502045

Software Engineering

Chapter 03

Lesson 03: Requirement Process

Topics covered

- ✧ Define Requirement
- ✧ Functional and non-functional requirements
- ✧ Requirement specification
- ✧ Requirements engineering processes:
  - ▪ Requirements elicitation and analysis
  - ▪ Requirements validation
  - ▪ Requirements management
  - ▪ Develop SRS (The software requirements document)

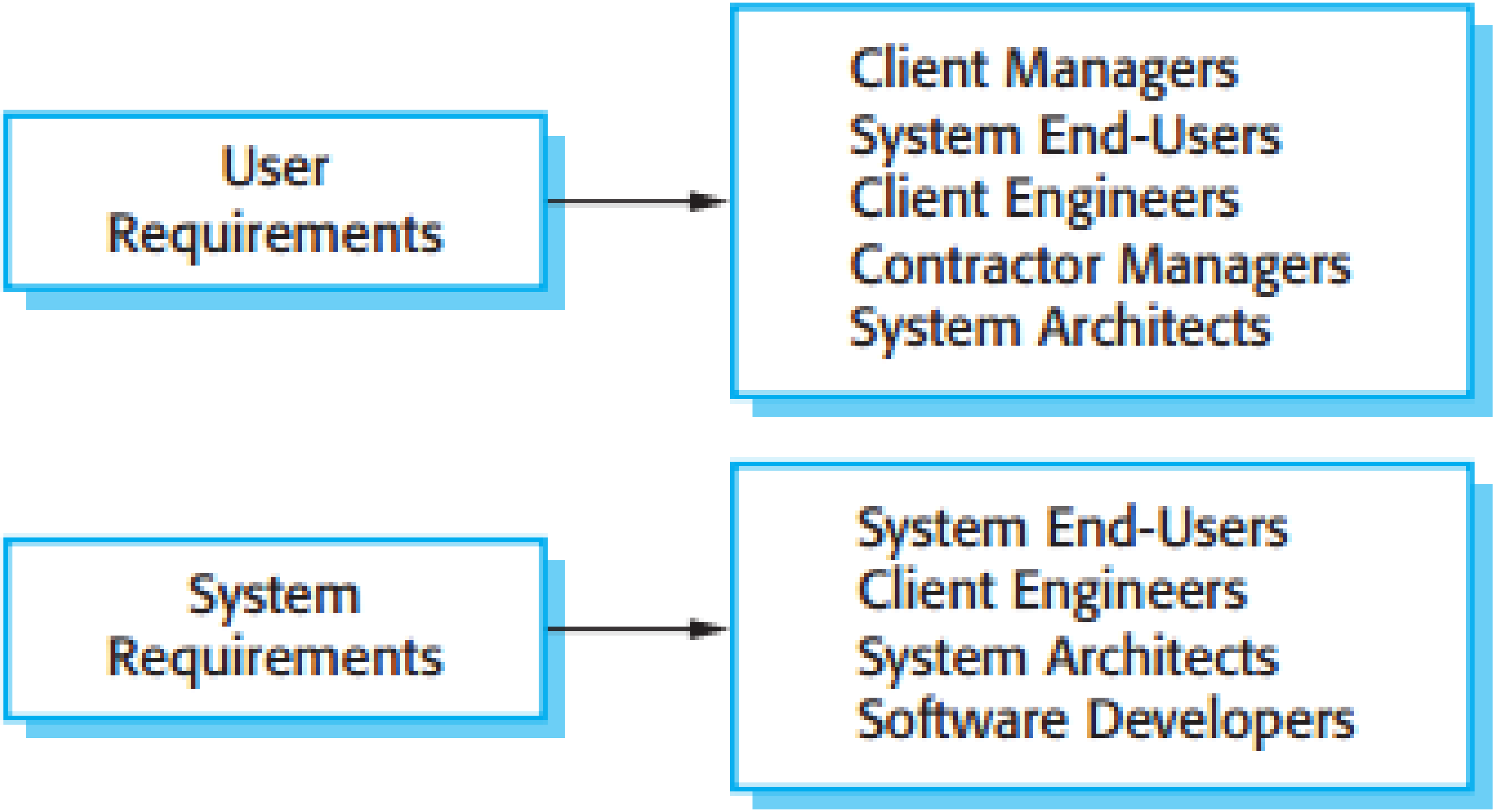
What is a requirement?

