Chapter 4 – Requirements Engineering

Topics covered

- ✧ Business requirements
- ✧ User requirements
- ✧ Functional and non-functional requirements
- ✧ Requirements engineering processes
- ✧ Requirements elicitation
- ✧ Requirements specification
- ✧ Requirements validation
- ✧ Requirements change

Requirements

- ✧ Requirements are the descriptions of the services provided by the system and its operational constraints.
- ✧ The process of finding out, analyzing, documenting and checking these services and constraints is called requirements engineering (RE)

Requirements engineering

- ✧ Every software development process goes through a requirements engineering phase.
- ✧ The process of establishing the services that a customer requires from a system and the constraints under which it operates and is developed.
- ✧ Requirements describe goals or tasks the users must be able to perform with the product that will provide value to someone.
- ✧ 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.
- ✧ 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.

Why Requirements?

- What are the advantages of a complete set of documented requirements?
  - Ensures that the user (not the developer) drives system functionality
  - Helps avoid confusion and arguments about development process
  - Helps minimize changes after development begins

Why Requirements?

- ✧ Changes in requirements
  - ▪ Requirements changes are expensive: it costs
    - 3 times as much during the design phase
    - 5-10 times as much during implementation
    - 10-100 times as much after release
  - ▪ A careful requirements process doesn't mean there will be no changes later
    - Average project experiences about a 25% change in requirements
    - This accounts for 70-80 % of the rework on the project
    - Important to plan for requirements changes

User and System req.

Requirements may be at different levels of abstraction:

✧ User requirements: from user's point of view

- Statements in natural language plus diagrams of the services the system provides and its operational constraints. Written for customers.

✧ System requirements: from developer's point of view

- 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.
- Also called functional specification

