Software Engineering

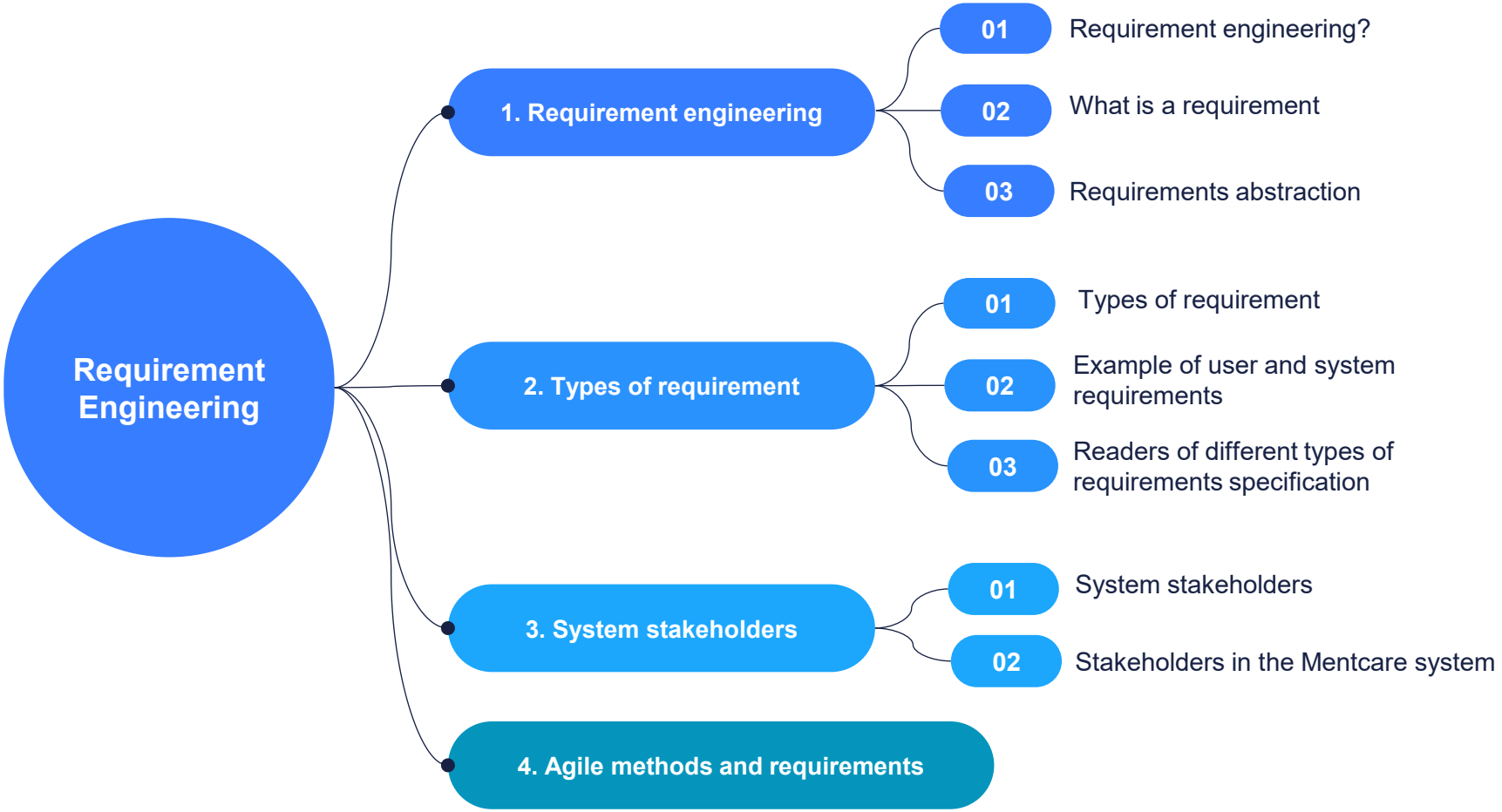
4
Requirement Engineering

School of Computer Science
Prof. Euijong Lee

Topics covered

Chapter4. Requirement Engineering

4-1: Requirement Engineering



Requirement engineering

[Requirement engineering?]

- The process of establishing the services
  - a **customer requires** from a system
  - the **constraints** under which it operates and is developed.

- The **system requirements**
  - the descriptions of the **system services**
  - **constraints** that are generated during the requirements engineering process.

Requirement engineering

[What is a requirement?]

- Requirement 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.

Requirement engineering

[Requirements abstraction (Davis)]

"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 of these documents may be called the **requirements document** for the system."

Types of requirement

[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.

Types of requirement

[Example of user and system requirements]

User requirements definition

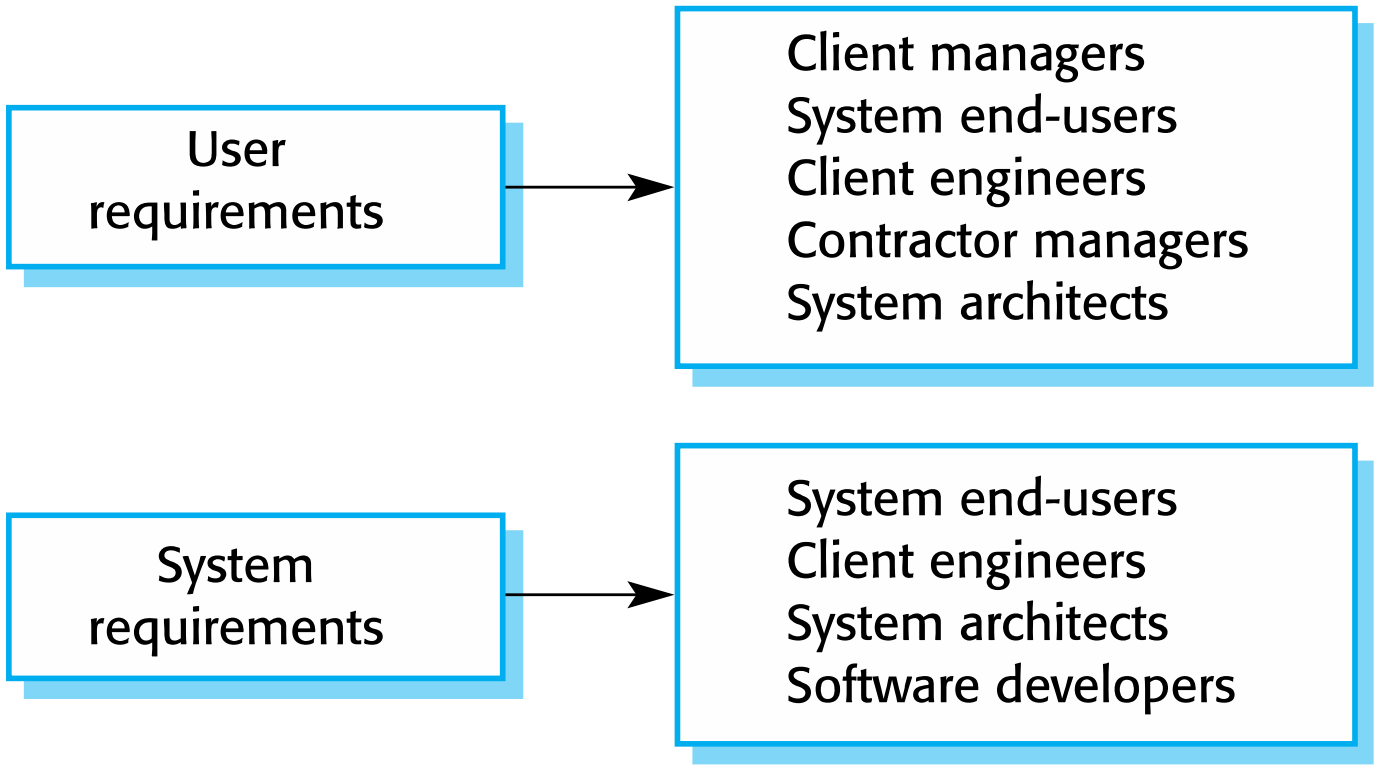
1. The Mentcare system shall generate monthly management reports showing the cost of drugs prescribed by each clinic during that month.

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.

Types of requirement

[Readers of different types of requirements specification]



System stakeholders

[System stakeholders (이해당사자)]

- 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

System 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.

System stakeholders

[Stakeholders in the Mentcare system (cont.)]

