Chapter 4 - Requirements Engineering

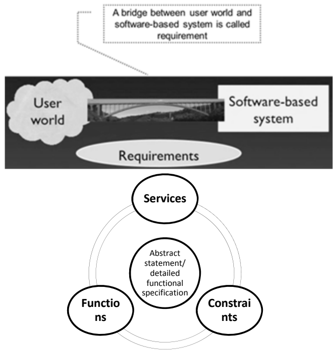
Contents

Chapter 4 – Requirements Engineering

Lecture 1 – Requirements (user, System, domain)

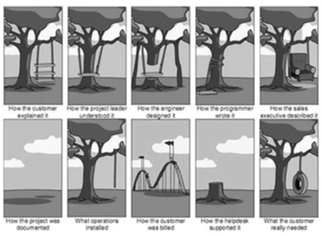
What is Requirement?

- It's a high-level abstract statement or a detailed mathematical functional specification of a system's services, functions and constraints
- The requirement should be
  - Open to interpretation and
  - Detailed enough to understand



Requirements Engineering

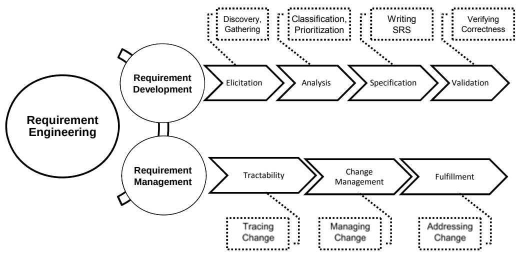
- The process of establishing services that the customer requires from a system and the constraints under which the system is operates and developed
- Requirements themselves are the descriptions of the system services and constraints



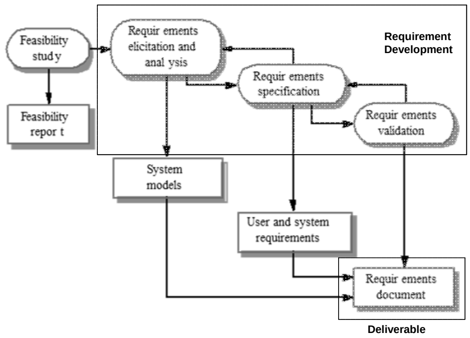
Importance of requirements Engineering

http://www.knovelblogs.com/2012/08/30/the-importance-of-requirements-engineering/

Requirements Engineering Process

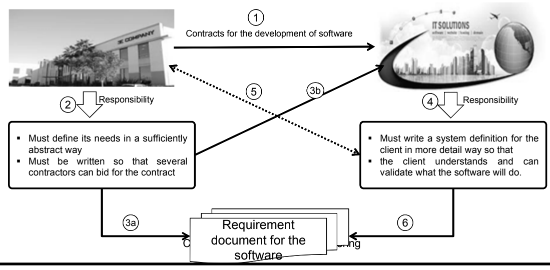


Requirements Development Process

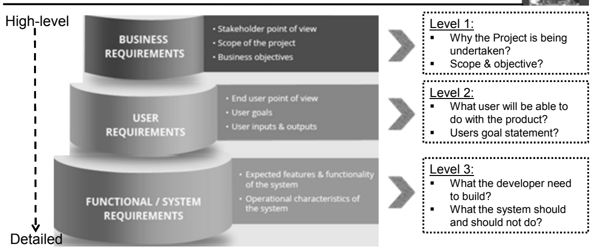


Requirements Abstraction (Davis)

- "A company wishes to let a contract for the development of a large software project from an IT Solution Company"
- Company must define its needs in a sufficiently abstract way
- The contractor must write a system definition for the client
- Both of these documents may be called the requirements document for the system



Types of Requirements

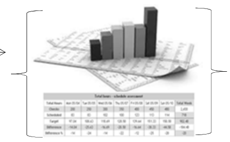


- **User requirements:** Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.
- **System requirements:** A structured document setting out detailed descriptions of the system's functions, services and operational constraints. Defines what should be implemented so may be part of a contract between client and contractor.
- **Business requirements:** A documents describing why the project is being started and what is its scope and objective

http://www.netsolutionsindia.com/blog/business-and-functional-requirements-what-is-the-differance-and-why-should-you-care/

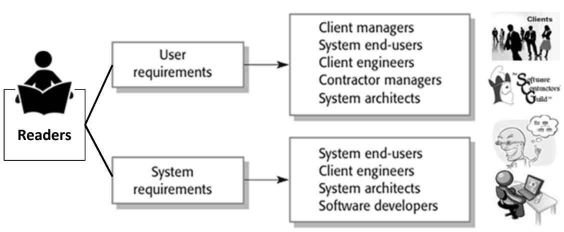
User and system requirements

- User requirements
  - Monthly and weekly reports showing cost of drugs prescribed by each clinic