User and system requirements

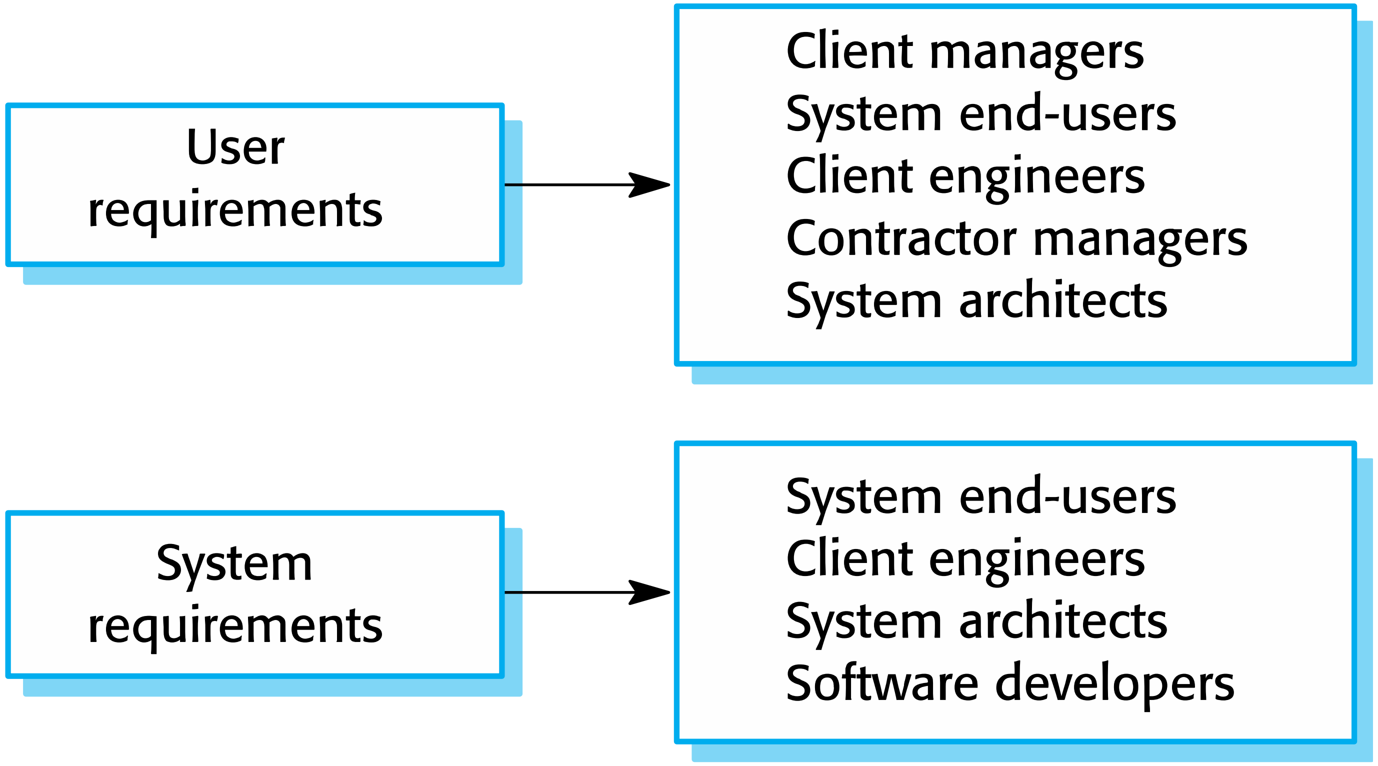
User requirements definition

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 different types 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
- ✧ A **custome**r is an individual or organization that derives either direct or indirect benefit from a product. Software customers could request, pay for, select, specify, use, or receive the output generated by a software product

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.

Stakeholders in the Mentcare 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.

Agile methods and requirements

- ✧ Many agile methods argue that producing detailed system requirements is a waste of time as requirements change so quickly.
- ✧ The requirements document is therefore always out of date.
- ✧ Agile methods usually use incremental requirements engineering and may express requirements as ‘user stories’ (discussed in Chapter 3).
- ✧ This is practical for business systems but problematic for systems that require pre-delivery analysis (e.g. critical systems) or systems developed by several teams.

Kinds of Requirements

Feature

- ✧ Feature logically related system capabilities that provide value to a user and are described by a set of functional requirements.

Different types of requirements

- ✧ Business requirements:
  - describe ***why*** the organization is implementing the system – the business benefits the organization hopes to achieve.
  - Business req. are recorded in Vision and Scope Document
- ✧ User requirements:
  - describe goals or tasks the users must be able to perform with the product that will provide value to someone.
  - It includes description of product characteristics important for user satisfaction. Can be represented with use cases, user stories, and event-response tables.
  - It describes what the user will be able to do with the system.

Different types of 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.
  - They describe *what* the developers must implement to enable users to accomplish their tasks (user req.), satisfying business req.
  - They are written using “shall” statements.
- 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

- A user shall be able to search the appointments lists for all clinics.
- The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
- 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 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 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, 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 classifications

Basic principle: requirements need to cover all relevant activities in the scope of the development project.

When defining requirements, try to consider the problem from all possible angles.

