TOPIC 5

Requirements Engineering

- Requirements Engineering (4.1, 4.2)
- Requirements Engineering Processes (4.3-4.7)

Requirements Engineering

Requirements Types

- User requirements
  - High-level description of what the customer needs a system to do
  - Uses English statements and diagrams.
    - user's language
    - basis for contract bidding
- System requirements
  - Document detailing precisely what should be implemented
  - Often more formal and technical than the user requirements.
    - Developer's language
    - part of the actual contract

Requirements Types

User requirements definition

1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

System requirements specification

1.1 On the last working day of each month, a summary of the drugs prescribed, their cost and the prescribing clinics shall be generated.
1.2 The system shall generate the report for printing after 17.30 on the last working day of the month.
1.3 A report shall be created for each clinic and shall list the individual drug names, the total number of prescriptions, the number of doses prescribed and the total cost of the prescribed drugs.
1.4 If drugs are available in different dose units (e.g. 10mg, 20mg, etc) separate reports shall be created for each dose unit.
1.5 Access to drug cost reports shall be restricted to authorized users as listed on a management access control list.

Functional and Nonfunctional Requirements

- Functional requirements
  - Services the system should provide
  - May state what the system should not do

what the system should do & ↙

- Non-functional requirements
  - Constraints on the system such as timing, development process, or standards compliance. (usually applies to the whole system & not indivisual features)

→ App sleeping on render

→ if you push code it runs instantly we can't schedule

Functional Requirements

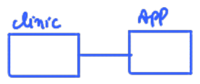
- Functional user requirements:

- Functional system requirements:

- Problems arise when requirements are not precisely stated

  – Ambiguous requirements may be interpreted differently by developers and users.

Functional Requirements



- Functional requirements for the MHC-PMS
  - A user shall be able to search the appointments lists for all clinics.
  - Each staff member shall be uniquely identified by his or her 8-digit employee number.
- Consider the term ‘search’:
  - User intention:
    - search for a patient across all appointments in all clinics.
  - Developer interpretation:
    - search for a patient in any one specific clinic.

Stakeholders - could refer to consumers but also other parties interested in the application.[external]

External - Developers, managers in the hospital

Requirements Completeness and Consistency

- In principle, requirements should be both complete and consistent.

Complete - describes all reqd functionality

Consistent - No conflicts/contradictions

eg→ collect student info, if we want to search a birthday, but legally you can't know someone's birthday

Practically impossible to produce a complete and consistent requirements document.

Non-functional Requirements

- Non-Functional requirements:
  - Defines system level properties and constraints
  - reliability, response time and storage space.
  - programming language or development method.
- Non-functional requirements may be more critical than functional requirements.
- If they are not met, the system may be useless.

Non-functional Requirements

prototyping is not good for non-functional

- Types:
  - Product Requirements

    efficiency, security, safety, up-time
  - Organizational Requirements

    workflow, operations,
  - External Requirements

    regulatory (laws)

Non-functional Requirements Implementation

- Non-functional requirements may affect the overall system architecture
  - Ex: organize system to minimize communication to meet performance requirements.
- A non-functional requirement may impose a number of functional requirements
  - Ex: security needs may dictate numerous features to meet those needs.

- encryption , no credit card info .

Non-functional requirements in MHC-PMS

- Product requirement
  - Specify that the delivered product must behave in a particular way

"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
  - Constraints from organizational policies

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

Non-functional requirements in MHC-PMS

- External requirement
  - From factors external to the system

  "The system shall implement patient privacy provisions as set out by law in HStan-03-2006-priv."

Non-functional Requirements Challenges

- Non-functional requirements may be
  - difficult to state precisely
  - Difficult to verify

- Goal: Write non-functional requirements quantitatively
  - User goal:

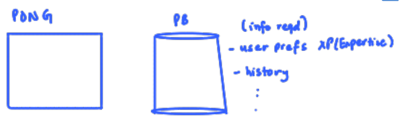
    "The system should be easy to use and organized such that user errors are minimized."
  - Verifiable requirement:

    "After 4 hours of training, average user error shall be less than 2 per hour."