- It may range from a high-level abstract statement of a service or of a system constraint to a detailed mathematical functional specification.
- This is inevitable as requirements may serve a dual function
  - May be the basis for a bid for a contract - therefore must be open to interpretation;
  - May be the basis for the contract itself - therefore must be defined in detail;
  - Both these statements may be called requirements.

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(nguoinhanthau).

Readers of different types of requirements specification



User Requirements

Client Managers
System End-Users
Client Engineers
Contractor Managers
System Architects

System Requirements

System End-Users
Client Engineers
System Architects
Software Developers

Topics covered

- ✧ Define Requirement
- ✧ Functional and non-functional requirements
- ✧ Requirement specification
- ✧ Requirements engineering processes:
  - ▪ Requirements elicitation and analysis
  - ▪ Requirements validation
  - ▪ Requirements management
  - ▪ Develop SRS (The software requirements document)

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.

Requirements imprecision

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

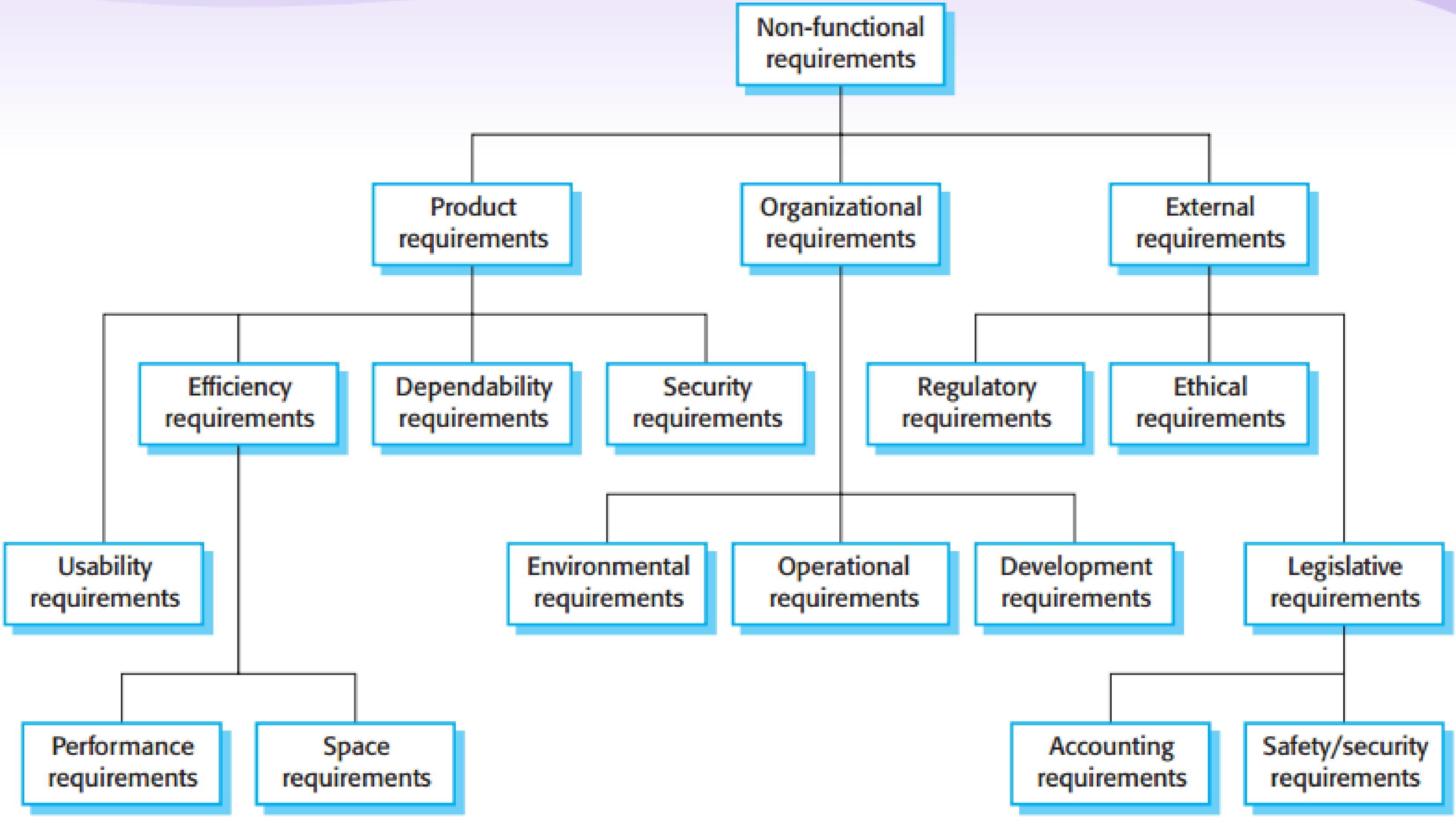
Non-functional requirements implementation

