OWL-Based User Preference and Behavior Routine Ontology for Ubiquitous System

Kim Anh Pham Ngoc, Young-Koo Lee, and Sung-Young Lee

Department of Computer Engineering, Kyung Hee University, Korea
anhpnk@oslab.khu.ac.kr, yklee@khu.ac.kr, sylee@oslab.khu.ac.kr

Abstract. In ubiquitous computing, behavior routine learning is the process of mining the context-aware data to find interesting rules on the user's behavior, while preference learning tries to utilize the user's behavior information to infer user interests, intention and desires. An intelligent environment should be adaptive, i.e. it is should be able to learn the routine and preference of user, then provide user with the suitable service. Developing intelligent ubiquitous environment requires not only good learning algorithms but also appropriate reusable models of user preference and behavior routine, which are not fully covered by current projects. In this paper, we propose a formal and comprehensive ontology-based model of user preference and behavior routine. The implementation of the ontology using OWL[14] enhances the expressiveness, support inference, knowledge reuse and knowledge sharing, which we can not achieve by normal models. The main benefit of this model is the ability to reason over context data to predict what the user wants the system to do. Based on our model, we also present a rule learning mechanism to learn the preference and behavior rules from context data. [1]

1 Introduction

Intelligent ubiquitous computing focuses on merging intelligent and agent-based system with the ubiquitous computing paradigm. An intelligent environment is a space where ordinary human activities mix seamlessly with computation in a way that enhances the functions of both system and user. That means when a user enters a smart-space, the system can recognize who the user is, what he is doing, "guess" what he intends to do, and how he desires the system to assist him. By other words, the system should be able to learn about user preference and behavior routine so that it can provide services to the user seamlessly & invisibly without any explicit user intervention.

There are many methods to learn user preference and user routine. One of them is association rule mining. User preference and routine learning is considered as associating different contexts to each other. These associations are derived as IF-THEN rules, or association rules. Furthermore, we can apply Bayesian net, or Hidden Markov model for preference and routine learning [7][10]. Many prototyping systems and architectures have successfully introduces novel algorithms for learning user routine from GPS location [7]. In those approaches, important places are identified by

[1] This work was supported by MIC Korea. Dr. S.Y.Lee is the corresponding author.

R. Meersman and Z. Tari (Eds.): CoopIS/DOA/ODBASE 2005, LNCS 3761, pp. 1615 – 1622, 2005.

monitoring the user's travel patterns and learning his frequented locations. Then these important places will be named by user.

However, a formal model to memorize the routines of user has *not* been defined; or by other words, there is *no* formal model for user routine. Similarly, the term "user preference" so far is just related to the user interest for some very specific subjects, such as the web links, music, presentation material, etc, as well as situation independent [10]. *None* of current user preference model fully represents the preferences of user in a ubiquitous environment, where the user interest, desire and intention vary by time and place. The requirement of having a formal model of user preference and behavior routine to use in ubiquitous computing systems is obvious. SOUPA [5] has already defined a user preference model. Nevertheless, this model is specified only for meeting room scenario, and rather simple to be considered as a *formal* and *general* user preference model.

Recently, many systems model context data using ontology and semantic web technique [6][12]. Web Ontology Language OWL [14] is preferred due to its ability to represent explicitly semantics associated with the knowledge, and to provide reasoning capabilities used by intelligent systems and agents to infer useful contexts. Our CAMUS middleware follows this ontology-based modeling approach. We have already defined and used OWL ontologies for basic entities in context-aware systems including agent, time, location, device and environment, as well as for domain data representing [1].

In this paper, to address the issue of user preference and behavior routine formal modeling, we propose additional *OWL-based user preference ontology* includes modular component vocabularies to represent user beliefs, desires, and intentions related to different times and places, together with the *behavior routine ontology* which is a sequence of location with the expected interval for each location, and allows developer to express the recurrence of the routine.

The rest of this paper is organized as follows. In section 2, we describe the user preference and behavior routine model and its ontological structure. Section 3 gives a detailed description of our idea by discussing the learning and reasoning mechanism with the support of ontology and OWL. We conclude our paper with a summary and outlook in section 4.

