

SWVE3002-42: Introduction to Software Engineering

Lecture 4 – Requirements Engineering

Sooyoung Cha

Department of Computer Science and Engineering

Topics covered

01 Requirements engineering

02 Functional and non-functional requirements

03 Requirements engineering processes

04 Requirements elicitation

05 Requirements specification

06 Requirements validation

07 Requirements change

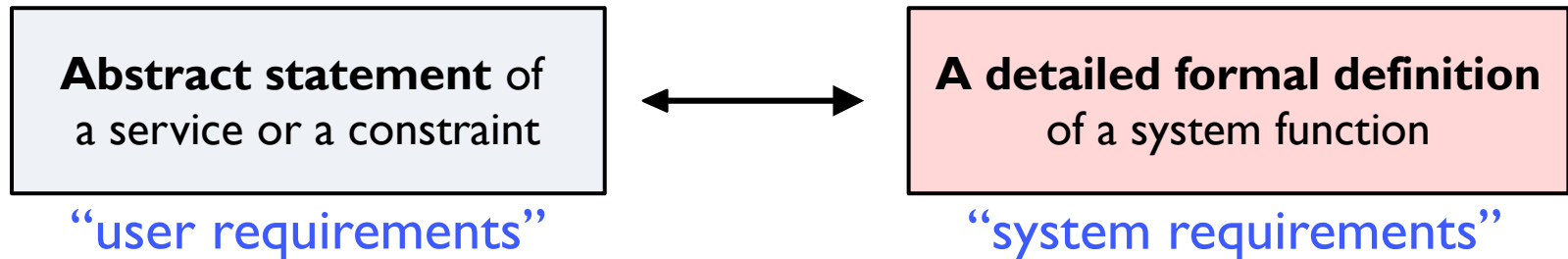


Chapter 4. (p.101 ~ p.137)

Requirements engineering

[Requirement]

- **The descriptions of the services** that a system should provide and the constraints on its operation.



[Requirement Engineering (RE)]

- **The process** of finding out, analyzing, documenting and checking these services and constraints.

(**The first stage** of the software engineering process)

(The very first stage, feasibility study, of the software engineering process)

User and system requirements

[Example of user and system requirements]

- MentCare: A medical **system that maintains information about patients** suffering from mental health problems and **their treatments**.

User requirements definition

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

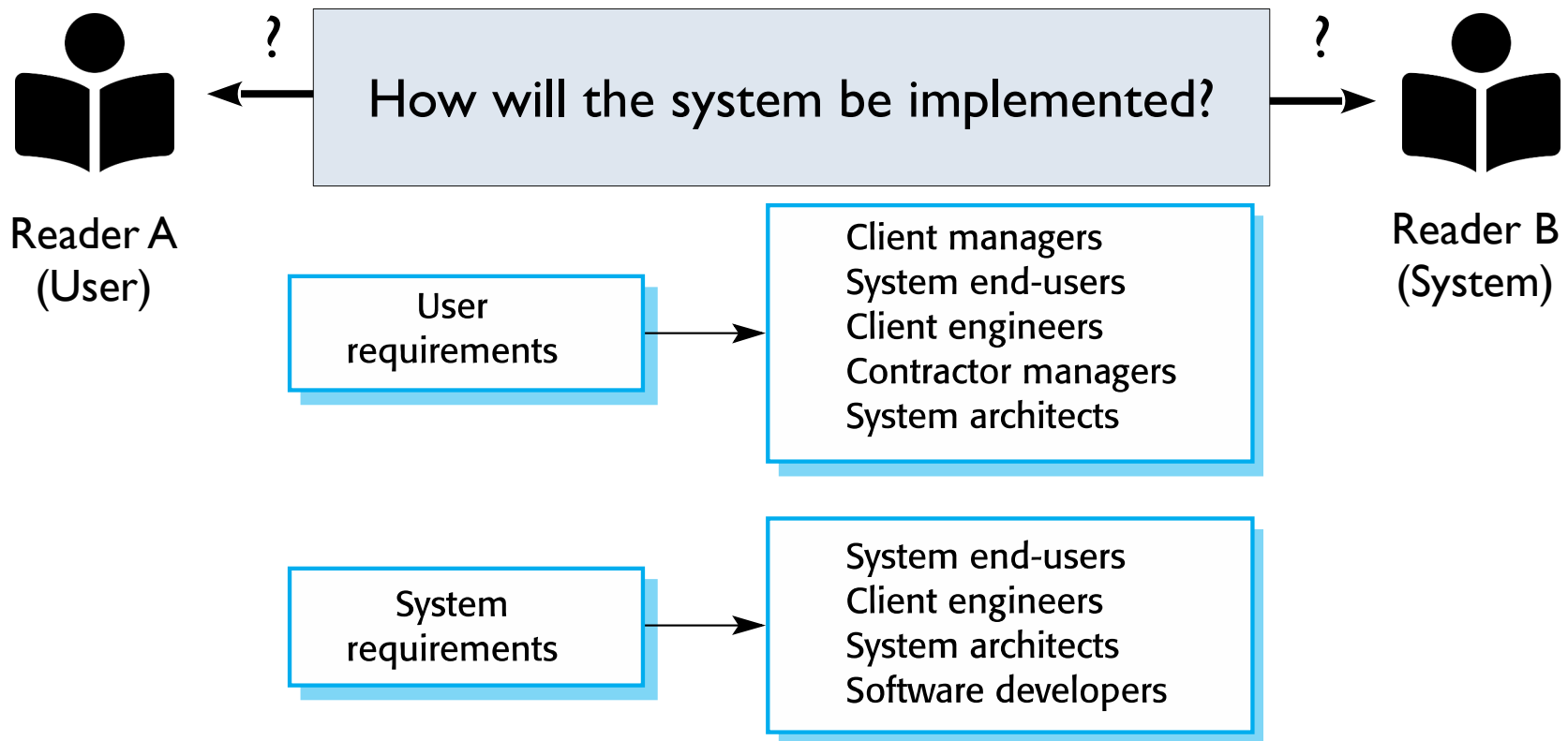
System requirements specification

- 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.