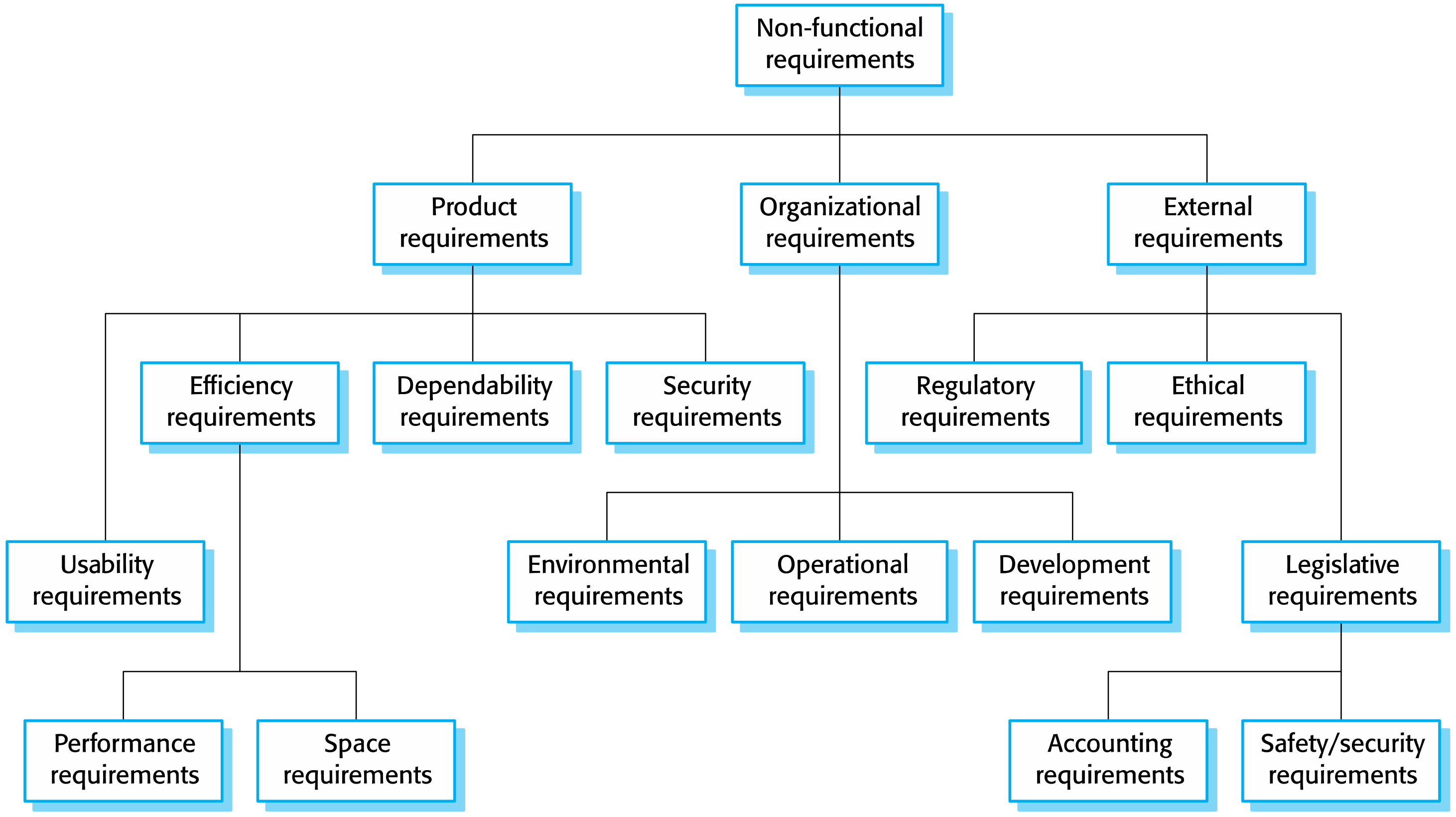
Examples of nonfunctional requirements in the Mentcare system

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.

Types of nonfunctional requirement



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 a number of related functional requirements that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

User requirements

User requirements are usually the first attempt to describe the desired functionality of the system.

Often suffer from deficiencies

✧ Lack of clarity: Language may be ambiguous and imprecise

✧ Requirements confusion: Functional req, non-functional requirements, system goals, and design information may not be clearly distinguished.

✧ Requirements amalgamation: several requirements may be expressed together as a single requirement

✧ Incomplete: important cases may not be covered

✧ Inconsistent: Requirements may contradict themselves

Part of the goal in refining user requirements to system requirements or a system specification is to eliminate these issues.

Usability requirements

- ✧ 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)

User Requirements

To minimize deficiencies in user requirements, some simple guidelines are helpful:

- ✧ Separate requirements
- ✧ Include a rationale for each requirement (especially useful when evaluating potential changes to requirements)
- ✧ Invent a standard format (requirements are easier to check)
- ✧ Distinguish between mandatory and desirable requirements
- ✧ Emphasize key parts of requirement
- ✧ Avoid technical jargon if possible (Keep user requirements simple.)

System Requirements

- ✧ System requirements elaborate user requirements to get a complete and precise description of the system.
- ✧ Depending on the project and process, there may be no separation into user and system requirements. It may be helpful to think of user requirements as the early versions of system requirements.
- ✧ An important consideration is the notation for system requirements.

Metrics for specifying nonfunctional 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

Requirements engineering processes

Requirements engineering processes

- There are a number of generic activities common to all processes
  - Requirements elicitation;
  - Requirements analysis;
  - Requirements specification;
  - Requirements validation;
  - Requirements management.
- In practice, RE is an iterative activity in which these processes are interleaved.