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

Non-functional classifications (Self Study)

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

Types of NF (Self Study)



Goals and requirements

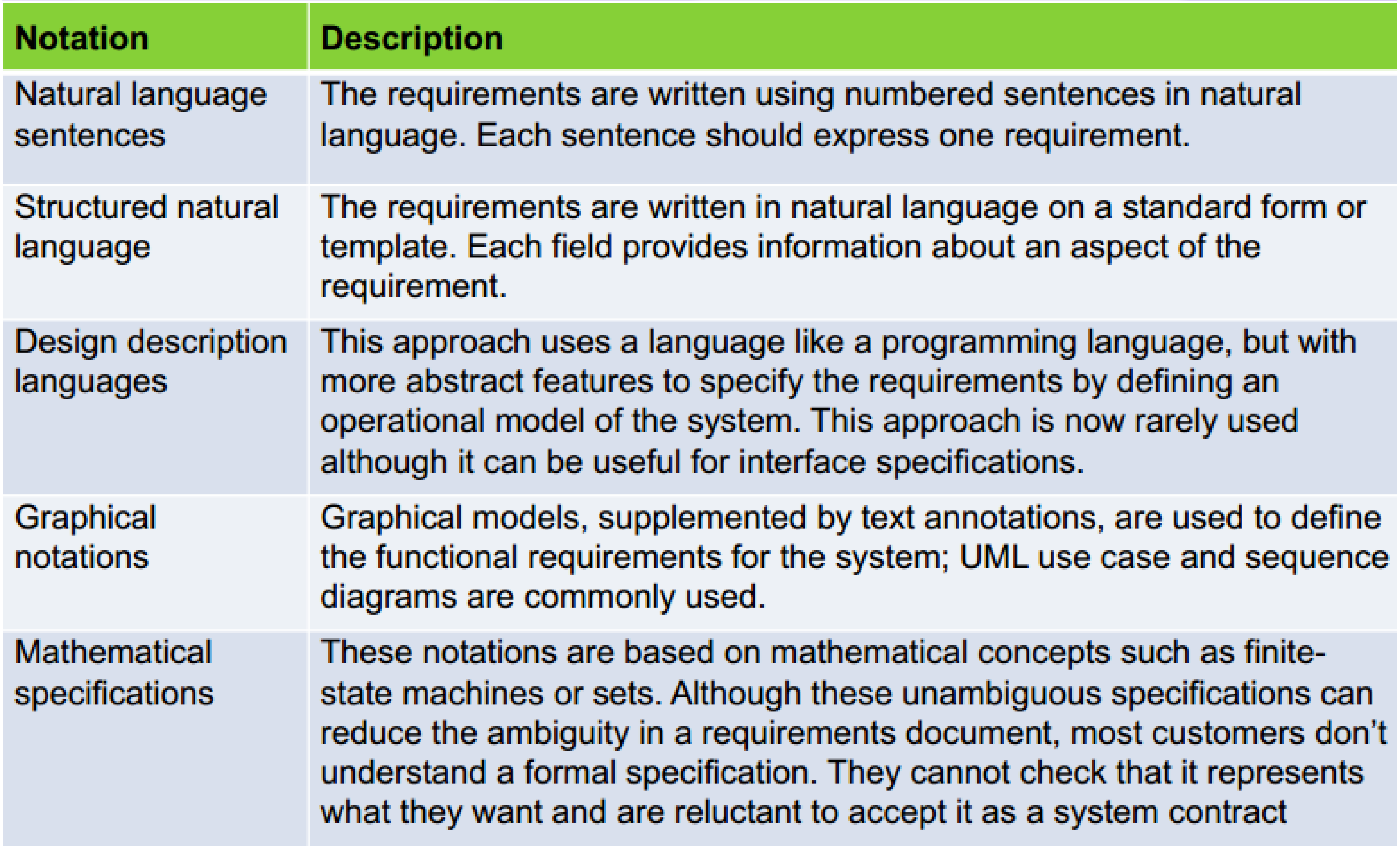
- ✧ A common problem with non-functional requirements is that users or customers often propose these requirements as general goals, such as ease of use, the ability of the system to recover from failure, or rapid user response.
- ✧ Whenever possible, you should write non-functional requirements quantitatively so that they can be objectively tested
- ✧ Goals are helpful to developers as they convey[truyền-đạt] the intentions[ý-đồ] of the system users.

