Osnovi softverskog inženjerstva

P-04: Inženjering zahtjeva

2024

Topics covered

- ✧ Introduction to requirements engineering
- ✧ Functional and non-functional requirements
- ✧ Requirements engineering processes
- ✧ Requirements elicitation
- ✧ Requirements specification
- ✧ Requirements validation
- ✧ Requirements change

Introduction to requirements engineering

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.
- 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 because 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.

Introduction to requirements engineering

Requirements types

✧ 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

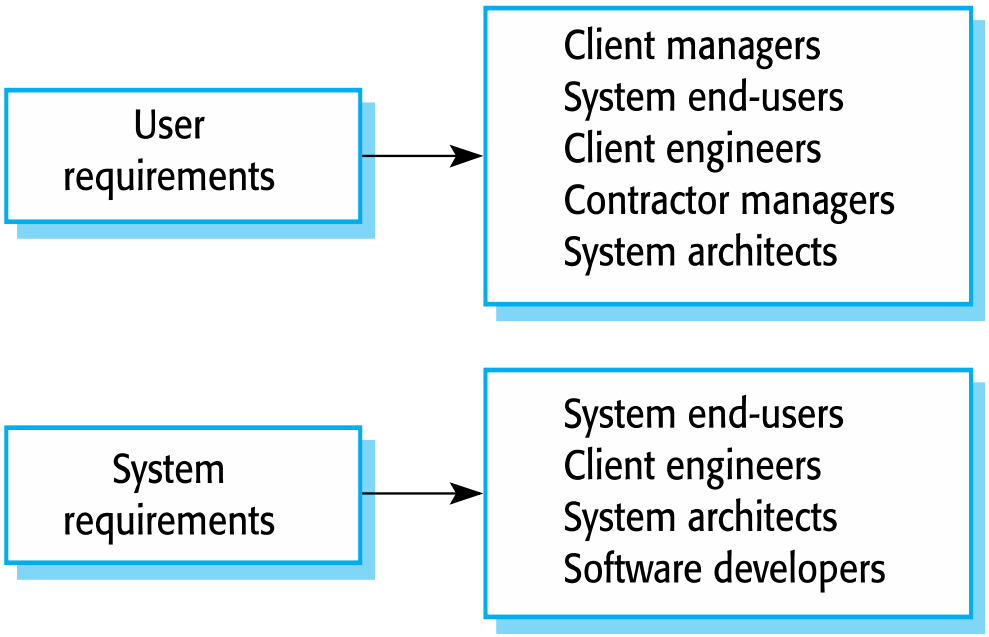
1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

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.

Introduction to requirements engineering

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**

Introduction to requirements engineering

Example: 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 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.

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 (FRs)

- **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.

Example FRs: Mentcare system

- 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.

