COMP 433 Software Engineering

Module 3: *Requirements Engineering*

Ahmed Tamrawi

atamrawi atamrawi.github.io ahmedtamrawi@gmail.com

Acknowledgment Notice

Part of the slides are based on content from Ian Somerville's Software Engineering book.

Requirements Engineering

- *The process of establishing the services that a customer requires from a system and the constraints under which it operates and is developed.*
- These requirements reflect the **needs of customers** for a system that serves a certain purpose such as controlling a device, placing an order, or finding information.
- The system requirements are the descriptions of the system services and constraints that are generated during the requirements engineering process.

What is a Requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a **detailed mathematical functional specification**.
  - The requirements are first written in **high-level**. *This provides a chance for other software companies to provide detailed solutions for the customer organization*.
  - Later on, a **detailed description** of the system requirements should be generated. *These detailed requirements serve as a contract between development company and customer.*
- This is inevitable as requirements may serve a dual function:
  - May be the basis for a bid for a contract - therefore must be open to interpretation;
  - May be the basis for the contract itself - therefore must be defined in detail;
  - Both these statements may be called requirements.

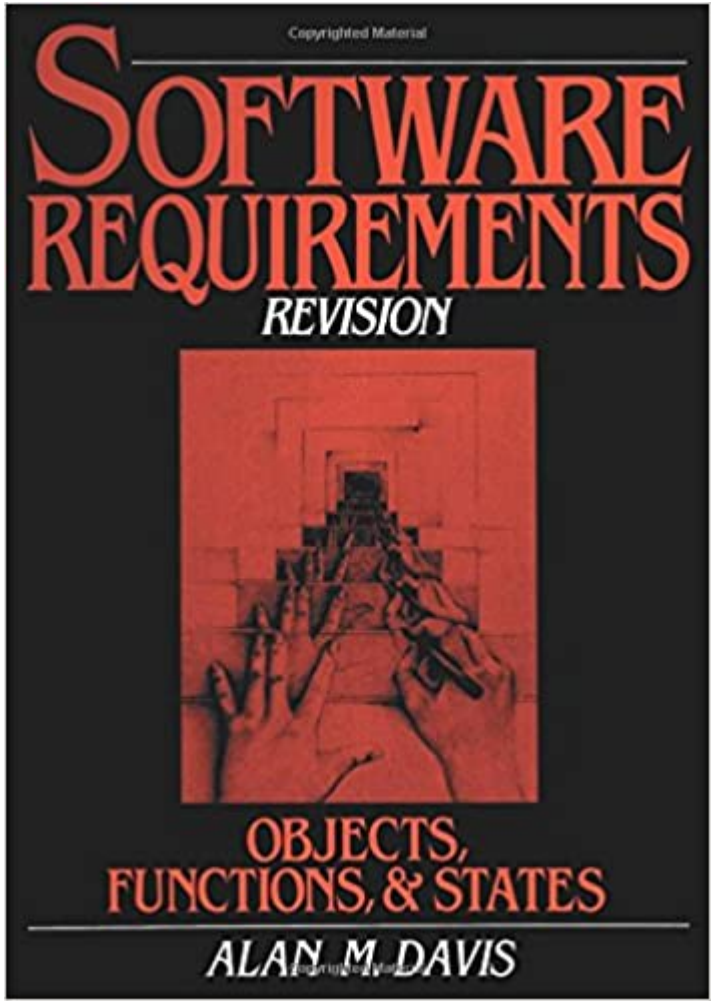
Requirements Abstraction

"If a company wishes to let a contract for a large software development project, ***it must define its needs in a sufficiently abstract way that a solution is not pre-defined****.*

The requirements must be written so that ***several contractors can bid for the contract****, offering, perhaps, different ways of meeting the client organization's needs.*

Once a contract has been awarded, the contractor must write a system definition for the client in more detail so that the client understands and can validate what the software will do.

Both documents may be called the requirements document for the system."