Types of requirements

[Why are these two requirements necessary?]

- Different types of readers use the requirements in different ways.



Types of requirements

[System stakeholders for MentCare]

- Any person or organization who is affected by the system in some way and who has a legitimate interest

Patients

whose information is recorded in the system.

Doctors

who are responsible for assessing and treating patients.

Nurses

who coordinate the consultations with doctors.

Receptionists

who manage patients' appointments.

IT staff

who are responsible for installing the system.

Functional and non-functional requirements

[SW system requirements]

- Functional requirements.
 - Statements of services **the system should provide**. (e.g., input/output)
 - Statements of **what the system should not do**.

A statement limiting access to authorized users?


- Non-functional requirements.
 - **Constraints on the services** offered by the system. (e.g., timing constraint)
 - **Constraints that apply to the entire system** rather than individual system features.

Functional requirements

[Functional requirements]

- Describing what the system should do.
- Varying from general requirements to very specific requirements.

[Example of functional requirements for Mentcare system]

- 
- (R1) A user is able to search the appointments lists for all clinics.
 - (R2) For each clinic and day, the system generates a list of patients who are expected to attend appointments that day.
 - (R3) Each staff member is uniquely identified by his eight-digit employee number.

Functional requirements

[Functional requirements]

- **Imprecision** in the requirements
 - **Leading to disputes** between customers and SW developers.
 - **Delaying** system **delivery** and increases costs.
- Example of Imprecision in the requirements for Mentcare system
 - (RI) A user is able to “**search**” the appointments lists for all clinics.

Search for the name in all appointments at all clinics.

After **choosing** the clinic user attended, **search** for the name in the clinic.

Functional requirements

[(Ideal) functional requirements]

- **Completeness** in requirements
 - All services required by the user should be defined.
- **Consistency** in requirements
 - Requirements should not be contradictory.



In practice, **it is impossible** to achieve requirements consistency and completeness **for large systems**.

Easy to make mistakes.

Have many stakeholders.

Non-functional requirements

[Non-functional requirements]

- Constrain characteristics of the system **as a whole**.
 - Response time, memory use, reliability, ...
- Non-functional requirements **>** individual functional requirements.
 - **Failing** to meet the requirement mean that **the whole system is unusable**.
- Non-functional requirements arise through user needs.
 - ex) budget constraints, organizational policies, safety regulations, privacy legislation.

Non-functional requirements

[Types of non-functional requirements]

- Product requirements
 - Specifying **the runtime behavior** of the software.
ex) execution speed, memory usage, acceptable failure rate, ...
- Organizational requirements
 - Deriving from **policies** in the customer's and developer's organizations.
ex) programming language, process standards to be used, ...
- External requirements
 - Deriving from factors **external to the system**.
ex) nuclear safety authority, legislative requirements, ...

Non-functional requirements

[Examples of non-functional requirements]

- Mentcare system

Product requirement

The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 08:30–17:30).

Downtime within normal working hours shall not exceed five seconds in any one day.

