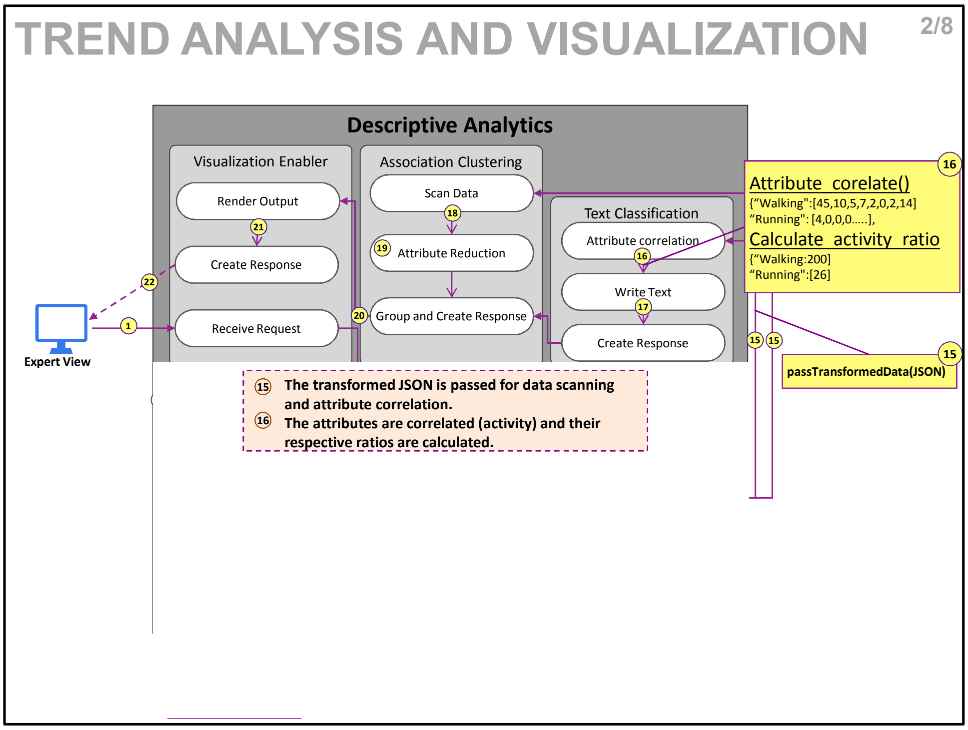


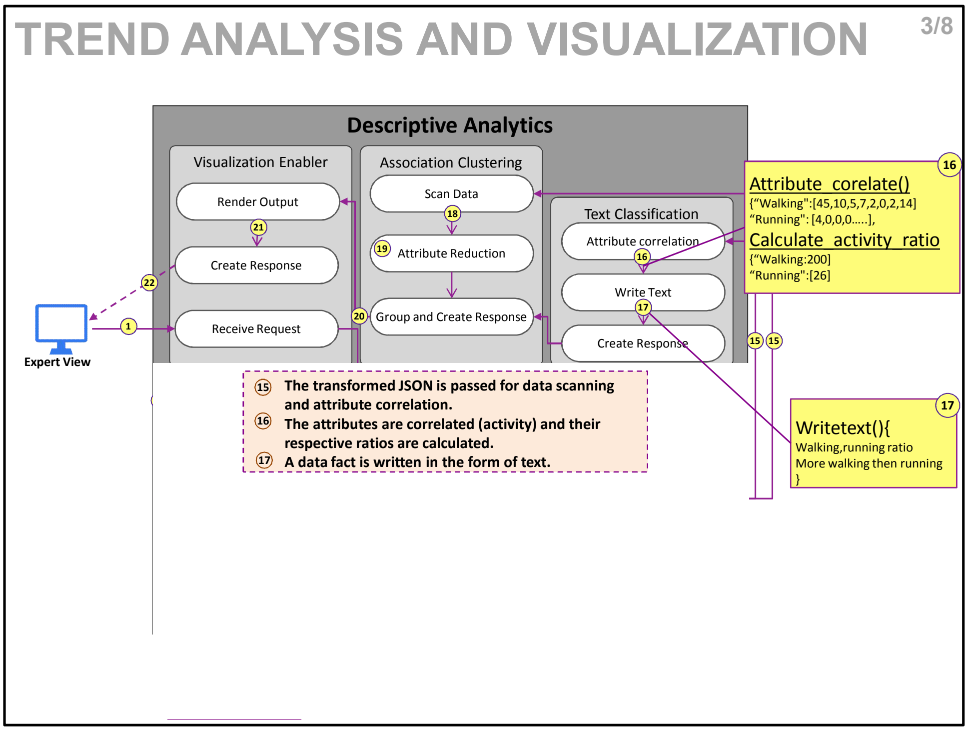

REQUEST DATA AND TRANSFORM
1/3
12 The data received is parsed according to the original query
Expert View
1
Query Creation Interface
Query attribute breakdown
Parameter Loader
Parse query
Query Library
Select
Data transformation
Map data to format
Parse Data
Transform data
Data Store Interface
Request Web service
Receive Result Set
Validate Data
Send Data
Parse_data()
Date,Duration,Activity,
12/1/2016,5m,walk
12/1/2016,120,sit
12/1/2016,5m,walk
12/1/2016,120,sit
Senddata(validated_object)
TAPACROSS
Data Curation Layer

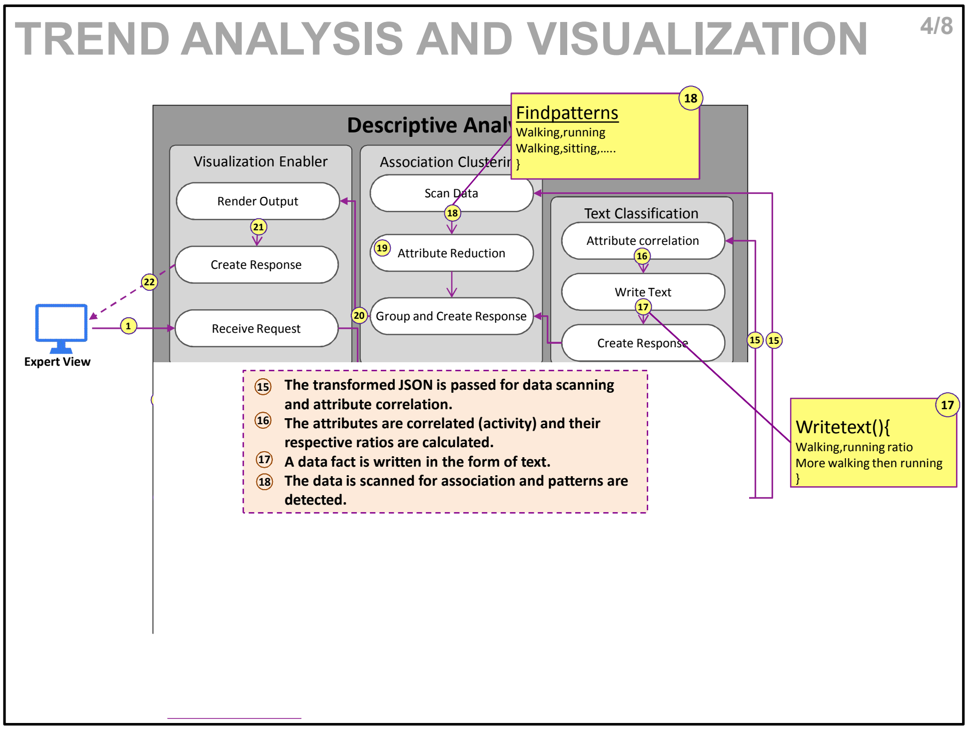

REQUEST DATA AND TRANSFORM
2/3
12 The data received is parsed according to the original query.
13 The complex object is then mapped to the json format for faster processing and graph plotting
Expert View
Map(object,json)
{"Walking":45
"Running":4
"Sitting":120
"Walking":10
"Sitting":600
Query Creation Interfa
transformation
Query attribute breakdo
Parameter Loader
Query Library
Select
Map data to format
Transform data
Data Store Interface
Request Web service
Receive Result Set
Validate Data
Send Data
Parse_data()
Date,Duration,Activity,
12/1/2016,5m,walk
12/1/2016,120,sit
12/1/2016,5m,walk
12/1/2016,120,sit
TAPACROSS
Data Curation Layer

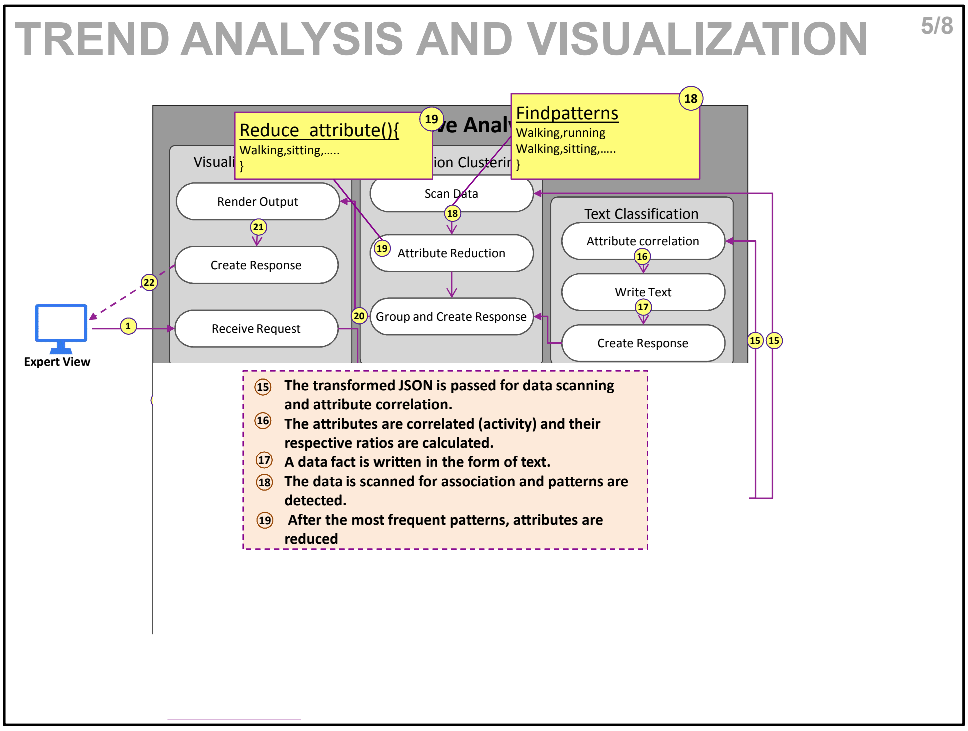

REQUEST DATA AND TRANSFORM
3/3
12 The data received is parsed according to the original query.
13 The complex object is then mapped to the json format for faster processing and graph plotting
14 The mapped JSON is then forwarded for trend analysis
Expert View
1
2
Query Creation Interfa
Query attribute breakdo
Parameter Loader
4
5
Query Library
6
Select
7
13
Map(object,json)
{"Walking":45
"Running":4
"Sitting":120
"Walking":10
"Sitting":600
transformation
Map data to format
13
14
Transform data
14
Send JSON()
{"Walking":[45,10,5,7,2,0,2,14]
"Running": [4,0,0,0.....],
"Sitting":[120,60,30,21]
}
12
Data Store Interface
Request Web service
Receive Result Set
10
Validate Data
11
Send Data
8a
8b
9a
9b
TAPACROSS
Data Curation Layer

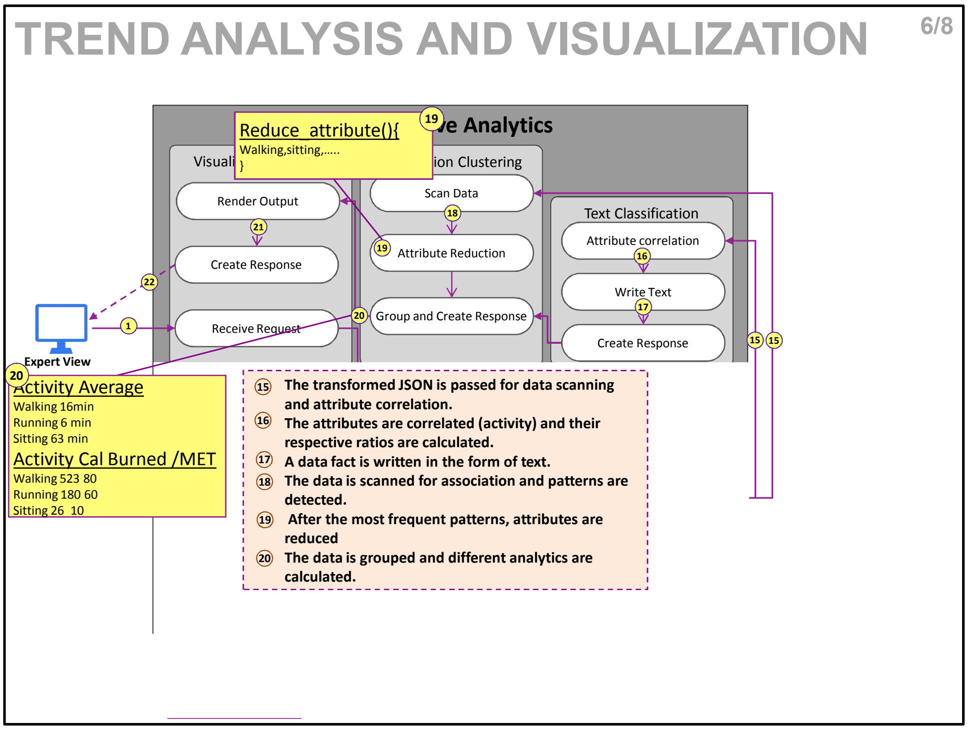

TREND ANALYSIS AND VISUALIZATION
1/8
Descriptive Analytics
Visualization Enabler
Render Output
21
Create Response
22
Receive Request
Expert View
1
Association Clustering
Scan Data
18
19 Attribute Reduction
20
Group and Create Response
Text Classification
Attribute correlation
16
Write Text
17
Create Response
15 15
15
passTransformedData(JSON)
15 The transformed JSON is passed for data scanning and attribute correlation.
14
Send JSON()
{"Walking":[45,10,5,7,2,0,2,14]
"Running": [4,0,0,0.....],
"Sitting":[120,60,30,21]
}