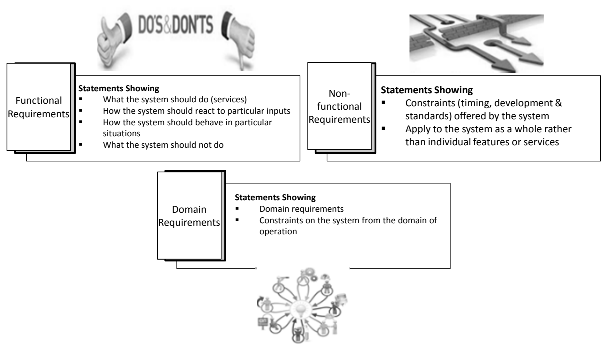


- System requirements
  - The system should have the capabilities to: (a) generate summary at the last day of month (b) automatic printing of the reports (c) unauthorized access control etc.

Readers of Different Types of Requirements Specification



Functional, Non-functional & Domain Requirements



Functional Requirements

Functional Requirements

- Describe functionality or services of the system
- High-level statements of what the system should and should not do
- How the system behave in different Situations

Example (medical system)

- A user shall search appointments lists for clinics
- System shall generate each day, for each clinic, a list of patients

Dependency

- Type of software,
- Expected users and
- The type of system where the software is used

Functional requirements for the MHC-PMS

- A user shall be able to search the appointments lists for all clinics.
- Generate Reports:
  - The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
- Authorization:
  - Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Functional Requirement

- To understand functional requirement, watch
  http://www.youtube.com/watch?v=a0BSre06ENg&list=PLpQUJygQ38u_YwDYil6u0cY_ti2_eHfKe

Requirements Imprecision

Imprecise Requirements

- The problem arise when requirements are not precisely stated.
- Requirements are ambiguously defined

Examples Scenario

- Consider the term 'search' in requirement 1
  - User intention – search for a patient name across all appointments in all clinics;
  - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

Requirements completeness and consistency

Good Requirement

Complete
- All the information
- No one guessing

Example
On power loss battery must support
- Operations (incomplete)
- Operations for 2h (complete)

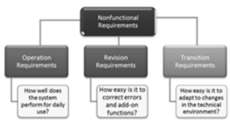
Consistent
- Do not conflict/no duplicate
- No contradiction
- Same terminology

Example
Change student :
- Address (conflict)
- Email address (no conflict)

- ✧ In principle, requirements should be both complete and consistent.
- ✧ Complete: They should include descriptions of all facilities required.
- ✧ Consistent: There should be no conflicts or contradictions in the descriptions of the system facilities.
- ✧ In practice, it is impossible to produce a complete and consistent requirements document.

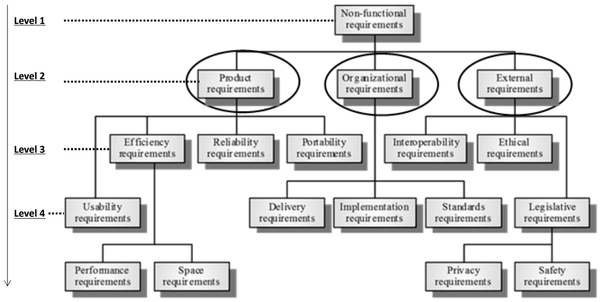
Non-functional Requirements

- Definition (non-functional requirements)
  - The requirements that tell how well does the system perform for daily use, how easy is it to correct errors and add-on function and how easy is it to adapt to change in the technical environment?
- It defines
  - System's properties : e.g. Reliability, response time and storage requirements
  - System constraints e.g. I/O device capability, system representations, etc.
  - Process requirements e.g. Particular IDE, programming language or development method



http://edevtech.com/blog/save-your-project-elicit-nonfunctional-requirements-effectively/

Types of Non-functional Requirements



Non-functional requirements implementation

Effects overall system rather than individual component

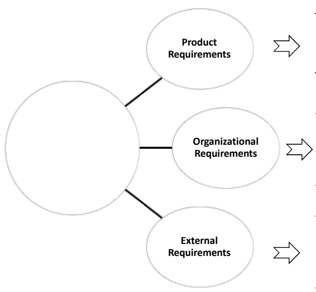
- For example, to ensure that performance requirements are met, you may have to organize the system to minimize communications between components.