Types of Requirement

- **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.***

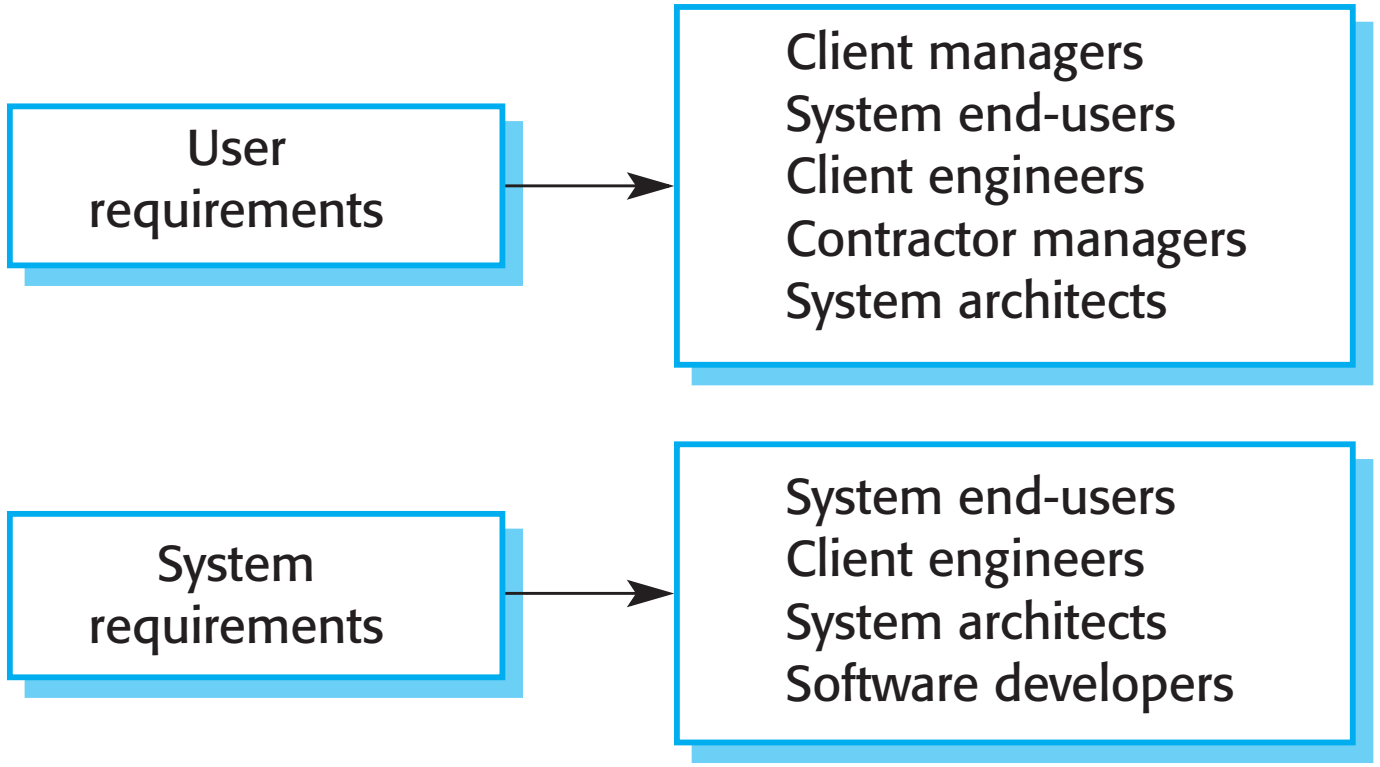
User requirements definition

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.

Readers of Requirements Specification



System Stakeholders

- *Any person or organization who is affected by the system in some way and so who has a legitimate interest*
- Stakeholder types:
  - End users
  - System managers
  - System owners
  - External stakeholders

Stakeholders in the Mentcare System

- **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.
- 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.

Functional and Non-functional Requirements

Functional and Non-functional Requirements

- **Functional requirements**
  - Statements of services the system should provide, how the system should react to particular inputs and how the system should behave in particular situations.
  - May state what the system should not do.
- **Non-functional requirements**
  - Constraints on the services or functions offered by the system such as timing constraints, constraints on the development process, standards, etc.
  - Often apply to the system as a **whole** rather than individual features or services.
- **Domain requirements**
  - Constraints on the system from the domain of operation

Functional Requirements

- Describe **functionality** or **system services**.
- Depend on the type of software, expected users and the type of system where the software is used.
- Functional **user requirements** may be *high-level statements of what the system should do*.
- Functional **system requirements** should *describe the system services in detail*.

Mentcare System: *Functional Requirements*

- **UR.1** A user shall be able to **search** the appointments lists for all clinics.
- **UR.2** The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
- **UR.3** Each staff member using the system shall be uniquely identified by his or her 8-digit employee number.

Requirements Imprecision

- Problems arise when functional requirements are *not precisely stated*.
- **Ambiguous** requirements may be interpreted in different ways by developers and users.
- Consider the term ‘search’ in requirement UR.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

- In principle, requirements should be **correct**, **complete** and **consistent**.
  - **Correct**: they should add value to the customer.
  - **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, because of system and environmental complexity, it is impossible to produce a complete and consistent requirements document.