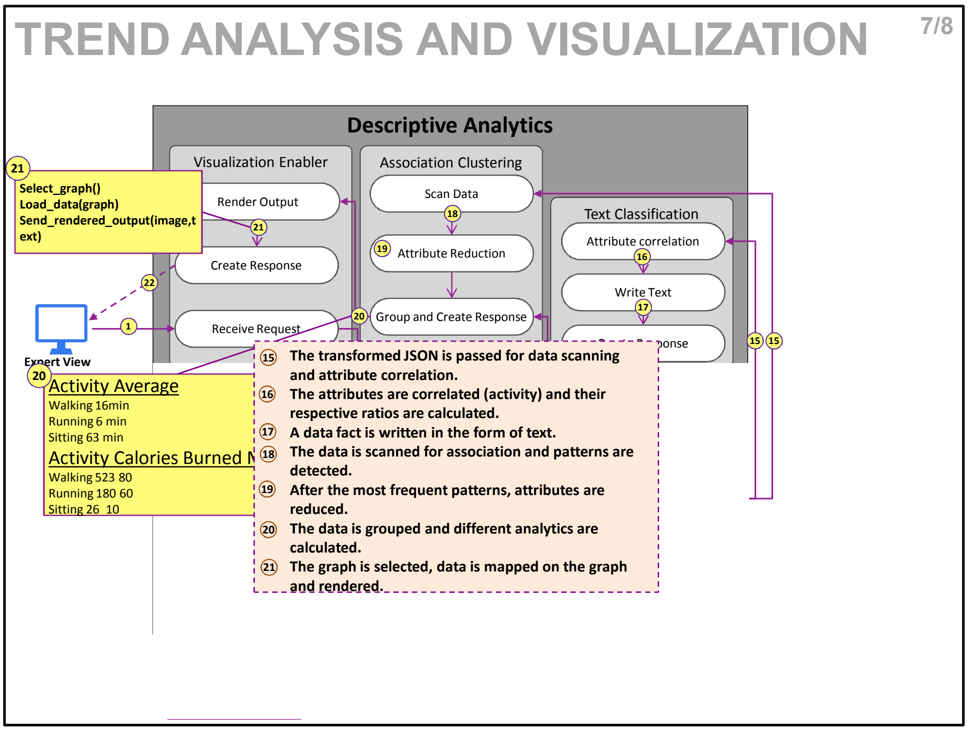

TREND ANALYSIS AND VISUALIZATION
2/8
Descriptive Analytics
Visualization Enabler
Render Output
Create Response
Receive Request
Association Clustering
Scan Data
Attribute Reduction
Group and Create Response
Text Classification
Attribute correlation
Write Text
Create Response
Expert View
Attribute_corelate()
{"Walking":[45,10,5,7,2,0,2,14]
"Running": [4,0,0,0.....],
Calculate_activity_ratio
{"Walking:200]
"Running":[26]
passTransformedData(JSON)
The transformed JSON is passed for data scanning and attribute correlation.
The attributes are correlated (activity) and their respective ratios are calculated.


TREND ANALYSIS AND VISUALIZATION
3/8
Descriptive Analytics
Visualization Enabler
Render Output
Create Response
Receive Request
Association Clustering
Scan Data
Attribute Reduction
Group and Create Response
Text Classification
Attribute correlation
Write Text
Create Response
Expert View
Attribute_corelate()
{"Walking":[45,10,5,7,2,0,2,14]
"Running": [4,0,0,0.....],
Calculate_activity_ratio
{"Walking:200]
"Running":[26]
Writetext(){
Walking,running ratio
More walking then running
}
The transformed JSON is passed for data scanning and attribute correlation.
The attributes are correlated (activity) and their respective ratios are calculated.
A data fact is written in the form of text.

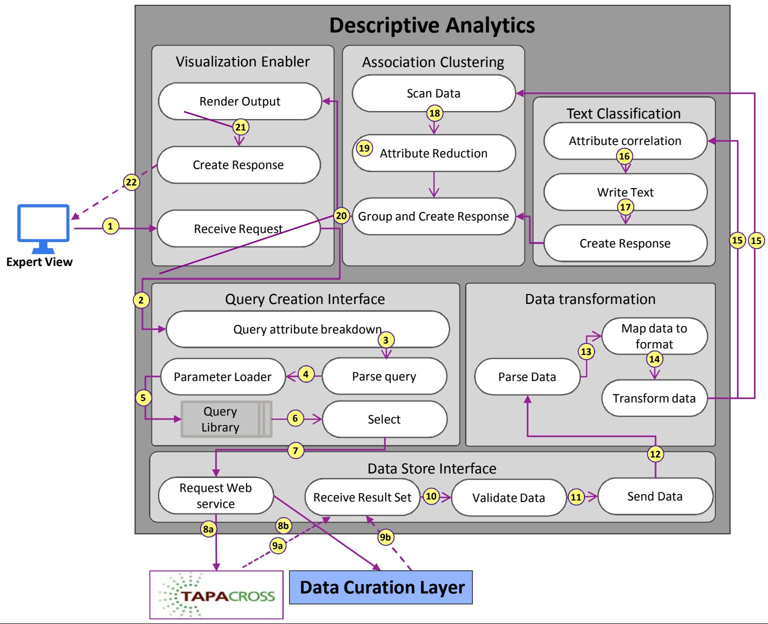

TREND ANALYSIS AND VISUALIZATION
4/8
Descriptive Anal
Findpatterns
Walking,running
Walking,sitting,.....
}
18
Visualization Enabler
Association Clusterir
Render Output
21
Create Response
22
Receive Request
1
Expert View
Scan Data
18
19 Attribute Reduction
20 Group and Create Response
Text Classification
Attribute correlation
16
Write Text
17
Create Response
15 15
15 The transformed JSON is passed for data scanning and attribute correlation.
16 The attributes are correlated (activity) and their respective ratios are calculated.
17 A data fact is written in the form of text.
18 The data is scanned for association and patterns are detected.
Writetext(){
Walking,running ratio
More walking then running
}
17


TREND ANALYSIS AND VISUALIZATION
5/8
Reduce_attribute(){
Walking,sitting,.....
}
19
ve Anal
Findpatterns
Walking,running
Walking,sitting,.....
}
18
Visuali
ion Clusterir
Render Output
21
Create Response
22
Receive Request
1
Expert View
Scan Data
18
19 Attribute Reduction
20 Group and Create Response
Text Classification
Attribute correlation
16
Write Text
17
Create Response
15 15
15 The transformed JSON is passed for data scanning and attribute correlation.
16 The attributes are correlated (activity) and their respective ratios are calculated.
17 A data fact is written in the form of text.
18 The data is scanned for association and patterns are detected.
19 After the most frequent patterns, attributes are reduced


TREND ANALYSIS AND VISUALIZATION
6/8
Reduce_attribute(){
Walking,sitting,.....
}
ve Analytics
Visuali
ion Clustering
Render Output
Create Response
Receive Request
Scan Data
Attribute Reduction
Group and Create Response
Text Classification
Attribute correlation
Write Text
Create Response
Expert View
Activity Average
Walking 16min
Running 6 min
Sitting 63 min
Activity Cal Burned /MET
Walking 523 80
Running 180 60
Sitting 26 10
(15) The transformed JSON is passed for data scanning and attribute correlation.
(16) The attributes are correlated (activity) and their respective ratios are calculated.
(17) A data fact is written in the form of text.
(18) The data is scanned for association and patterns are detected.
(19) After the most frequent patterns, attributes are reduced
(20) The data is grouped and different analytics are calculated.


TREND ANALYSIS AND VISUALIZATION
7/8
Descriptive Analytics
Visualization Enabler
Association Clustering
Select_graph()
Load_data(graph)
Send_rendered_output(image,text)
Render Output
Create Response
Receive Request
Scan Data
Attribute Reduction
Group and Create Response
Text Classification
Attribute correlation
Write Text
Expert View
Activity Average
Walking 16min
Running 6 min
Sitting 63 min
Activity Calories Burned
Walking 523 80
Running 180 60
Sitting 26 10
(15) The transformed JSON is passed for data scanning and attribute correlation.
(16) The attributes are correlated (activity) and their respective ratios are calculated.
(17) A data fact is written in the form of text.
(18) The data is scanned for association and patterns are detected.
(19) After the most frequent patterns, attributes are reduced.
(20) The data is grouped and different analytics are calculated.
(21) The graph is selected, data is mapped on the graph and rendered.

TREND ANALYSIS AND VISUALIZATION

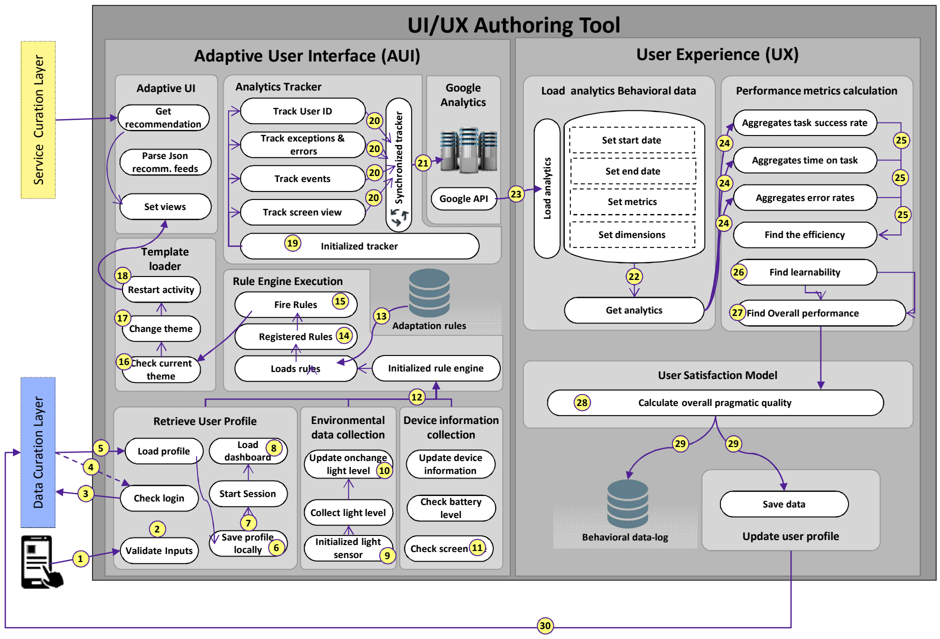


TREND ANALYSIS AND VISUALIZATION

STENCILS for SL Micro-Animation

UI/UX Authoring Tool

UI/UX Detailed Architecture

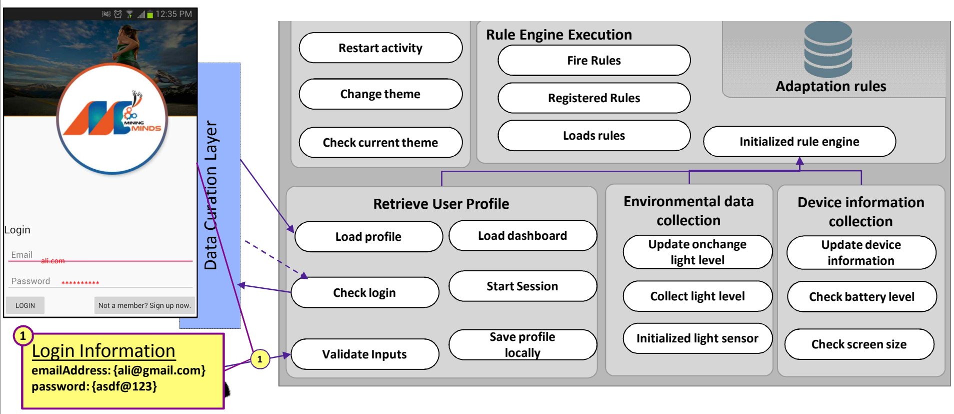


Scenario 1: Adaptive User Interface

Desc. : **User information**, **environmental** information, and device information are acquired for **rule engine** to **adapt the UI** accordingly.