Functional and non-functional requirements

Requirements imprecision

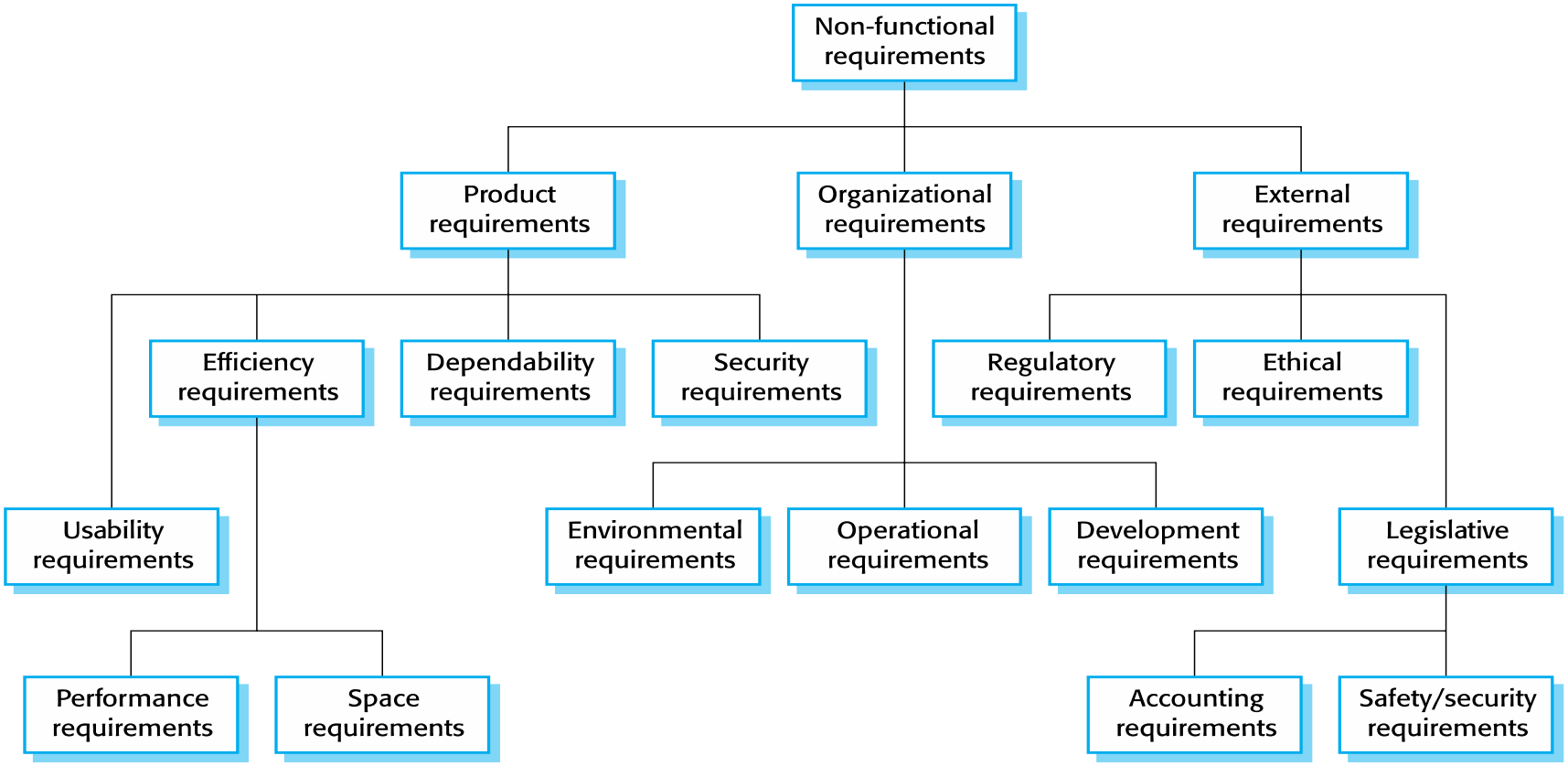
- **Problems arise when FRs are not precisely stated**.
- Ambiguous requirements **may be interpreted in different ways** by developers and users.
- Consider the term ‘search’ in requirement 1
  - User intention – search for a patient name across all appointments in all clinics;
  - Developer interpretation – search for a patient name in an individual clinic. User chooses clinic then search.

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 classifications

- **Product** NFRs
  - Requirements which specify that the delivered product must behave in a particular way (e.g. execution speed, reliability, etc.)
- **Organisational** NFRs
  - Requirements which are a consequence of organisational policies and procedures (e.g. implementation requirements, etc.)
- **External** NFRs
  - Requirements which arise from external factors and its development process (e.g. interoperability , legislative, etc.)



Functional and non-functional requirements

Examples of NFRs 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.

NFRs implementation

- **NFRs 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 NFR** such as a security requirement, **may generate a number** of related FRs s that define system services that are required.
  - It may also generate requirements that restrict existing requirements.

Functional and non-functional requirements

Goals and requirements

- NFRs 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 NFR**
  - **A statement using some measure that can be objectively tested**.
- Goals are helpful to developers as they convey the intentions of the system users.