Multiple functional requirements are generated for a single non-functional requirement

- For example, security may generate a number of related functional requirements that define system services that are required.

Non-functional classification

Non-functional Requirements | **Definition and Example**



Product Requirements
- Specify that the delivered product must behave in a particular way e.g.
  - Execution speed, reliability, etc.

Organizational Requirements
- Are a consequence of organizational policies and procedures e.g.
  - Process standards used, implementation requirements, etc.

External Requirements
- Arise from factors which are external to the system and its development process e.g.
  - Interoperability requirements, legislative requirements, etc.

Examples of nonfunctional requirements in the MHC-PMS

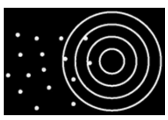
Product requirement
The MHC-PMS shall be available to all clinics during normal working hours (Mon–Fri, 0830–17.30). Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement
Users of the MHC-PMS system shall authenticate themselves using their health authority identity card.

External requirement
The system shall implement patient privacy provisions as set out in HStan-03-2006-priv.

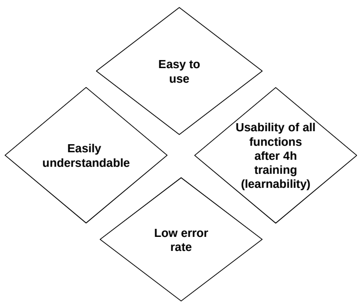
Goals and requirements

- Non-functional requirements may be very ddifficult to state precisely
  - imprecise requirements are difficult to verify
- Goal
  - A general intention of the goal is user's "ease of use"
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested



Usability requirements

- The system should be easy to use, produce low errors rate, should be easily understandable, increase learnability with experience
- Medical staff shall be able
  - to use all the system functions after four hours of training.
  - after this training, the average number of errors made by experienced users shall not exceed two per hour of system use.

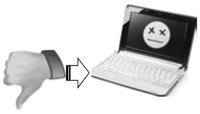


Metrics for Specifying Non-functional Requirements

Property	Measure
Speed	▪ Processed transactions/second ▪ User/event response time ▪ Screen refresh time
Size	▪ Mbytes ▪ Number of ROM chips
Ease of use	▪ Training time ▪ Number of help frames
Reliability	▪ Mean time to failure ▪ Probability of unavailability ▪ Rate of failure occurrence ▪ Availability
Robustness	▪ Time to restart after failure ▪ Percentage of events causing failure ▪ Probability of data corruption on failure
Portability	▪ Percentage of target dependent statements ▪ Number of target systems

Domain Requirements

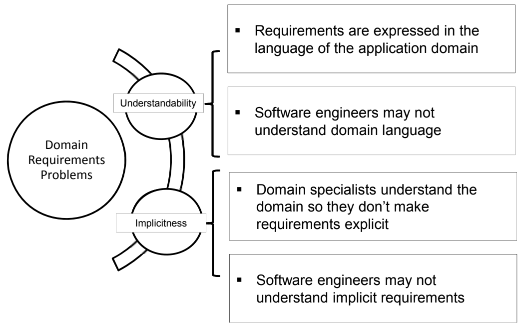
- Requirements imposed by a system's operational domain
  - For example. a train control system has to take into account the braking characteristics in different weather conditions
- How they are defined?
  - Can be either new functional requirements or constraints on existing requirements
- Why use domain requirements?
  - To make the system workable
  - If not satisfied, the system may not work



Train protection system

- This is a domain requirement for a train protection system:
- The deceleration of the train shall be computed as:
  - Dtrain = Dcontrol + Dgradient
  - where Dgradient is 9.81ms2 * compensated gradient/alpha and where the values of 9.81ms2 /alpha are known for different types of train.
- It is difficult for a non-specialist to understand the implications of this and how it interacts with other requirements.

Domain Requirements Problems



Key Points

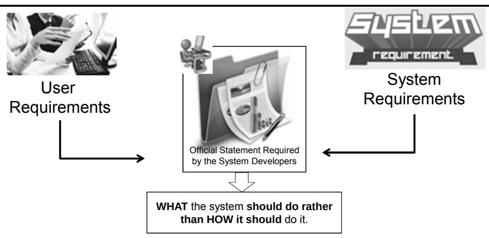
- Requirements set out
  - What the system should do and
  - What are the constraints on operation and implementation
- Functional requirements describe
  - Services of the system
- Non-functional requirements describe
  - Constrain imposed by the system
- Both are applied to the system