Non-functional Requirements Metrics

Property	Measure
Speed	Response time
Size	Mb.
Ease of use	training time
Reliability	Rate of failure
Robustness	time to recover from failure

Example



	Functional Requirement	Non-Functional Requirement
User Requirement	- Initialize the game upto user XP	- Initialize game in a reasonable time
System Requirement	- Query DB for user info	- Query done in less than 100 ms

Example

	Functional Requirement	Non-Functional Requirement
User Requirement		
System Requirement		

Summary

- Requirements define
  - what the system should do and constraints on its operation and implementation.
- Functional requirements:
  - the services that the system must provide.
- Non-functional requirements:
  - constrain the system or development process.
  - Often relate to emergent properties of the system.
  - Apply to the system as a whole.

Domain Requirements Problems

Domain requirement problems typically stem from improper communication with the customer for example:

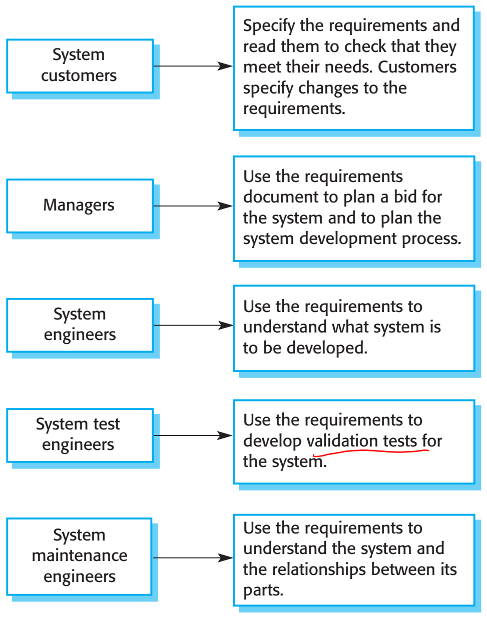
- Understandability Problems
- Implicitness Problems

Requirements Engineering Processes

Software Requirements Document

- Software requirements document:
  - Can Include Both:
    - a definition of user requirements and
    - a specification of the system requirements.
  - NOT a design document:

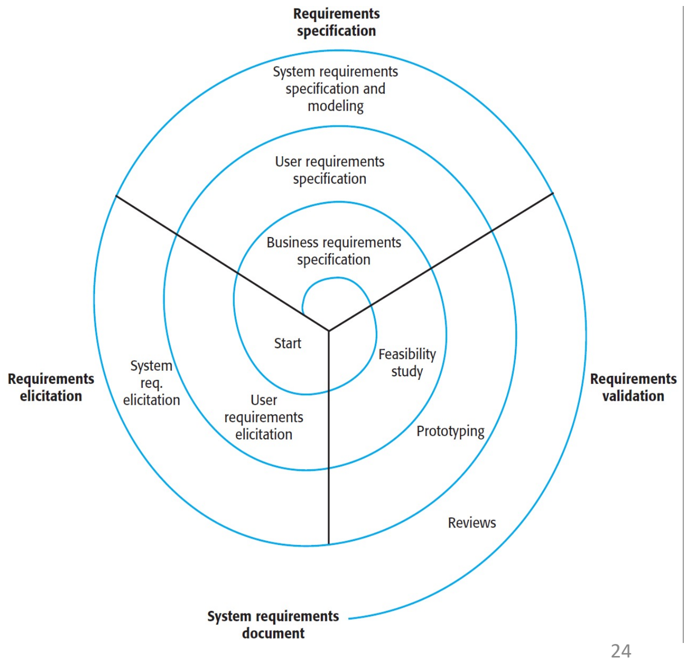
Users of a Requirements Document


System customers
Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers
Use the requirements document to plan a bid for the system and to plan the system development process.
System engineers
Use the requirements to understand what system is to be developed.
System test engineers
Use the requirements to develop validation tests for the system.
System maintenance engineers
Use the requirements to understand the system and the relationships between its parts.