Examples of usability requirements

- **Goal:**
  - The system should be easy to use by medical staff and should be organized in such a way that user errors are minimized.
- **NFR:**
  - 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.

Functional and non-functional requirements

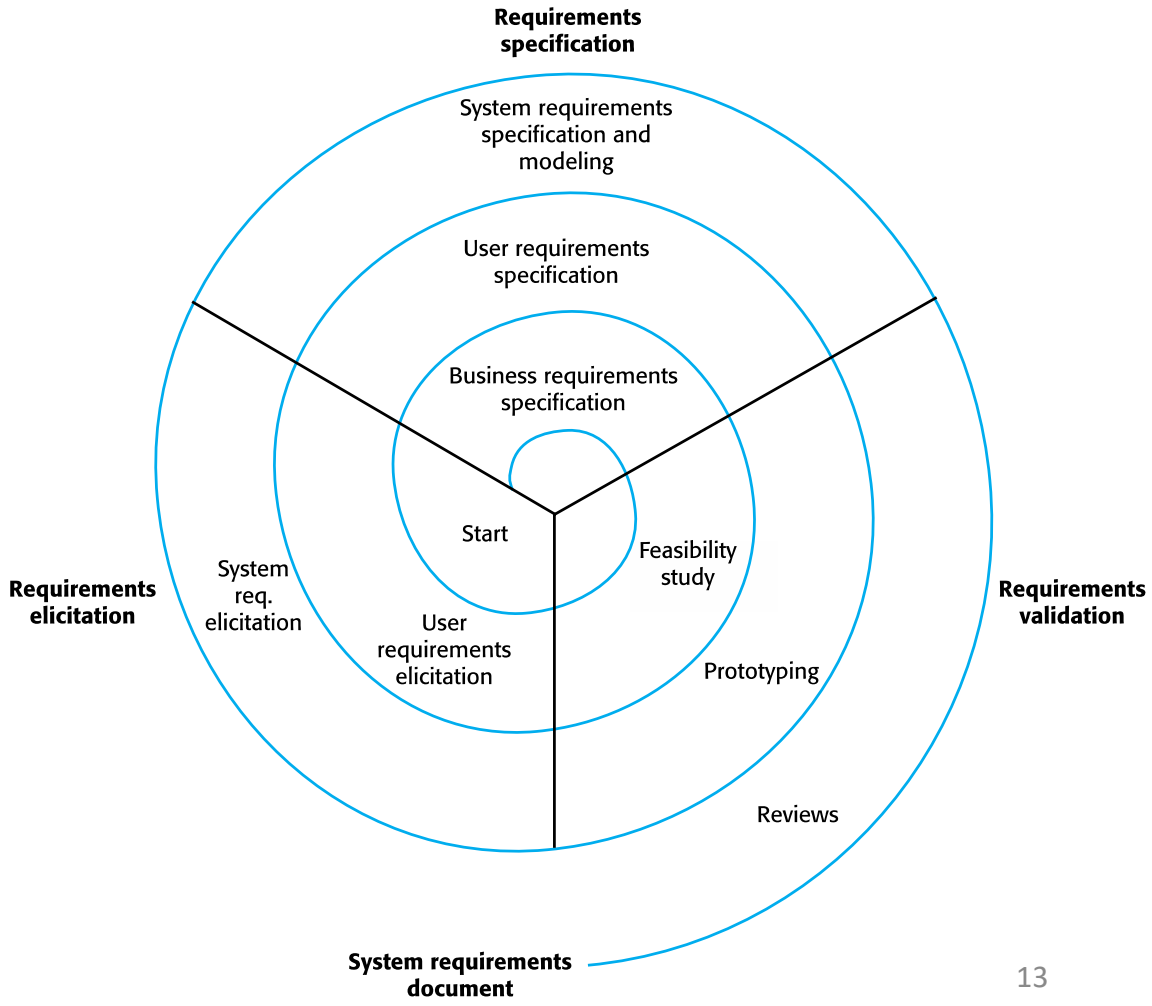
Metrics for specifying NFRs

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

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

A spiral view of the RE process



Requirements elicitation

Requirements elicitation

- Sometimes called **requirements gathering / 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.***
- 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.