Organizational requirement

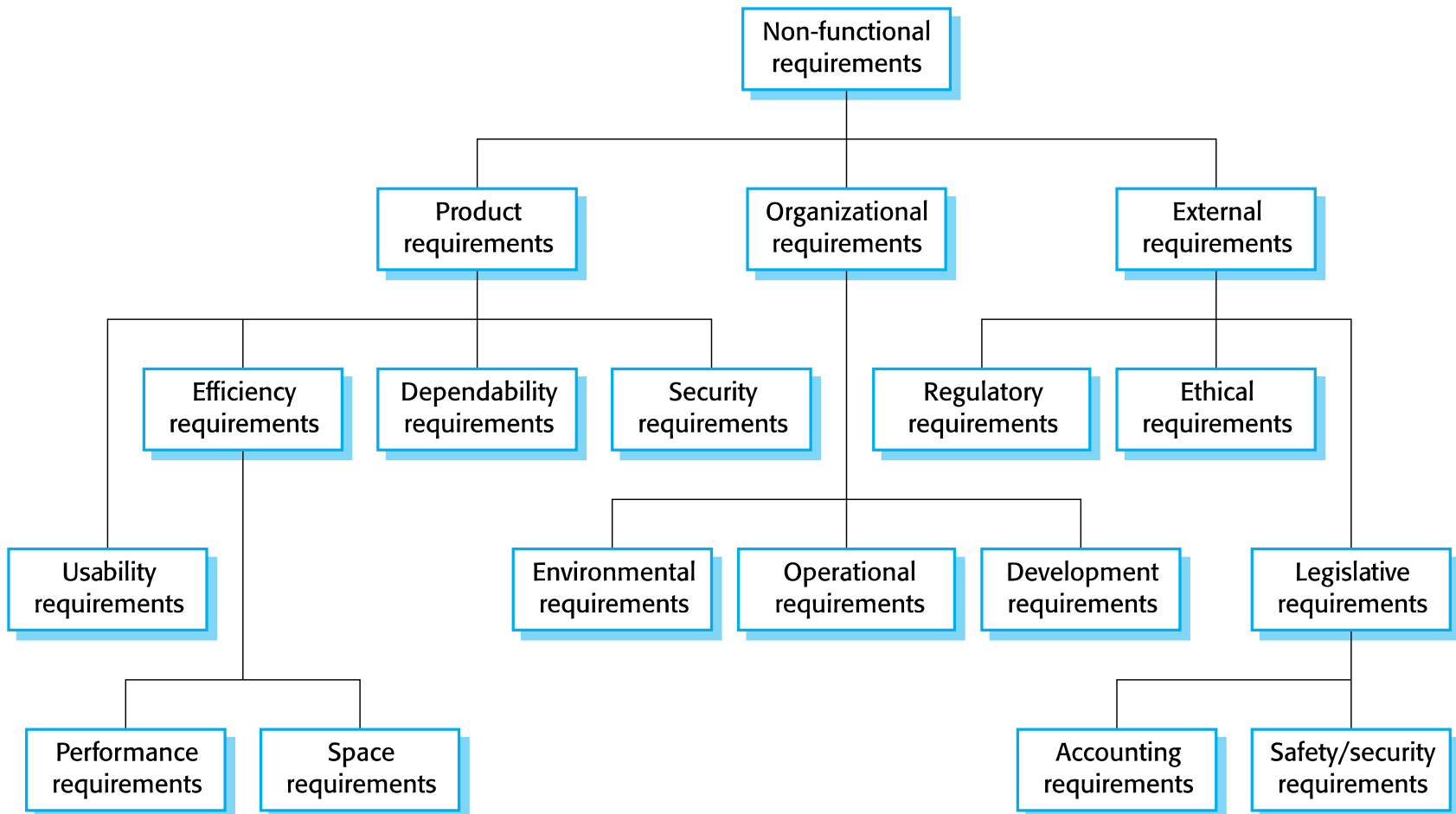
Users of the Mentcare system shall authenticate themselves using their health authority identity card.

External requirement

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

Non-functional requirements

[Types of non-functional requirements]



Non-functional requirements

[Problems in non-functional requirements]

- Stakeholders propose requirements as **general goals**.
 - ex) ease of use, rapid user response, ...

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

↓ “testable” non-functional requirement.

*Medical staff shall be able to use all the system functions **after two hours of training**. After this training, **the average number of errors** made by experienced users shall not exceed **two per hour of system use**.*

Non-functional requirements

[Solutions in non-functional requirements]

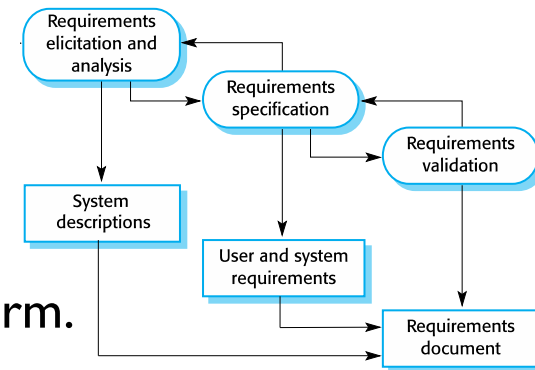
- Writing non-functional requirements **quantitatively** for testing.
- Translating the customer's goals into **measurable requirements**.

Property	Measure
Speed	Processed transactions/second, Screen refresh time, User/event response 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,
Robustness	Time to restart after failure, Percentage of events causing failure, Probability of data corruption
Portability	Percentage of target dependent statements, Number of target systems

Requirements engineering processes

[Three key activities in requirements engineering]