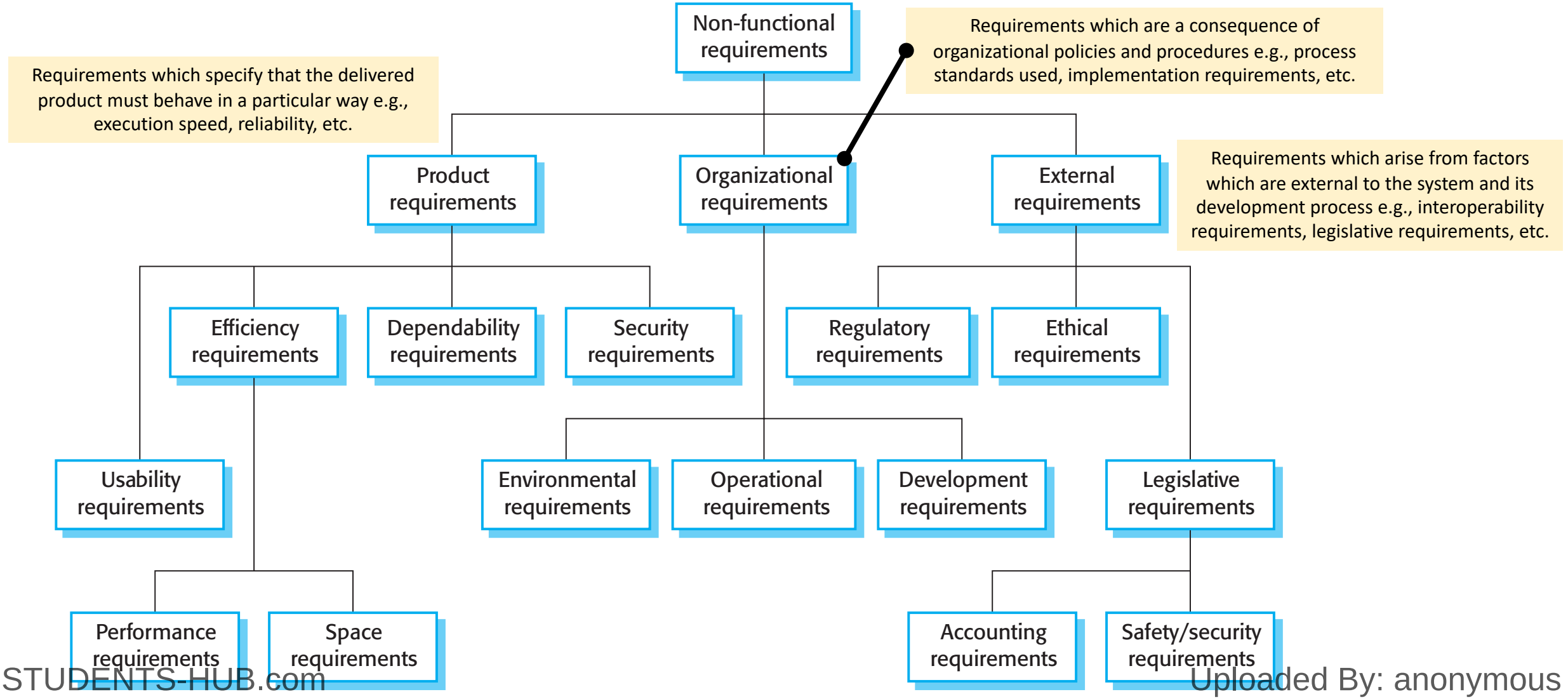
Non-functional Requirements

- These define **system properties and constraints** e.g. reliability, response time and storage requirements. Constraints are I/O device capability, system representations, etc.
- Process requirements may also be specified mandating a particular IDE, programming language or development method.
- *Non-functional requirements may be more critical than functional requirements*. If these are not met, the system may be useless.

Non-functional Requirements Implementation

- Non-functional requirements may affect the **overall architecture** of a system rather than the individual components.
  - For example, to ensure that **performance** requirements are met, you may have to organize the system to minimize communications between components.
- A single non-functional requirement, such as a security requirement, may generate several related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

Types of Non-functional Requirement



Examples of Non-functional Requirements in the Mentcare System

Product requirement
The Mentcare system 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 Mentcare 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 difficult to state precisely and imprecise requirements may be difficult to verify.
- Goal: A general intention of the user such as ease of use.
- Verifiable non-functional requirement
  - A statement using some measure that can be objectively tested.
- Goals are helpful to developers as they convey the intentions of the system users.

Usability Requirements

- Non-Functional Requirements Should be Quantifiable
  - The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (**Goal**)
  - 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. (**Testable non-functional requirement**)

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

Ambiguous Terms to Avoid in Requirements Specifications

acceptable, adequate	Define what constitutes **acceptability** and how the system can judge this.
as much as, practicable	Don't leave it up to the developers to determine what's **practicable**. Make it TBD and set a date to find out
at least, at a minimum, not more than, not to exceed	Specify the **minimum** and **maximum** acceptable values.
between	Define whether the **end points** are included in the range.
depends on	Describe the nature of the **dependency**. Does another system provide input to this system, must other software be installed before your software can run, or does your system rely on another one to perform some calculations or services?
efficient	Define how **efficiently** the system uses resources, how quickly it performs specific operations, or how easy it is for people to use?
fast, rapid	Specify the **minimum acceptable speed** at which the system performs some action.
flexible	Describe the ways in which the system must change in response to changing conditions or business needs.
improved, better, faster, superior	Quantify how much **better** or **faster** constitutes adequate improvement in a specific functional area.
including, including but not limited to, and so on, etc., such as	The list of items should include **all possibilities**. Otherwise, it can't be used for design or testing.
maximize, minimize, optimize	State the **maximum** and **minimum** acceptable values of some parameter.

Domain Requirements

- Derived from the **application domain** and describe system **characteristics** and **features** that reflect the **domain**.
- May generate new functional requirements, (non-functional) constraints on existing requirements or define specific computations.
- If domain requirements are not satisfied, the system may be unworkable!

- **Example 1:**
  R.7.1 a train control system has to take into account the braking characteristics in different weather conditions.

- **Example 2:**
  8.4.3 a PMS has to enforce all confidentiality rules in accordance with national medical domain practices