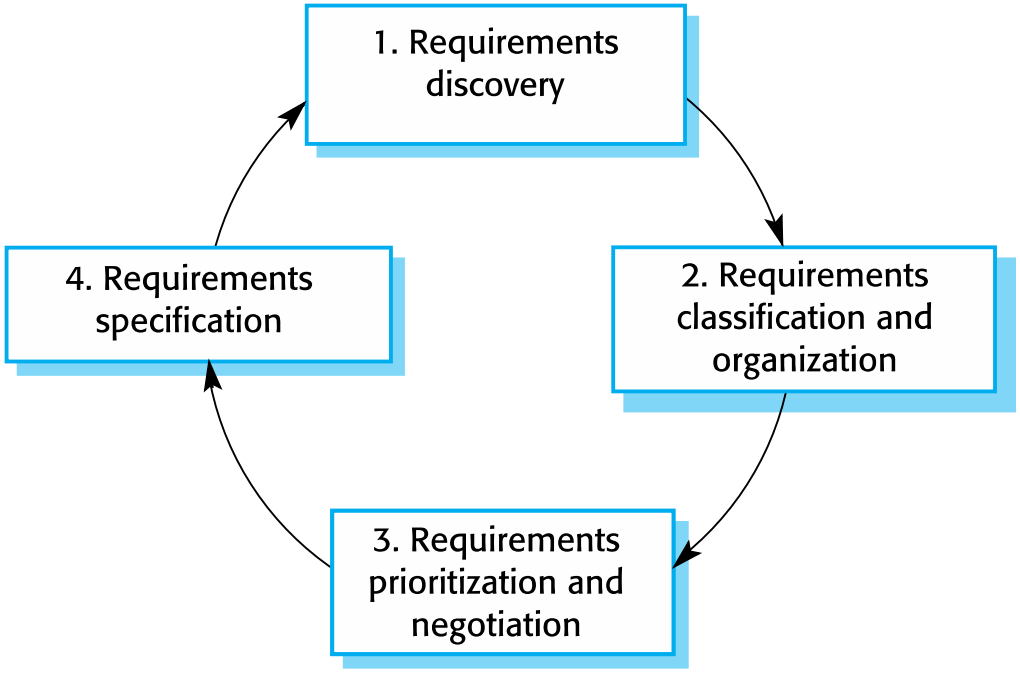
Problems of requirements elicitation

- **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

Activities and process

- **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

Requirements discovery

- The **process of gathering information** about the required and existing systems and distilling the user and system requirements from this information.
- **Interaction is with system stakeholders** from managers to external regulators.
- Systems normally have a range of stakeholders.
- **Elicitation techniques**
  - **interviews**
  - **questionnaires**
  - **observation**
  - **prototypes**
  - **group sessions (workshops)**
  - **…**

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

Interviews in practice

- **Normally a mix of closed and open-ended interviewing**.
- Interviews are good for getting an overall understanding of what stakeholders do and how they might interact with the system.
- Interviewers need to be open-minded without pre-conceived ideas of what the system should do
- **You need to prompt the use to talk** about the system **by suggesting requirements rather than simply asking them what they want**.

Problems with interviews

- Application specialists may use language to describe their work that isn't easy for the requirements engineer to understand.
- **Interviews are 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.

Requirements elicitation

Observation (Ethnography)

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

Scope of ethnography

- **Requirements that are derived from the way that people actually work** rather than the way that process definitions suggest.
- Requirements that are derived from cooperation and awareness of other people's activities.
  - 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.

Requirements elicitation

Stories and scenarios

- Scenarios and user stories are **real-life examples of how a system can be used**.
- Stories and scenarios are a description of how a system may be used for a particular task.
- Because they are based on a practical situation, stakeholders can relate to them and can comment on their situation with respect to the story.