Chapter 4 – Requirements Engineering

Lecture 2 – Software Requirement Document and its Representation

Software Requirements Document

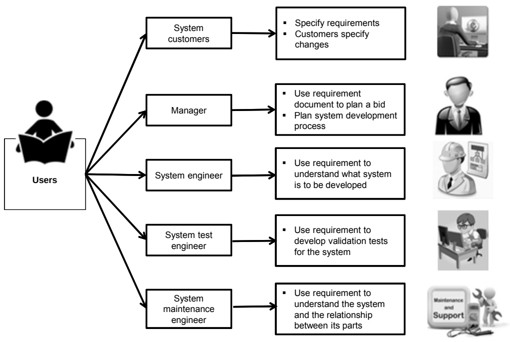
- The software requirements document is an official written statement of what the software will do.
- Include both a definition of user requirements and a specification of the system requirements.
- As far as possible, it should set of WHAT the system should do rather than HOW it should do it.



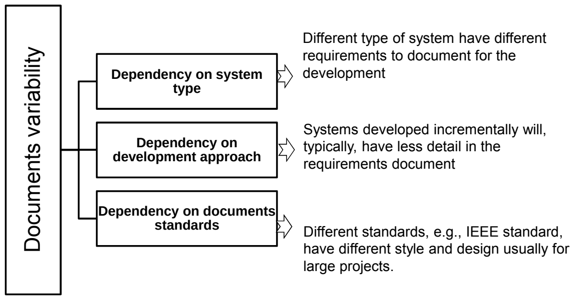
Agile Methods and Requirements

- Agile approach to requirements
  - Producing a requirements document using agile method is waste of time as requirements change so quickly.
  - The document is therefore always out of date.
- Problems of Agile Methods for Requirements
  - Practical for business systems but problematic for systems that require a lot of pre-delivery analysis (e.g. critical systems)

Users of a Requirements Document



Requirements Document Variability



The Structure of a Requirements Document

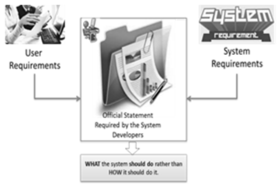
Chapter	Description
Preface	▪ defines expected readership of the document ▪ describes its version history, ▪ rationale for the creation of a new version ▪ summary of the changes made in each version
Introduction	▪ describe the need for the system ▪ briefly describe the system's functions and explain how it will work ▪ describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	▪ defines technical terms used in the document. ▪ make assumptions about the experience or expertise of the reader.
User requirements definition	▪ describe the services provided for the user. ▪ Describes nonfunctional system requirements using natural language, diagrams, etc. ▪ Specifies product and process standards that must be followed.
System architecture	▪ Describes a high-level overview of the anticipated system architecture, ▪ Distribution of functions across system modules. ▪ Architectural components that are reused should be highlighted.

The Structure of a Requirements Document

Chapter	Description
System requirements specification	▪ further detail of nonfunctional requirements. ▪ Interfaces should be added to more describe the functional and nonfunctional requirements
System models	▪ includes graphical system models ▪ shows relationships between the system components. ▪ Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	▪ describes fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. ▪ useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	▪ provide detailed, specific information related to application being developed; ▪ for example, hardware and database descriptions and database requirements
Index	▪ Several indexes to the document may be included. ▪ As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Requirements Specification

- Definition (specification)
  - The process of writing down the user and system requirements in a requirements document.
  - Its a detailed description of the requirements that may include more technical information.
- Characteristics
  - Should be understandable by end-users and customers who do not have a technical background.
  - Should be complete enough to model.

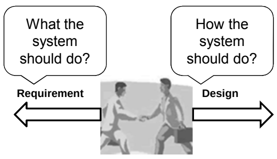


Ways of Writing a System Requirements Specification

Notation	Description
Natural language	▪ numbered sentences in natural language are used ▪ Each sentence should express one requirement.
Structured natural language	▪ standard form or template of natural language is used. ▪ Each field provides information about an aspect of the requirement.
Design description languages	▪ language like a programming language, but with more abstract features is used
Graphical notations	▪ Graphical models, supplemented by text annotations, are used to define the functional requirements for the system; ▪ UML use case and sequence diagrams are commonly used.
Mathematical specifications	▪ finite-state machines or sets are used. ▪ these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification.

Requirements and Design

- Requirement vs. Design
  - Requirements should state what the system should do and the design should describe how it does this.
- Requirements and design are inseparable
  - A system architecture may be designed to structure the requirements;
  - The system may inter-operate with other systems that generate design requirements;
  - The use of a specific architecture to satisfy non-functional requirements may be a domain requirement.
  - This may be the consequence of a regulatory requirement.