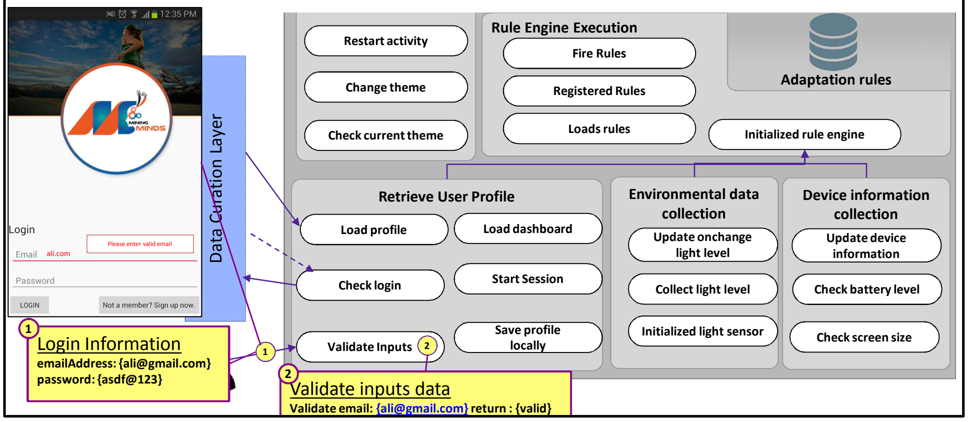
RETRIEVE USER PROFILE

1 First user input the login credential information



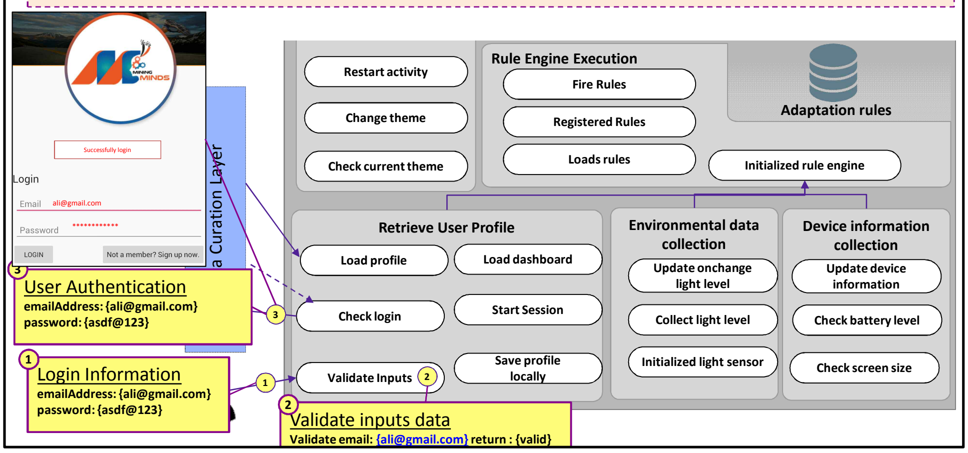
RETRIEVE USER PROFILE

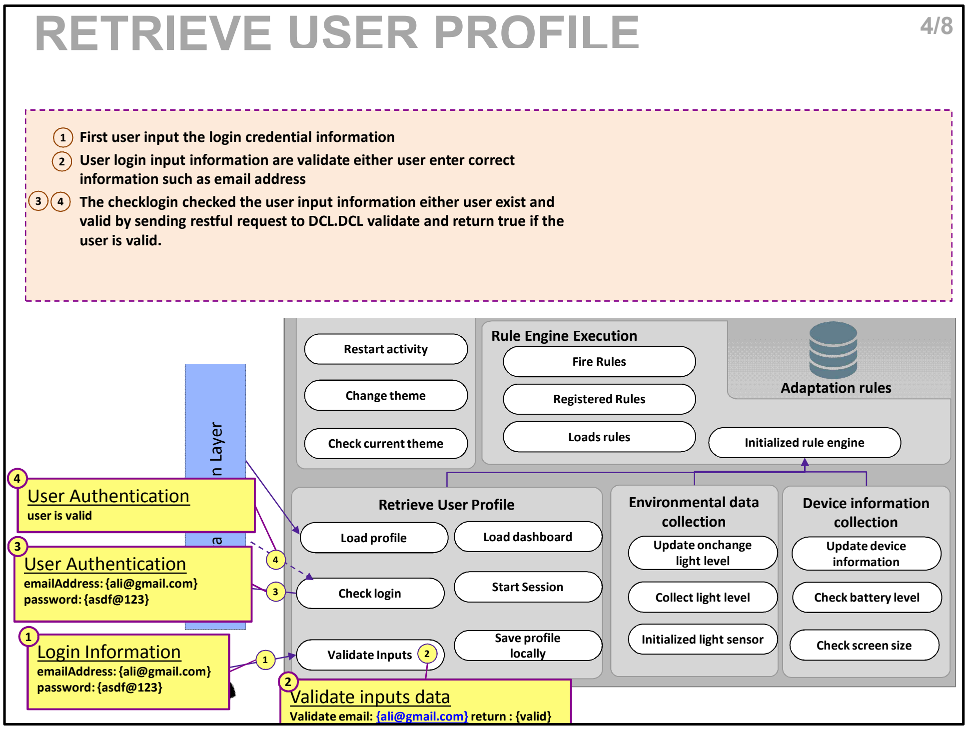
1. First user input the login credential information
2. User login input information are validate either user enter correct information such as email address

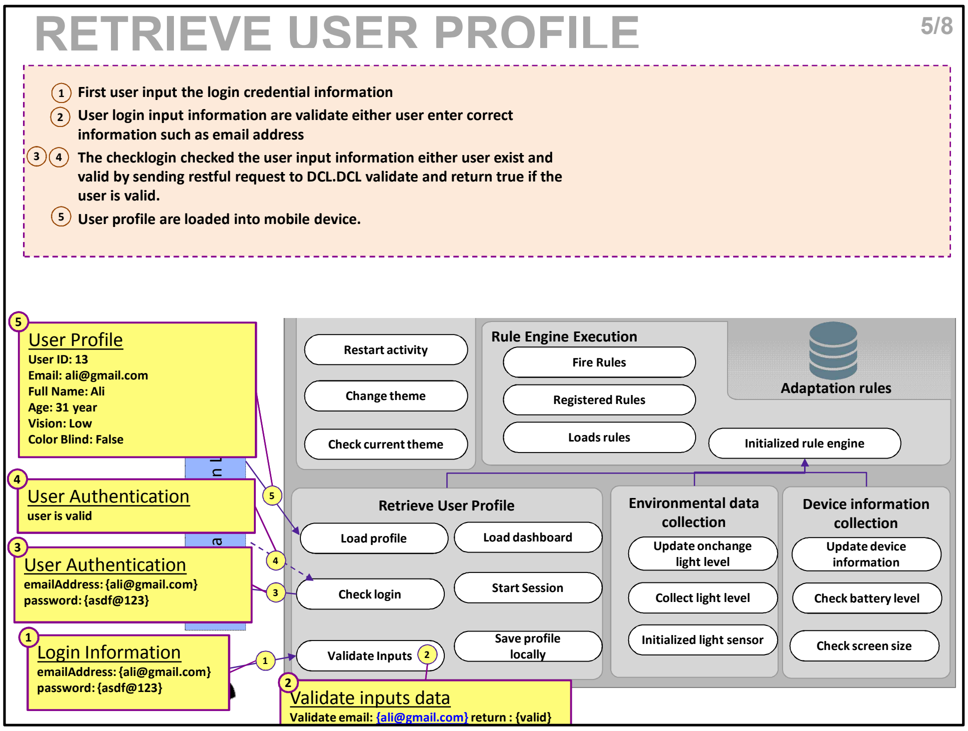


RETRIEVE USER PROFILE

1. First user input the login credential information
2. User login input information are validate either user enter correct information such as email address
3. The checklogin- checked the user input information either user exist and valid by sending restful request to DCL.DCL validate and return true if the user is valid.




RETRIEVE USER PROFILE
4/8
1 First user input the login credential information
2 User login input information are validate either user enter correct information such as email address
3 4 The checklogin checked the user input information either user exist and valid by sending restful request to DCL.DCL validate and return true if the user is valid.
Restart activity
Change theme
Check current theme
Rule Engine Execution
Fire Rules
Registered Rules
Loads rules
Initialized rule engine
Adaptation rules
Retrieve User Profile
Load profile
Load dashboard
Check login
Start Session
Validate Inputs
Save profile locally
Environmental data collection
Update onchange light level
Collect light level
Initialized light sensor
Device information collection
Update device information
Check battery level
Check screen size
n Layer
4 User Authentication
user is valid
3 User Authentication
emailAddress: {ali@gmail.com}
password: {asdf@123}
1 Login Information
emailAddress: {ali@gmail.com}
password: {asdf@123}
2 Validate inputs data
Validate email: {ali@gmail.com} return : {valid}

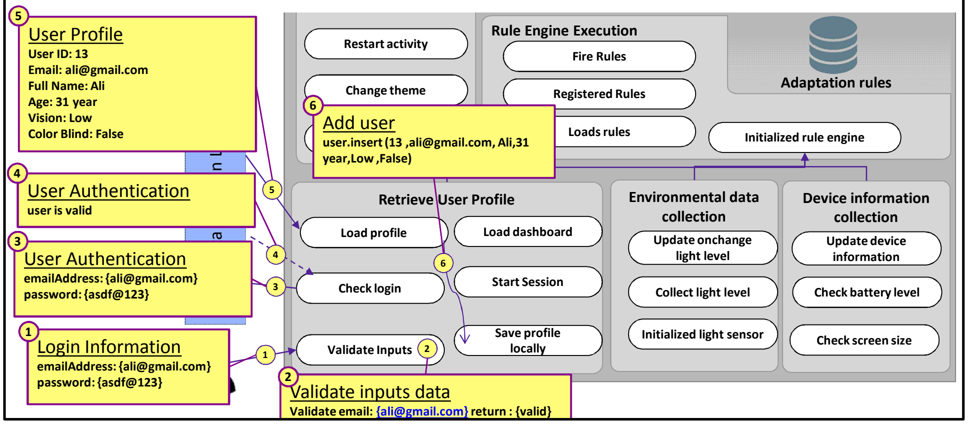

RETRIEVE USER PROFILE
5/8
1 First user input the login credential information
2 User login input information are validate either user enter correct information such as email address
3 4 The checklogin checked the user input information either user exist and valid by sending restful request to DCL.DCL validate and return true if the user is valid.
5 User profile are loaded into mobile device.
5 User Profile
User ID: 13
Email: ali@gmail.com
Full Name: Ali
Age: 31 year
Vision: Low
Color Blind: False
Restart activity
Change theme
Check current theme
Rule Engine Execution
Fire Rules
Registered Rules
Loads rules
Initialized rule engine
Adaptation rules
Retrieve User Profile
Load profile
Load dashboard
Check login
Start Session
Validate Inputs
Save profile locally
Environmental data collection
Update onchange light level
Collect light level
Initialized light sensor
Device information collection
Update device information
Check battery level
Check screen size
4 User Authentication
user is valid
3 User Authentication
emailAddress: {ali@gmail.com}
password: {asdf@123}
1 Login Information
emailAddress: {ali@gmail.com}
password: {asdf@123}
2 Validate inputs data
Validate email: {ali@gmail.com} return : {valid}

RETRIEVE USER PROFILE

1. First user input the login credential information
2. User login input information are validate either user enter correct information such as email address

3 4 The checklogin checked the user input information either user exist and valid by sending restful request to DCL.DCL validate and return true if the user is valid.