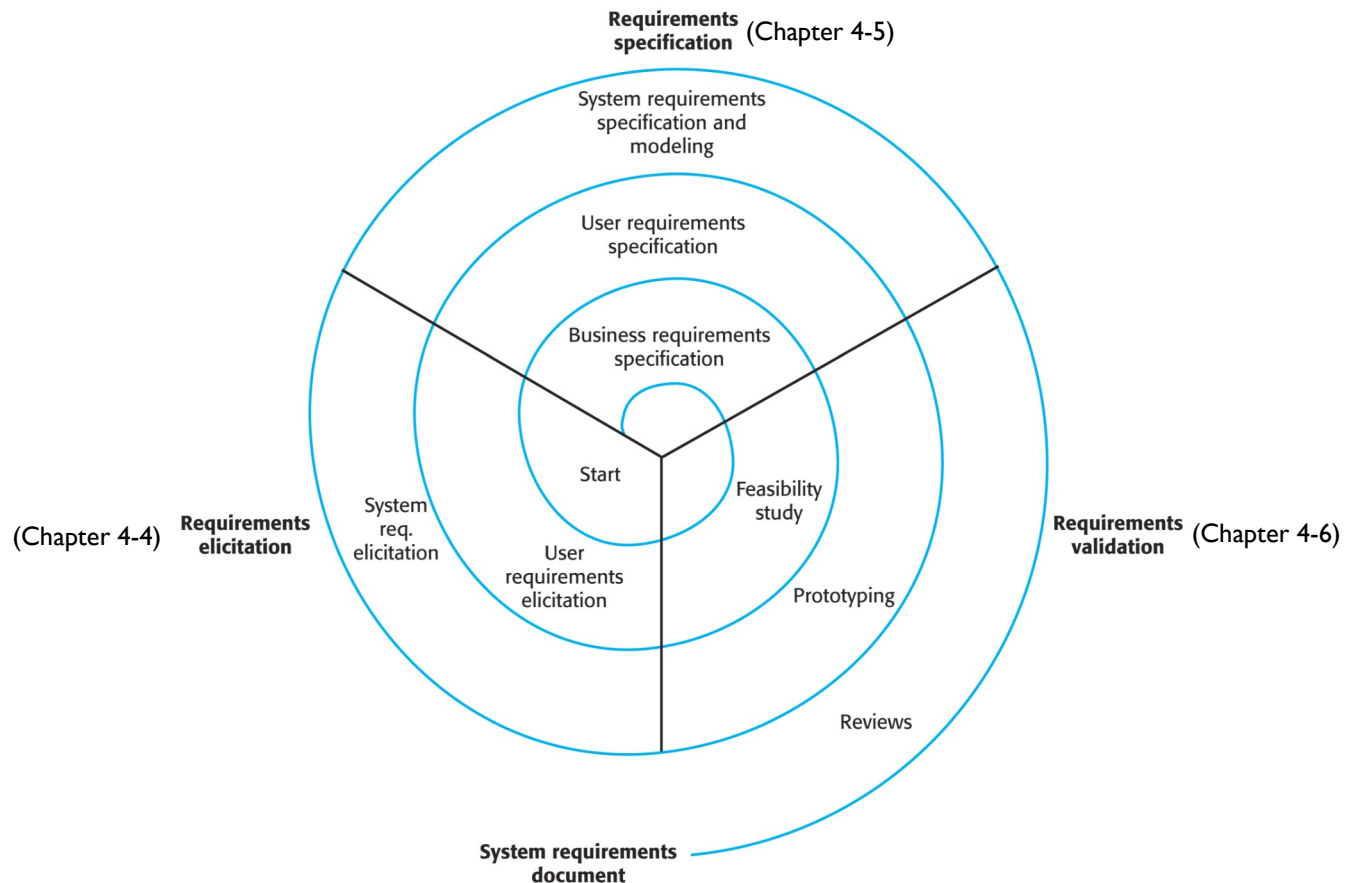
- Elicitation and analysis
 - Interacting with stakeholders.
- Specification
 - Converting these requirements into a standard form.
- Validation
 - Checking that the requirements define the system that customer wants.



In practice, RE is an iterative process in which the activities are interleaved.

Requirements engineering processes

[A spiral view of the requirements engineering process]



Requirements elicitation

[Goal]

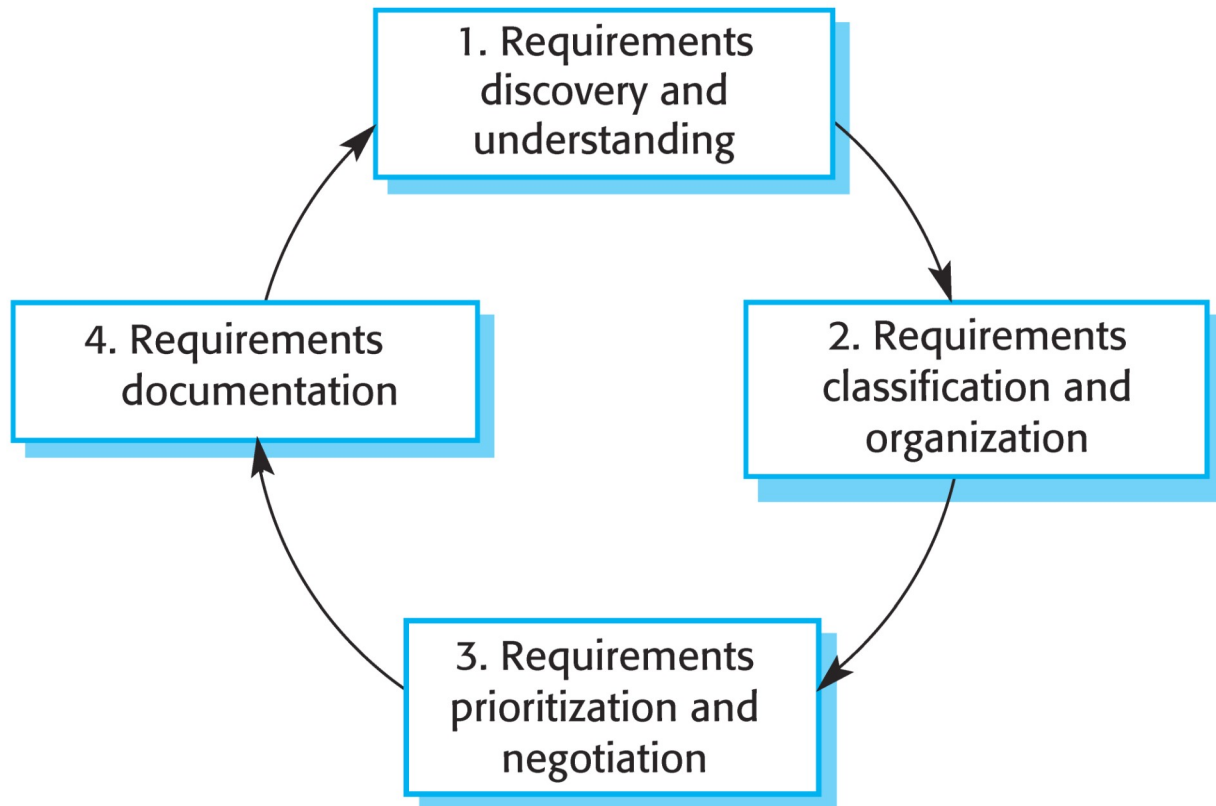
- **Understanding** the work that stakeholders do and how they use a new system to help support that work.