Domain Requirements Problems

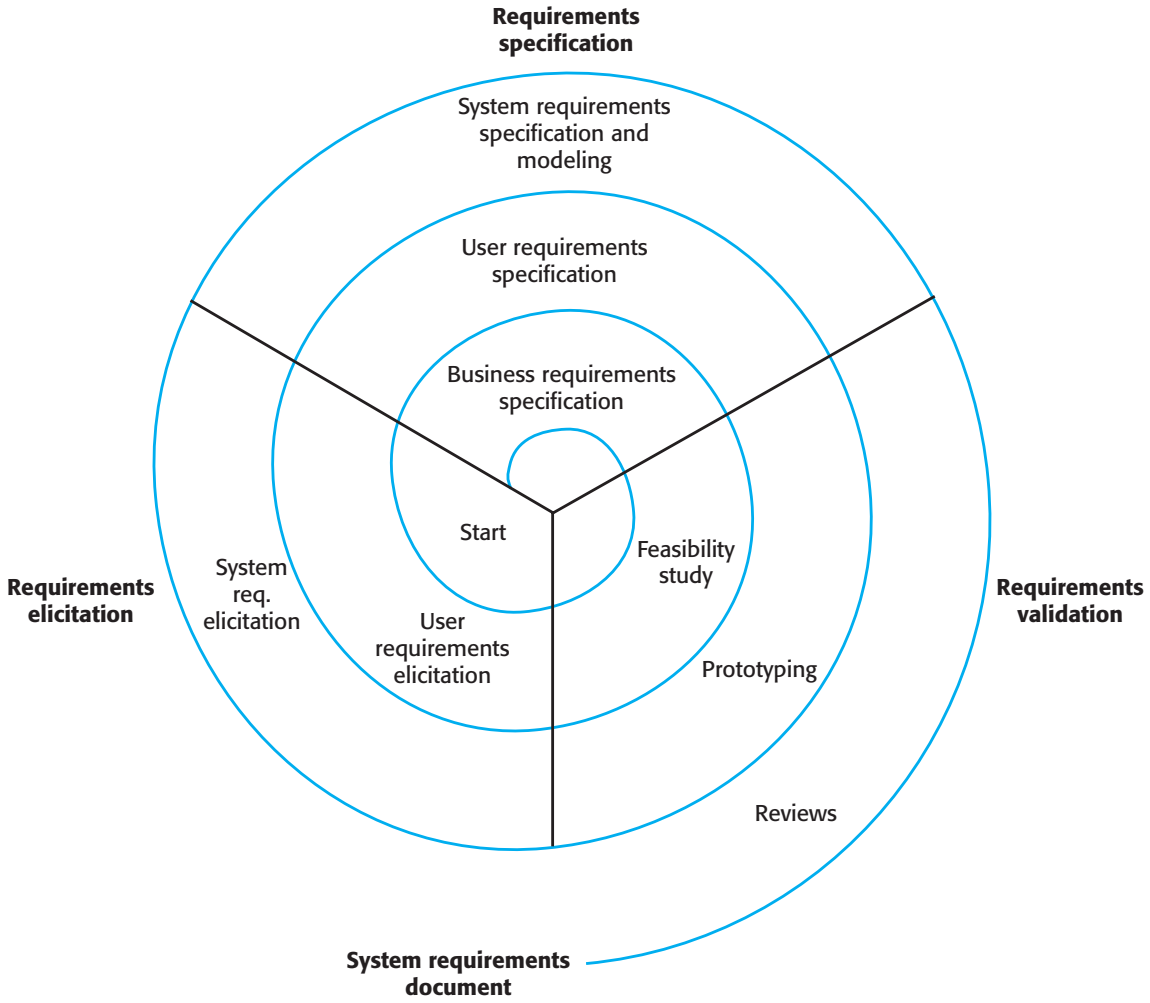
- **Understandability**
  - Requirements are expressed in the language of the application domain. This is often not understood by software engineers developing the system.
- **Implicitness**
  - Domain specialists understand the area so well that they do not think of making the domain requirements explicit.

Requirements Engineering Processes

Requirements Engineering Processes

- The processes used for RE vary widely depending on the **application domain**, the **people** involved and the **organization** developing the requirements.
- However, there are a number of generic activities common to all processes:
  - Requirements elicitation and analysis;
  - Requirements validation;
  - Requirements management.
- In practice, RE is an **iterative activity** in which these processes are interleaved.

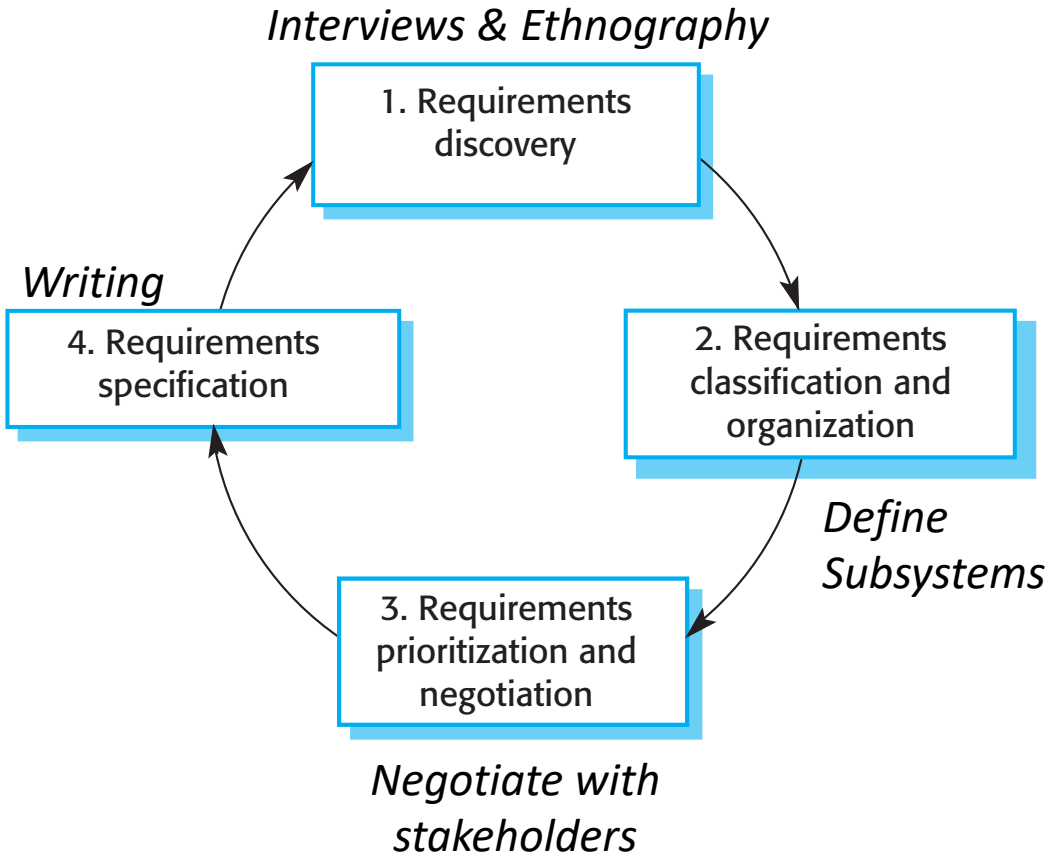
Spiral View of Requirements Engineering Process