Natural language specification

Specification using natural language

- The requirements are written as natural language sentences supplemented by diagrams and tables

Reason for using natural language as specification

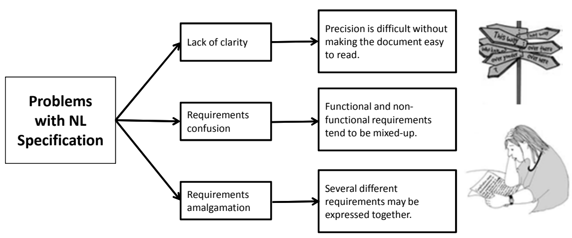
- it is expressive, intuitive and universal.
- can be understood by users and customers easily.

Guidelines for writing requirements

- General Guidelines (NL)
  - Use standard format for all requirements.
  - Use language in a consistent way.
  - Use shall for mandatory requirements
  - Use should for desirable requirements.
  - Use text highlighting
  - Avoid the use of computer jargon.
  - Include an explanation (rationale) of why a requirement is necessary.

Standard format
Consistent language
Use of shall
Use of should
Highlighting
Avoid jargon
Explanation

Problems with natural language



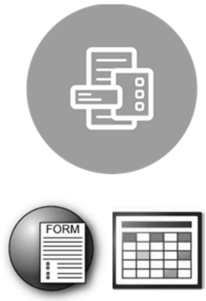
Example requirements for the insulin pump software system

3.2 The system shall measure the blood sugar and deliver insulin, if required, every 10 minutes. *(Changes in blood sugar are relatively slow so more frequent measurement is unnecessary; less frequent measurement could lead to unnecessarily high sugar levels.)*

3.6 The system shall run a self-test routine every minute with the conditions to be tested and the associated actions defined in Table 1. *(A self-test routine can discover hardware and software problems and alert the user to the fact the normal operation may be impossible.)*

Structured Specifications

- Definition (Structured Specifications)
  - An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- Applicability/Suitability
  - Suitable for embedded control system but
  - Too rigid for writing business system requirements.
- Types
  - Form-based specification
  - Tabular-based specification



Form-based specifications

- Definition of the function or entity.
- Description of function
- Inputs and their source.
- Outputs and where they go to.
- Action to be taken.
- Pre and post conditions (if appropriate).
- The side effects (if any) of the function.

Insulin Pump Example

Insulin Pump/Control Software/SRS/3.3.2

Function Compute insulin dose: safe sugar level.

Description

Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

Inputs Current sugar reading (r2); the previous two readings (r0 and r1).

Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose—the dose in insulin to be delivered.

Destination Main control loop.

Action

CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.

Requirements

Two previous readings so that the rate of change of sugar level can be computed.

Pre-condition

The insulin reservoir contains at least the maximum allowed single dose of insulin.

Post-condition r0 is replaced by r1 then r1 is replaced by r2.

Side effects None.

A structured specification of a requirement for an insulin pump

Insulin Pump/Control Software/SRS/3.3.2

Function Compute insulin dose: safe sugar level.

Description

Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.

Inputs Current sugar reading (r2); the previous two readings (r0 and r1).

Source Current sugar reading from sensor. Other readings from memory.

Outputs CompDose—the dose in insulin to be delivered.

Destination Main control loop.

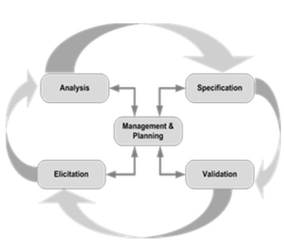
Tabular specification

- Used to supplement natural language.
- Particularly useful when a number of possible alternative courses of action are available.
- Example,
  - Insulin pump systems bases its computations on the rate of change of blood sugar level as shown in the tabular specification below.

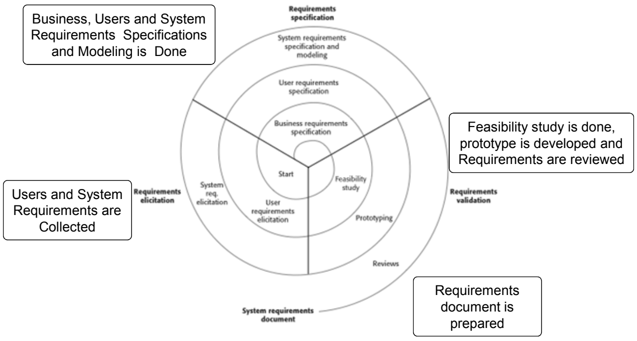
Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2 – r1) < (r1 – r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ((r2 – r1) ≥ (r1 – r0))	CompDose = round ((r2 – r1)/4) If rounded result = 0 then CompDose = MinimumDose

Requirements Engineering Processes

- Widely depends on application domain, people involved and organisation developing the requirements.
- A number of generic activities, common to all processes are:
  - Requirements elicitation;
  - Requirements analysis;
  - Requirements validation;
  - Requirements Specification.
- Is an iterative activity in which these processes are interleaved.