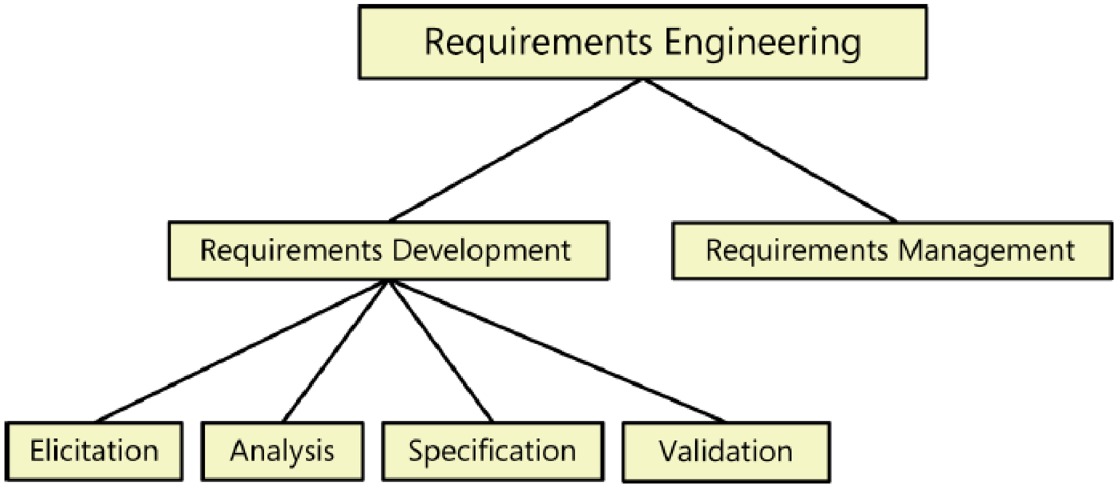
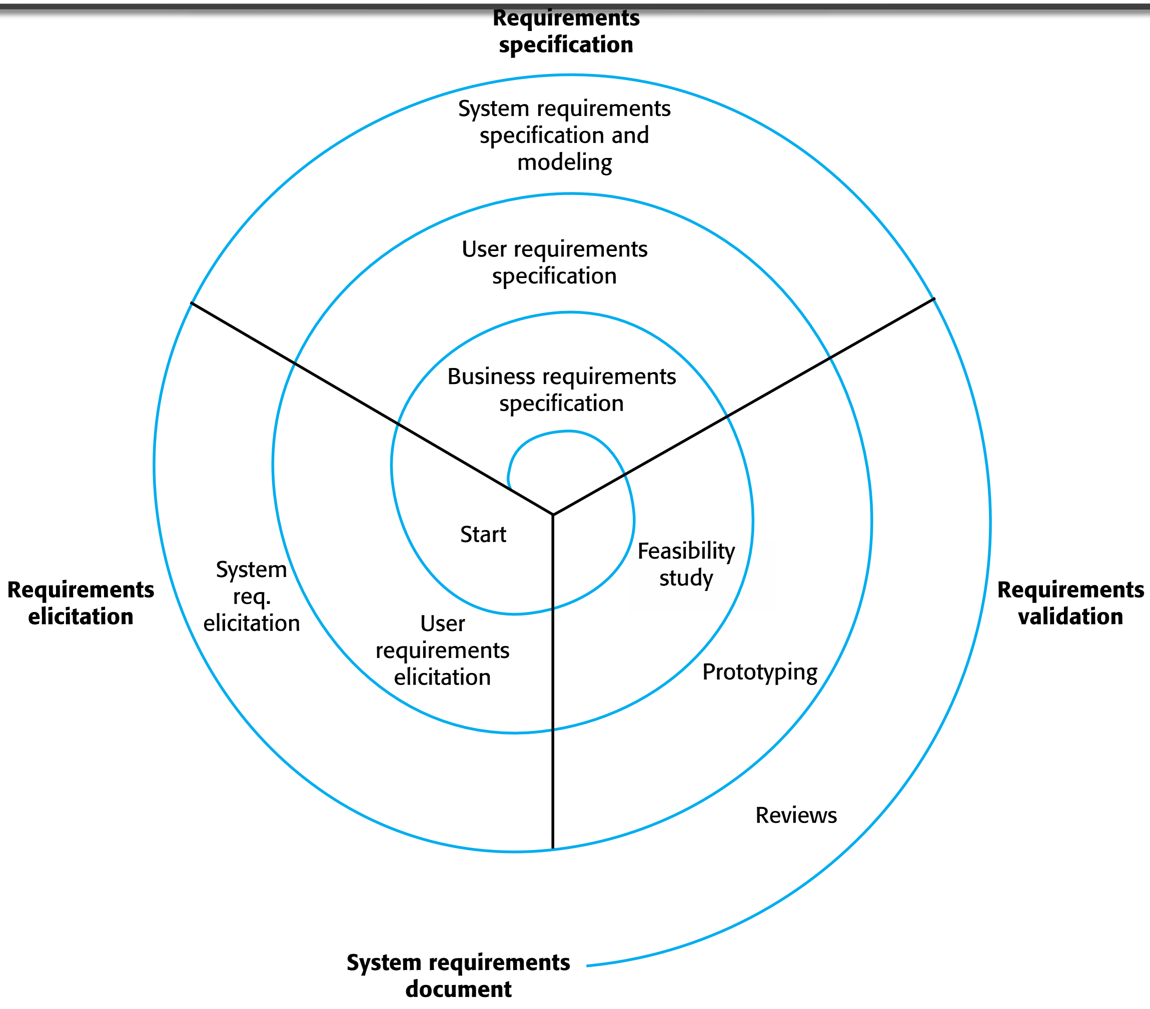