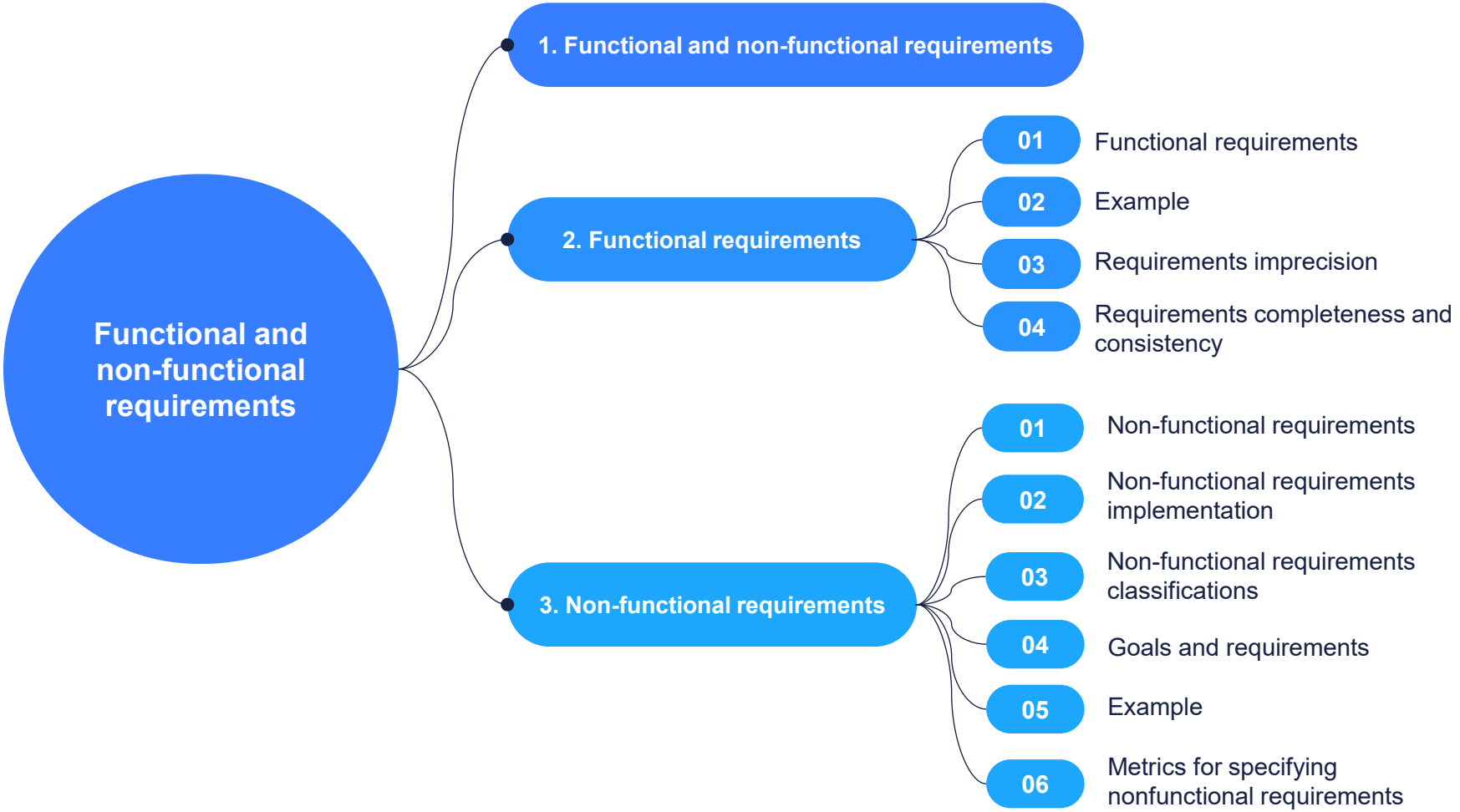
A medical ethics manager	who must ensure that the system meets current ethical guidelines for patient care.
Health care managers	who obtain management information from the system.
Medical records staff	who are responsible for ensuring that system information can be maintained and preserved, and that 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 previous lecture).
- 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**.

Therefore, we will learn 'formal' view of the requirement engineering in this chapter

4-2: 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
  - 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
  - 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

[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**.

Functional requirements

[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.

Functional requirements

[Requirements imprecision (요구사항의 부정확성)]

- **Problems** arise when functional requirements are not precisely stated.
- Ambiguous requirements may be interpreted in different ways by developers and users.
- Example: 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.

Functional requirements

[Requirements completeness and consistency]

- In principle, requirements should be both complete and consistent.
- **Complete** (완전성)
  - They should include descriptions of **all facilities required**.
- **Consistent** (일관성)
  - 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

[Non-functional requirements]

- These define **system properties and constraints**
  - Reliability
  - Response time
  - 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

[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
  - For example; 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.

Non-functional requirements

[Non-functional classifications]

Product requirements (제품 요구사항)

Requirements which **specify** that the delivered product **must behave** in a particular way : e.g. execution speed, reliability, etc.

Organisational requirements (조직 요구사항)

Requirements which are a consequence of **organisational policies** and **procedures**

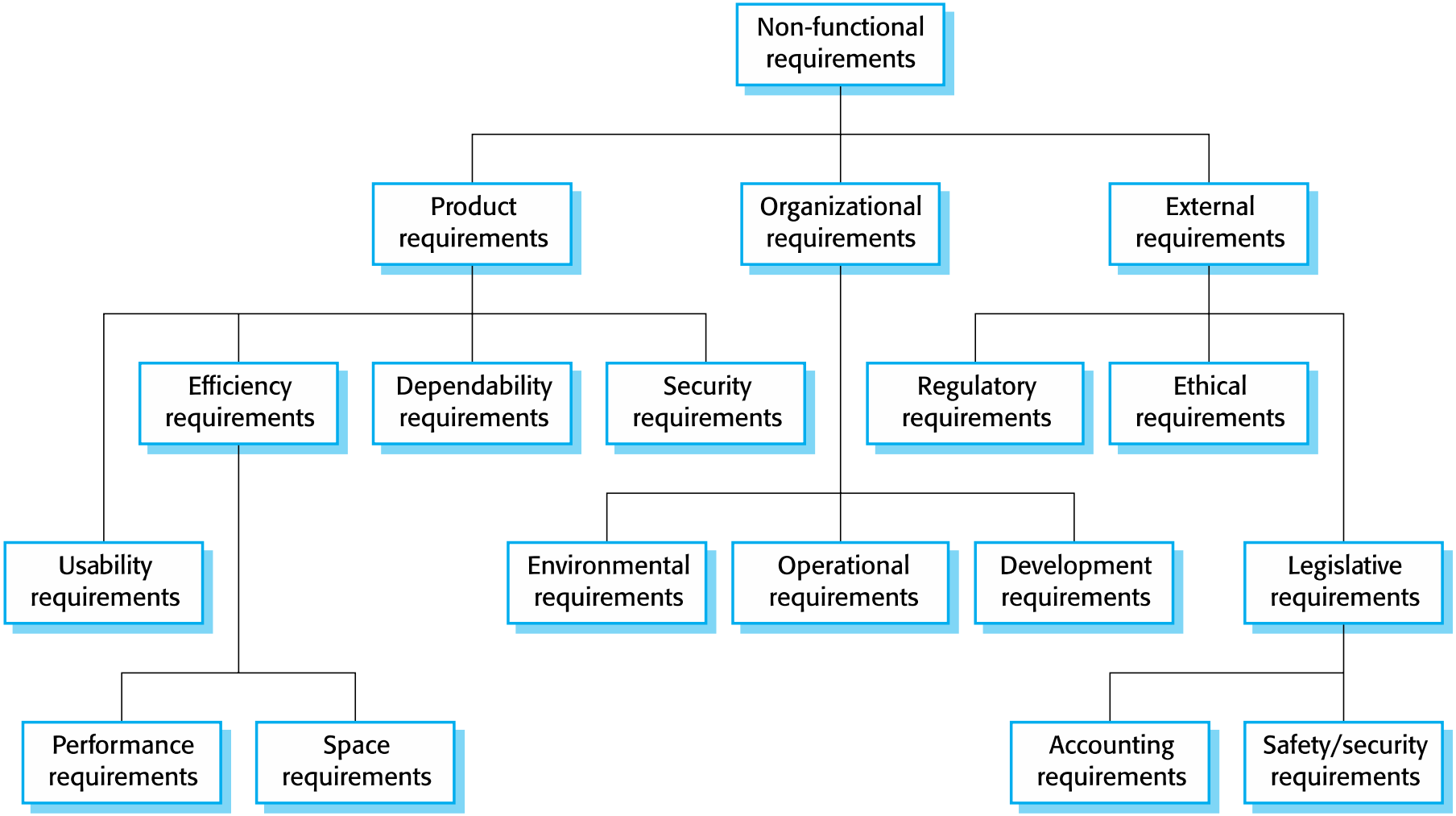
- e.g. process standards used, implementation requirements, etc.

External requirements (외부 요구사항)

Requirements which arise from factors which are **external** to the system and its development process

- e.g. interoperability requirements, legislative requirements, etc.

Non-functional requirements



Non-functional requirements

[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.

Non-functional requirements

[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.