Requirements Elicitation and Analysis

Requirements Elicitation and Analysis

- Sometimes called **requirements elicitation** or **requirements discovery**.
- Involves technical staff working with customers to find out about the application **domain**, the **services** that the system should **provide** and the **system's operational constraints**.
- May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc. These are called *stakeholders.*



Process Activities

- **Requirements Discovery**
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage. (Interviewing and Ethnography)
- **Requirements Classification and Organization**
  - Groups related requirements and organizes them into coherent clusters.
- **Prioritization and Negotiation**
  - Prioritizing requirements and resolving requirements conflicts.
- **Requirements Specification**
  - Requirements are documented and input into the next round of the spiral.

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 may change.

Requirements Elicitation and Analysis

- There are many requirement engineering techniques for requirement elicitation and analysis, some of the often-used ones:
  - Interviewing
  - Scenario generation
  - Use case analysis
  - Ethnography

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.
  - Focused interviews, with clusters of stakeholders
- Effective interviewing
  - Be open-minded, avoid pre-conceived ideas about the requirements and are willing to listen to stakeholders.
  - Prompt the interviewee to get discussions going using a springboard question, a requirements proposal, or by working together on a prototype system.

Interviewing

- Meeting introductory protocol
  - Ensure cultural introduction protocols are followed
- First meeting
  - Aim: to understand the business and its context with a clear aim to understand business processes and services.
- Effective meetings:
  - Ensure a chair is assigned at the beginning, to keep time-controlled progress
  - Ensure an agenda is defined with clear objectives of the target outcome of the meeting
  - Ensure a timescale is set for each agenda item and is kept/controlled by the chair
  - Ensure clear actions and decisions (and who is responsible for and by when) are identified and reached by the end of the meeting
  - Ensure the actions and decisions are summarized at the end of the meeting

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.

Scenario for Collecting Medical History

INITIAL ASSUMPTION:
The patient has seen a medical receptionist who has created a record in the system and collected the patient's personal information (name, address, age, etc.). A nurse is logged on to the system and is collecting medical history.

NORMAL:
The nurse searches for the patient by family name. If there is more than one patient with the same surname, the given name (first name in English) and date of birth are used to identify the patient.

The nurse chooses the menu option to add medical history.

The nurse then follows a series of prompts from the system to enter information about consultations elsewhere on mental health problems (free text input), existing medical conditions (nurse selects conditions from menu), medication currently taken (selected from menu), allergies (free text), and home life (form).

WHAT CAN GO WRONG:
The patient's record does not exist or cannot be found. The nurse should create a new record and record personal information.

Patient conditions or medication are not entered in the menu. The nurse should choose the 'other' option and enter free text describing the condition/medication.

Patient cannot/will not provide information on medical history. The nurse should enter free text recording the patient's inability/unwillingness to provide information. The system should print the standard exclusion form stating that the lack of information may mean that treatment will be limited or delayed. This should be signed and handed to the patient.

OTHER ACTIVITIES:
Record may be consulted but not edited by other staff while information is being entered.

SYSTEM STATE ON COMPLETION:
User is logged on. The patient record including medical history is entered in the database, a record is added to the system log showing the start and end time of the session and the nurse involved.

Ethnography

- A social scientist spends a considerable time observing and analyzing how people work.
- People do not have to explain or articulate their work.
- Social and organizational factors of importance may be observed.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

Requirements Specification

- *The process of writing down the user and system requirements in a requirements document.*
- At this stage, focus on **what** the system should do, not **how** the system will do it (design)
- **User requirements** must be understandable by end-users and customers who do not have a technical background.
- **System requirements** are more detailed requirements and may include more technical information.
- The requirements may be part of a contract for the system development
  - It is therefore important that these are as complete as possible.

Writing System Requirements Specification

Notation	Description
Natural language sentences	The requirements are written using numbered sentences in natural language. Each sentence should express one requirement.
Structured natural language	The requirements are written in natural language on a standard form or template. Each field provides information about an aspect of the requirement.
Design description languages	This approach uses a language like a programming language, but with more abstract features to specify the requirements by defining an operational model of the system. This approach is now rarely used although it can be useful for interface specifications.
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	These notations are based on mathematical concepts such as finite-state machines or sets. Although these unambiguous specifications can reduce the ambiguity in a requirements document, most customers don't understand a formal specification. They cannot check that it represents what they want and are reluctant to accept it as a system contract.

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.*)

Guidelines for Writing Requirements

- Invent a standard format and use it for all requirements
- Use language in a consistent way.
  - Use **shall** for mandatory (or forceful) requirements,
  - Use **should** for desirable requirements
  - Use text highlighting to identify key parts of the requirement
- Include an explanation (rationale) of why a requirement is necessary
- Avoid the use of computer jargon!
- In principle, **requirements** should state **what** the system should do, and the **design** should describe **how** the system does it.

Problems with Natural Language

- **Lack of clarity**
  - Precision is difficult without making the document difficult to read.
- **Requirements confusion**
  - Functional and non-functional requirements tend to be mixed-up.
- **Requirements amalgamation**
  - Several different requirements may be expressed together.

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.
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.