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

Topics covered

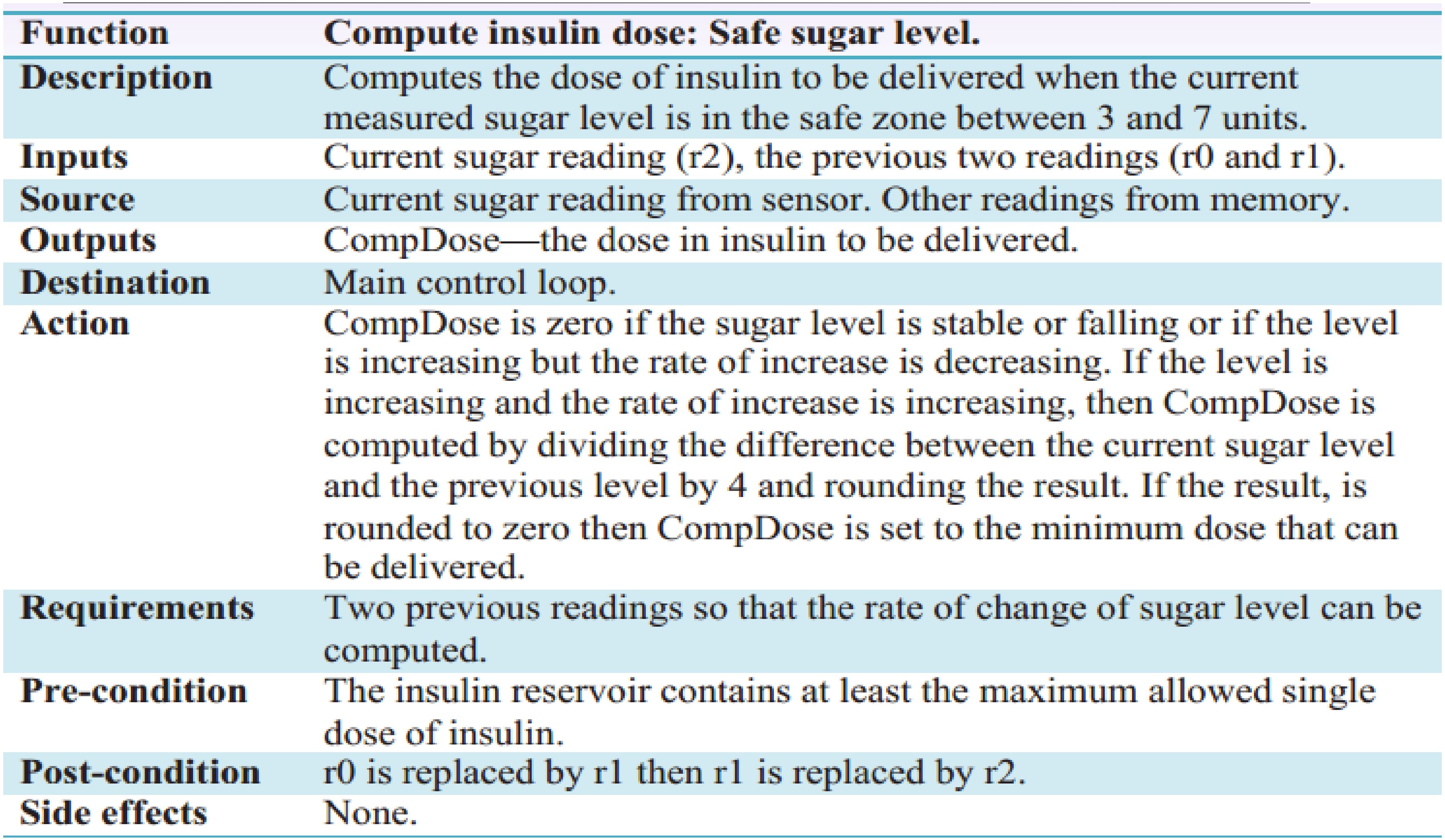
- ✧ Define Requirement
- ✧ Functional and non-functional requirements
- ✧ Requirement specification
- ✧ Requirements engineering processes:
  - ▪ Requirements elicitation and analysis
  - ▪ Requirements validation
  - ▪ Requirements management
  - ▪ Develop SRS (The software requirements document)