[Problems]

- Stakeholders **don't know** what they really want from a SW system.
- Stakeholders express requirements in **their own terms**.
- **Different stakeholders** express their requirements in **different ways**.
- **Political factors** may influence the system requirements.
- The requirements **dynamically change** during the analysis process.

Requirements elicitation

[The requirements elicitation and analysis process]



Requirements elicitation

[The requirements elicitation and analysis process]

Requirements discovery and understanding

- **Interacting** with stakeholders to **discover their requirements**.

Requirements classification and organization

- **Grouping** related requirements and organizing them into **coherent clusters**.

Requirements prioritization and negotiation

- **Prioritizing** requirements and **resolving** requirements **conflicts**.

Requirements documentation

- **Documenting** requirements. (e.g., early draft of SW requirement, whiteboard, wikis)

Requirements elicitation techniques

[Two fundamental approaches to requirements elicitation]

- Interviewing (= Talking to people about what they do.)
 - **Closed interviews**, where the stakeholder answers predefined questions.
 - **Open interviews**, in which there is no predefined agenda.
- Observation or Ethnography
 - Watching people doing their job to see what artifacts they use, how they use them, and so on

“Pay attention to what users do, not what they say.”

Jakob Nielsen, co-founder of NN/g.

Requirements elicitation techniques

[Eliciting domain knowledge through interviews is difficult]

- Usage of domain specific terminologies.
- Difficulty for explaining some familiar domain knowledge.
- Omitting to mention fundamental knowledge to stakeholders.

Stories and scenarios

[Stories and scenarios]

- A description of how the system can be used for a particular task.
 - Stories: narrative text for a high-level description of system use.
 - Scenarios: specific information collected such as inputs and outputs.

[The purpose of Stories]

- Facilitating discussion of how the system might be used.
- Acting as a starting point for eliciting the requirements for that system.

Stories and scenarios

[iLearn: The stories for photo sharing in the classroom]

Jack is a primary school teacher in northern Scotland. He has decided that a class project should be focused on the fishing industry in the area, looking at the history, development, and economic impact of fishing. As part of this project, pupils are asked to gather and share reminiscences from relatives, use newspaper archives, and collect old photographs related to fishing and fishing communities in the area. Pupils use an iLearn wiki to gather together fishing stories and SCRAN to access newspaper archives and photographs. However, Jack also needs a photo-sharing site because he wants pupils to take and comment on each other's photos and to upload scans of old photographs that they may have in their families.

Jack sends an email to a primary school teachers' group, which he is a member of, to see if anyone can recommend an appropriate system. Two teachers reply, and both suggest that he use KidsTakePics, a photo-sharing site that allows teachers to check and moderate content. As KidsTakePics is not integrated with the iLearn authentication service, he sets up a teacher and a class account. He uses the iLearn setup service to add KidsTakePics to the services seen by the pupils in his class so that when they log in, they can immediately use the system to upload photos from their mobile devices and class computers.

Stories and scenarios

[Scenarios]

- A structured form of the user stories.
- The common characteristics of a scenario.
 - A description of what the system expect when the scenario starts.
 - A description of the normal flow of events in the scenario.
 - A description of what can go wrong.
 - Information about other activities that are going on at the same time.
 - A description of the system state when the scenario ends.

Stories and scenarios

[The scenario for uploading photos to KidsTakePics]

◆ **Initial assumption:**

A user or a group of users have one or more digital photographs to be uploaded to the picture sharing site. These are saved on either a tablet or laptop computer. They have successfully logged on to KidsTakePics.

◆ **Normal:**

The user chooses to upload photos and is prompted to select the photos to be uploaded on the computer and to select the project name under which the photos will be stored. Users should also be given the option of inputting keywords that should be associated with each uploaded photo. Uploaded photos are named by creating a conjunction of the user name with the filename of the photo on the local computer.

On completion of the upload, the system automatically sends an email to the project moderator asking them to check new content and generates an on-screen message to the user that this has been done.

Stories and scenarios

[Uploading photos to KidsTakePics]

◆ **What can go wrong:**

No moderator is associated with the selected project. An email is automatically generated to the school administrator asking them to nominate a project moderator. Users should be informed that there could be a delay in making their photos visible.

Photos with the same name have already been uploaded by the same user. The user should be asked if they wish to re-upload the photos with the same name, rename the photos or cancel the upload. If they chose to re-upload the photos, the originals are overwritten. If they chose to rename the photos, a new name is automatically generated by adding a number to the existing file name.

◆ **Other activities:** The moderator may be logged on to the system and may approve photos as they are uploaded.

◆ **System state on completion:** User is logged on. The selected photos have been uploaded and assigned a status 'awaiting moderation'. Photos are visible to the moderator and to the user who uploaded them.

Requirements specification

[Requirements specification]

- The process of **writing down** the user and system requirements.
- User requirements
 - They are usually written in **natural language**
 - They have to be understandable by end-users.
- System requirements
 - **Expanded versions** of the user requirements.
 - The part of the contract for the implementation of the system.