5. User profile are loaded into mobile device.
6. Then user profile data are locally save in mobile device

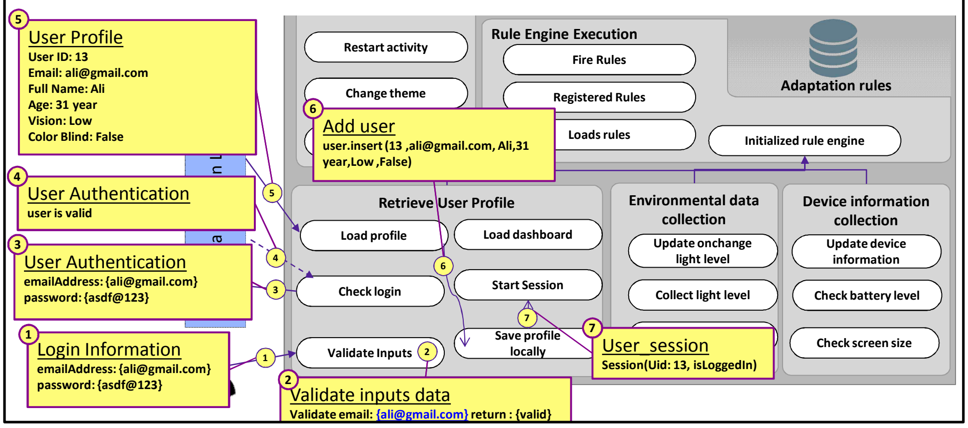


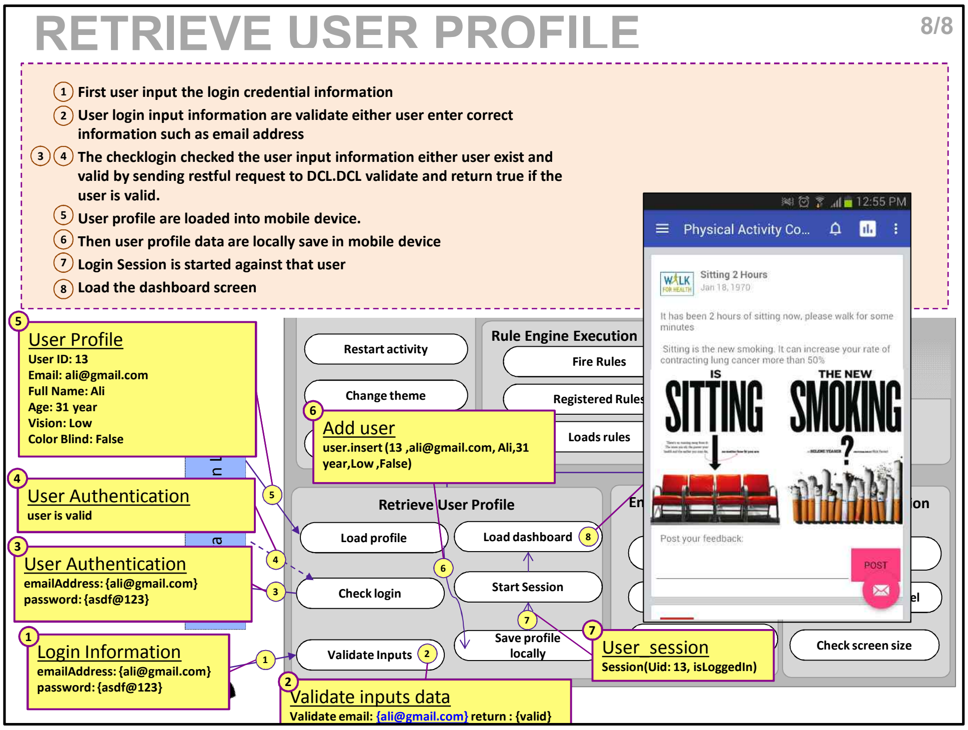
RETRIEVE USER PROFILE

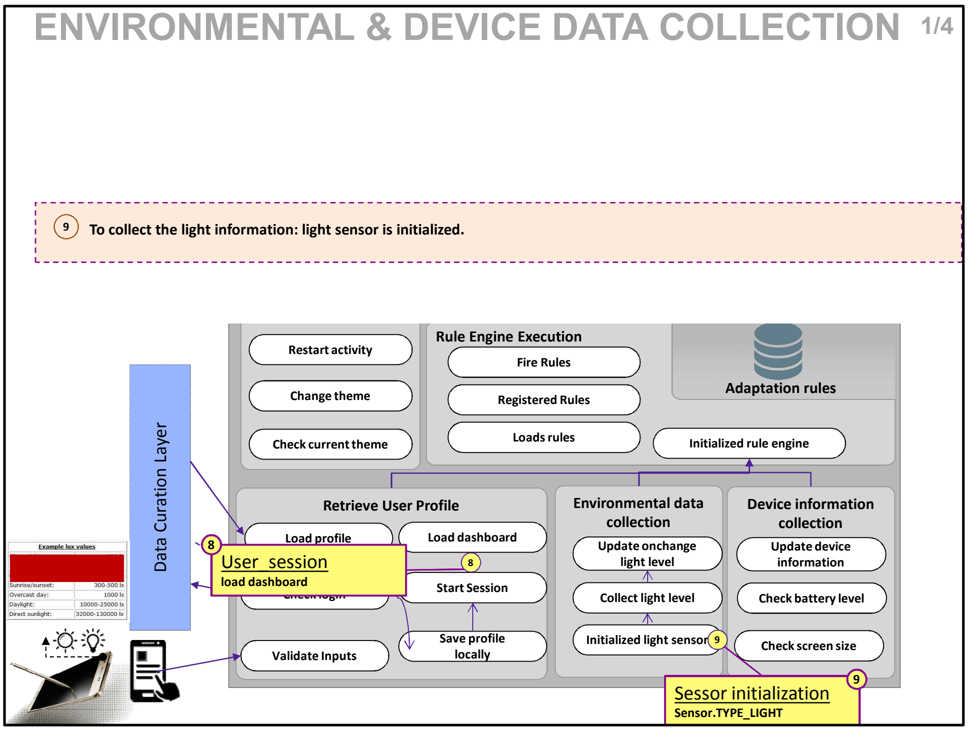
1. First user input the login credential information
2. User login input information are validate either user enter correct information such as email address

3 4 The checklogin checked the user input information either user exist and valid by sending restful request to DCL.DCL validate and return true if the user is valid.

5. User profile are loaded into mobile device.
6. Then user profile data are locally save in mobile device
7. Login Session is started against that user



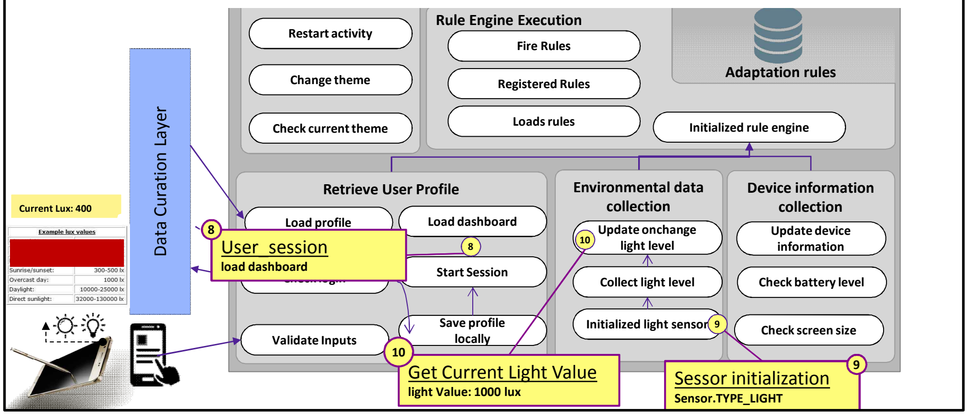

RETRIEVE USER PROFILE
8/8
1 First user input the login credential information
2 User login input information are validate either user enter correct information such as email address
3 4 The checklogin checked the user input information either user exist and valid by sending restful request to DCL.DCL validate and return true if the user is valid.
5 User profile are loaded into mobile device.
6 Then user profile data are locally save in mobile device
7 Login Session is started against that user
8 Load the dashboard screen
5 User Profile
User ID: 13
Email: ali@gmail.com
Full Name: Ali
Age: 31 year
Vision: Low
Color Blind: False
4 User Authentication
user is valid
3 User Authentication
emailAddress: {ali@gmail.com}
password: {asdf@123}
1 Login Information
emailAddress: {ali@gmail.com}
password: {asdf@123}
Restart activity
Change theme
6 Add user
user.insert (13 ,ali@gmail.com, Ali,31 year,Low ,False)
Rule Engine Execution
Fire Rules
Registered Rules
Loads rules
Retrieve User Profile
Load profile
Load dashboard 8
Check login
Start Session
Validate Inputs 2
Save profile locally
7 User_session
Session(Uid: 13, isLoggedIn)
2 Validate inputs data
Validate email: {ali@gmail.com} return : {valid}
Check screen size
12:55 PM
Physical Activity Co...
Sitting 2 Hours
Jan 18, 1970
It has been 2 hours of sitting now, please walk for some minutes
Sitting is the new smoking. It can increase your rate of contracting lung cancer more than 50%
IS
SITTING
THE NEW
SMOKING
?
Post your feedback
POST


ENVIRONMENTAL & DEVICE DATA COLLECTION
1/4
9 To collect the light information: light sensor is initialized.
Restart activity
Change theme
Check current theme
Rule Engine Execution
Fire Rules
Registered Rules
Loads rules
Adaptation rules
Initialized rule engine
Data Curation Layer
Retrieve User Profile
Load profile
Load dashboard
8 User_session
load dashboard
Start Session
Validate Inputs
Save profile locally
Environmental data collection
Update onchange light level
Collect light level
Initialized light sensor 9
Device information collection
Update device information
Check battery level
Check screen size
9 Sessor initialization
Sensor.TYPE_LIGHT
Example lux values
Sunrise/sunset:
Overcast day:
Daylight:
Direct sunlight:

ENVIRONMENTAL & DEVICE DATA COLLECTION 2/4

(9) To collect the light information: light sensor is initialized.

(10) Light sensor start collecting the light information in the form of light intensity(Lx)

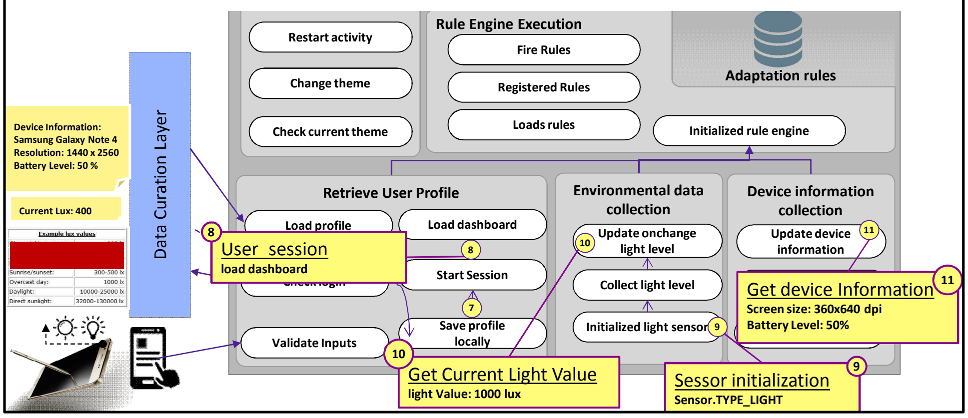


ENVIRONMENTAL & DEVICE DATA COLLECTION 3/4

(9) To collect the light information: light sensor is initialized.

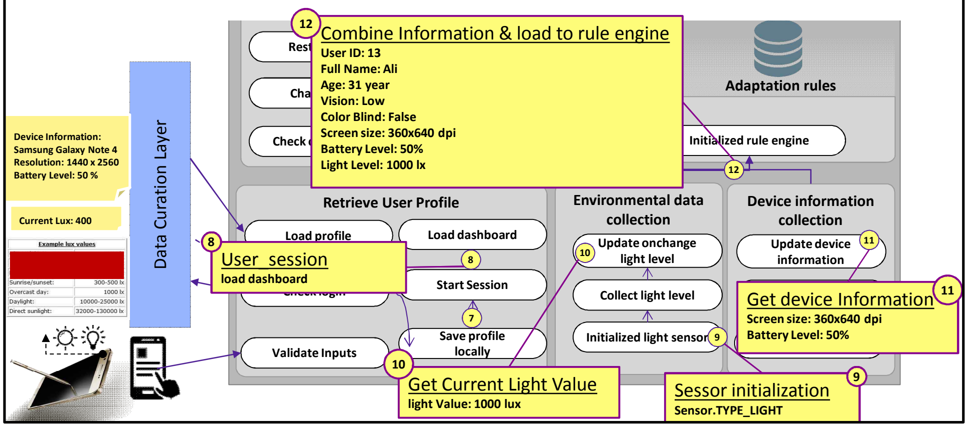
(10) Light sensor start collecting the light information in the form of light intensity(Lx)

(11) It is responsible to identify several device characteristics such as Screen Size , Battery Level

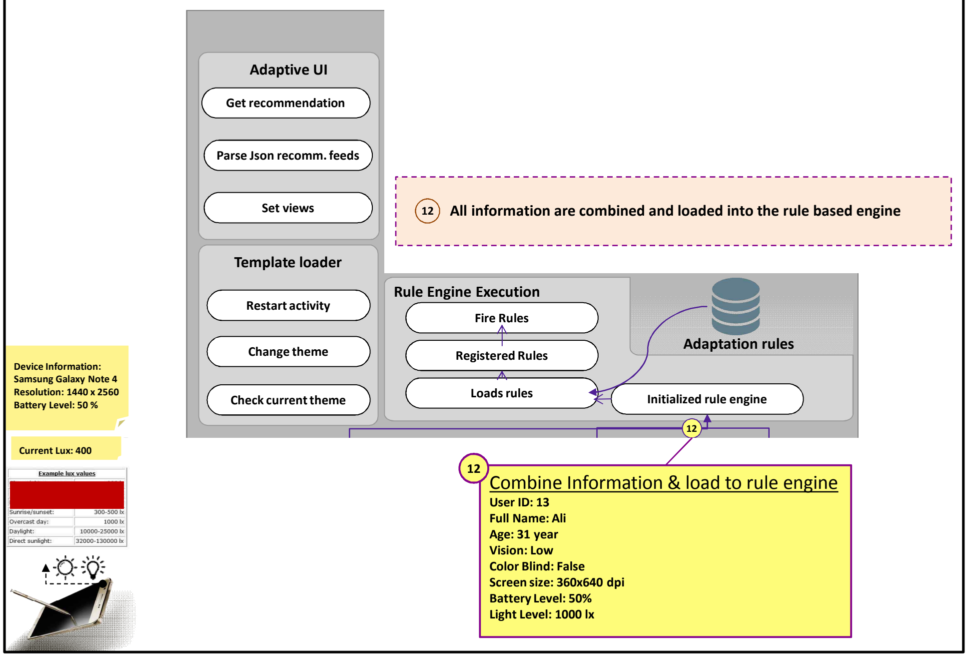


ENVIRONMENTAL & DEVICE DATA COLLECTION 4/4

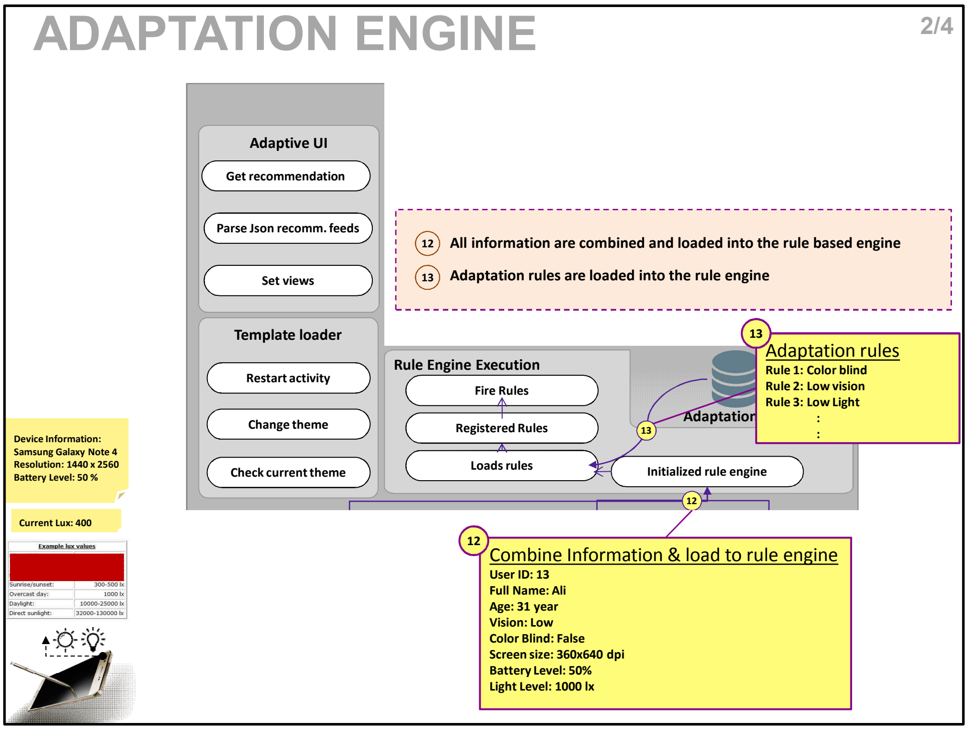
9. To collect the light information: light sensor is initialized.
10. Light sensor start collecting the light information in the form of light intensity(Lx)
11. It is responsible to identify several device characteristics such as Screen Size , Battery Level
12. All information are combined and loaded into the rule based engine.