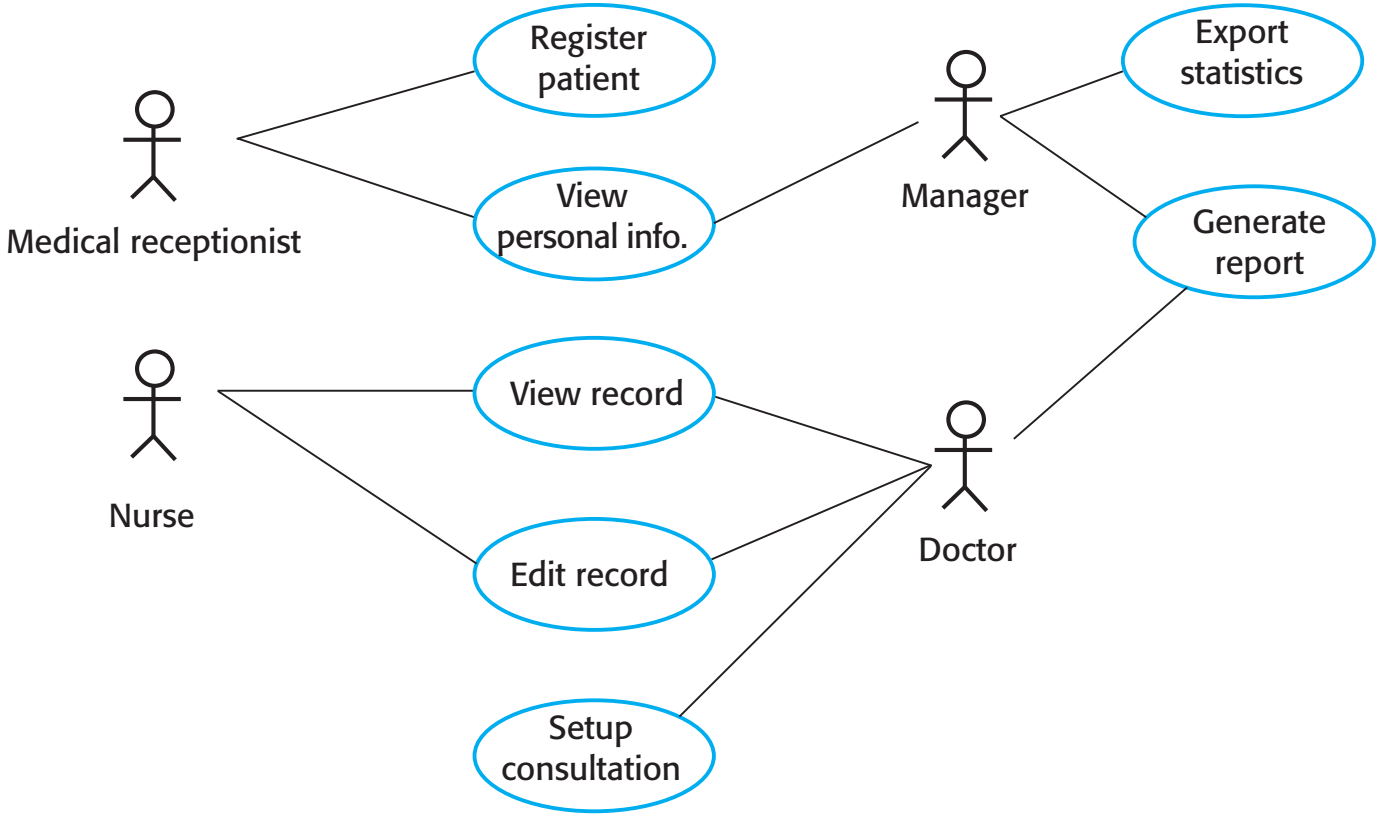
Tabular Specification of Computation for an Insulin Pump

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

Use Cases

- Use-cases are a kind of scenario that are included in the UML.
- Use cases 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.
- **UML sequence diagrams** may be used to add detail to use-cases by showing the sequence of event processing in the system.