Requirements Engineering Processes

- Generic activities common to all RE processes:
  - Requirements elicitation
  - Requirements analysis
  - Requirements validation
  - Requirements management

Spiral View of RE Processes


Requirements specification
System requirements specification and modeling
User requirements specification
Business requirements specification
Start
Feasibility study
Requirements elicitation
System req. elicitation
User requirements elicitation
Prototyping
Requirements validation
Reviews
System requirements document

Requirements Elicitation

Requirements Elicitation and Analysis

- Software developers work with a range of stakeholders to gather requirements
- Requirements Discovery:
  - Gathering information about the system and extracting user and system requirements

Requirements Elicitation Problems

- Stakeholders don’t know what they really want
- Stakeholders express requirements in their own terms.
- The requirements change during the analysis process.
- Different stakeholders may have conflicting requirements

- How can we elicit information from the customer?

Requirements Discovery Techniques

- Stakeholder interviews common in RE process.
- Types of interview
  – Closed Interviews
    - based on predetermined list of questions
  – Open interviews
    - explore various issues with stakeholders without use of predetermined questions

- Both are often used together.

Requirements Discovery Techniques

- Ethnography
  - Analyst immerses him/herself in work environment where system will be used.
  - Analyst observes current workflow;
    people don't explain it to him/her.

Requirements Discovery Scenarios

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

Scenario: collecting medical history

Starting Situation:

Patient seen by receptionist who created record in system and collected patient's personal information (name, address, age).
A nurse is logged in and is collecting medical history.

Normal Workflow:

Nurse searches for the patient by family name. If more than one patient returned, use given name and date of birth.

Nurse chooses the menu option to add medical history.

Nurse follows a series of prompts from the system to enter
- other consultations on mental health problems (free text),
- existing medical conditions (selects conditions from menu),
- medication currently taken (selected from menu),
- allergies (free text), and home life (form).

Scenario: collecting medical history

What can go wrong:

Patient's record not exist: nurse creates a new record.

Patient conditions or medication not in menu: nurse chooses 'other' and enter free text describing the condition/medication.

Patient cannot/will not provide information on medical history: nurse records patient's refusal and prints exclusion form.

Concurrent activities:

Others can read but not edit record while being entered.

System state on completion:

User is logged on.
Patient record (with medical history) entered in the database,
System log shows nurse, start and end time of the session.

UML: Unified Modeling Language

- Object-Oriented Paradigm modelling notation
- Clear and effective way to model many aspects of a software system using a commonly understood language
- Programming language independent
- Enables a variety or analysis and design techniques
- A subset of UML will be used in this course
  - Use Case Diagrams are used to model system functionality

Requirements Discovery: Use Cases

- Use-cases are a scenario based technique
- To model interactions in UML
  - Graphically shows actors in interaction with:
    - Success cases
    - Failure cases
- The set of use cases should describe all possible interactions with the system.
  - Does not show sequence of actions.

Use Case: Actor

- Entity outside the software system
  - interacts with the system
  - Operates on objects in the system but cannot be operated upon by objects in the system.

- Represents coherent role played by users

Use Case: Actor

- A user of software system may take on more than 1 role, usually at different times