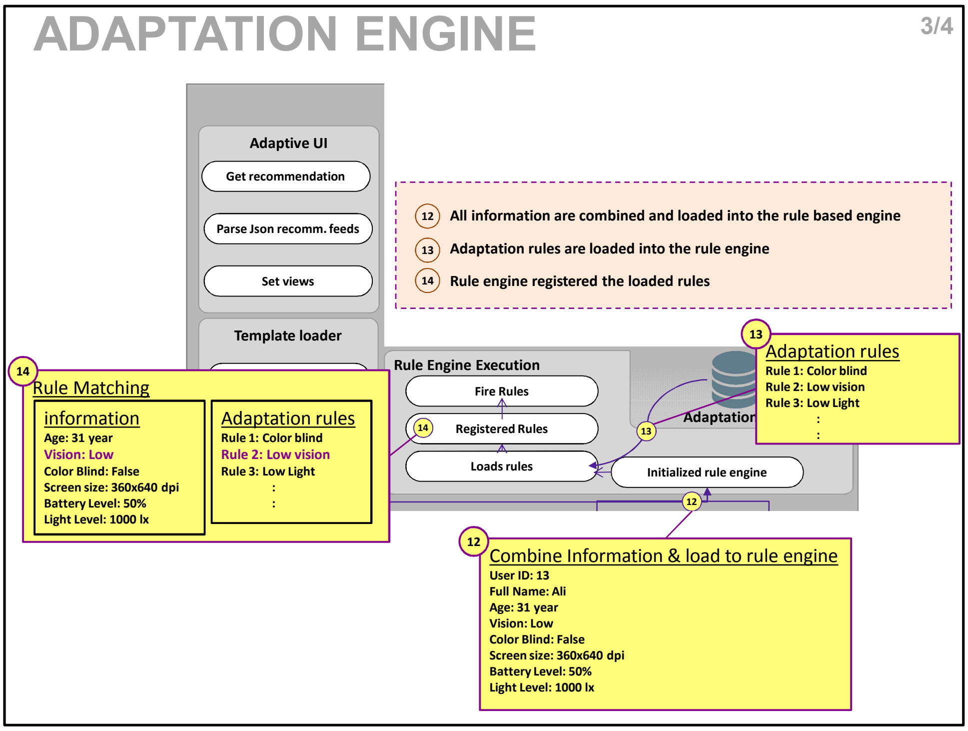


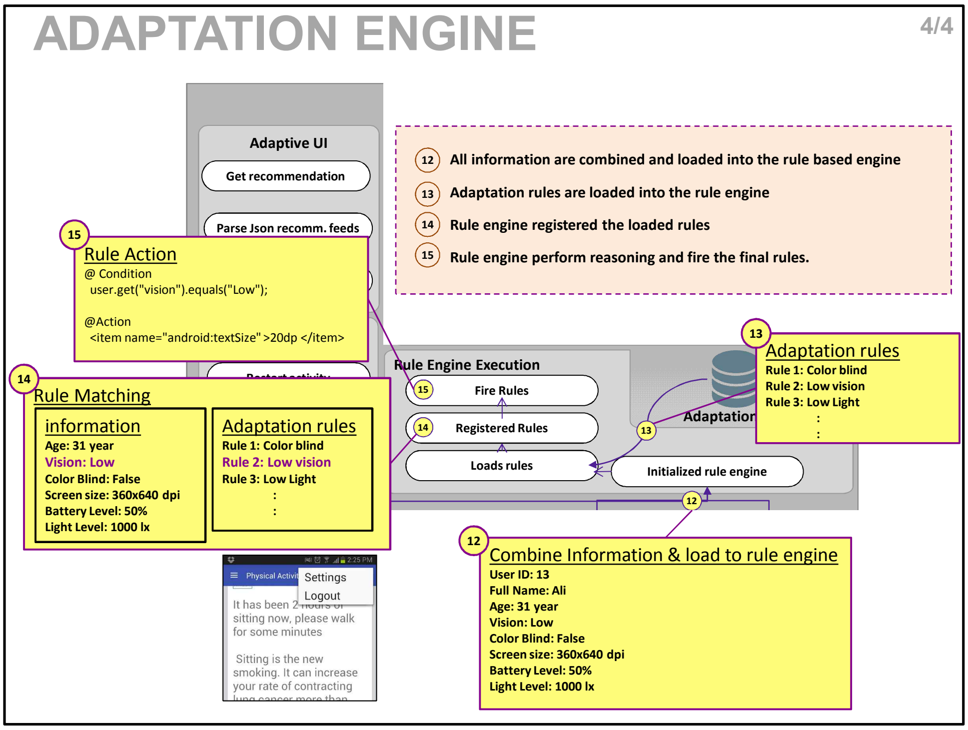
ADAPTATION ENGINE 1/4

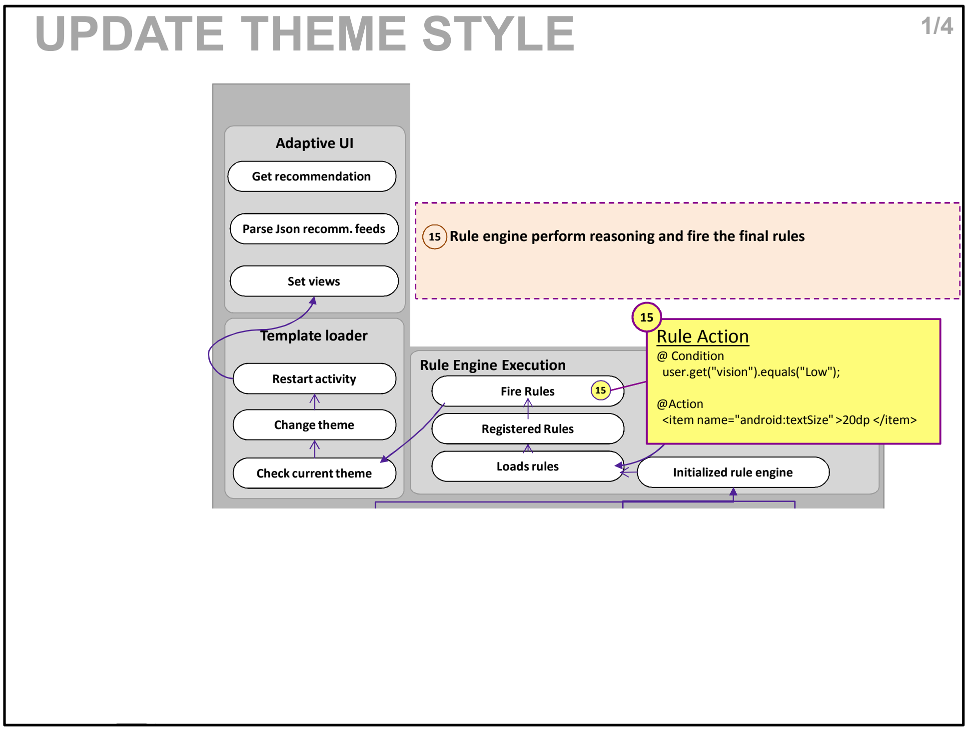


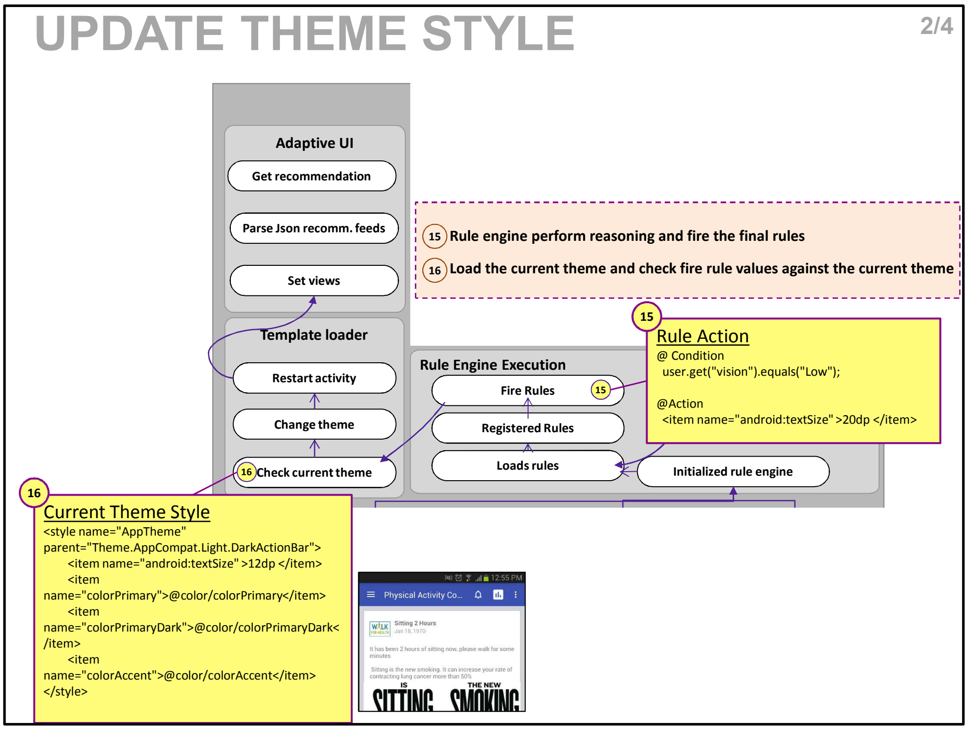
12. All information are combined and loaded into the rule based engine

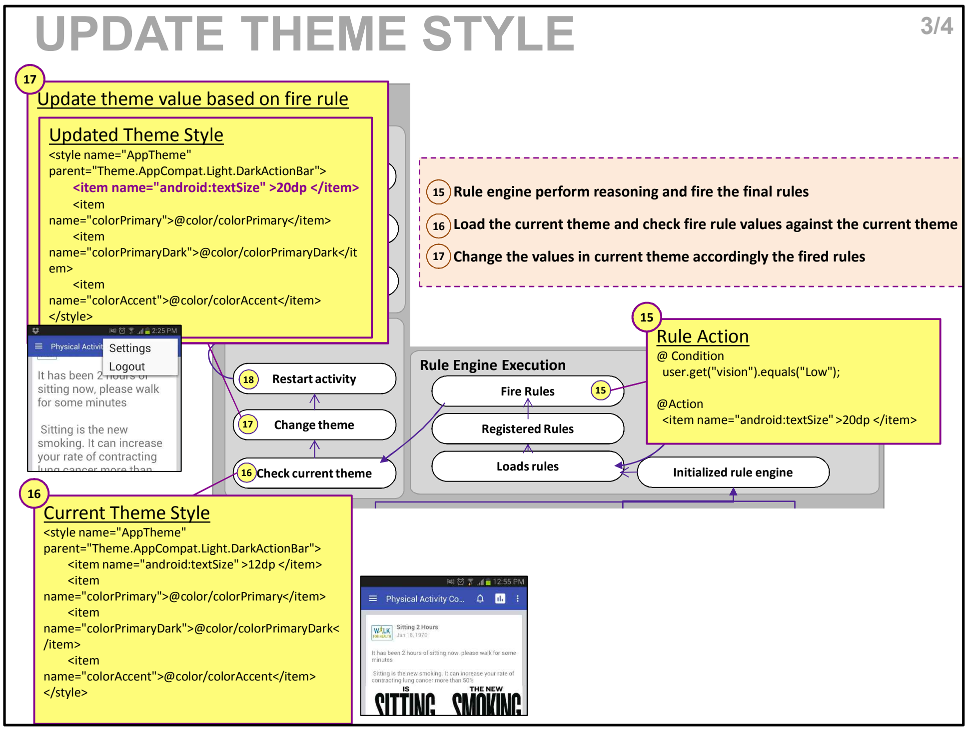

ADAPTATION ENGINE
2/4
Adaptive UI
Get recommendation
Parse Json recomm. feeds
Set views
Template loader
Restart activity
Change theme
Check current theme
12 All information are combined and loaded into the rule based engine
13 Adaptation rules are loaded into the rule engine
Rule Engine Execution
Fire Rules
Registered Rules
Loads rules
Initialized rule engine
Adaptation
Adaptation rules
Rule 1: Color blind
Rule 2: Low vision
Rule 3: Low Light
Device Information:
Samsung Galaxy Note 4
Resolution: 1440 x 2560
Battery Level: 50 %
Current Lux: 400
Example lux values
Sunrise/sunset: 300-500 lx
Overcast day: 1000 lx
Daylight: 10000-25000 lx
Direct sunlight: 32000-130000 lx
Combine Information & load to rule engine
User ID: 13
Full Name: Ali
Age: 31 year
Vision: Low
Color Blind: False
Screen size: 360x640 dpi
Battery Level: 50%
Light Level: 1000 lx