Requirements specification

[Notations for writing system requirements]

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.
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.

Natural language specification

[Natural language]

- The most widely used way of specifying SW requirements.
 - (+) expressive, intuitive, and universal.
 - (-) **vague, ambiguous.**

[Guidelines for writing natural language requirements]

- Use language consistently to **distinguish between mandatory and desirable** requirements. (e.g., shall vs should)
- Use **text highlighting** to identify key parts of the requirement.
- **Do not assume** that readers understand technical SE language.
- Associate **a rationale** with each user requirement. (e.g., why? who?)

Natural language specification

[Example requirements for the insulin pump SW 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.)*

Structured specifications

[Structured specifications]

- A way of writing system requirements where requirements are written in a standard way rather than as free-form text.
 - A description of the function or entity being specified.
 - A description of **its inputs** and **outputs**.
 - The information **needed for the computation**.
 - A description of the action to be taken.
 - A precondition, post-condition.
 - A description of the side effects (if any) of the operation.

Structured specifications

[The structured specification of a requirement for an insulin pump]

Insulin Pump/Control Software/SRS/3.3.2

Function	<u>Compute insulin dose</u> : Safe sugar level.
Description	<u>Computes the dose of</u> 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:	<u>CompDose is zero if the sugar level is stable</u> 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. (see Figure 4.14)
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Precondition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Postcondition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

Structured specifications

[The tabular specification]

- Useful when there are a number of possible alternative situation.
- Useful when complex computations are to be specified.

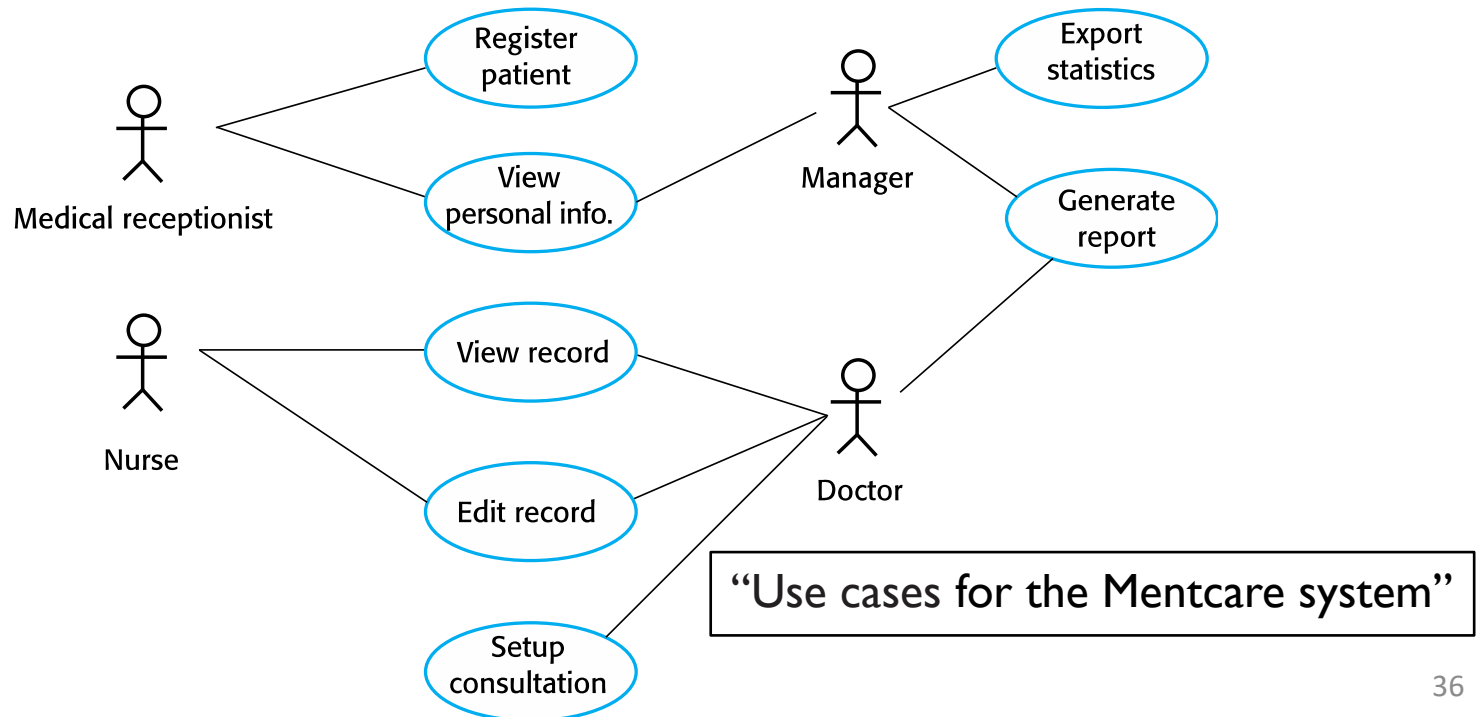
Condition	Action
Sugar level falling ($r_2 < r_1$)	CompDose = 0
Sugar level stable ($r_2 = r_1$)	CompDose = 0
Sugar level increasing and rate of increase decreasing ($(r_2 - r_1) < (r_1 - r_0)$)	CompDose = 0
Sugar level increasing and rate of increase stable or increasing $r_2 > r_1$ & $((r_2 - r_1) \geq (r_1 - r_0))$	CompDose = round $((r_2 - r_1)/4)$ If rounded result = 0 then CompDose = MinimumDose