- An actor may represent more than one user

Primary and Secondary Actors

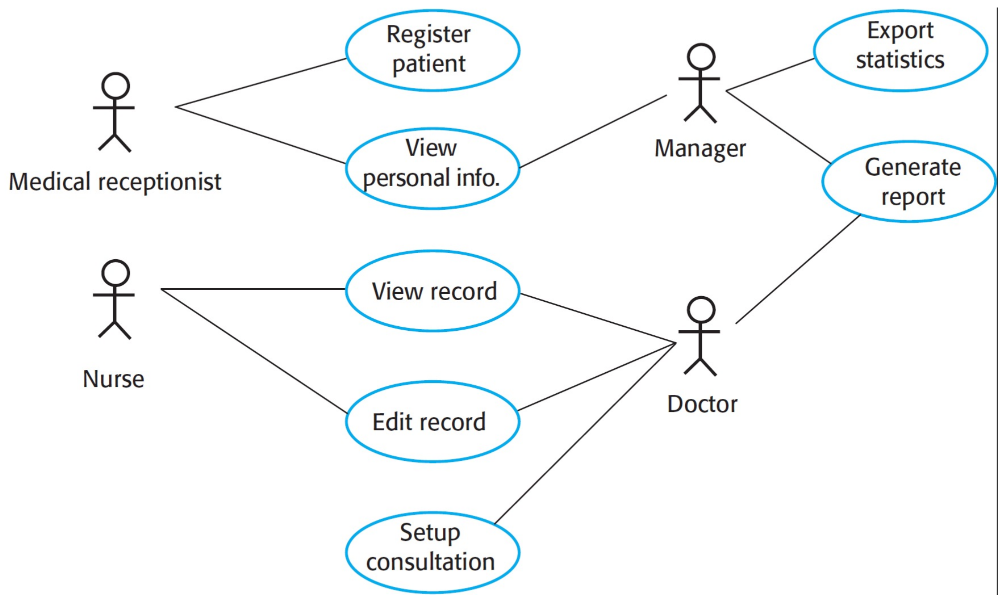
- Primary Actors
  - Actors who initiate a scenario (use case) causing the system to achieve a goal

- Secondary Actors
  - Actors supporting the system so primary users goals can be completed (do not initiate the use case or scenario)

Primary and Secondary Actors

- It is possible for an Actor to be a primary (initiating) actor in one scenario and a (non-initiating) or secondary actor in another scenario in the same system

MHC-PMS Use Cases



Requirements Specification

Writing System Requirements Specification

Notation	Description
Natural language sentences	Requirements are numbered sentences in plain English.
Structured natural language	Requirements are in English, but written based on a form or template
Graphical Notations	Diagrams and text to describe the system.
Mathematical specification	Unambiguous, but hard for customers to understand.

Natural Language

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 Specification: Insulin Pump

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.

Structured Specification: Insulin Pump

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.

Tabular Specification: Insulin Pump

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

The structure of a requirements document

Chapter	**Description**
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the 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	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
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 Validation

Requirements Validation

- The process of demonstrating that the requirements collected define the system that the customer really wants
- Checks:
  - Validity
  - Consistency
  - Completeness
  - Realism
  - Verifiability

Requirements Validation Techniques

- Requirements reviews
  - Systematic manual analysis of the requirements
  - Involve both developers and customer while requirements are being formulated.

- Prototyping
  - Using an executable model of the system to check requirements.
- Test-case generation
  - Developing tests for requirements to feasibility

Requirements Change Management

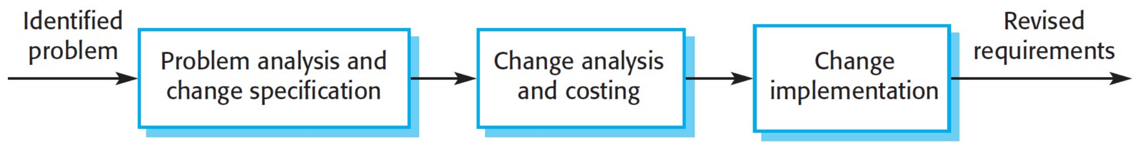
Requirements Management Planning

Decisions to be made at the planning stage:

Requirement identification	Each requirement must be uniquely identified so that it can be cross-referenced with other requirements
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	These policies define the relationships between each requirement and between the requirements and the system design that should be recorded
Tool Support	Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

Requirements Management

- Activity during the requirements engineering process and system development.
- Set of activities to assess impact and cost of changes.


Identified problem
Problem analysis and change specification
Change analysis and costing
Change implementation
Revised requirements