ADAPTATION ENGINE
3/4
Adaptive UI
Get recommendation
Parse Json recomm. feeds
Set views
Template loader
12 All information are combined and loaded into the rule based engine
13 Adaptation rules are loaded into the rule engine
14 Rule engine registered the loaded rules
Rule Matching
information
Age: 31 year
Vision: Low
Color Blind: False
Screen size: 360x640 dpi
Battery Level: 50%
Light Level: 1000 lx
Adaptation rules
Rule 1: Color blind
Rule 2: Low vision
Rule 3: Low Light
Rule Engine Execution
Fire Rules
Registered Rules
Loads rules
Initialized rule engine
Adaptation
Adaptation rules
Rule 1: Color blind
Rule 2: Low vision
Rule 3: Low Light
Combine Information & load to rule engine
User ID: 13
Full Name: Ali
Age: 31 year
Vision: Low
Color Blind: False
Screen size: 360x640 dpi
Battery Level: 50%
Light Level: 1000 lx


ADAPTATION ENGINE
4/4
Adaptive UI
Get recommendation
Parse Json recomm. feeds
12 All information are combined and loaded into the rule based engine
13 Adaptation rules are loaded into the rule engine
14 Rule engine registered the loaded rules
15 Rule engine perform reasoning and fire the final rules.
15 Rule Action
@ Condition
user.get("vision").equals("Low");
@Action
<item name="android:textSize" >20dp </item>
14 Rule Matching
information
Age: 31 year
Vision: Low
Color Blind: False
Screen size: 360x640 dpi
Battery Level: 50%
Light Level: 1000 lx
Adaptation rules
Rule 1: Color blind
Rule 2: Low vision
Rule 3: Low Light
Rule Engine Execution
Fire Rules
Registered Rules
Loads rules
Initialized rule engine
Adaptation
13 Adaptation rules
Rule 1: Color blind
Rule 2: Low vision
Rule 3: Low Light
12 Combine Information & load to rule engine
User ID: 13
Full Name: Ali
Age: 31 year
Vision: Low
Color Blind: False
Screen size: 360x640 dpi
Battery Level: 50%
Light Level: 1000 lx
Settings
Logout
sitting now, please walk
for some minutes
Sitting is the new
smoking. It can increase
your rate of contracting

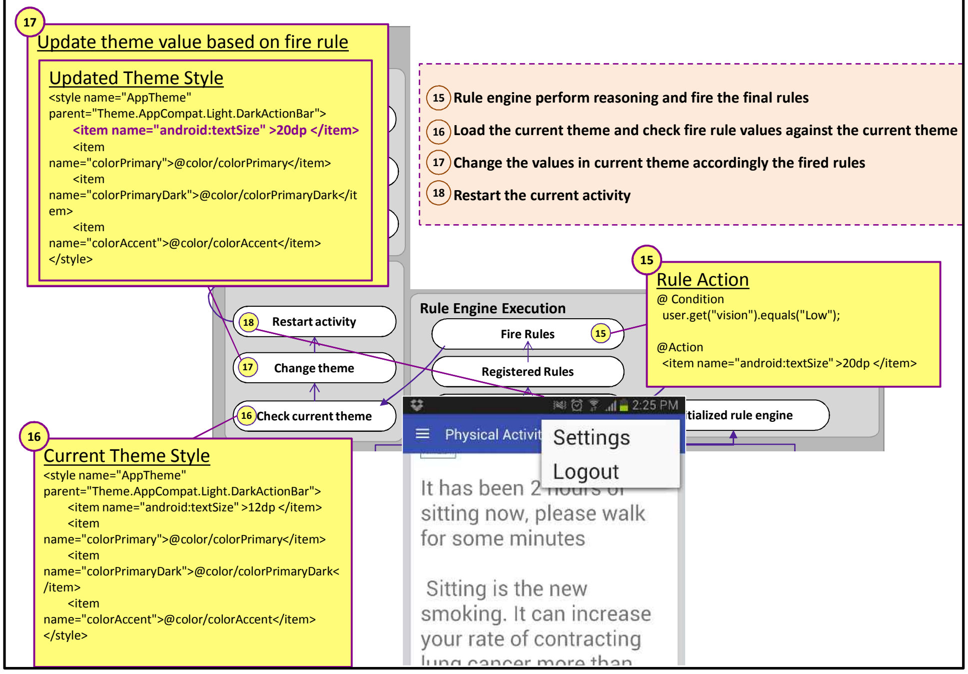

UPDATE THEME STYLE
1/4
Adaptive UI
Get recommendation
Parse Json recomm. feeds
Set views
Template loader
Restart activity
Change theme
Check current theme
15 Rule engine perform reasoning and fire the final rules
15 Rule Action
@ Condition
user.get("vision").equals("Low");
@Action
<item name="android:textSize" >20dp </item>
Rule Engine Execution
Fire Rules
Registered Rules
Loads rules
Initialized rule engine


UPDATE THEME STYLE
2/4
Adaptive UI
Get recommendation
Parse Json recomm. feeds
Set views
Template loader
Restart activity
Change theme
16 Check current theme
15 Rule engine perform reasoning and fire the final rules
16 Load the current theme and check fire rule values against the current theme
15
Rule Action
@ Condition
user.get("vision").equals("Low");
@Action
<item name="android:textSize" >20dp </item>
Rule Engine Execution
Fire Rules 15
Registered Rules
Loads rules
Initialized rule engine
16
Current Theme Style
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
<item name="android:textSize" >12dp </item>
<item name="colorPrimary">@color/colorPrimary</item>
<item name="colorPrimaryDark">@color/colorPrimaryDark</item>
<item name="colorAccent">@color/colorAccent</item>
</style>
Physical Activity Co...
Sitting 2 Hours


UPDATE THEME STYLE
3/4
17
Update theme value based on fire rule
Updated Theme Style
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
<item name="android:textSize" >20dp </item>
<item name="colorPrimary">@color/colorPrimary</item>
<item name="colorPrimaryDark">@color/colorPrimaryDark</item>
<item name="colorAccent">@color/colorAccent</item>
</style>
Settings
Logout
sitting now, please walk for some minutes
Sitting is the new smoking. It can increase your rate of contracting
15 Rule engine perform reasoning and fire the final rules
16 Load the current theme and check fire rule values against the current theme
17 Change the values in current theme accordingly the fired rules
15
Rule Action
@ Condition
user.get("vision").equals("Low");
@Action
<item name="android:textSize" >20dp </item>
Rule Engine Execution
Fire Rules 15
Registered Rules
Loads rules
Initialized rule engine
18 Restart activity
17 Change theme
16 Check current theme
16
Current Theme Style
<style name="AppTheme" parent="Theme.AppCompat.Light.DarkActionBar">
<item name="android:textSize" >12dp </item>
<item name="colorPrimary">@color/colorPrimary</item>
<item name="colorPrimaryDark">@color/colorPrimaryDark</item>
<item name="colorAccent">@color/colorAccent</item>
</style>
Physical Activity Co...
Sitting 2 Hours

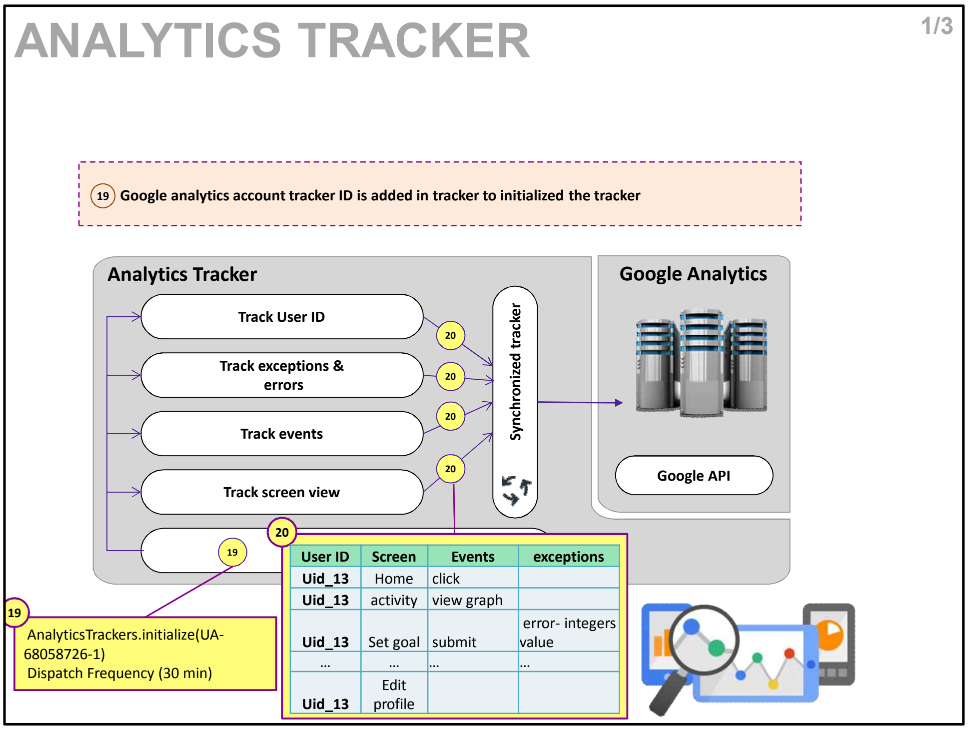
UPDATE THEME STYLE

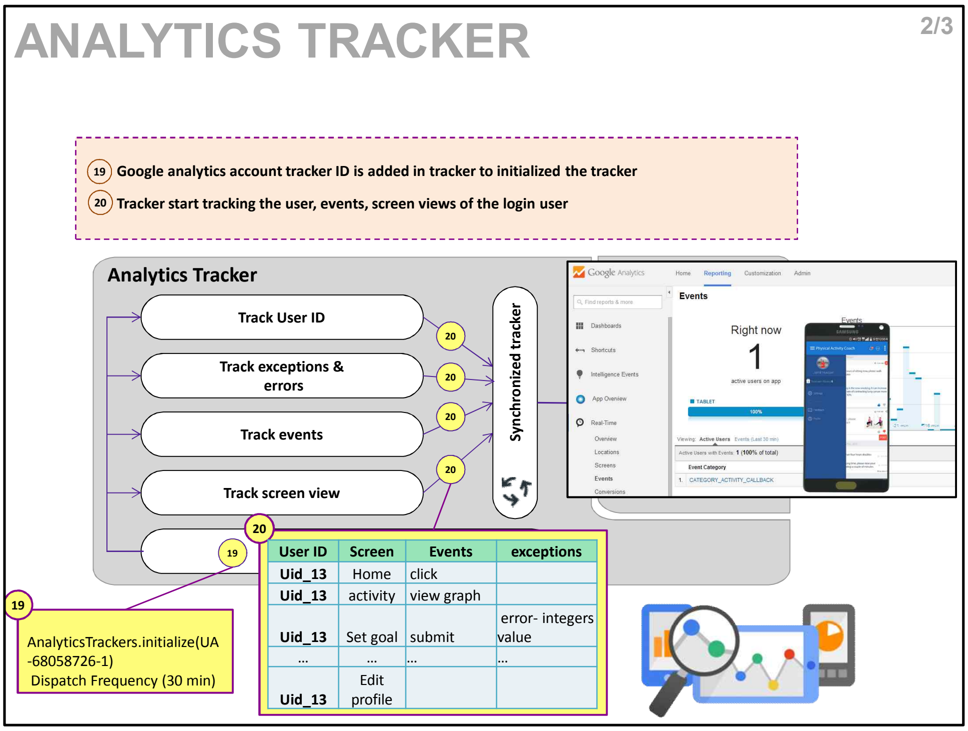
4/4



Scenario 2: User Experience evaluation

Desc.: **User experience evaluation** based on user **interaction** data that collects by **analytics tracker**, to finding the different **usability metric**. Finally the **user satisfaction** is calculating based on those metrics.