Non-functional requirements

[Usability requirements in Mentcare system]

Goal

- 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

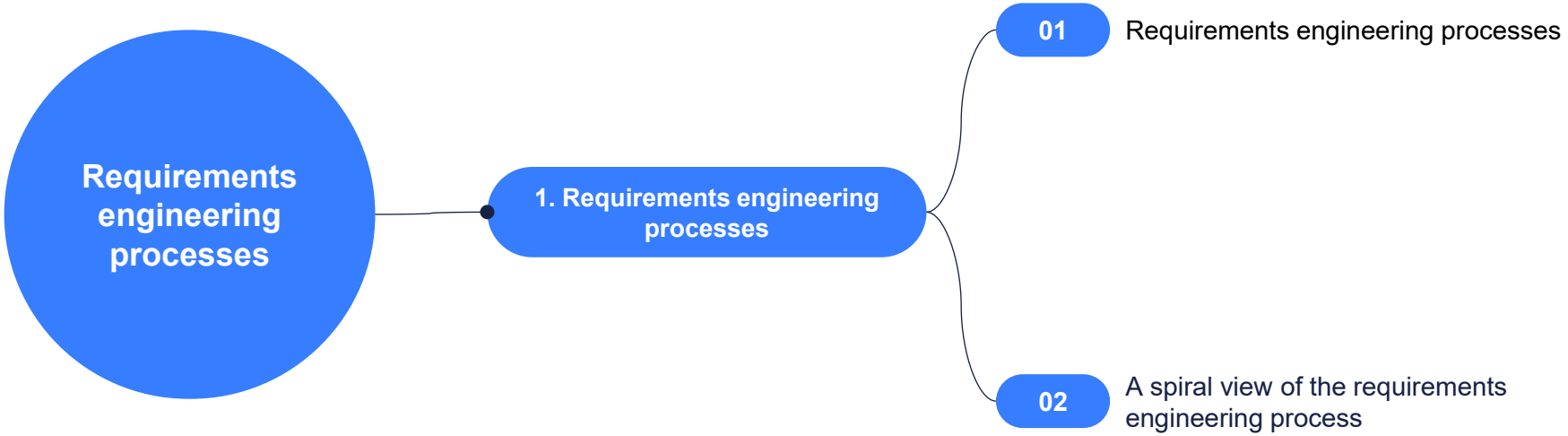
- Medical staff shall be able to use all the system functions after four hours of training.
- After this training, the average number of errors made by experienced users shall not exceed two per hour of system use

Non-functional requirements

[Metrics for specifying nonfunctional requirements]

Property	Measure
Speed (속도)	Processed transactions/second User/event response time Screen refresh time
Size (크기)	Mbytes Number of ROM chips
Ease of use (사용 편리성)	Training time Number of help frames
Reliability (신뢰성)	Mean time to failure Probability of unavailability Rate of failure occurrence Availability
Robustness (견고성)	Time to restart after failure Percentage of events causing failure Probability of data corruption on failure
Portability (이식성)	Percentage of target dependent statements Number of target systems

4-3: Requirements engineering processes



Requirements engineering processes

[Requirements engineering processes]

- The processes used for **RE vary widely depending on the application domain**,
  - The people involved
  - The organisation developing the requirements.

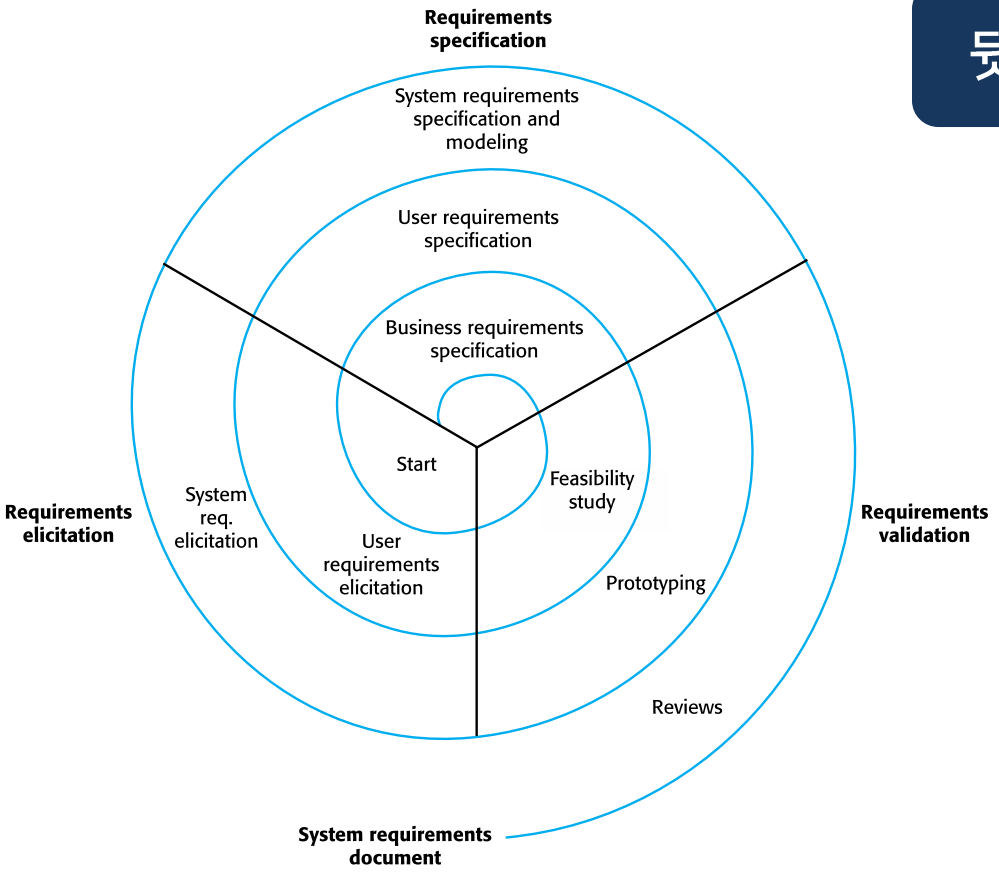
- However, there are a number of generic activities common to all processes
  - Requirements **elicitation** (도출)
  - Requirements **analysis** (분석)
  - Requirements **validation** (검증)
  - Requirements **management** (관리)

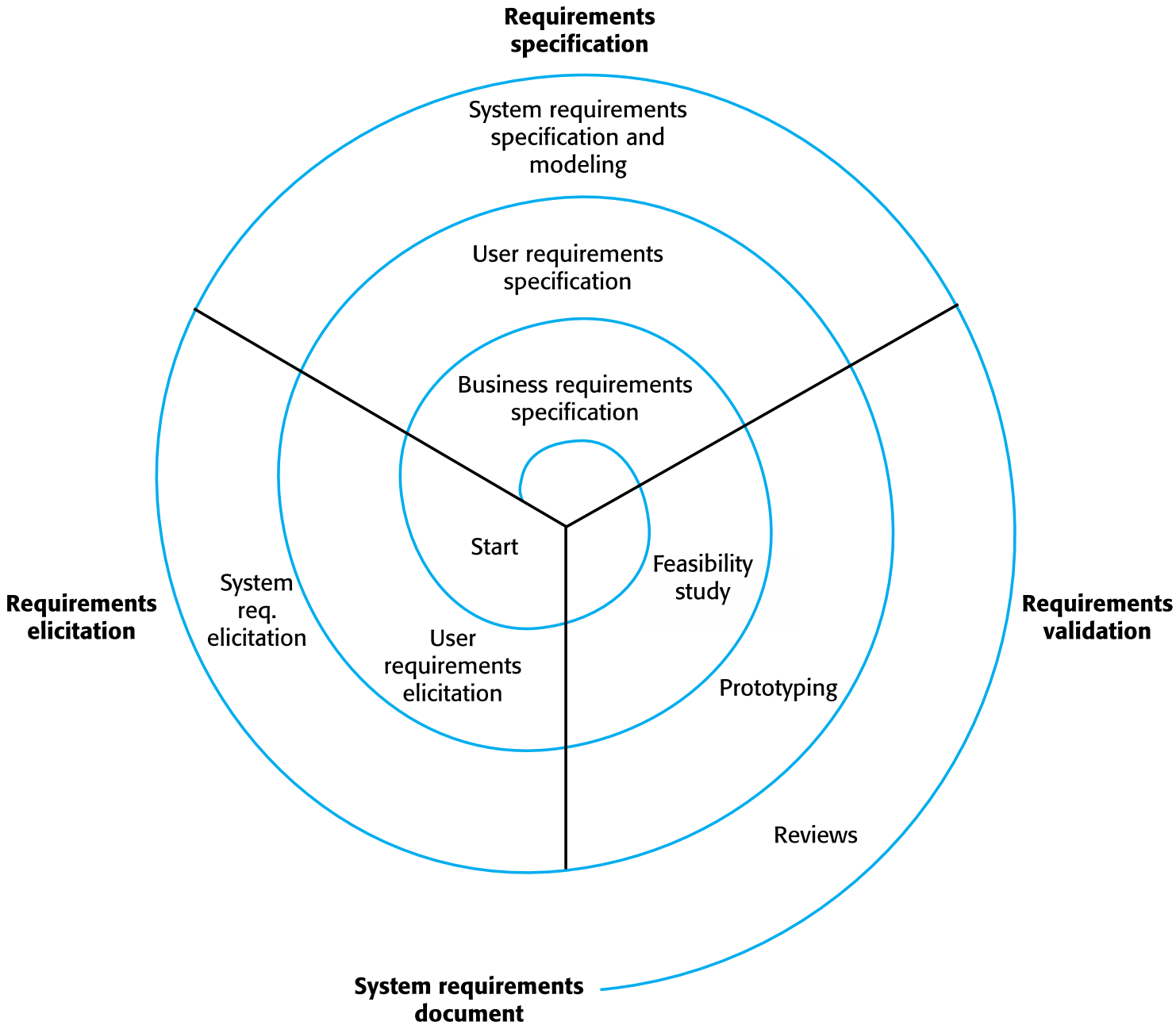
In practice,
RE is an iterative activity in which these processes are interleaved.