2 Spatio-Temporal Ontology of User Preference and User Routine

2.1 Using OWL Ontologies for Formal Context Modeling

Within the domain of knowledge representation, the term ontology refers to the formal, explicit description of concepts, which are often conceived as a set of entities, relations, instances, functions, and axioms, leading to shared and common understanding that can be communicated between people and application systems [2]. Traditionally, ontologies are only used to describe domains (as mentioned above) but in W3C's OWL (web ontology language) [14], the horizon of ontology has been broadened to include instance data as well.

There are several potential advantages for developing context models based on Semantic Web Ontology, such as its e*xpressiveness*, the capability of *Knowledge Sharing* and *Knowledge Reuse*; the support to various existing *logic inference* mechanisms, and lastly, its e*xtensibility.*

2.2 Spatio-Temporal Ontology of User Preference (STOUP)

In many personalized e-applications, the preference model is merely a strict partial order of a set of attributes, expressing "for attribute A, value *y* is better than *x*". However, in a ubiquitous system, not only the pure preference of users, but also the interest, desire and intention of the users should be considered. Besides, situation also plays an important role, i.e., user preference alternates from time to time and from place to place. A formal user preference model should cover all these aspects.

Our user preference and routine model is an additional part of an existing context model for ubiquitous system, Contel, which is already defined by our research group, and currently used in CAMUS [1]. In Contel, all the entities in a context-aware system are categorized into agents, devices, environment, location and time. These categories consist of following main classes (or concepts): Agent, Activity, Devices, Environment, Location Description, Place, Time (Time Interval and Time Instant) and Event. There are also many auxiliary classes, generalized and specialized classes to enrich the semantic capability.

To represent the user preference model, we add a *Preference* class, which has relationship with all main classes, and its subclasses as illustrated in Fig.1.

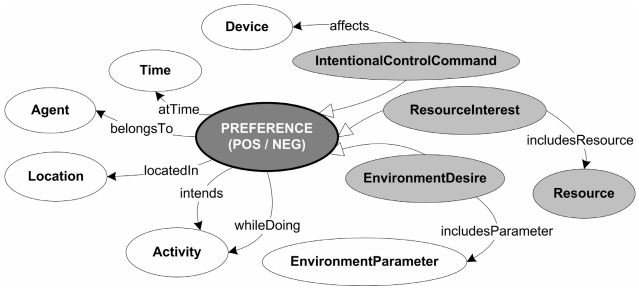

Fig. 1. STOUP structure. Preference class and its subclasses are new classes added to Contel.

An agent (which is a user, group or organization) can have *POS* or *NEG Preference*, which represents the like and dislike, or the best choice and the worst choice. Each preference is related to a certain time and place. Preference also depends on what the agent is doing, and/or what it intends to do next. There are 3 sub-classes of Preference: *ResourceInterest*, *EnvironmentDesire* and *IntentionalControlCommand*. *ResourceInterest* represents the interest of agent in some Resources, such as *MusicGenre* or *TVChannel*. *EnvironmentDesire* expresses how an agent wants the environment to be, for example the desired temperature or the preferred light. *IntentionalControlCommand* specifies the operations which an agent wants the devices to perform, such as turning on the television or rolling down the curtain.

All the properties of *Preference* class which are related to *Agent*, *Time*, *Location* and *Activity* have minimal cardinality 0. The absence of any specified relationship is understood as "for all kind of this". For example, if a preference is defined without any location, it means that this preference can be applied everywhere. *Preference* class also has the *noAgent* property to make it become a general rule which will be applied whenever there is no user around, such as turning off the light or change the

software program running on computer into stand-by mode. Similarly, the *noActivity* property denotes the idle state of agents.

Another significant property of *Preference* class is the *probability* property, which expresses the importance or the priority of a preference. This property has major influence on system decision making process. For example, if the probability of a user giving "Light.TurnOff" control command before going to bed is 99%, the system can infer that whenever the user is sleeping, the light should be off, i.e. the light intensity should be Dark. Or that the equal probability of user watching Music channel or Movie channel at night makes the system ask the user before selecting one of those channel.