Example requirements for the insulin pump software system

Use cases

[Use cases]

- A way of describing interactions between users and a system.
 - Actor (human or other systems), Each class of interaction



The software requirements document

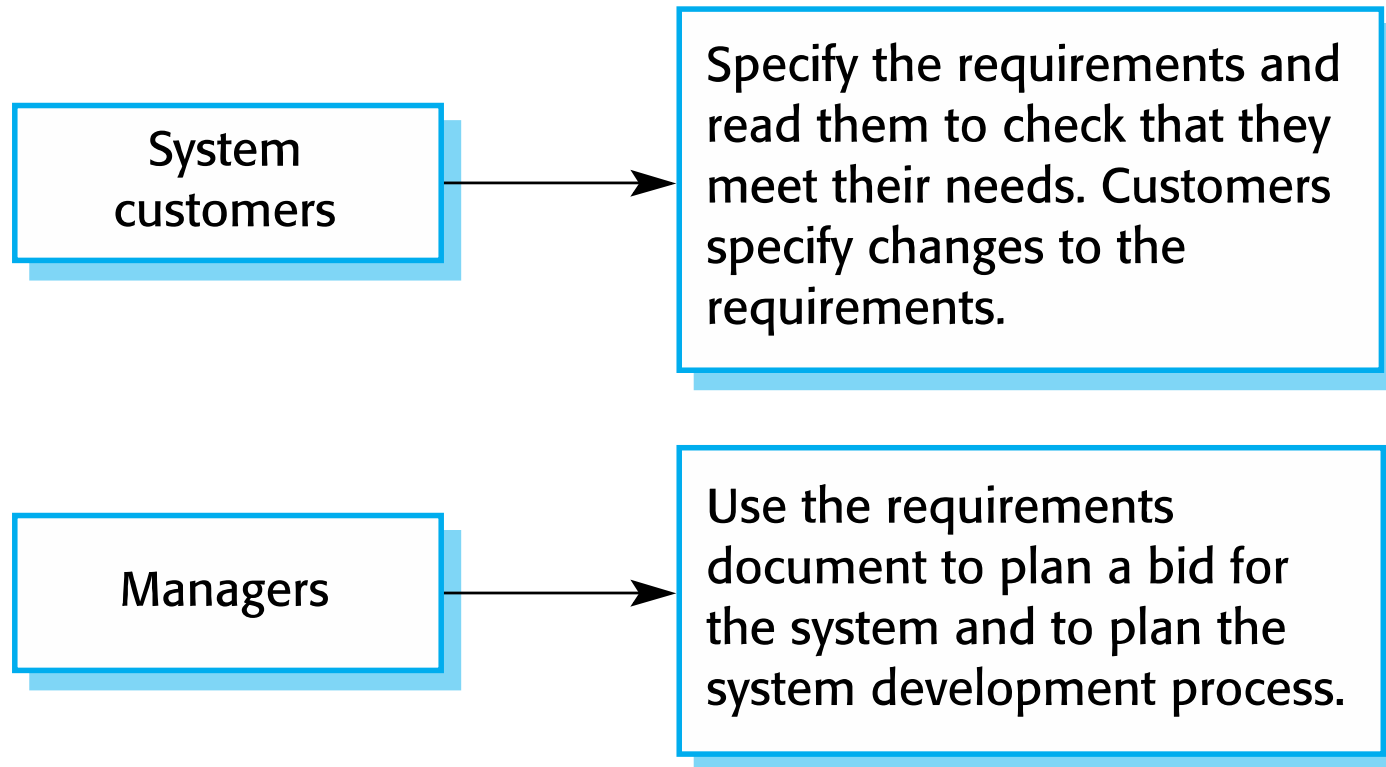
[The software requirements document]

- An official statement of what the system developers should implement. (The software requirements specification or **SRS**)
- Including both the user and system requirements.
- Having a diverse set of users.
 - Customers, managers, engineers, test engineers,

The software requirements document

[Users of a requirements document]

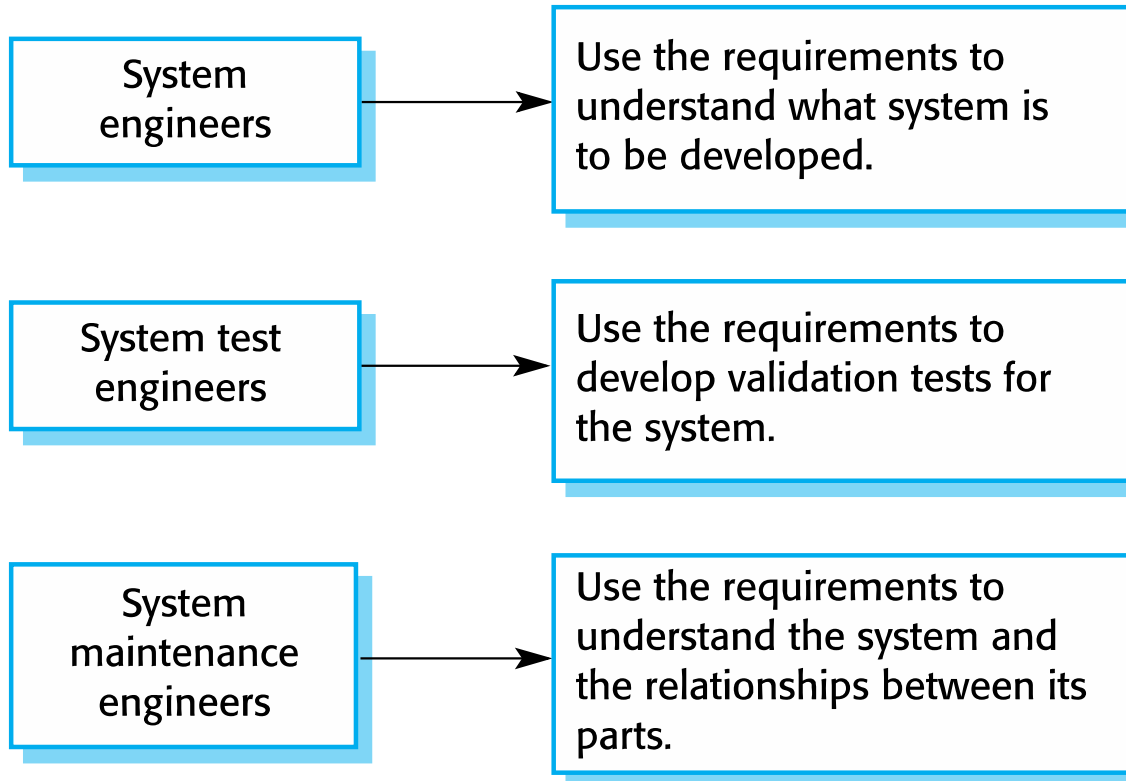
- Showing possible users of the document and how they use it.