Requirements engineering processes

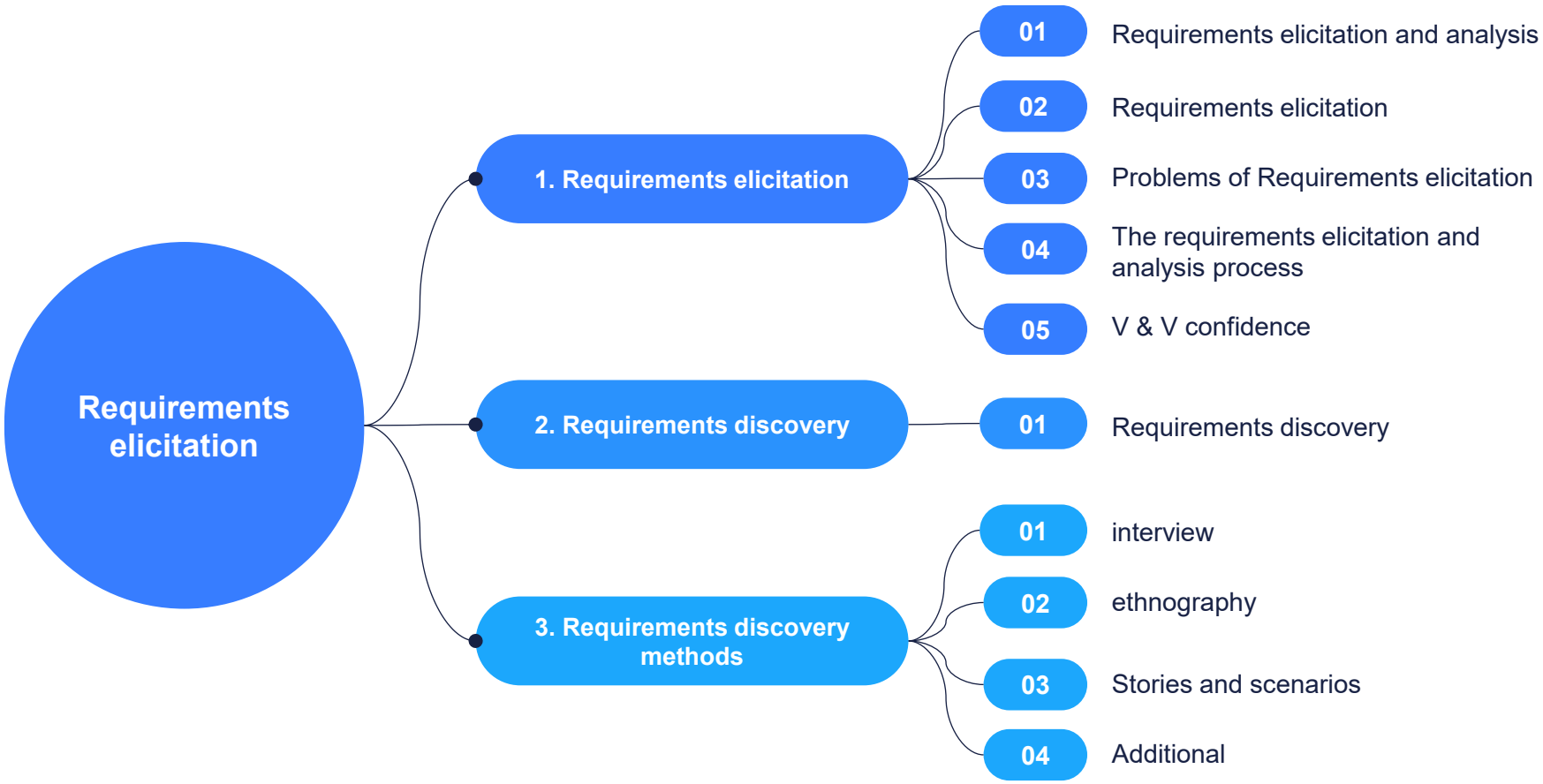
[A spiral view of the requirements engineering process]

뒷 페이지 참고!




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

4-4: Requirements elicitation



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
  - The system's operational constraints.

- **Stakeholders** (이해관계자 / 이해당사자)
  - May involve end-users, managers, engineers involved in maintenance, domain experts, trade unions, etc.

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 (발견 및 이해),
  - Requirements classification and organization (분류 및 구성),
  - Requirements prioritization and negotiation (우선순위 부여 및 협상),
  - Requirements specification (구체화/문서화).

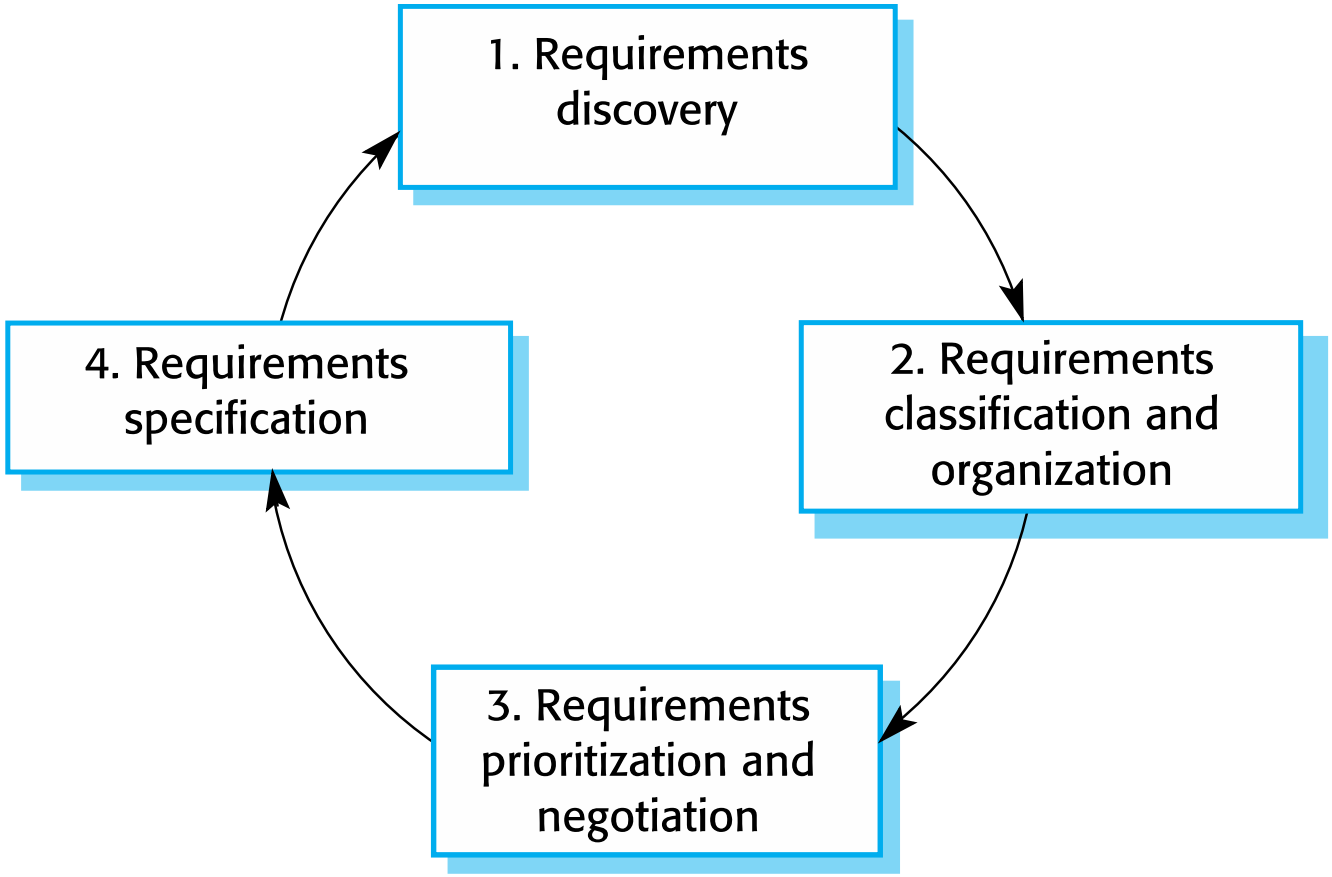
Requirements elicitation

[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. (e.g., 게임셧다운제)
- The **requirements change** during the analysis process.
  - New stakeholders may emerge and the business environment may change.

Requirements elicitation

[The requirements elicitation and analysis process]



Requirements elicitation

[Process activities]

Requirements discovery (발견 및 이해)

- Interacting with stakeholders to **discover their requirements**
- **Domain requirements** are also discovered at this stage.

Requirements classification and organization (분류 및 구성)

- **Groups** related requirements and organizes them into **coherent clusters**.