FIGURE 1-4 Subdisciplines of software requirements engineering.

A spiral view of the requirements engineering process


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

Requirements elicitation

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.*

Requirements elicitation

Requirements elicitation

- ✧ 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.

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.

Process activities

- ✧ **Requirements discovery**
  - Interacting with stakeholders to discover their requirements. Domain requirements are also discovered at this stage.
- ✧ **Requirements classification and organisation**
  - Groups related requirements and organises them into coherent clusters.
- ✧ **Prioritisation and negotiation**
  - Prioritising requirements and resolving requirements conflicts.
- ✧ **Requirements specification**
  - Requirements are documented and input into the next round of the spiral.

Requirements Elicitation and Analysis

✧ Approaches:
- Interviewing
- Scenarios
- Use-cases

Requirements Elicitation and Analysis: Interviewing

- ✧ **Meeting with stakeholders**
  - Sit down together and ask questions
  - Listen to what is said and unsaid
  - Have a set of fixed questions
  - Be prepared to ask flexible follow-up questions
- ✧ **Master-apprentice approach**
  - Have them teach you what they do
  - Go to workplace and watch the process
  - Learn requirements first-hand
- ✧ **Get Details**
  - Ask for copies of reports, logs, emails, etc.
  - Details may support, complete, or contradict what the stakeholder said

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 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.

Requirements Elicitation and Analysis: Scenarios

It is often easier for people to relate to real-life examples than abstract descriptions.

Scenarios describe a particular sequence of events that could take place within the proposed system. It is a structured form of user story.

Scenarios typically include:

- A description of what the system and users expect when the scenario starts
- A description of the normal flow of events in the scenario
- A description of what can go wrong and how this is handled
- Information about other activities that might be going on concurrently
- A description of the state of the system when the scenario finishes

Requirements Elicitation and Analysis: Use-Cases

Use-cases are a scenario-based technique for describing requirements.

Use-cases:

- identify the type of interaction and the actors involved
- may describe a particular scenario or a set of possible scenarios
- are typically illustrated with UML or similar diagrams (more on this later)
- are most effective at capturing *functional requirement*

A use-case based requirements methodology must take care to cover all possible interactions within the system.

Requirements specification

Requirements specification

- ✧ The process of writing down the user and system requirements in a requirements document.
- ✧ User requirements have to 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.

Ways of writing a system requirements specification

Notation	Description
Natural language	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

Requirements and design

- ✧ In principle, requirements should state **what** the system should do and the design should describe **how** it does this.
- ✧ In practice, 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

- ✧ Requirements are written as natural language sentences supplemented by diagrams and tables.
- ✧ Used for writing requirements because it is expressive, intuitive and universal. This means that the requirements can be understood by users and customers.