A Spiral View of the Requirements Engineering Process



Requirements Elicitation and Analysis

- Definition (Elicitation and Analysis)
  - The process in which software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, system performance, and hardware constraints, etc.
- People involved (*Stakeholders*)
  - Technical staff
  - End-users, managers, engineers, domain experts, trade unions

Problems of Requirements Analysis

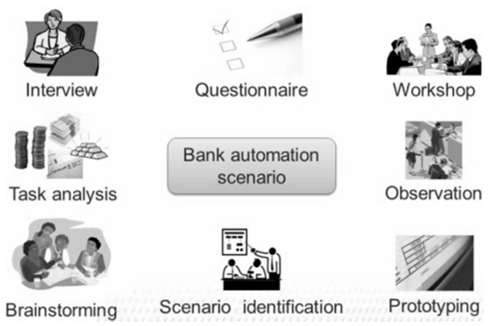
- Stakeholders don't know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- The requirements change during the analysis process.
- New stakeholders may emerge and the business environment change.

Unclear objective/goal
Own spec. of requirement
Conflicting requirements
Organ. and political influence
Change in requirement
Change in environment

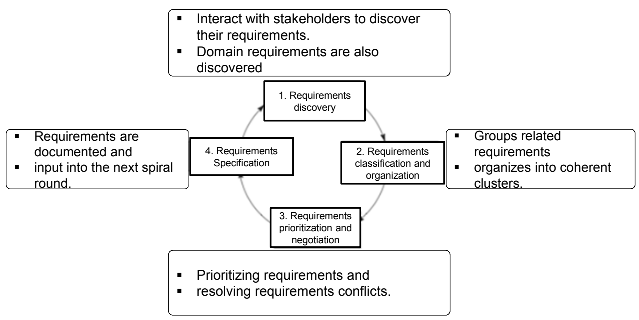
Requirements elicitation and analysis

- Software engineers work with a range of system stakeholders to
  - find out about the application domain,
  - the services that the system should provide,
  - the required system performance, hardware constraints, other systems, etc.
- Stages include:
  - Requirements discovery,
  - Requirements classification and organization,
  - Requirements prioritization and negotiation,
  - Requirements specification.

Requirements Elicitation Techniques



Requirements Elicitation and Analysis Process



Problems of Requirements Elicitation

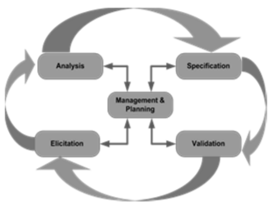
- Stakeholders don't know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- Organisational and political factors may influence the system requirements.
- The requirements change during the analysis process.
- New stakeholders may emerge and the business environment change.

Key Points (Lecture 2)

- The software requirements document is an agreed statement of the system requirements.
- The requirements engineering process is an iterative process used to represent the requirements
- Different specification methods including natural language, form-based, tabular-based, other etc. are used

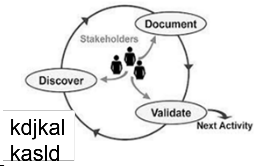
Chapter 4 – Requirements Engineering

Lecture 3 - Requirements Engineering and Management Processes



Requirements Discovery

- Definition (requirements discovery)
  - The process of gathering information about the required and existing systems and filtering the user and system requirements from this information.
- Stakeholders involved
  - Stakeholders ranges from managers to external regulators.
- Systems normally have a range of stakeholders.



kdjkal kasld k

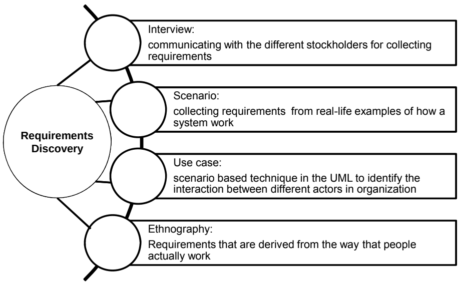
Stakeholders in the MHC-PMS

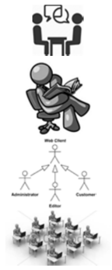
- Patients whose information is recorded in the system.
- Doctors who are responsible for assessing and treating patients.
- Nurses who coordinate the consultations with doctors and administer some treatments.
- Medical receptionists who manage patients' appointments.
- IT staff who are responsible for installing and maintaining the system.