Prioritisation and negotiation (우선순위 및 협상)

- **Prioritising** requirements and **resolving** requirements **conflicts**.

Requirements specification (구체화 / 문서화)

- Requirements are **documented** and input into the next round of the spiral.

Requirements discovery

- Requirement discovery process are”
  - **Gathering information** about the required and existing systems
  - **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.

Requirements discovery methods : interview

[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 discovery methods : interview

[Interviews in practice]

- Normally a **mix of closed and open-ended** interviewing.
- **Interviews** are good for :
  - Getting an **overall understanding** of what stakeholders do
  - **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.

Requirements discovery methods : interview

[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 discovery methods : ethnography

[Definition of ethnography]

- Ethnography is the branch of anthropology in which different cultures are studied and described.

- 에스노그래피
  :사회 일반 사회와 문화의 여러 가지 현상을 정량적이고 정성적인 조사 기법을 이용한 현장 조사를 통하여 연구하는 학문 분야.
  ⇒규범 표기는 미확정이다

Requirements discovery methods : ethnography

[Ethnography in RE]

- 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.

Requirements discovery methods : ethnography

[Scope of ethnography]

- **Requirements** that are derived from the **way that people actually work** rather than the way which process definitions suggest that they ought to work.

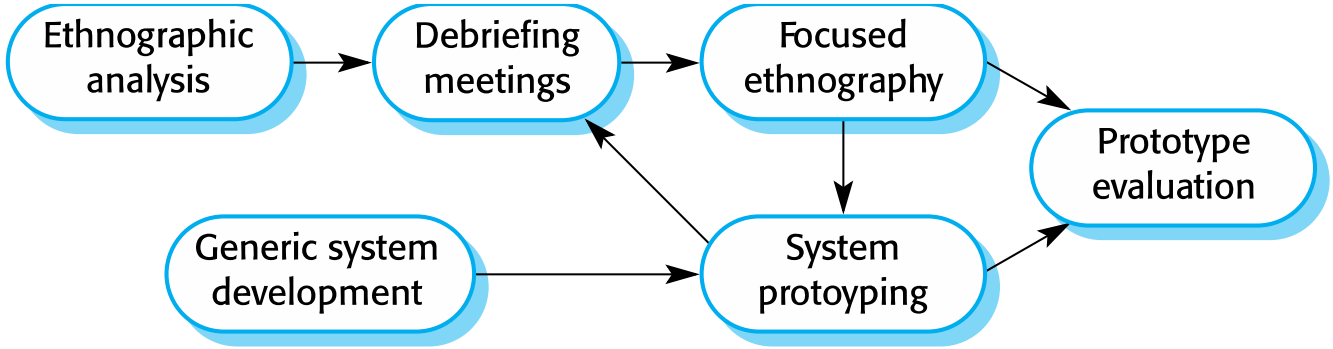
- **Requirements** that are **derived** from cooperation and awareness of other **people's activities**.
  - Awareness of what other people are doing leads to changes in the ways in which we do things.

- Ethnography is effective for **understanding existing processes** but **cannot identify new features** that should be added to a system.

Requirements discovery methods : ethnography

[Focused ethnography]

- **Combines ethnography with prototyping**
  - Prototype development results in unanswered questions which focus the ethnographic analysis.
  - Fig. Ethnography and prototyping for requirements analysis



- The **problem** with ethnography is that **it studies existing practices** which may have some historical basis which is no longer relevant.
  (= 혁신에 취약함)
  - e.g., Nokia and Apple in mobile phone market

Requirements discovery methods : Stories and scenarios

[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 stories and scenarios
  - Stakeholders can comment on their situation with respect to the story.

Requirements discovery methods : Stories and scenarios

[Example: Photo sharing in the classroom (iLearn)]

Jack is a primary school teacher in Ullapool (a village in northern Scotland). He has decided that a class project should be focused around the fishing industry in the area, looking at the history, development and economic impact of fishing. As part of this, 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 (a history resources site) to access newspaper archives and photographs. However, Jack also needs a photo sharing site as he wants pupils to take and comment on each others' 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 uses 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.

Requirements discovery methods : Stories and scenarios

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 discovery methods : Stories and scenarios

[Example: Uploading photos (iLearn)]

- **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 upload photos and they are prompted to select the photos to be uploaded on their computer and to select the project name under which the photos will be stored. They 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.

Requirements discovery methods : Stories and scenarios

[Example: Uploading photos (iLearn) (cont.)]

- **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 discovery methods : Additional

Presentation of customer

- Developer can understand requirement in early stage
- Provide useful guideline of the system

Research documents

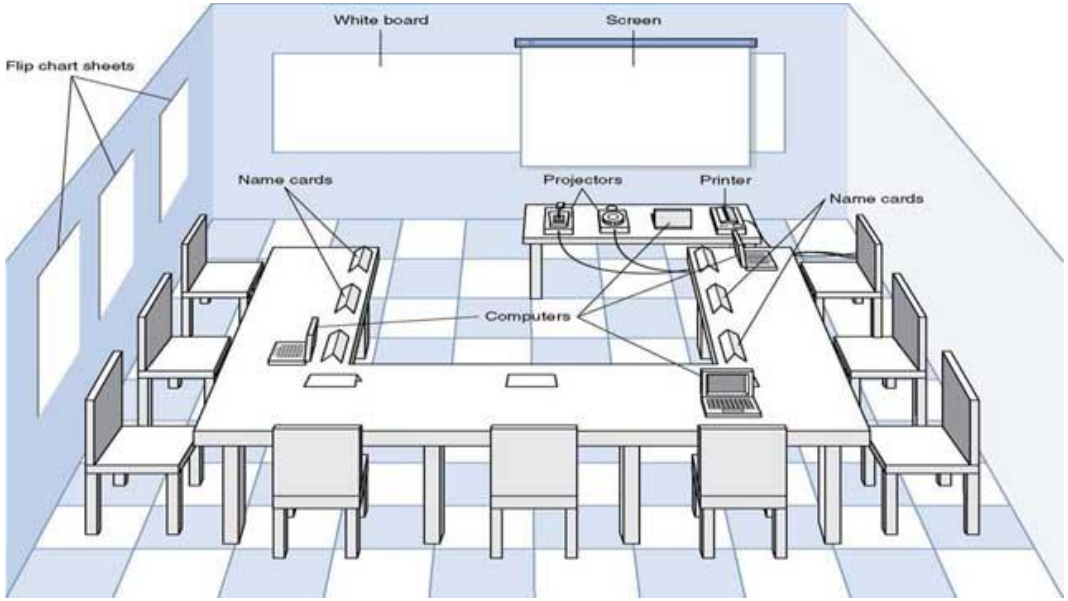
- Research related documents like:
  - Similar project
  - Document and forms that are used in the domain fields
    - e.g., receipt form, order forms, etc.
  - Standards
  - Government regulations

Requirements discovery methods : Additional

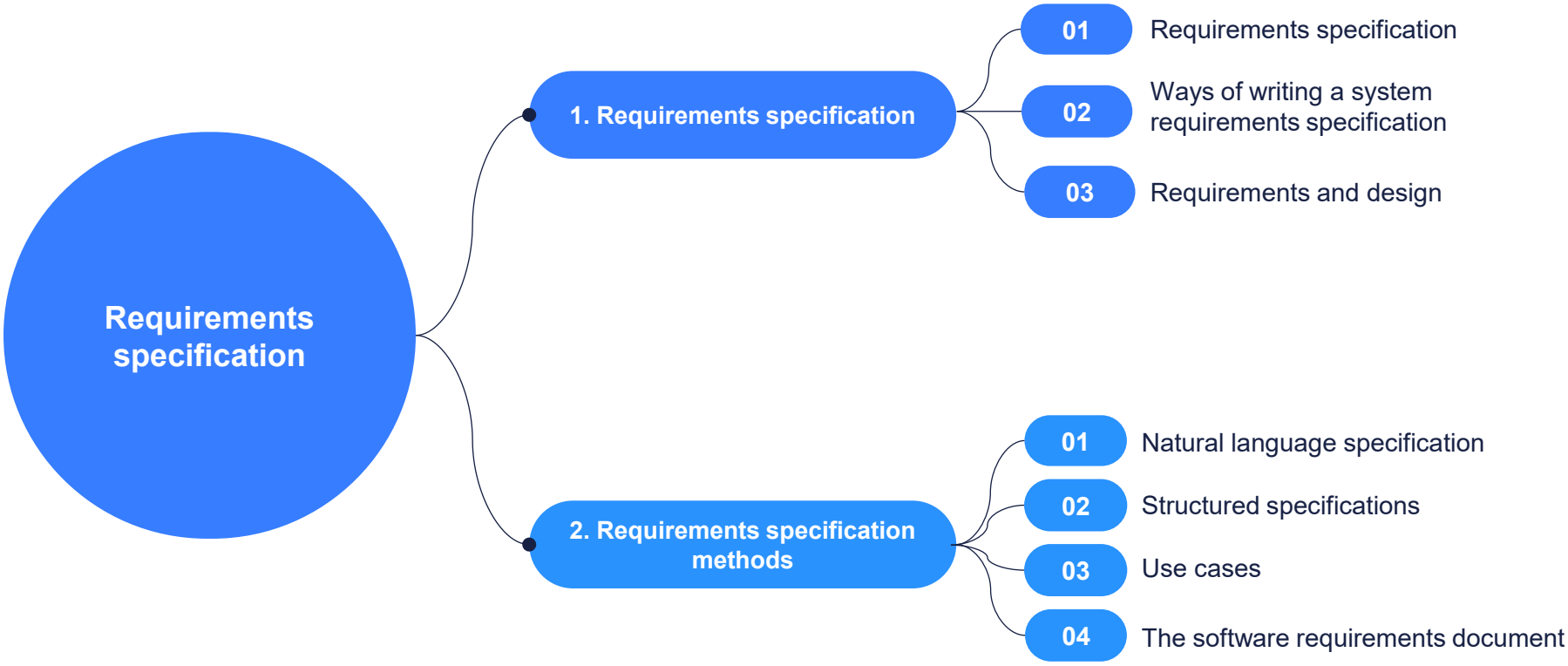
Brain storming

- Brainstorming is a **group creativity technique** by which efforts are made to find a conclusion for a specific problem by gathering a list of ideas spontaneously contributed by its members.
- JAD (Joint Application Development)