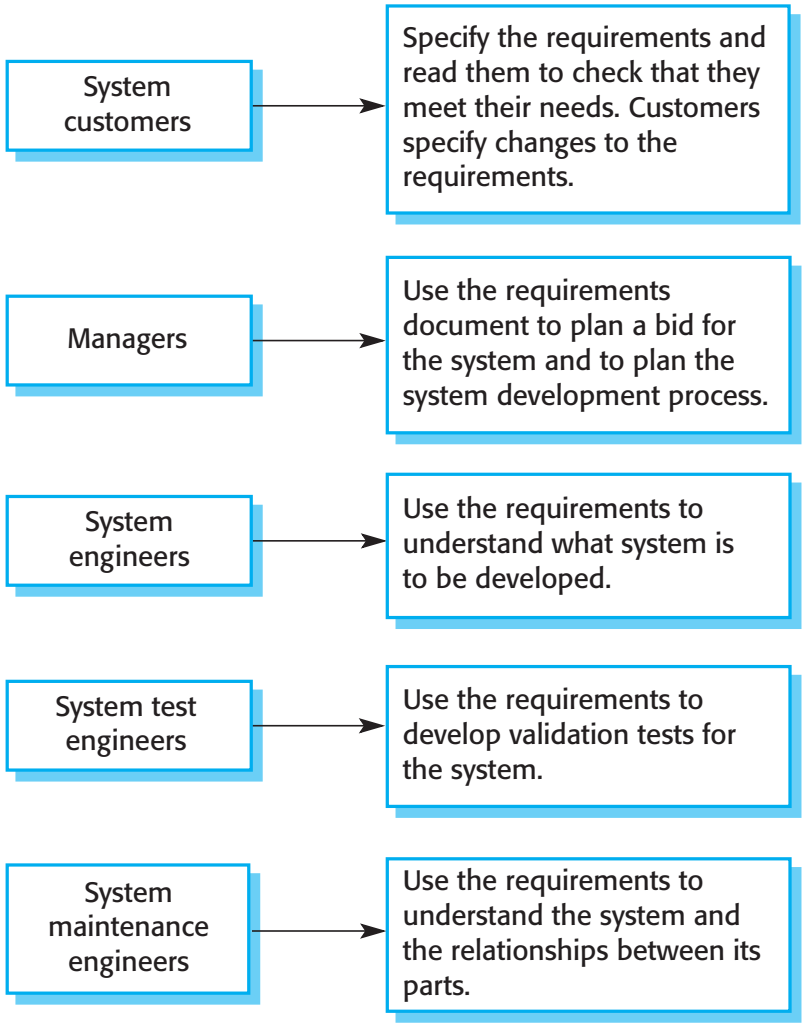
Use Cases for the Mentcare System



The Software Requirements Document

- The software requirements document is the **official statement of what is required of the system developers**.
- Should include both a **definition of user requirements** and a **specification of the system requirements**.
- It is **NOT** a design document. As far as possible, it should set of **WHAT** the system should do rather than **HOW** it should do it.

Users of a Requirements Document



Requirements Document Variability

- Information in requirements document depends on type of system and the approach to development used.
- Systems developed incrementally will, typically, have less detail in the requirements document.
- Requirements documents standards have been designed e.g., IEEE standard. These are mostly applicable to the requirements for large systems engineering projects.

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.

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

- *Concerned with demonstrating that the requirements define the system that the customer really wants.*
- 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

- **Requirements reviews**: *Systematic manual analysis of the requirements.*
  - **Verifiability**: *Is the requirement realistically testable?*
  - **Comprehensibility**: *Is the requirement properly understood?*
  - **Traceability**: *Is the origin of the requirement clearly stated?*
  - **Adaptability**: *Can the requirement be changed without a large impact on other requirements?*
- **Prototyping**
  - Using an executable model of the system to check requirements.
- **Test-case generation**
  - Developing tests for requirements to check testability.

In-class Activity

Requirements Discovery and Analysis

Activity: *Requirements Analysis*

- For the upcoming sets of requirements, answer the following questions:
  - What is the type of each of the set of requirements?
  - Validate each requirement set based on the following characteristics, identify the ones that do not validate.
    - Correctness
    - Unambiguous
    - Completeness
    - Consistency
    - Traceability
    - Realistic/Feasibility

Small Library Information System

- **UR 1** The library system should allow users to search for books online and see their availability and status.
  - **SR 1.1** System users namely librarians, members, and managers shall be able to search for and view books information.
  - **SR 1.2** System users can search for books using any combination of book title, author, category, and year.
  - **SR 1.3** The search results should contain the following book information: title, year, authors, number of pages and book status.
  - **SR 1.4** System shall show results in no more than 3 seconds.
  - **SR 1.5** Book status can be one of the following: available, rented, reserved, or pending.
  - **SR 1.6** Clicking on a book title, the system shall display complete book information along with book cover picture.

Video Rental System

- **UR 3.8** The system shall email due notices to members.
  - **SR 3.8.1** The system shall send due notices one day before due date of any rented video.
  - **SR 3.8.2** The system shall send overdue notices every third day after the due date.
  - **SR 3.8.3** The system shall email a 60-day notice of charges for non-returned videos.
  - **SR 3.8.4** The due notice shall display the currently due videos, the elapsed due time, and corresponding charges as well as the videos that are due within 2 days of the date of the notice.

XML Editor

- **UR1** The XML Editor shall switch between displaying and hiding nonprinting characters instantaneously.
  - **SR 1.1** The user shall be able to toggle between displaying and hiding all XML tags in the document being edited with the activation of a specific triggering mechanism.
  - **SR 1.2** The display shall change in 0.1 second or less.
- **UR2** The XML parser shall produce a markup error report that allows quick resolution of errors when used by XML novices.
  - **SR 2.1** After the XML Parser has completely parsed a file, it shall produce an error report that contains the line number and text of any XML errors found in the parsed file and a description of each error found.
  - **SR 2.2** If no parsing errors are found, the parser shall not produce an error report.
- **UR3** The editor shall not offer search and replace options that could have disastrous results.
  - **SR 3.1** The editor shall require the user to confirm global text changes, deletions, and insertions that could result in data loss.
  - **SR 3.2** The application shall provide a multilevel undo capability limited only by the system resources available to the application.

Student Registration Subsystem

- **UR 1** The system shall allow the Registrar to modify student information in the registration system.
  - **SR 1.1** The system shall allow the Registrar to search for an existing student using student id number.
  - **SR 1.2** In case the id does not correspond to an existing student, the system should display an error message. The Registrar can then type in a different id number or cancel the operation.
  - **SR 1.3** In case the entered student id corresponds to an exiting student, the system retrieves the student information and displays it on the screen.
  - **SR 1.4** Student information include name, date of birth, student id, graduation date, and address.
  - **SR 1.5** The Registrar can modify one or more of the student information fields: name, date of birth, student id number, status, graduation date, and address.
  - **SR 1.6** When changes are complete, the Registrar selects "save."
  - **SR 1.7** The system validates data, then updates the student information.

Safe Home System

- **UR 2** System detects an emergency, sends an alert to the correct authority and primary caregiver.
  - **SR 2.1** The system detects health emergency via its constant monitoring sensors defined in **UR 1**.
  - **SR 2.2** The system asks the customer if the user needs assistance as in **UR 45**.
  - **SR 2.3** If the customer tells the system to send an alert:
    - **SR 2.3.1** The system notifies local authorities and primary caregiver that the customer needs assistance as stated in **UR 30**.
    - **SR 2.3.2** The system notifies the customer that authorities have been notified.
  - **SR 2.4** If the customer informs the system not to send an alert:
    - **SR 2.4.1** The system continues to monitor emergency as in **UR 1**.
    - **SR 2.4.2** The system re-alerts if no action is taken as in **UR 2**.
  - **SR 2.5** If the customer does not respond to the system, the system notifies local authorities and primary caregiver as stated in **UR 30**.