Scenarios

- A **structured form of user story**
- **Scenarios 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**.

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.

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.

Requirements specification

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 specification

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**.

Problems with natural language

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

Guidelines for writing reqs.

- **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.

Example requirements for the insulin pump software system:

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.)*

Requirements specification

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.

Requirements specification

Example: A structured requirement specification 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 b
measured sugar level is in the safe

Inputs Current sugar reading (r2);
and r1).

Source Current sugar reading
from memory.

Outputs CompDose—the dose i

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.

Requirements specification

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.

Example: Tabular specification of computation for an insulin pump

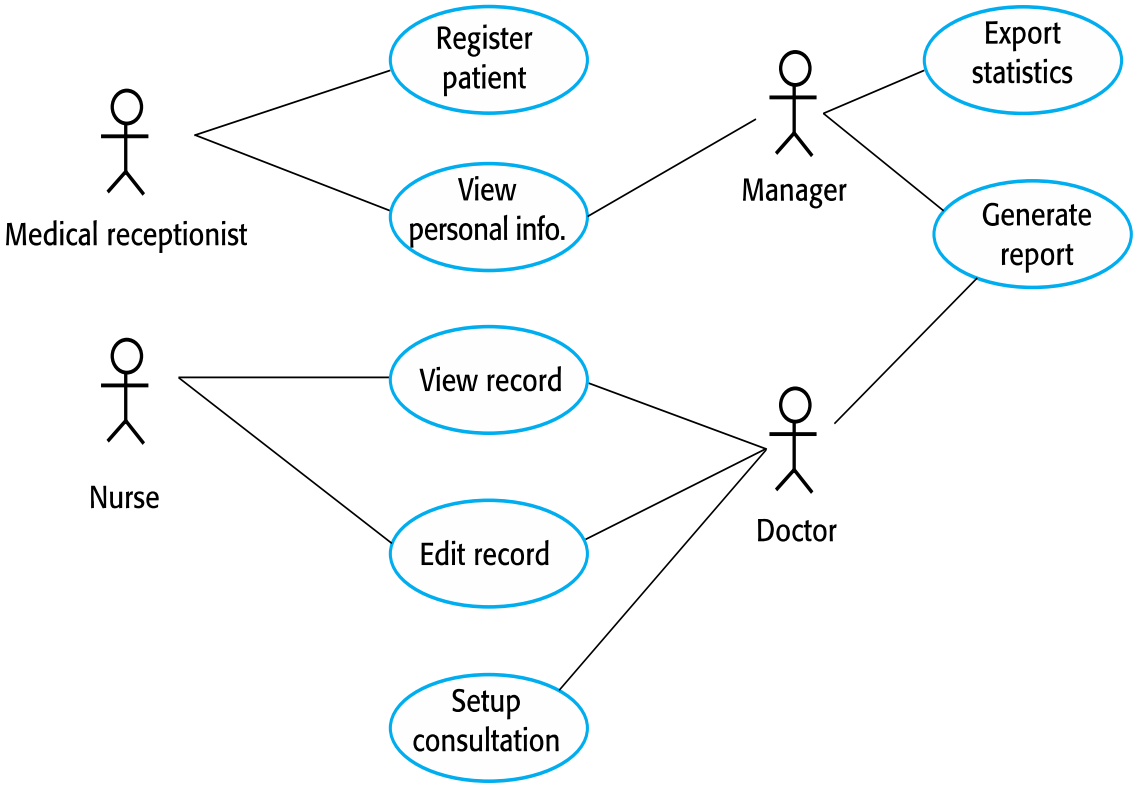
Condition	Action
Sugar level falling (r2 < r1)	CompDose = 0
Sugar level stable (r2 = r1)	CompDose = 0
Sugar level increasing and rate of increase decreasing ((r2 – r1) < (r1 – r0))	CompDose = 0
Sugar level increasing and rate of increase stable or increasing ((r2 – r1) ≥ (r1 – r0))	CompDose=round ((r2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