Source: Davis, K., Titchkosky, L., & Werbicki P. (2002) Joint Application Design, Participatory Design.



4-5: Requirements specification



Requirements specification

[Requirements specification (요구사항 명세)]

- 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 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.

Requirements specification

[Ways of writing a system requirements specification (cont.)]

Notation	Description
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 m**athematical 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

[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 methods : 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.

Requirements specification methods : Natural language specification

[Guidelines for writing requirements]

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

Requirements specification methods : Natural language specification

[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.)*

Requirements specification methods : Natural language specification

[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.

Requirements specification methods : Structured specifications

[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.

Requirements specification methods : Structured specifications

[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 methods : Structured specifications

[Example
- 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.

Requirements specification methods : Structured specifications

[Example (cont.)
- 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.

Requirements specification methods : Structured specifications

[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.

Requirements specification methods : Structured specifications

[Tabular specification]

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

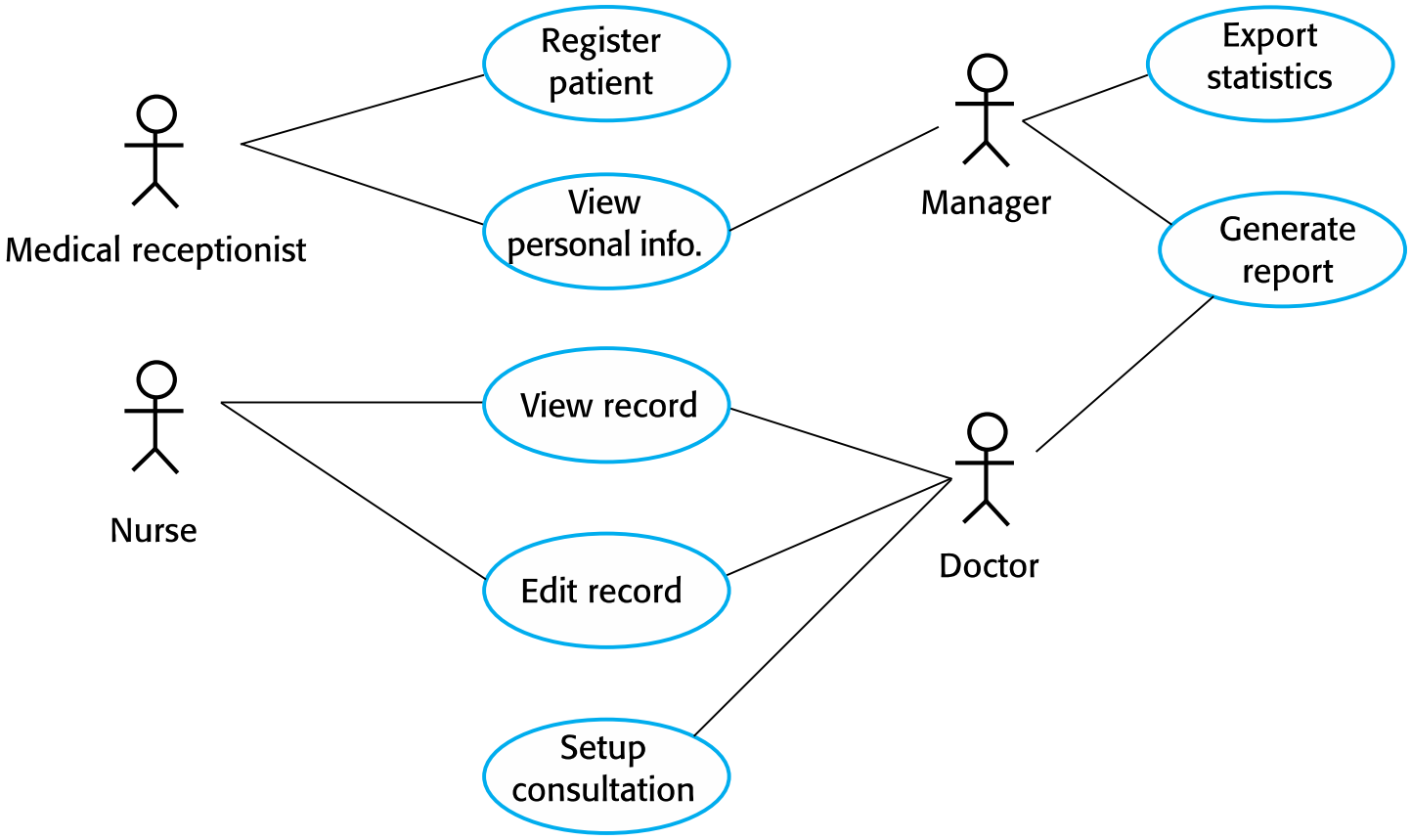
Requirements specification methods : Use cases

[Use cases]

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

Requirements specification methods : Use cases

[Use cases for the Mentcare system]



Requirements specification methods : The software requirements document

[The software requirements document (소프트웨어 요구사항 문서)]

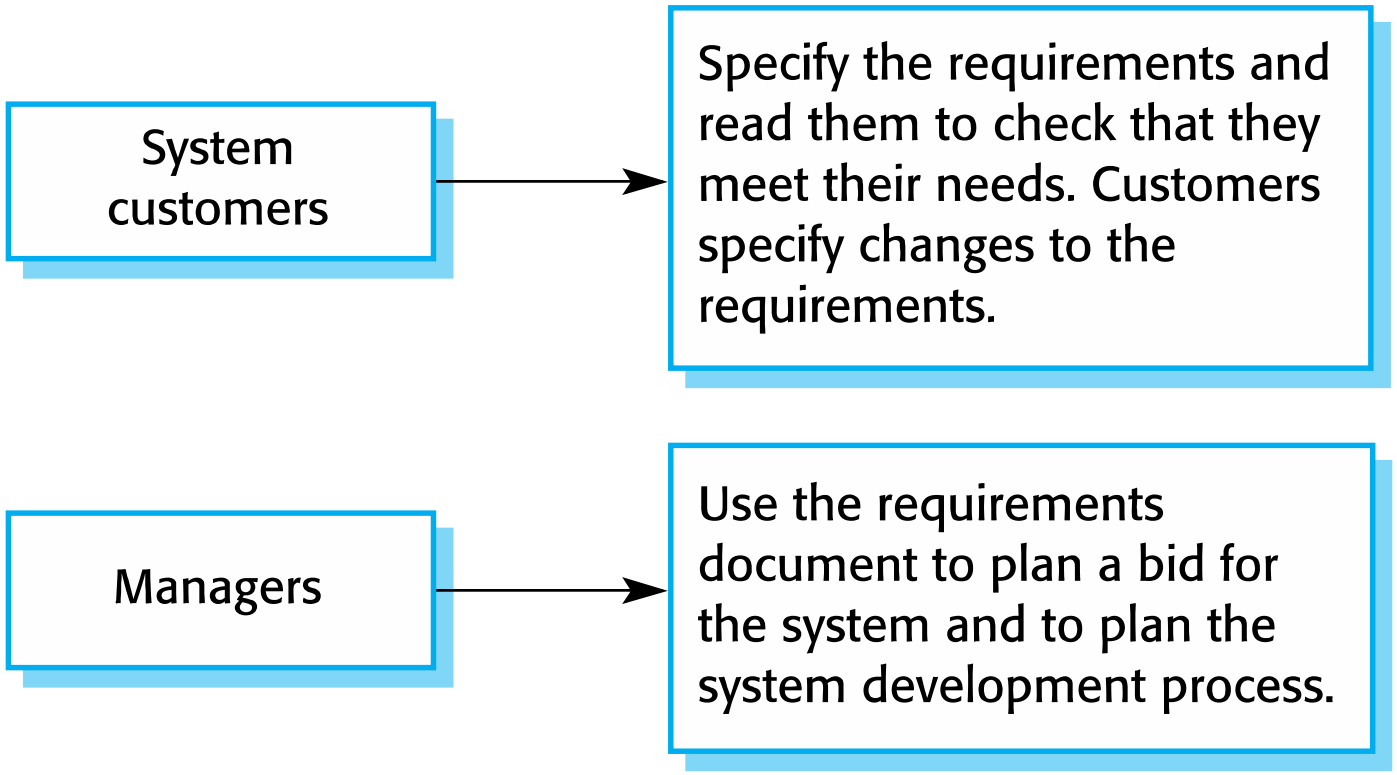
- The software requirements document is the **official statement** of what is required of the system developers.