Stakeholders in the MHC-PMS

- A medical ethics manager who must ensure that the system meets current ethical guidelines for patient care.
- Health care managers who obtain management information from the system.
- Medical records staff who are responsible for ensuring that system information can be maintained and preserved, and that record keeping procedures have been properly implemented.

Requirements Discovery Methods





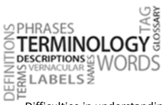
Interviewing

- Formal or informal interviews with stakeholders are part of most RE processes.
- Types of interview
  - Closed interviews based on pre-determined list of questions
  - Open interviews where various issues are explored with stakeholders.
- Effective interviewing
  - Be open-minded, avoid pre-conceived ideas about the requirements
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

Interviews in Practice

- Normally a mix of closed and open-ended interviewing.
- Good about interview
  - Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- Bad about interview
  - Not good for understanding domain requirements
  - Requirements engineers cannot understand specific domain terminology;
  - Some domain knowledge is so familiar that people find it hard to articulate or think that it isn't worth articulating.

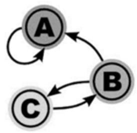

Understanding

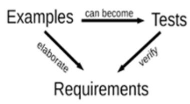


Difficulties in understanding domain terminology

Scenarios

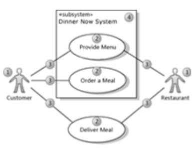
- Scenarios are real-life examples of how a system can be used.
- They should include
  - A description of the starting situation;
  - A description of the normal flow of events;
  - A description of what can go wrong;
  - Information about other concurrent activities;
  - A description of the state when the scenario finishes.

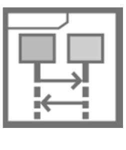




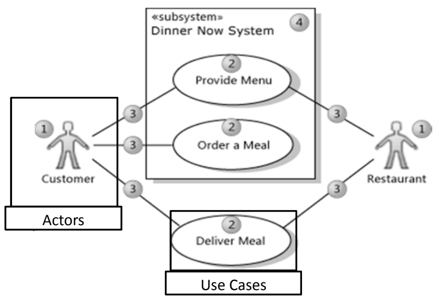
Use Cases

- Use-cases are a scenario based technique in the UML which identify the actors in an interaction and which describe the interaction itself.
- A set of use cases should describe all possible interactions with the system.
- High-level graphical model supplemented by more detailed tabular description
- Sequence diagrams may elaborate use-cases by adding sequence





Use Cases for a Dinner Order System



Ethnography

- Definition (Ethnographic Requirement Gathering)
  - Requirements that are derived from the way people actually working rather than the way at which process definitions suggest that they ought to work.
  - A social scientist spends a considerable time observing and analysing how people actually work.
- Shortcomings
  - Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

Scope of ethnography

Actual working of people

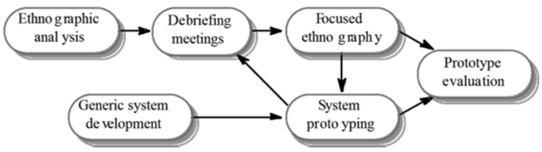
- Requirements that are derived from the way that people actually work

Cooperation and awareness of people

- Awareness of what other people are doing leads to changes in the ways in which we do things.

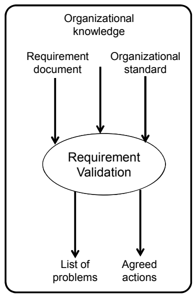
✧ Ethnography is effective for understanding existing processes but cannot identify new features that should be added to a system.

Ethnography and Prototyping for Requirements Analysis



Requirements Validation

- Definition (requirements validation)
  - The process of determining whether the requirements defined (i.e., documents, etc.) is complete as well as verifiable
  - Concerned with demonstrating that the requirements define the system that the customer really wants.
- Why requirements validation?
  - Requirements error costs are high so validation is very important
  - Fixing a requirements error after delivery may cost up to 100 times the cost of fixing an implementation error.



Requirements Checking

- **Validity**: Does the system provide the functions which best support the customer's needs?
- **Consistency**: Are there any requirements conflicts?
- **Completeness**: Are all functions required by the customer included?
- **Realism**: Can the requirements be implemented given available budget and technology
- **Verifiability**: Can the requirements be checked?