Requirement Set 1

- The system shall provide a service for users (students) to register and create an account.
- Users shall be able to submit queries
- The system shall adhere to the guidelines set by the ministry of higher education
- Users should be able to listen to music when using the website
- The system shall allow only the registered users to use the services of the website.

Requirement Set 2

- **UR1** Registered users shall be able to submit a new application to study at the university during normal working hours, adhering to the education submission procedures.
- **UR2** The system shall create an application template and opens it in a new web page when users press new application button, from the “create new application” web page. The template should have the following data fields: Full Name, DoB, address, telephone numbers, Tawjihi Grade, and three Subjects to be studied in the order of preference.
- **UR3** The system shall fill the template automatically and detect and extract, intelligently, user details, as per the template, using advanced detection technologies, e.g. biometric.
- **UR4** The system shall check all the applications’ data fields are complete and valid before submission and within a reasonable time. The system shall check number data fields contain only number values and text data fields contain at least some text values.

Requirement Set 3

- The Background Task Manager shall provide status messages at regular intervals not less than every 60 seconds.

Requirement Set 4

- **UR1** The Background Task Manager (BTM) shall display status messages in a designated area of the user interface.
  - **SR1.1** The messages shall be updated every 60 plus or minus 10 seconds after background task processing begins.
  - **SR1.2** The messages shall remain visible continuously.
  - **SR1.3** Whenever communication with the background task process is possible, the BTM shall display the percent completed of the background task.
  - **SR1.4** The BTM shall display a "Done" message when the background task is completed.
  - **SR1.5** The BTM shall display a message if the background task has stalled.

Requirements Change

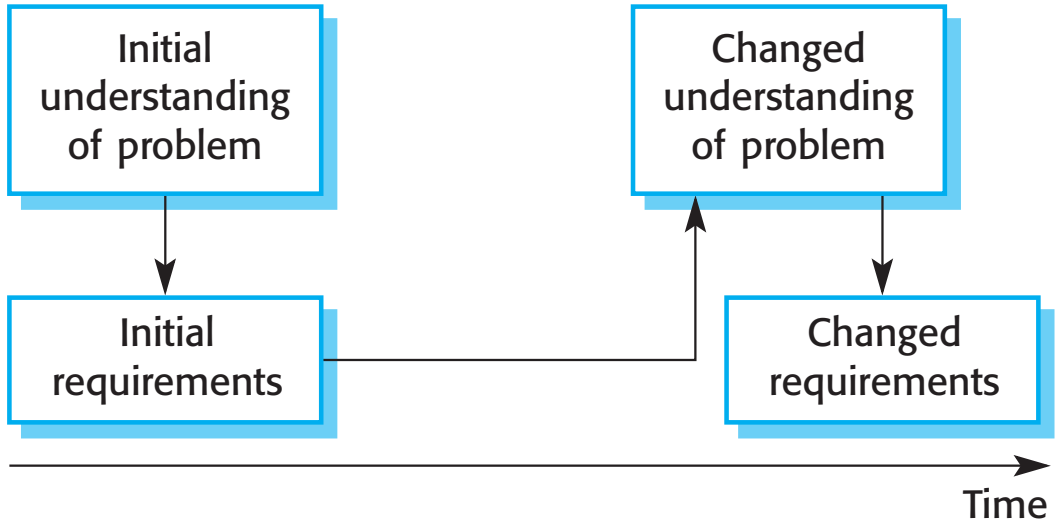
Changing Requirements

- The business and technical environment of the system always changes after installation.
  - New hardware may be introduced, it may be necessary to interface the system with other systems, business priorities may change (with consequent changes in the system support required), and new legislation and regulations may be introduced that the system must necessarily abide by.
- The people who pay for a system and the users of that system are rarely the same people.
  - System customers impose requirements because of organizational and budgetary constraints. These may conflict with end-user requirements and, after delivery, new features may have to be added for user support if the system is to meet its goals.

Changing Requirements

- Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.
  - The final system requirements are inevitably a compromise between them, and, with experience, it is often discovered that the balance of support given to different users has to be changed.

Requirements Evolution



Requirements Management

- Requirements management is *the process of managing changing requirements during the requirements engineering process and system development*.
- New requirements emerge as a system is being developed and after it has gone into use.
- You need to keep track of individual requirements and **maintain links between dependent requirements** so that you can assess the impact of requirements changes.
- You need to establish a **formal process for making change proposals and linking these to system requirements**.

Requirements Management Planning

- Establishes the level of requirements management detail that is required.
- Requirements management decisions:
  - *Requirements identification* Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.
  - *A change management process* This is the set of activities that assess the impact and cost of changes.
  - *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 Change Management

- Deciding if a requirements change should be accepted
  - *Problem analysis and change specification*
    - During this stage, the problem or the change proposal is analyzed to check that it is valid. This analysis is fed back to the change requestor who may respond with a more specific requirements change proposal or decide to withdraw the request.
  - *Change analysis and costing*
    - The effect of the proposed change is assessed using traceability information and general knowledge of the system requirements. Once this analysis is completed, a decision is made whether or not to proceed with the requirements change.
  - Change implementation
    - The requirements document and, where necessary, the system design and implementation, are modified. Ideally, the document should be organized so that changes can be easily implemented.

Requirements Change Management