- 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.

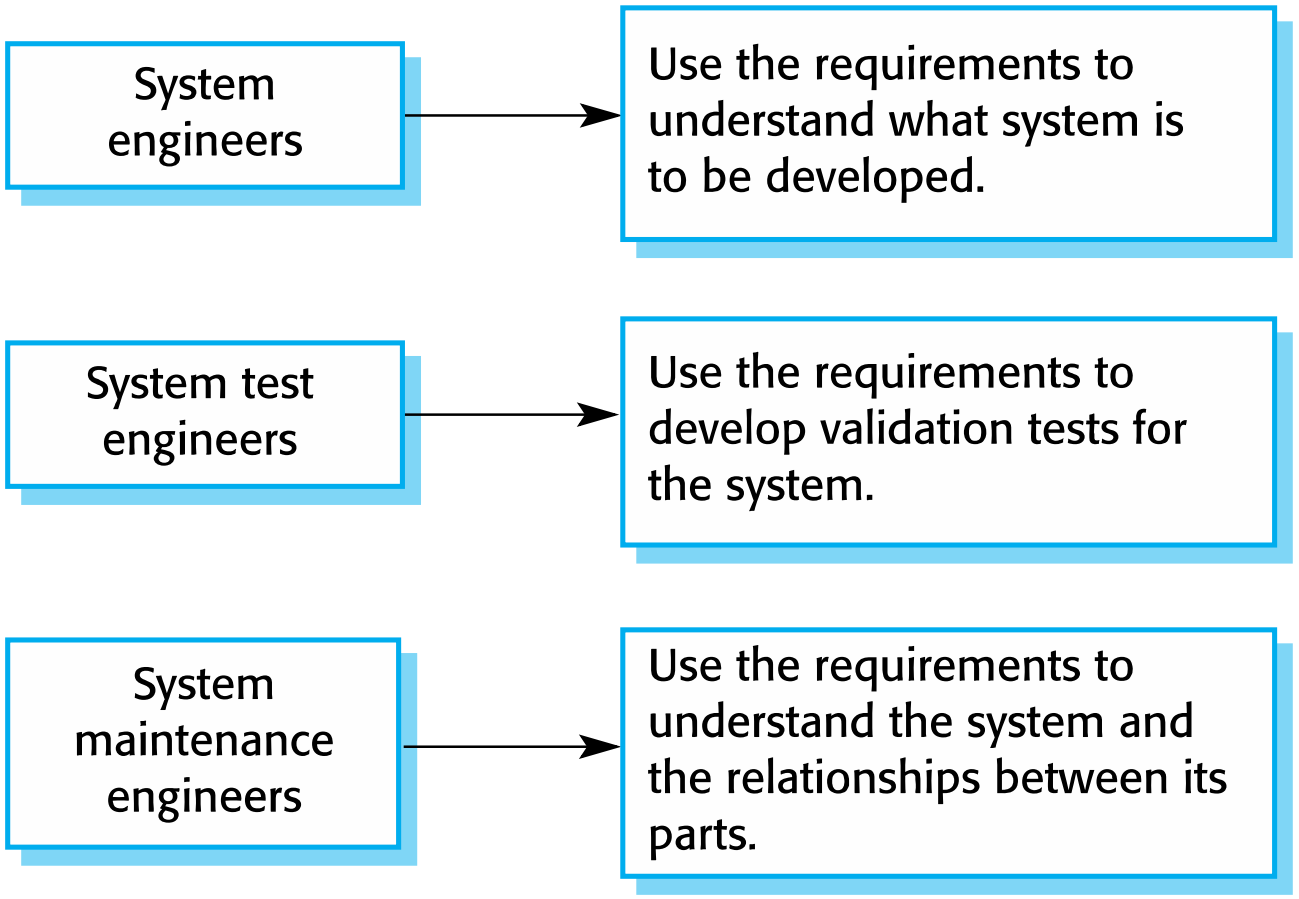
Requirements specification methods : The software requirements document

[Users of a requirements document]



Requirements specification methods : The software requirements document

[Users of a requirements document (cont.)]



Requirements specification methods : The software 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.

Requirements specification methods : The software requirements document

[The structure of a requirements document]

Chapter	Description
Preface (개요)	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 (도입)	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 (용어사전)	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.

Requirements specification methods : The software requirements document

[The structure of a requirements document (cont.)]

Chapter	Description
User requirements definition (사용자 요구사항 정의)	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 (시스템 아키텍처)	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 methods : The software requirements document

[The structure of a requirements document (cont.)]

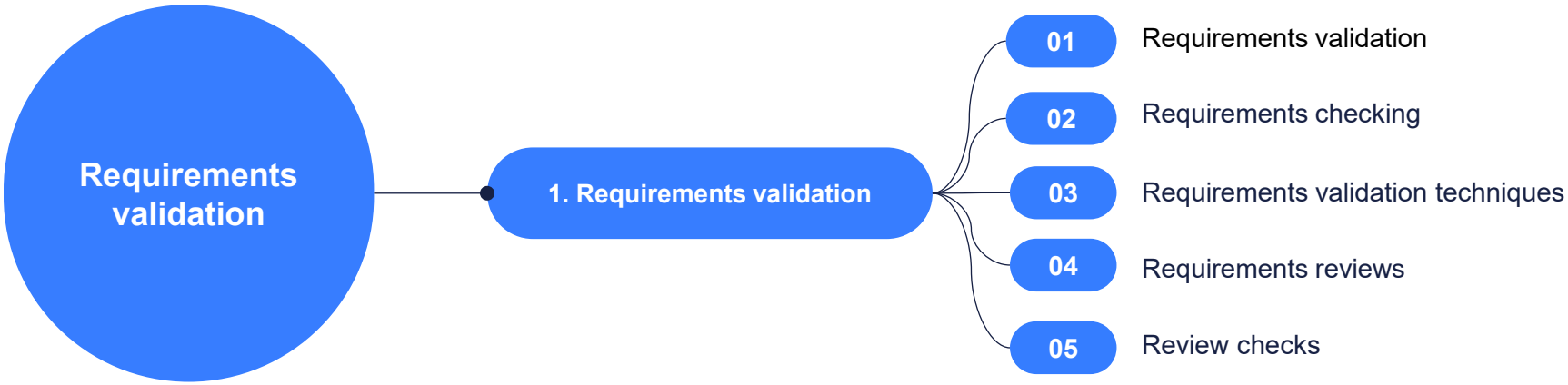
Chapter	Description
System requirements specification (시스템 요구사항 명세)	This should describe the functional and nonfunctional requirements in more detail. If necessary, further detail may also be added to the nonfunctional requirements. Interfaces to other systems may be defined.
System models (시스템 모델)	This 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 (시스템 진화)	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.

Requirements specification methods : The software requirements document

[The structure of a requirements document (cont.)]

Chapter	Description
Appendices (부록)	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 (색인)	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.

4-5: Requirements validation



Requirements validation

[Requirements validation (요구사항 검증)]

- 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

[Requirements checking]

Validity (유효성)	Does the system provide the functions which best support the customer's needs?
Consistency (일관성)	Are there any requirements conflicts?
Completeness (완전성)	Are all functions required by the customer included?
Realism (실현성)	Can the requirements be implemented given available budget and technology
Verifiability (검증가능성)	Can the requirements be checked?

Requirements validation

[Requirements validation techniques]

Requirements reviews

- Systematic manual analysis of the requirements.

Prototyping

- Using an executable model of the system to check requirements. Covered in previous lecture.

Test-case generation

- Developing tests for requirements to check testability.