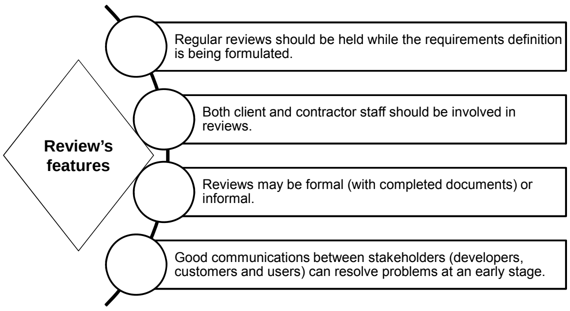


Requirements Validation Techniques

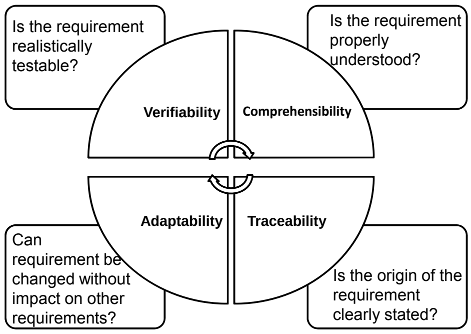
- **Review** — Requirements reviews: Systematic manual analysis of the requirements
- **Prototype** — Prototyping: Using an executable model of the system to check requirements. Covered in Chapter 2.
- **Test-case** — Test-case generation: Developing tests for requirements to check testability.



Requirements Reviews



Review Checks



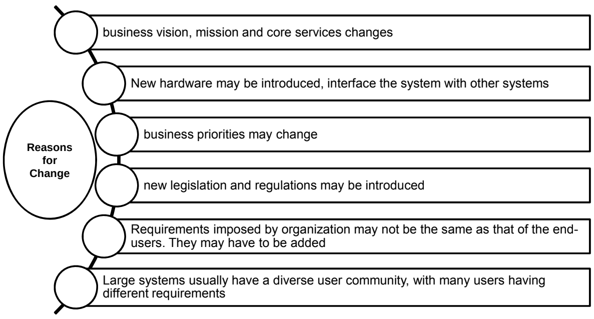
Requirements Management

- Definition (requirements management)
  - The process of managing changing requirements during the requirements engineering process and system development.
- Why we need RM?
  - New requirements emerge as a system is being developed and after it has gone into use.
- Maintaining change history
  - need to keep track of individual requirements and maintain links between dependent requirements so that you can assess the impact of requirements changes.

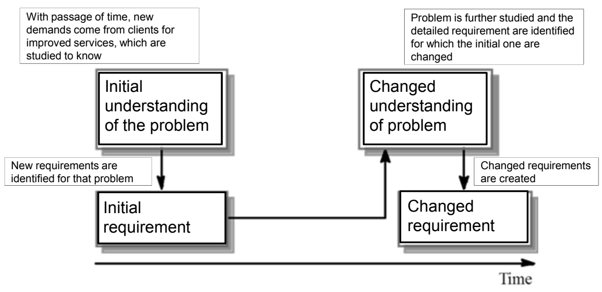
Changing Requirements

- The business and technical environment of the system always changes after installation.
- The business vision, mission and core services area changes with the rapid change in business world.
- This change in business strategy demands for change in requirements

Reasons for Change in Requirements



Requirements Evolution

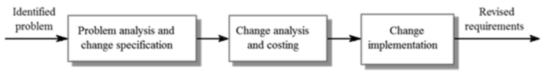


Requirements Management Planning

Requirements identification	*Requirements identification* • Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.
A change management process	*A change management process* • This is the set of activities that assess the impact and cost of changes. I discuss this process in more detail in the following section.
Traceability policies	*Traceability policies* • These policies define the relationships between each requirement and between the requirements and the system design that should be recorded.
Tool support	*Tool support* • Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

Requirements Change Management

- Deciding if a requirements change should be accepted



- The change is analyzed and checked for validity.
- The change requestor either further specifies or withdraw the request.

- The effect of the change is assessed.
- On completion, a decision is made whether or not to proceed with the requirements change.

- All the relevant documents are modified.
- The document are organized so that changes can be easily implemented.

Key points (Lecture 3)

- Requirement engineering process is an iterative process that includes elicitation, specification and validation.
- Requirements elicitation is a set of techniques used are: interviews, scenarios, use-cases and ethnography.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.
- Requirements changes business, organizational and technical changes inevitably lead to changes to the requirements for a software system.
- Requirements management is the process of managing and controlling these changes.

Hshs

kkll

Thanks