2.3 Spatio-Temporal Ontology of User Routine (STOUR)

There are some requirements for a user routine model in ubiquitous computing systems. First, a routine is recursive. It can be a daily routine, or weekly, or monthly routine, etc. Second, a user routine includes a sequence of user locations, or user activities, each of which has expected time interval, i.e. the average time interval user spends doing an activity at a location. Finally, the routine model should support reasoning, which means it should help predicting the next location, or the intended activity.

Fig. 2 illustrates the structure of our user routine ontology.

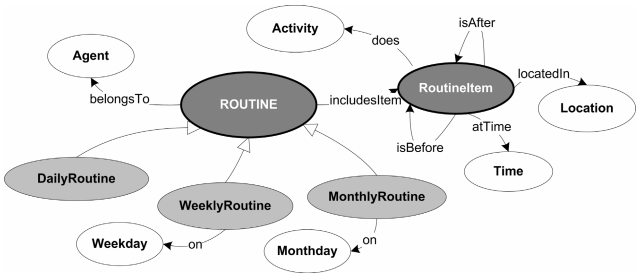

Fig. 2. STOUR structure. Routine, RoutineItem class and Routine's subclasses are new classes added to Contel.

An *Agent* can have one to many *Routine*. There are 3 types of routine: *DailyRoutine*, which is the most common, *WeeklyRoutine* and *MonthlyRoutine*. For *WeeklyRoutine*, some weekdays are included, for example the user goes to gym every Tuesday, Thursday and Saturday. Similarly, *MonthlyRoutine* comes with some days in the month.

Each *Routine* has a sequence of *RoutineItem*. Each item is a state in which user is located in a certain place or is doing a certain work, in an expected time interval. *RoutineItems* go in sequence, so each of them can be before of after another.

Because routine is also an uncertain concept, one *routineProbability* property is attached to each *RoutineItem*.

The following example shows the context ontology that describes a preference and routine of a user named Bilbo using OWL.

```
<POSResourceInterest rdf:ID="BilboMusicInterest">
 <probability>0.9</probability>
 <belongsTo
       rdf:resource="http://ucg.khu.ac.kr/ca#Bilbo"/ >
 <locatedIn
       rdf:resource="http://ucg.khu.ac.kr/cl#LivingRoom"/ >
 <atTime>
  <TimeInterval>
   <startTime>18:00</startTime>
   <endTime>20:00</endTime>
  <TimeInterval>
 </atTime>
<includesResource>
 <MusicResource>
    <resourcePropertyName>MusicGenre
    </resourcePropertyName>
    <resourcePropertyValue>Classical Music
    </resourcePropertyValue>
 </MusicResource>
 </includesResource>
</POSResourceInterest>
```

```
<WeeklyRoutine rdf:ID="BilboWeeklyMeetingRoutine">
 <belongsTo
       rdf:resource="http://ucg.khu.ac.kr/ca#Bilbo"/ >
 <on
       rdf:resource="http://ucg.khu.ac.kr/cr#Tuesday"/ >
 <includesItem
       rdf:resource="http://ucg.khu.ac.kr/br#TuesdayMeeting"/ >
</WeeklyRoutine >
<TuesdayMeeting>
  <routineProbability>0.86</routineProbability>
  <atTime>
  <TimeInterval>
   <startTime>14:00</startTime>
   <endTime>16:00</endTime>
  <TimeInterval>
 </atTime>
 <does rdf:resource="http://ucg.khu.ac.kr/cac#OSLabMeeting"/ >
 <locatedIn rdf:resource="http://ucg.khu.ac.kr/cl#Room505"/ >
</TuesdayMeeting>
```

Fig. 3. example of user preference and routine data in OWL

3 OWL-Based Reasoning and Learning Mechanism

In this session, to show the advantage of ontology-based modeling approach using OWL, we will illustrate the OWL-support reasoning mechanism, and how the system can learn user routine and preference through the control commands of user and the history context database.