Requirements validation

[Requirements reviews]

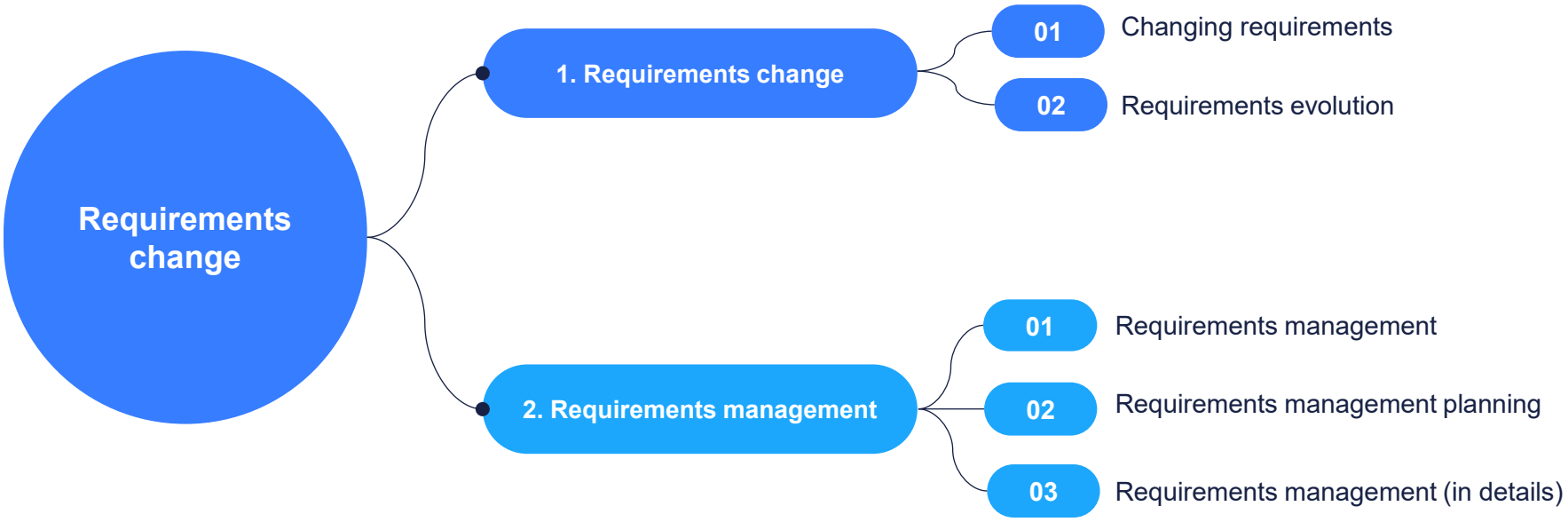
- Regular reviews should be held while the **requirements definition** is being formulated.
- **Both client and contractor staff** should be involved in reviews.
- Reviews may be **formal** (with completed documents) or **informal**.
  - Good communications between developers, customers and users can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

Requirements validation

[Review checks]

Verifiability (검증가능성)	Is the requirement realistically testable?
Comprehensibility (이해가능성)	Is the requirement properly understood?
Traceability (추적가능성)	Is the origin of the requirement clearly stated?
Adaptability (확장/적용 가능성)	Can the requirement be changed without a large impact on other requirements?

4-7: Requirements change



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)
  - New legislation and regulations may be introduced that the system must necessarily abide by.

Requirements change

[Changing requirements (cont.)]

- 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
  - After delivery, new features may have to be added for user support if the system is to meet its goals.

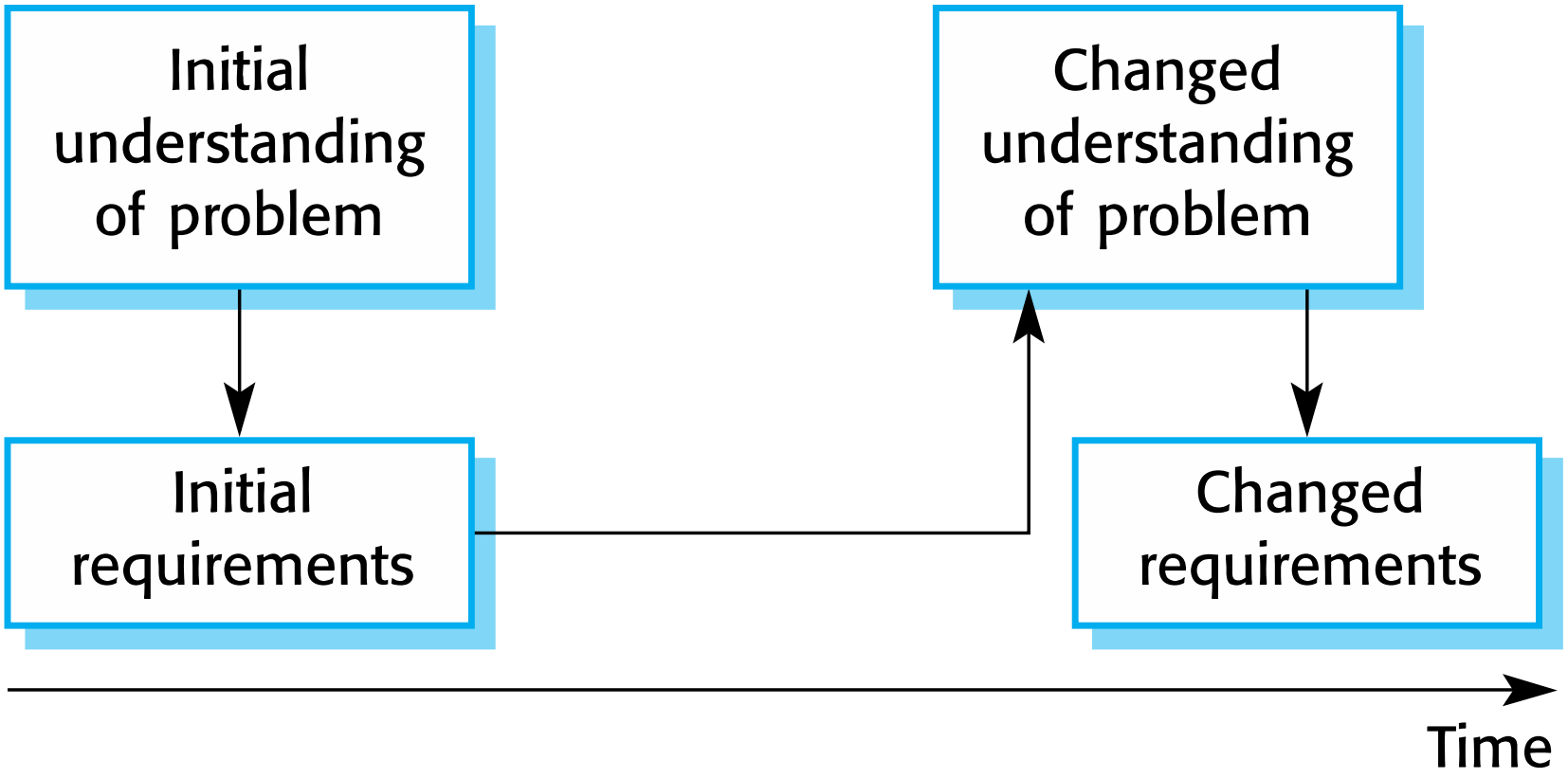
Requirements change

[Changing requirements (cont.)]

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

Requirements change

[Requirements evolution]



Requirements management

[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

[Requirements management planning]

- Establishes the level of requirements management detail that is required.

- Requirements management decisions:
  - **Requirements identification** (요구사항 식별)
    - Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.
  - **A change management process** (변경관리 프로세스)
    - This is the set of activities that assess the impact and cost of changes.
    - We will learn this process in more detail in the next slides

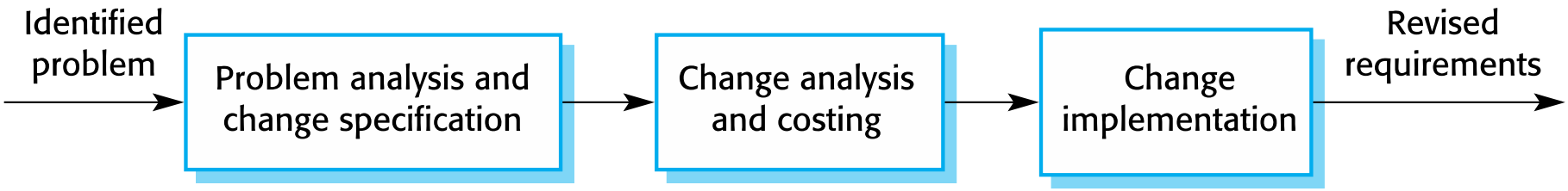
Requirements management

[Requirements management planning (cont.)]

- Requirements management decisions: (cont.)
  - **Traceability policies** (추적가능성 정책)
    - These policies define the relationships
      - » between each requirement
      - » between the requirements and the system design that should be recorded.
  - **Tool support** (도구 지원)
    - Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

Requirements management

[Requirements change management (in details)]



- Deciding if a requirements change should be accepted
  - **Problem analysis and change specification** (분석 문제 및 변경 명세)
    - 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.

[Requirements change management (in details) (cont.)]

- Deciding if a requirements change should be accepted (cont.)
  - **Change analysis and costing** (변경 분석 및 비용 산출)
    - 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** (변경 구현)
    - 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 (1)

- Requirements for a software system set out what the system should do and define constraints on its operation and implementation.
- Functional requirements are statements of the services that the system must provide or are descriptions of how some computations must be carried out.
- Non-functional requirements often constrain the system being developed and the development process being used.
- They often relate to the emergent properties of the system and therefore apply to the system as a whole.

Key Points (2)

- The requirements engineering 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 (3)

- ✧ Requirements specification is the process of formally documenting the user and system requirements and creating a software requirements document.

- ✧ The software requirements document is an agreed statement of the system requirements. It should be organized so that both system customers and software developers can use it.

Key Points (4)

- ✧ 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 these changes.