ANALYTICS TRACKER
1/3
19 Google analytics account tracker ID is added in tracker to initialized the tracker
Analytics Tracker
Track User ID
Track exceptions & errors
Track events
Track screen view
Synchronized tracker
Google Analytics
Google API
20
19
AnalyticsTrackers.initialize(UA-68058726-1)
Dispatch Frequency (30 min)
User ID | Screen | Events | exceptions
Uid_13 | Home | click |
Uid_13 | activity | view graph |
Uid_13 | Set goal | submit | error- integers value
… | … | … | …
Uid_13 | Edit profile | |


ANALYTICS TRACKER
2/3
19 Google analytics account tracker ID is added in tracker to initialized the tracker
20 Tracker start tracking the user, events, screen views of the login user
Analytics Tracker
Track User ID
Track exceptions & errors
Track events
Track screen view
Synchronized tracker
Events
Right now
1
active users on app
20
19
AnalyticsTrackers.initialize(UA-68058726-1)
Dispatch Frequency (30 min)
User ID | Screen | Events | exceptions
Uid_13 | Home | click |
Uid_13 | activity | view graph |
Uid_13 | Set goal | submit | error- integers value
… | … | … | …
Uid_13 | Edit profile | |

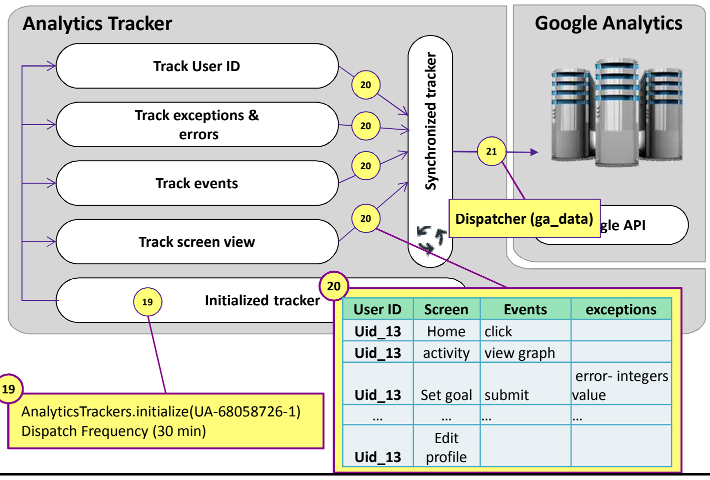
ANALYTICS TRACKER

3/3

(19) Google analytics account tracker ID is added in tracker to initialized the tracker

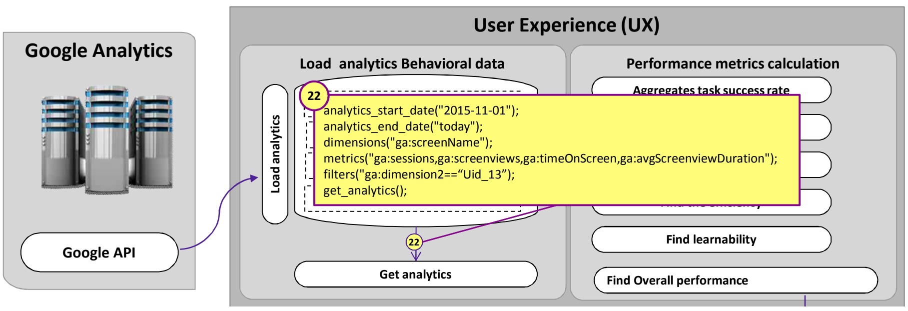
(20) Tracker start tracking the user, events, screen views of the login user

(21) Tracker synchronized the data by calling the dispatch function with google analytics server.

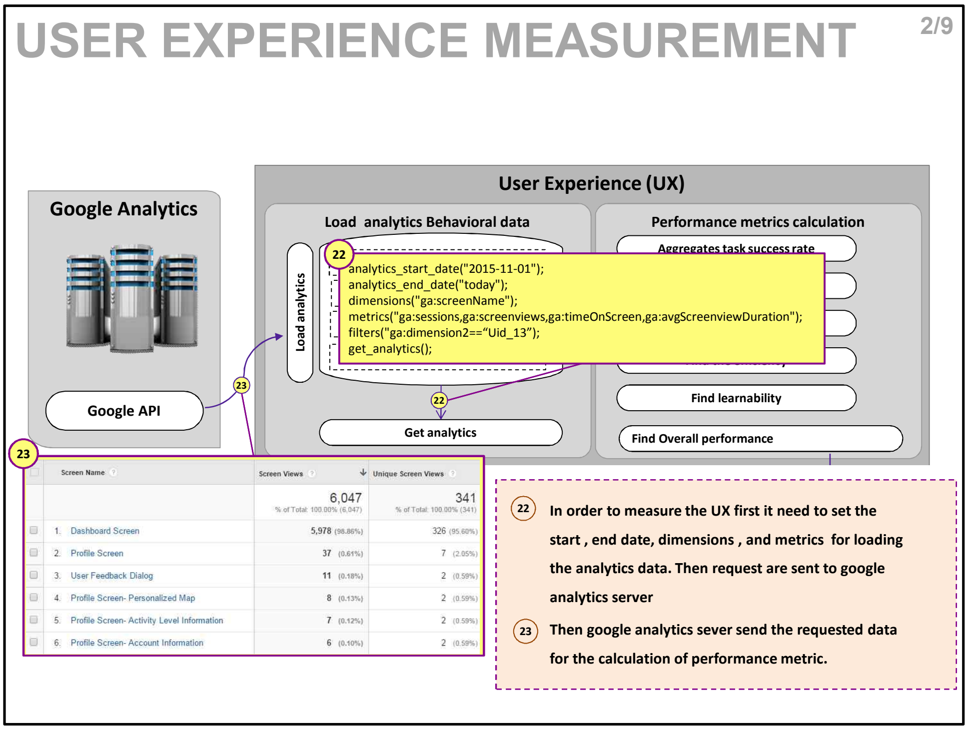


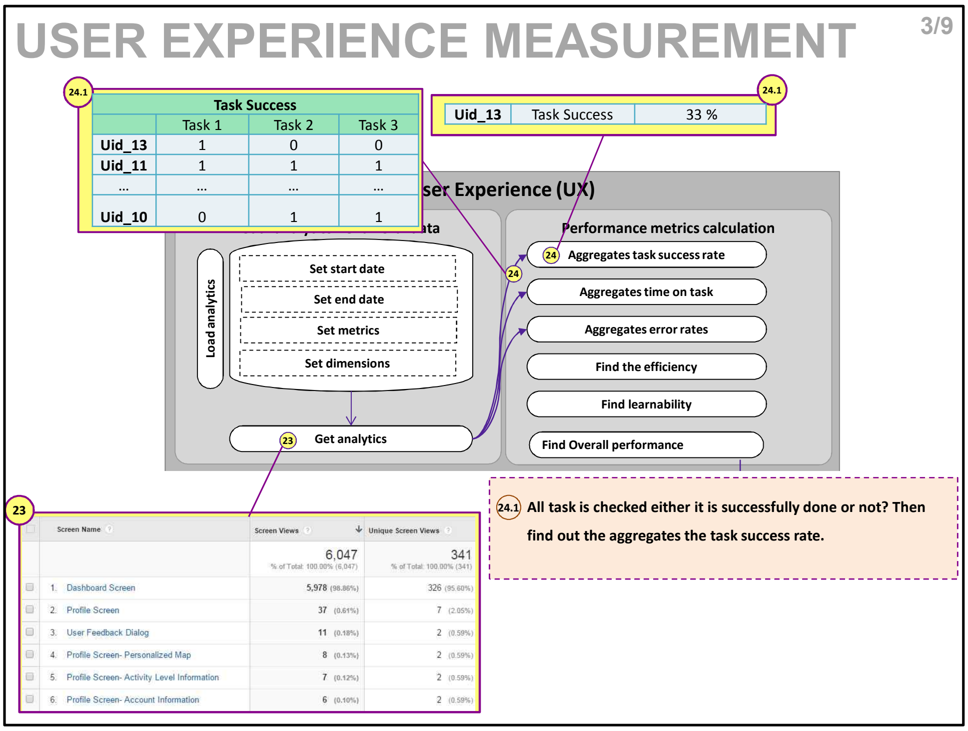
USER EXPERIENCE MEASUREMENT

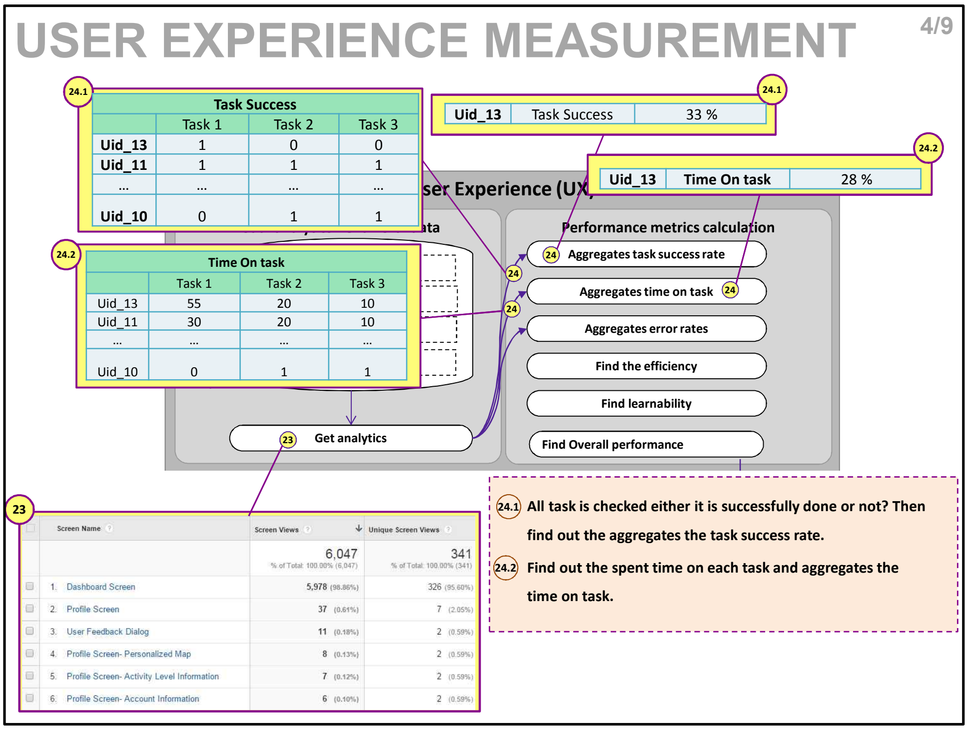
1/9

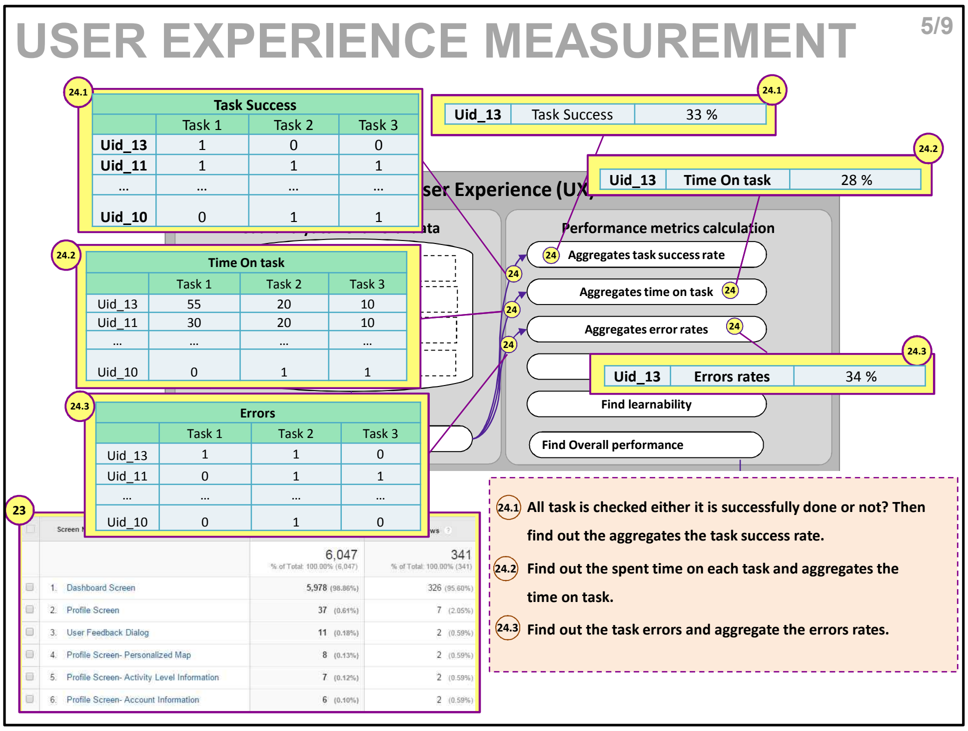


(22) In order to measure the UX first it need to set the start , end date, dimensions , and metrics for loading the analytics data. Then request are sent to google analytics server

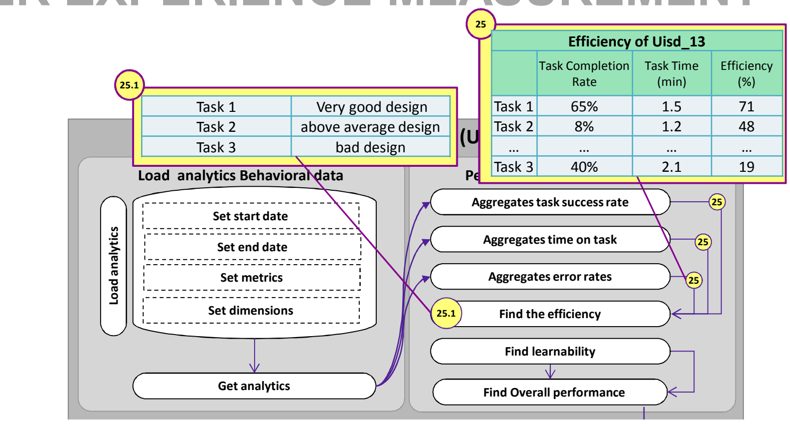

USER EXPERIENCE MEASUREMENT
2/9
User Experience (UX)
Google Analytics
Google API
Load analytics Behavioral data
Load analytics
analytics_start_date("2015-11-01");
analytics_end_date("today");
dimensions("ga:screenName");
metrics("ga:sessions,ga:screenviews,ga:timeOnScreen,ga:avgScreenviewDuration");
filters("ga:dimension2=="Uid_13");
get_analytics();
Get analytics
Performance metrics calculation
Aggregates task success rate
Find learnability
Find Overall performance
Screen Name
Screen Views
Unique Screen Views
6,047
341
1. Dashboard Screen 5,978 (98.86%) 326 (95.60%)
2. Profile Screen 37 (0.61%) 7 (2.05%)
3. User Feedback Dialog 11 (0.18%) 2 (0.59%)
4. Profile Screen- Personalized Map 8 (0.13%) 2 (0.59%)
5. Profile Screen- Activity Level Information 7 (0.12%) 2 (0.59%)
6. Profile Screen- Account Information 6 (0.10%) 2 (0.59%)
22 In order to measure the UX first it need to set the start , end date, dimensions , and metrics for loading the analytics data. Then request are sent to google analytics server
23 Then google analytics sever send the requested data for the calculation of performance metric.