3.1 OWL-support Reasoning Mechanism

3.1.1 Ontology Reasoning Mechanisms

High valued ontologies depend heavily on the availability of well-defined semantics and powerful reasoning modules. The expressive power and the efficiency of reasoning provided by OWL, (the semantics of OWL can be defined via a translation into an expressive Description Logics (DL)), make it an ideal candidate for ontology constructs. The facts gathered from context entities make a factual world in OWL, consisting of individuals and their relationships asserted through binary relations.

Ontology reasoning helps us to find subsumption relationships (between subconcept-superconcept), instance relationships (an individual i is an instance of concept C), and consistency of context knowledge base. In the design phase of formalizing the context entities, OWL reasoning services (such as satisfiability and subsumption) can test whether concepts are non-contradictory and can derive implied relations between concepts.

Let us take an example to see how ontology reasoning can help deducing implied context. In preference ontology, the class PianoMusic is a subClassOf ClassicalMusic. So when knowing that Bilbo is interestedIn ClassicalMusic, and "Hungarian Sonata" is a song which has type PianoMusic, the system can deduce that Bilbo is interestedIn "Hungarian Sonata".

3.1.2 Context Reasoning Mechanisms

However, many types of contextual information cannot be easily deduced using only ontology inference. In addition to ontology reasoning, we can also use logic inference. A set of rules can be defined to assert additional constraints for context entity instances when certain conditions (represented by a concept term) are met.

Over the concepts and relations defined in our ontologies, we can do a lot of reasoning based on many types of logics, such as description logic, description temporal logic, and spatial logic.

There are many reasoning engines work over OWL format data, such as Racer[13], SWI-Prolog[11], Pellet[9], etc. A list of OWL implementations can be found at [8]. In our current implementation, we use Jena library [4] to handle OWL format context data and ontologies. Therefore we exploit the Jena generic rule reasoner [4] to make inference over our context data.

Following is an example of Jena rule to infer the control command which user intends to give in a certain situation.

[r1: (?user agt:personName "Bilbo"), (?x time:currentTime ?t),
(?user act:currentActivity ?curact), (?curact rdf:type act:WatchingTV),
(?curact act:actionObject ?tv), (?user act:intendedActivity ?intact),
(?intact rdf:type act:GoingToWork) AND 520 <=?t AND ?t <= 530
-> [(?cmd dev:cmdObject ?tv), (?cmd dev:cmdTime ?t)
<- (makeInstance(?user,agt:givesCommand, dev:TVTurnOff, ?cmd)]][2] (1)

with *agt, time, act, rdf, dev* are the aliases for the namespaces of ontologies (agent, time, activity, RDF, device) which are currently used to define data models in our system. Details of those ontologies are described in [1]. This rule is matched when there are OWL markups about user Bilbo watching TV and intending to go to work, then a new instance of class TVTurnOff command is created and the properties cmdObject and cmdTime of that instance is assigned with the current watched television and current time.

However, most developers find building the rules like this the most difficult task in building ubiquitous computing systems, particularly in intelligent environments such as smart homes, where the system has to learn a lot about users preference, behavior, routine, etc. In order to minimize the burden for developer in building context-aware applications, our middleware architecture provide support to learn the inference rules from context data and build reasoning engines using those rules, as described in next section.

3.2 Learning the Rules for Context Reasoning

In this scenario, the user preference is learned through user control commands and responses to the messages from system. User can control the home devices by remote controls, or send command messages through some computer software interfaces. The commands are stored in the history database together with relevant information i.e.

[2] This is a temporal rule. In Jena generic reasoner[3], we can infer about time by continuously updating the currentTime property of Time class with current timestamp. The time value is converted into minutes (or second, or millisecond, depend on the purpose of system).

user location, timestamp, current activity, intended activity (if this information is available), environment state. The tuple {user=Bilbo; place=DiningRoom; timestamp = 2005/04/01 8:40; currentActivity=WatchingTV; intendedActivity= GoingToWork; command= TV.TurnOff} is an example.