How to write



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

Example



Function	Compute insulin dose: Safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered.
Destination	Main control loop.
Action	CompDose is zero if the sugar level is stable or falling or if the level is increasing but the rate of increase is decreasing. If the level is increasing and the rate of increase is increasing, then CompDose is computed by dividing the difference between the current sugar level and the previous level by 4 and rounding the result. If the result, is rounded to zero then CompDose is set to the minimum dose that can be delivered.
Requirements	Two previous readings so that the rate of change of sugar level can be computed.
Pre-condition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Post-condition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

Discussion (10’)

Topics covered

- ✧ Define Requirement
- ✧ Functional and non-functional requirements
- ✧ Requirement specification
- ✧ Requirements engineering processes:
  - ▪ Requirements elicitation and analysis
  - ▪ Develop SRS (The software requirements document)
  - ▪ Requirements validation
  - ▪ Requirements management

Requirements elicitation and analysis

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

Problems of requirements analysis

- ✧ 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[xuất-hiện] and the business environment may change.

Requirements elicitation and analysis

- ✧ Software engineers work with a range of system stakeholders to find out about the application domain, the services that the system should provide, the required system performance, hardware constraints, other systems, etc.
- ✧ Stages include:
  - ▪ Requirements discovery,
  - ▪ Requirements classification and organization,
  - ▪ Requirements prioritization and negotiation(damphan),
  - ▪ Requirements specification.

Topics covered

- ✧ Define Requirement
- ✧ Functional and non-functional requirements
- ✧ Requirement specification
- ✧ Requirements engineering processes:
  - ▪ Requirements elicitation and analysis
  - ▪ Develop SRS (The software requirements document)
  - ▪ Requirements validation
  - ▪ Requirements management

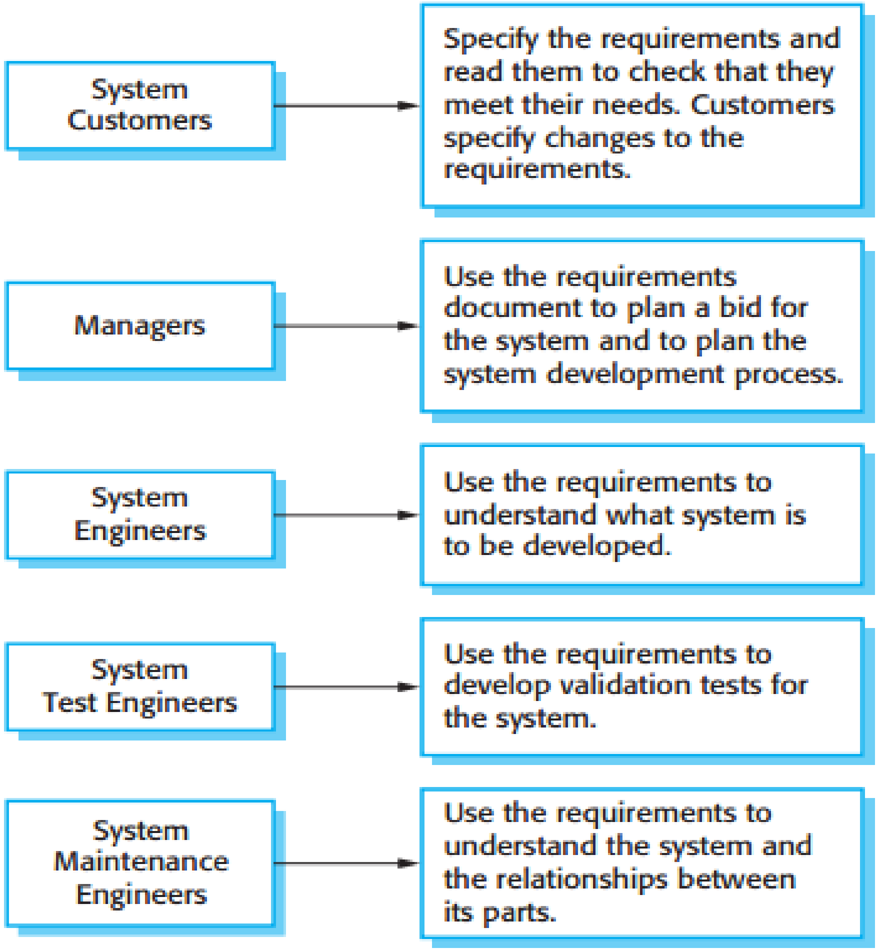
The software requirements document - SRS