Requirements specification

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



Requirements specification

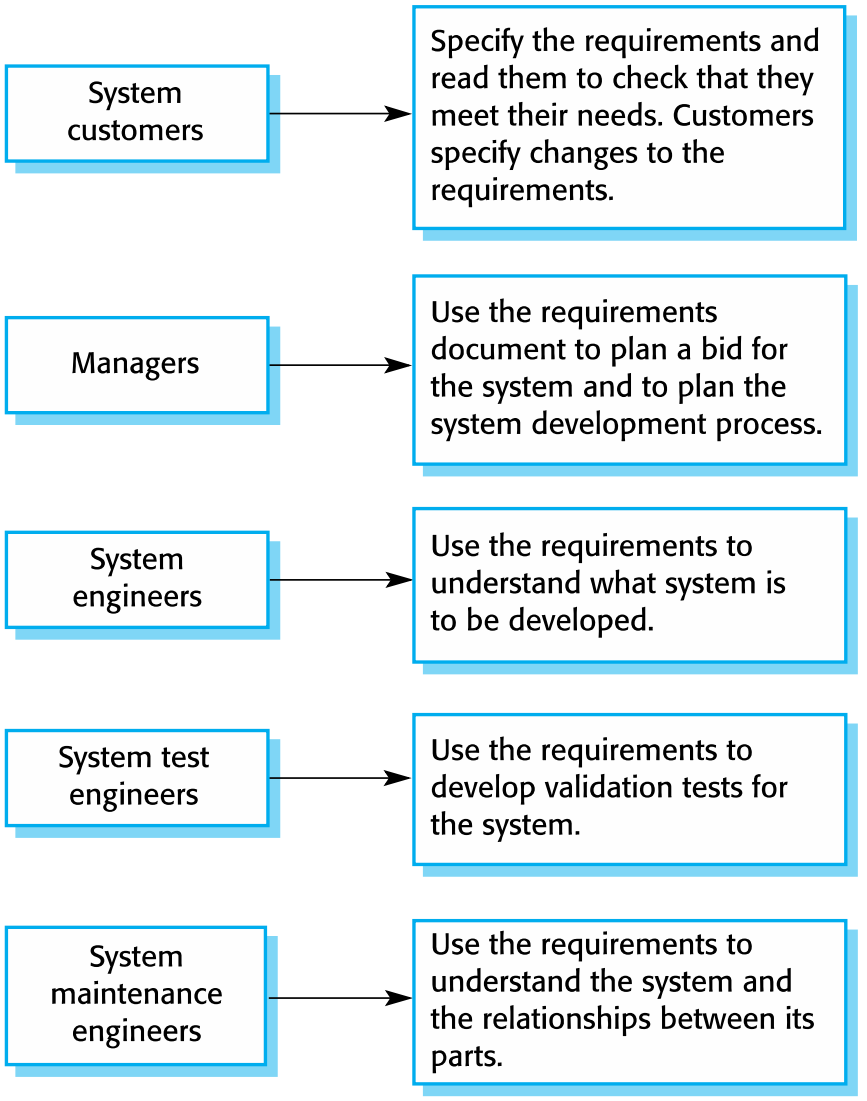
The software requirements document

- The **software requirements document** (**SRS – Software Requirements Specification**) 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 document variability

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

Users of a requirements document



Requirements specification

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 h**ow 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 (FRs) The **NFRs** 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.

Requirements specification

The structure of a requirements document (cont.)

Chapter	Description
System requirements specification	This **should describe the system FRs and NFRs in more detail**. If necessary, further detail may also be added to the NFRs. **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

- 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 validation techniques

- **Requirements reviews**
  - Systematic manual analysis of the requirements.
- **Prototyping**
  - Using an executable model of the system to check requirements.
- **Test-case generation**
  - Developing tests for requirements to check testability.