Using this kind of information as the training data, the system can learn rules like:

personName=Bilbo; currenttime=[8:40-8:50];
currentActivity=WatchingTV; intendedActivity=GoingToWork
$\Rightarrow$ *command=TV.TurnOff (Utility=0.86)*

The rules can then be converted into suitable format for the reasoning engine which is used. In our prototype system, a rule like (1) is produced.

Moreover, we store the rule into database as a Preference object so that the knowledge about user preference can be shared and reused.

There are many algorithms to learn a rule from example data set. Currently we apply two algorithms for two cases:

- *Rule learning when knowing the desired output*

If the output is defined, we learn the rule from example data set by an approach as in decision tree learning but by following the branch with best score in terms of splitting function.

The *Utility* of a new candidate can be computed using information-theoretic measures like entropy:

Utility(r) = entropy(the subset of examples covered by r)

- *Rule learning without knowing the desired output*

If the output is undefined, we can't use any classification algorithm to learn the rule sets. In this case, Apriori association rule mining algorithm [3] is more suitable. Among a large number of learned rules, we select the "right" rules by assigning a utility function to calculate the value of each rule based on confidence and support.

$$Utility(r) = \alpha.Conf(r) + \beta.Sup(r)$$

With *Conf(r)* is the confidence *Sup(r)* is the support of the rule.

The α and β coefficients are related to each other by $\alpha + \beta = 1$, and define the type of rule which is more interested. Normally $\alpha = 1$ and $\beta = 0$, showing that a rule which has high confidence will be chosen even if it rarely happens.

Only the rules with high utility will be selected.

4 Summary and Outlook

In this paper, we have presented a formal Spatio-Temporal Ontology of User Preference and Behavior Routine. We discussed how it can be used for heterogeneous ubiquitous computing environment to support knowledge sharing, reuse, and logical reasoning with the help of Smart Home scenario.

Our next steps include the integration of various machine learning techniques and reasoning engines into our framework. Different machine learning techniques have different input and output format, and different use. By implementing the wrapper for all the techniques and defining a common format for input data, we hope to enable the system developers to handle the machine learning techniques more easily.

References

[1] Anjum S., et al.: Formal Modeling in Context Aware Systems. In proceedings: Workshop on Modeling and Retrieval of Context, CEUR, ISSN 613-0073, Vol-114, 2004.
[2] J. Davies, et al.: Towards the Semantic Web, Ontology-Driven Knowledge Management, John Wiley & Sons. (Nov. 2002)
[3] Jiawei Han,Micheline Kamber. Data Mining: Concepts and Techniques
[4] Jena: A Semantic Web Framework for Java. http://jena.sourceforge.net/
[5] Harry C., et al.: SOUPA: Standard Ontology for Ubiquitous and Ubiquitous Applications. In Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous 2004), Boston, MA, August 22-26, 2004.
[6] Harry Chen et al., "Intelligent Agents Meet the Semantic Web in Smart Spaces", Article, IEEE Internet Computing, November 2004
[7] Liao L.,et al.: Learning and Inferring Transportation Routines. In Proceedings: AAAI-0, 2004.
[8] OWL Implementations. http://www.w3.org/2001/sw/WebOnt/impls
[9] Pellet OWL Reasoner. http://www.mindswap.org/2003/pellet/index.shtml
[10] Stefan H., et al.: Preference Mining: A Novel Approach on Mining User Preferences for Personalized Applications. PKDD 2003
[11] SWI-Prolog/XPCE Semantic Web Library. http://www.swi-prolog.org/packages/semweb. html
[12] T. Gu et. al. An Ontology-based Context Model in Intelligent Environments. In Proceedings of Communication Networks and Distributed Systems Modeling and Simulation Conference, pp. 270-275. San Diego, California, USA, January 2004
[13] Volker Haarslev et. al. Querying the Semantic Web with Racer + nRQL. Proceedings of the KI-2004 International Workshop on Applications of Description Logics (ADL'04), Ulm, Germany, September 24, 2004.
[14] W3C Web Ontology Working Group: The Web Ontology language: OWL. http://www.w3.org/2001/sw/WebOnt/