Guidelines for writing requirements

- ✧ Invent a standard format and use it for all requirements.
- ✧ Use language in a consistent way. Use shall for mandatory requirements, should for desirable requirements.
- ✧ Use text highlighting to identify key parts of the requirement.
- ✧ Avoid the use of computer jargon.
- ✧ Include an explanation (rationale) of why a requirement is necessary.

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.

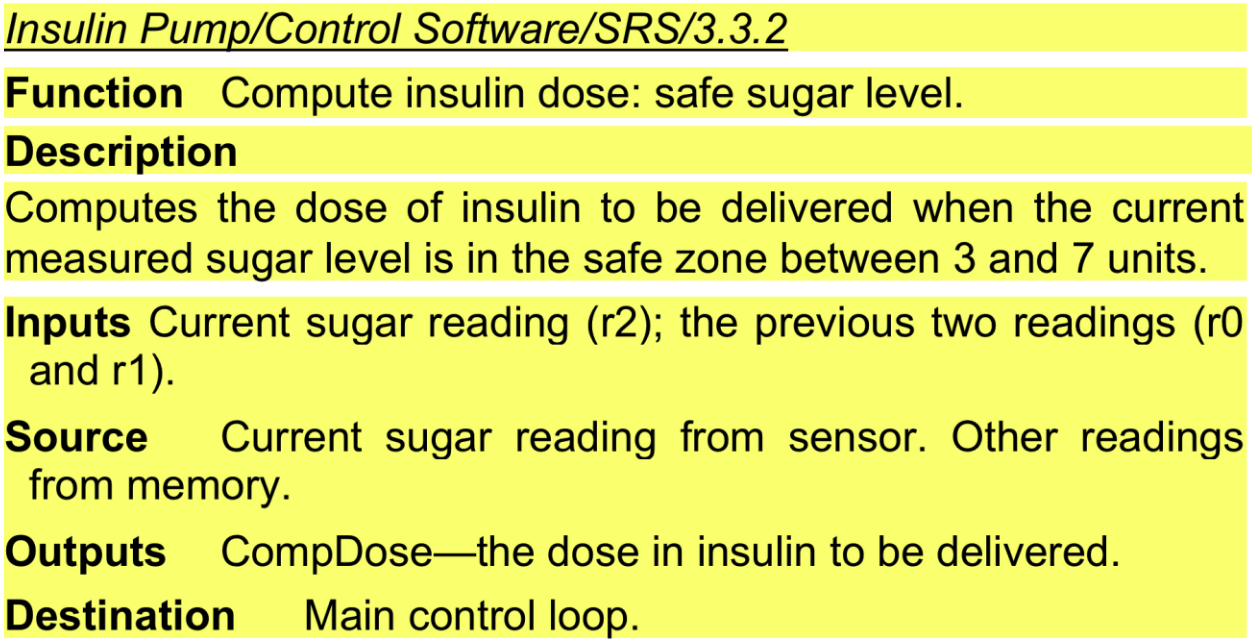
Structured specifications

- ✧ An approach to writing requirements where the freedom of the requirements writer is limited and requirements are written in a standard way.
- ✧ This works well for some types of requirements e.g. requirements for embedded control system but is sometimes too rigid for writing business system requirements.

Form-based specifications

- ✧ Definition of the function or entity.
- ✧ Description of inputs and where they come from.
- ✧ Description of outputs and where they go to.
- ✧ Information about the information needed for the computation and other entities used.
- ✧ Description of the action to be taken.
- ✧ Pre and post conditions (if appropriate).
- ✧ The side effects (if any) of the function.

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.

A structured specification of a requirement for an 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

- ✧ Used to supplement natural language.
- ✧ Particularly useful when you have to define a number of possible alternative courses of action.
- ✧ For example, the insulin pump systems bases its computations on the rate of change of blood sugar level and the tabular specification explains how to calculate the insulin requirement for different scenarios.