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

Requirements reviews

- **Regular reviews should be held** while the requirements definition is being formulated.
- Both **client and contractor** staff **should be involved** in reviews.
- Reviews **may be formal** (with completed documents) **or informal**. Good communications between developers, customers and users can resolve problems at an early stage.

Review checks

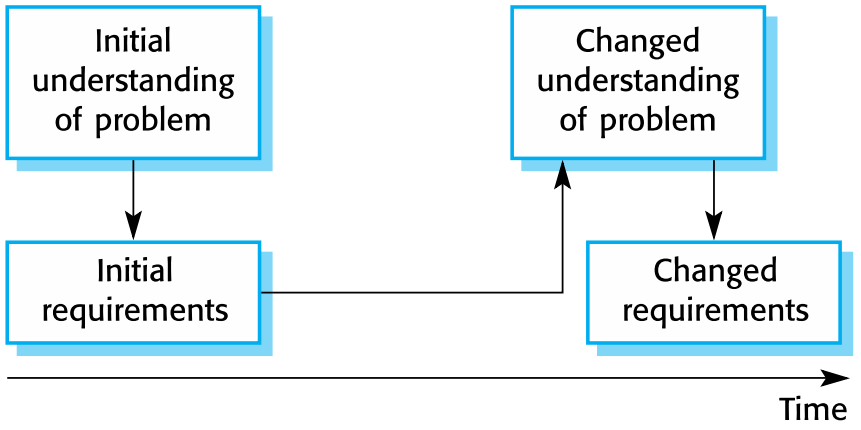
- **Verifiability**
  - Is the requirement realistically testable?
- **Comprehensibility**
  - Is the requirement properly understood?
- **Traceability**
  - Is the origin of the requirement clearly stated?
- **Adaptability**
  - Can the requirement be changed without a large impact on other requirements?

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.
- **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

Requirements evolution



Requirements change

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**.

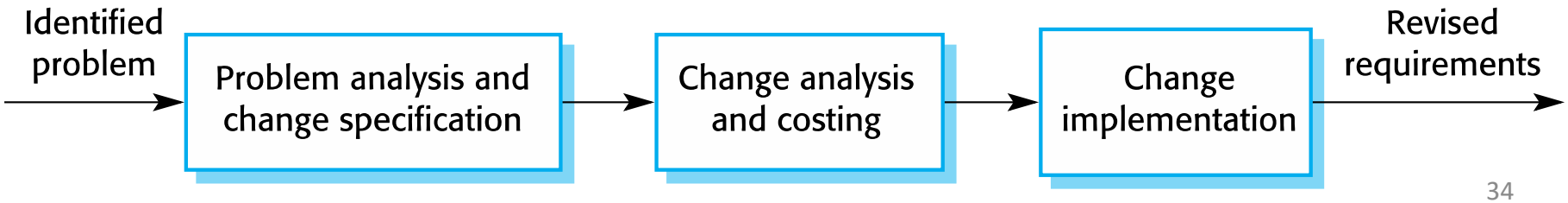
Requirements management planning

- 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

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.



Key points

- **Requirements for a software system set out what the system should do and define constraints on its operation and implementation**.
- **FRs are statements of the services that the system must provide or are descriptions of how some computations must be carried out**.
- **NFRs often constrain the system being developed and the development process being used**.
- **NFRs often relate to the emergent properties of the system and therefore apply to the system as a whole.**
- **The RE process is an iterative process that includes requirements elicitation, specification and validation.**
- **Requirements elicitation is an iterative process that can be represented as a spiral of activities** – requirements discovery, requirements classification and organization, requirements negotiation and requirements documentation.
- **You can use a range of techniques for requirements elicitation** including interviews and ethnography. User stories and scenarios may be used to facilitate discussions.

Key points (cont.)

- Requirements specification is the process of formally documenting the user and system requirements and creating a software requirements document.
- The software requirements document (SRS – Software Requirements Specification) is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it.
- Requirements validation is the process of checking the requirements for validity, consistency, completeness, realism and verifiability.
- 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 changes of the requirements.