USER EXPERIENCE MEASUREMENT
3/9
Task Success
Task 1 Task 2 Task 3
Uid_13 1 0 0
Uid_11 1 1 1
... ...
Uid_10 0 1 1
Uid_13 Task Success 33 %
User Experience (UX)
Performance metrics calculation
Load analytics
Set start date
Set end date
Set metrics
Set dimensions
Get analytics
Aggregates task success rate
Aggregates time on task
Aggregates error rates
Find the efficiency
Find learnability
Find Overall performance
Screen Name
Screen Views
Unique Screen Views
6,047
341
1. Dashboard Screen 5,978 (98.86%) 326 (95.60%)
2. Profile Screen 37 (0.61%) 7 (2.05%)
3. User Feedback Dialog 11 (0.18%) 2 (0.59%)
4. Profile Screen- Personalized Map 8 (0.13%) 2 (0.59%)
5. Profile Screen- Activity Level Information 7 (0.12%) 2 (0.59%)
6. Profile Screen- Account Information 6 (0.10%) 2 (0.59%)
24.1 All task is checked either it is successfully done or not? Then find out the aggregates the task success rate.


USER EXPERIENCE MEASUREMENT
4/9
24.1
Task Success
Task 1 Task 2 Task 3
Uid_13 1 0 0
Uid_11 1 1 1
... ...
Uid_10 0 1 1
Uid_13 Task Success 33 %
24.2
Uid_13 Time On task 28 %
ser Experience (UX
Performance metrics calculation
24 Aggregates task success rate
Aggregates time on task 24
Aggregates error rates
Find the efficiency
Find learnability
Find Overall performance
24.2
Time On task
Task 1 Task 2 Task 3
Uid_13 55 20 10
Uid_11 30 20 10
... ...
Uid_10 0 1 1
23 Get analytics
23
Screen Name
Screen Views
Unique Screen Views
6,047
341
1. Dashboard Screen 5,978 (98.86%) 326 (95.60%)
2. Profile Screen 37 (0.61%) 7 (2.05%)
3. User Feedback Dialog 11 (0.18%) 2 (0.59%)
4. Profile Screen- Personalized Map 8 (0.13%) 2 (0.59%)
5. Profile Screen- Activity Level Information 7 (0.12%) 2 (0.59%)
6. Profile Screen- Account Information 6 (0.10%) 2 (0.59%)
24.1 All task is checked either it is successfully done or not? Then find out the aggregates the task success rate.
24.2 Find out the spent time on each task and aggregates the time on task.


USER EXPERIENCE MEASUREMENT
5/9
24.1
Task Success
Task 1 Task 2 Task 3
Uid_13 1 0 0
Uid_11 1 1 1
... ...
Uid_10 0 1 1
Uid_13 Task Success 33 %
24.2
Uid_13 Time On task 28 %
ser Experience (UX
Performance metrics calculation
24 Aggregates task success rate
Aggregates time on task 24
Aggregates error rates 24
24.3
Uid_13 Errors rates 34 %
Find learnability
Find Overall performance
24.2
Time On task
Task 1 Task 2 Task 3
Uid_13 55 20 10
Uid_11 30 20 10
... ...
Uid_10 0 1 1
24.3
Errors
Task 1 Task 2 Task 3
Uid_13 1 1 0
Uid_11 0 1 1
... ...
Uid_10 0 1 0
23
6,047
341
1. Dashboard Screen 5,978 (98.86%) 326 (95.60%)
2. Profile Screen 37 (0.61%) 7 (2.05%)
3. User Feedback Dialog 11 (0.18%) 2 (0.59%)
4. Profile Screen- Personalized Map 8 (0.13%) 2 (0.59%)
5. Profile Screen- Activity Level Information 7 (0.12%) 2 (0.59%)
6. Profile Screen- Account Information 6 (0.10%) 2 (0.59%)
24.1 All task is checked either it is successfully done or not? Then find out the aggregates the task success rate.
24.2 Find out the spent time on each task and aggregates the time on task.
24.3 Find out the task errors and aggregate the errors rates.

USER EXPERIENCE MEASUREMENT

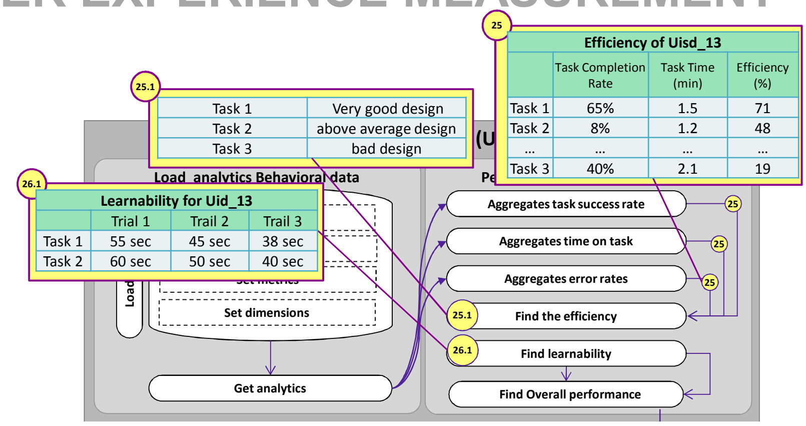


Efficiency of Uisd_13	Task Completion Rate	Task Time (min)	Efficiency (%)
Task 1	65%	1.5	71
Task 2	8%	1.2	48
...	...	...	...
Task 3	40%	2.1	19

Task 1	Very good design
Task 2	above average design
Task 3	bad design

25 All aggregated values are inputs to find out the efficiency

USER EXPERIENCE MEASUREMENT



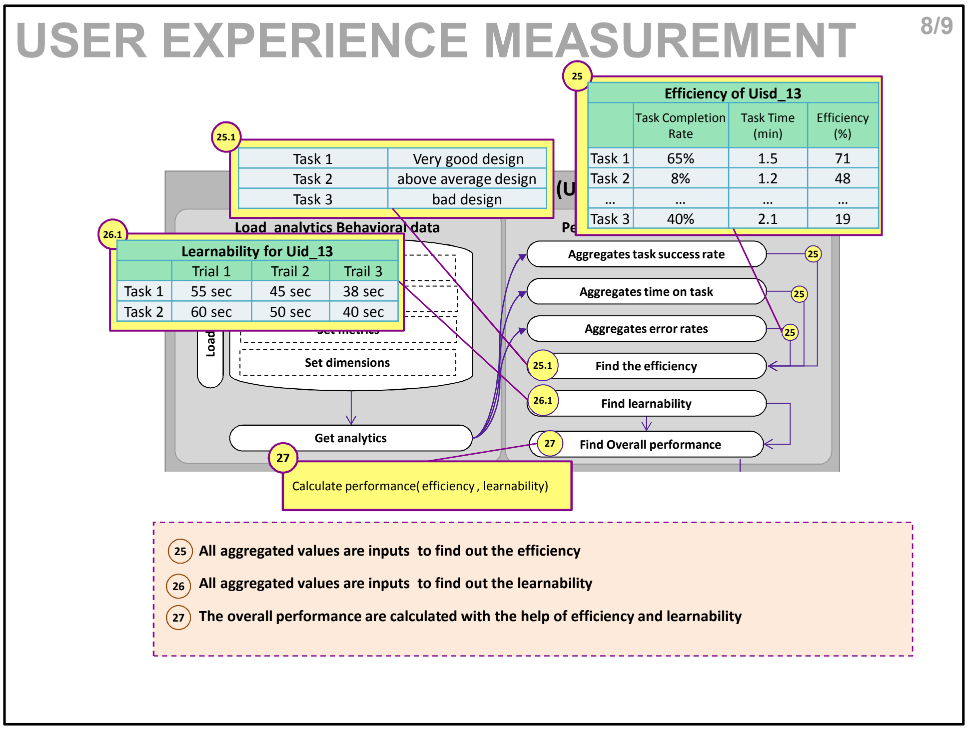
Efficiency of Uisd_13	Task Completion Rate	Task Time (min)	Efficiency (%)
Task 1	65%	1.5	71
Task 2	8%	1.2	48
...	...	...	...
Task 3	40%	2.1	19

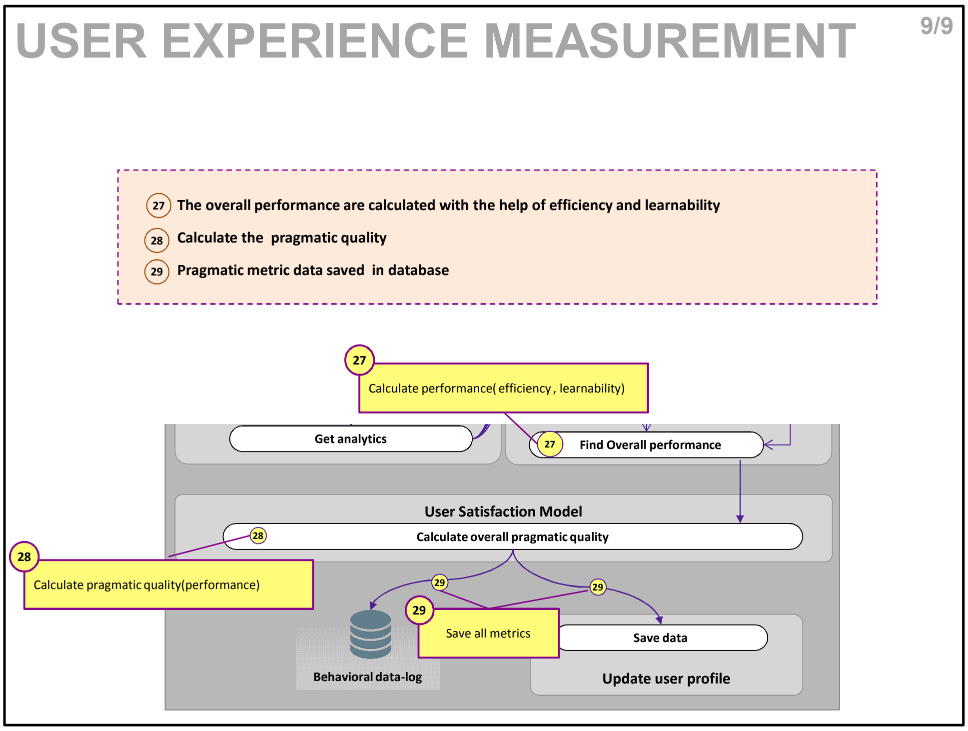
Task 1	Very good design
Task 2	above average design
Task 3	bad design

Learnability for Uid_13	Trial 1	Trail 2	Trail 3
Task 1	55 sec	45 sec	38 sec
Task 2	60 sec	50 sec	40 sec

25 All aggregated values are inputs to find out the efficiency

26 All aggregated values are inputs to find out the learnability


USER EXPERIENCE MEASUREMENT
8/9
25
Efficiency of Uisd_13
Task Completion Rate | Task Time (min) | Efficiency (%)
Task 1 | 65% | 1.5 | 71
Task 2 | 8% | 1.2 | 48
... | ... | ... | ...
Task 3 | 40% | 2.1 | 19
25.1
Task 1 | Very good design
Task 2 | above average design
Task 3 | bad design
26.1
Learnability for Uid_13
Trial 1 | Trail 2 | Trail 3
Task 1 | 55 sec | 45 sec | 38 sec
Task 2 | 60 sec | 50 sec | 40 sec
Load analytics Behavioral data
Set dimensions
Get analytics
Aggregates task success rate
Aggregates time on task
Aggregates error rates
Find the efficiency
Find learnability
Find Overall performance
27
Calculate performance(efficiency , learnability)
25 All aggregated values are inputs to find out the efficiency
26 All aggregated values are inputs to find out the learnability
27 The overall performance are calculated with the help of efficiency and learnability


USER EXPERIENCE MEASUREMENT
9/9
27 The overall performance are calculated with the help of efficiency and learnability
28 Calculate the pragmatic quality
29 Pragmatic metric data saved in database
27
Calculate performance(efficiency , learnability)
Get analytics
Find Overall performance
User Satisfaction Model
Calculate overall pragmatic quality
28
Calculate pragmatic quality(performance)
29
Save all metrics
Save data
Behavioral data-log
Update user profile