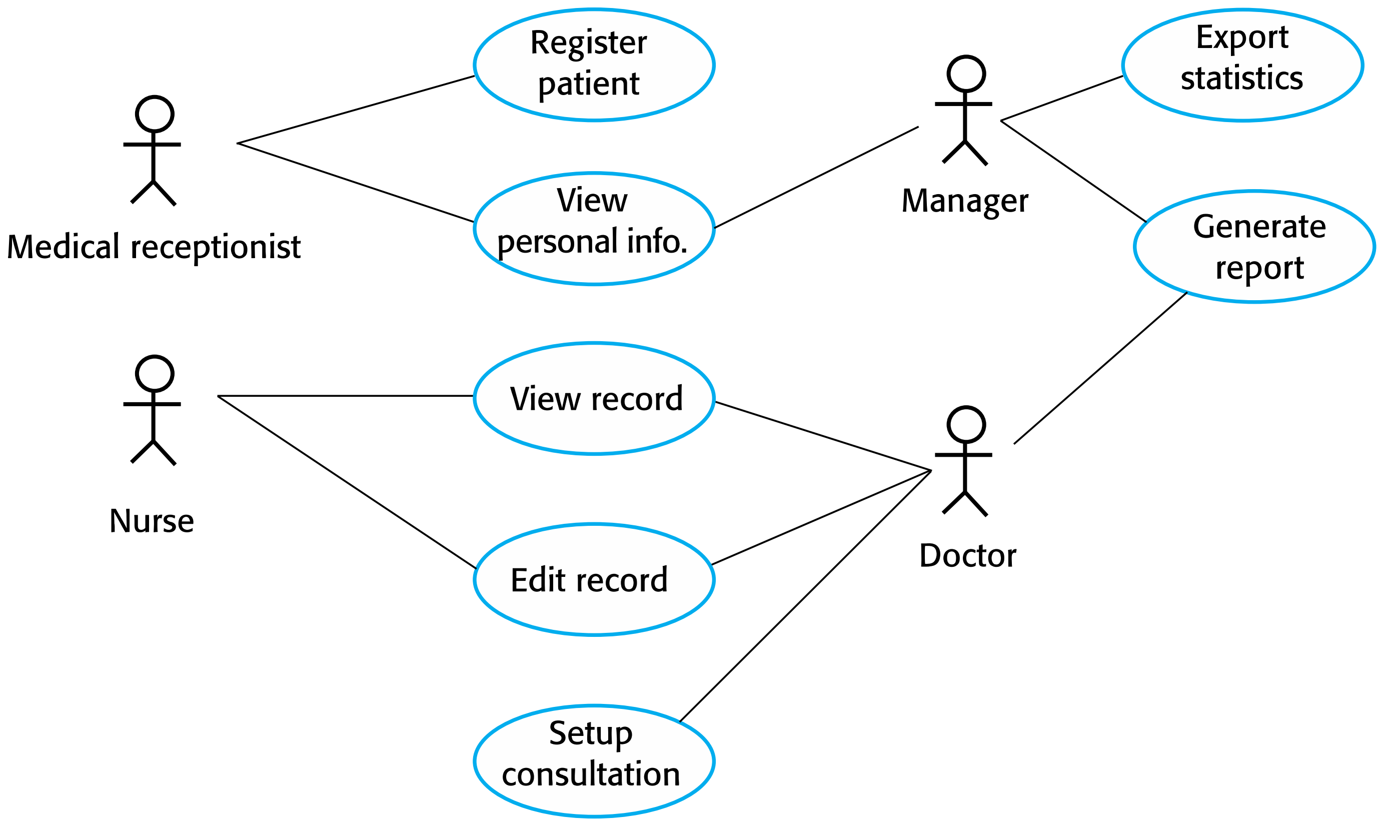
Tabular specification of computation for an insulin pump

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 (see Chapter 5).
- ✧ 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.

Requirements Validation

Once the requirements have been captured in a document, they should be checked in a number of different ways.

Important requirements checks

- V*alidity checks Make sure that the captured requirements are really what the* stakeholders want and need (i.e. get stakeholder feedback).
- *Consistency checks Requirements in the document should not conflict.*
- *Completeness checks The requirements should cover all possible situations* and behaviors for the system.
- *Realism checks The requirements should be double-checked for feasibility,* taking into account the budget and schedule.
- *Verifiability Requirements should be verifiable. There should be a way to* demonstrate that the system meets its requirements.