The software requirements document

[Users of a requirements document]

- Showing possible users of the document and how they use it.



The software requirements document

[The structure of a requirements document]

Chapter	Description
Preface	This should define the expected readership of the document and describe its version history, including a rationale for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system's functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.

The software requirements document

[The structure of a requirements document]

Chapter	Description
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

The software requirements document

[The structure of a requirements document]

Chapter	Description
System requirements specification	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components and the system and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.

The software requirements document

[The structure of a requirements document]

Chapter	Description
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Requirements validation

[Requirements validation]

- The process of checking whether requirements define the system that the customer really wants.
- Critical process in a requirements document.
 - The errors can lead to extensive rework costs when these problems are discovered after the system is in service.

[Requirements validation techniques]

Requirements reviews

Prototyping

Test-case generation

Requirements validation

[Different types of checks in requirements validation]

Validity

The requirements reflect the real needs of system users.

Consistency

Requirements in the document should not conflict.

Completeness

Requirements define all functions and the constraints.

Realism checks

Requirements can be implemented within the budget.

Verifiability

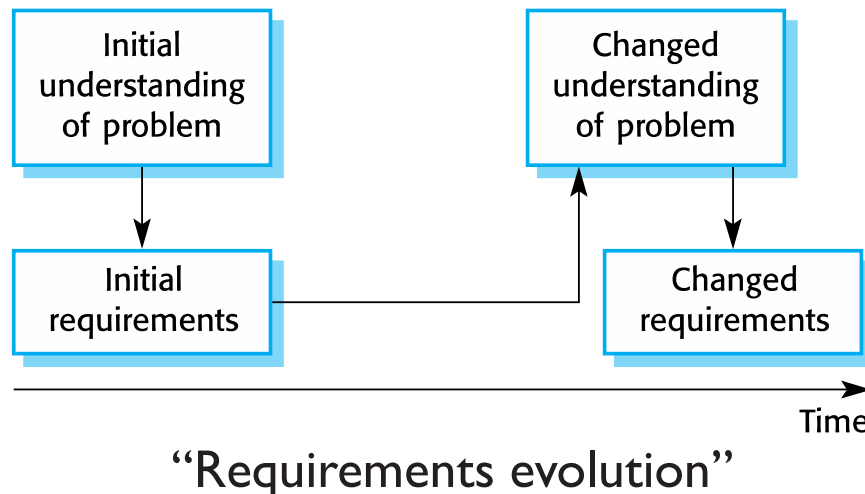
Requirements should always be written so that they are verifiable. (e.g., a set of test-cases)

Requirements Change

[Requirements Change]

- The requirements for large software systems are always changing.
 - Interfacing the system with other systems.
 - New legislation and regulations may be introduced.
 - People who pay for a system and the users of that system are not same.

...



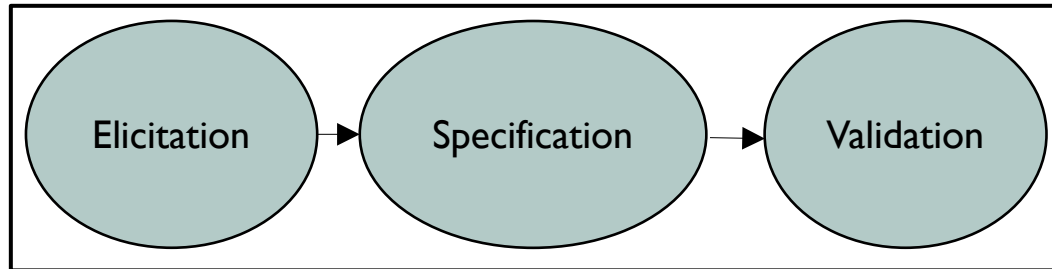
Requirements Change

[Issues for requirements management planning]

- 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.
- Tool support
 - Requirements management involves the processing of large amounts of information about the requirements. (e.g., shared spreadsheets)

Summary

Requirements Engineering



User
Requirements

System
Requirements

Functional
Requirements

Non-Functional
Requirements

Thank You