Requirements Validation

Techniques for requirements validation

✧ •*Requirements reviews A team which includes representative stakeholders* manually checks requirements.

✧ •*Prototyping An executable model is built based on the requirements.* Feedback is solicited from stakeholders

✧ *Test-case generation Requirements are converted into specific test cases.* This is the approach taken by XP.

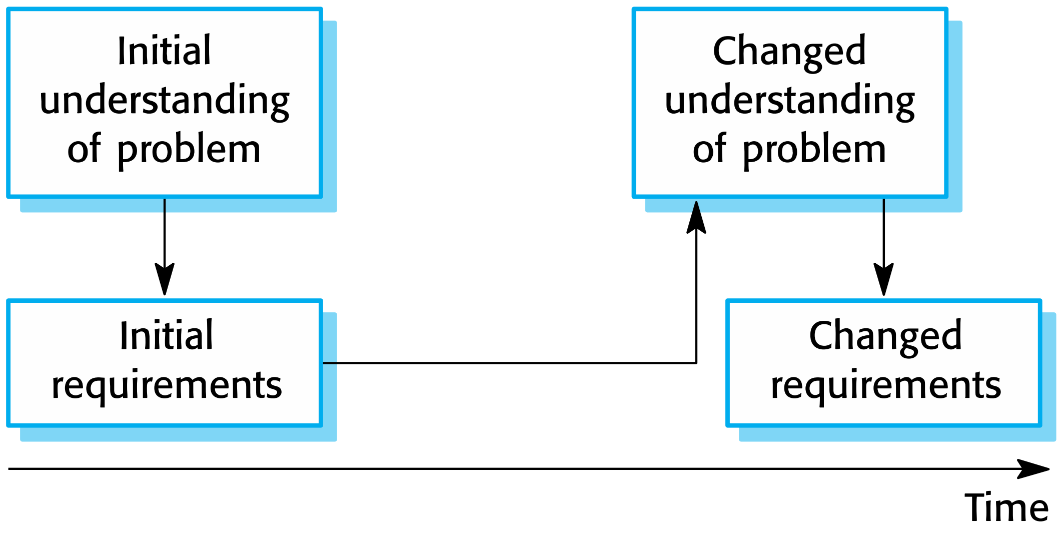
Requirements Management

Handling requirements changes

- ✧ *Make sure you understand requirements before moving on to design Good* design will not compensate for bad requirements. You will have to do it over again when the requirements are fixed.
- ✧ *Make sure everyone knows the cost and benefit of requirements changes* Clients get excited about new features. They need to be reminded that this will impact the cost and schedule. Changes that do not fit in with the business case should be rejected.
- ✧ *Set up a change-control procedure Plan for changes. Make sure all impacted* stakeholders have a say in proposed changes.
- ✧ *Use the right software process Evolutionary approaches more easily*

accommodate frequent changes.

Requirements evolution


Initial understanding of problem
Changed understanding of problem
Initial requirements
Changed requirements
Time

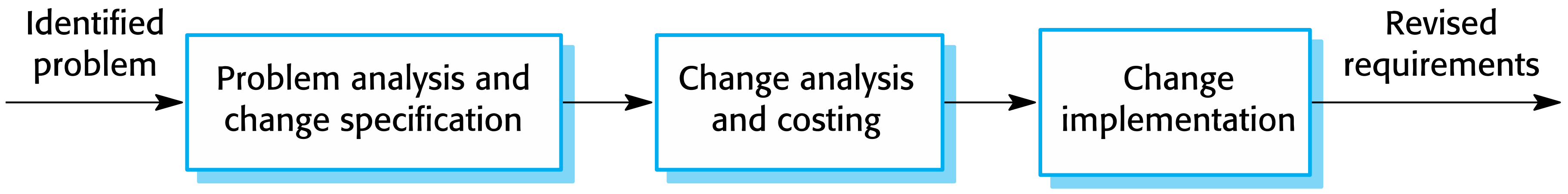
Requirements management

- ✧ 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. 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 change management


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