- ✧ The software requirements document is the official statement of what is required of the system developers should implement.
- ✧ 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.

Agile methods and requirements (Self Study)

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

Users of a requirements document


System Customers → Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.

Managers → Use the requirements document to plan a bid for the system and to plan the system development process.

System Engineers → Use the requirements to understand what system is to be developed.

System Test Engineers → Use the requirements to develop validation tests for the system.

System Maintenance Engineers → Use the requirements to understand the system and the relationships between its parts.

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.

The structure of a requirements document

Chapter	Description
Preface	This should define the expected readership[đọc-giả] of the document and describe its version history, including a rationale[lý-do] for the creation of a new version and a summary of the changes made in each version.
Introduction	This should describe the need for the system. It should briefly describe the system’s functions and explain how it will work with other systems. It should also describe how the system fits into the overall business or strategic objectives of the organization commissioning the software.
Glossary	This should define the technical terms used in the document. You should not make assumptions about the experience or expertise of the reader.
User requirements definition	Here, you describe the services provided for the user. The nonfunctional system requirements should also be described in this section. This description may use natural language, diagrams, or other notations that are understandable to customers. Product and process standards that must be followed should be specified.
System architecture	This chapter should present a high-level overview of the anticipated system architecture, showing the distribution of functions across system modules. Architectural components that are reused should be highlighted.

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.
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.

Requirement Process
Develop SRS – Techniques

- ✧ Specify requirements using **structured natural language** (forms, tables, etc.)
- ✧ **Functional requirements** can be specified using modeling - a combination of graphical notations and structured natural language
  - **Use cases, Use case diagrams, Use case specification**
  - **Activity Diagram, State Diagram**
  - **DFD, Concept ERD**
  - **Prototype: Screen Flow, Screen spec specification**
  - ...
- ✧ **Non-functional requirements** can't be modeled => specified using structured natural language only

Requirement Process
Develop SRS – Characteristics of good SRS 1/2

- ✧ Correct: requirement ~ what the software shall meet.
- ✧ Unambiguous:
  - Has only one interpretation (to both creator & user)
  - Use natural language & avoid the words like: maybe, generally, etc.
- ✧ Complete
  - Include all significant requirements.
  - Define all the software responses & include all the refs/labels.
  - Use of TBD: should avoid OR mention why, what to do, who, when.

Requirement Process
Develop SRS - Characteristics of good SRS 2/2

✧ Consistent: no conflict between individual requirements.

✧ Verifiable: reviewable & testable in finite cost-effective process.

✧ Traceable: clear origin & good reference for future develop/enhance documents.

Requirement Process
Develop SRS – SRS Review Checklist

✧ SRS Review Checklist

- To review the requirements by yourself
- Make sure you understood completely the requirements:
  - Organization and Completeness: adequate, no missing, etc.
  - Correctness: no conflict, verifiable, in scope, message, etc.
  - Non-functional requirements, quality attributes, etc.

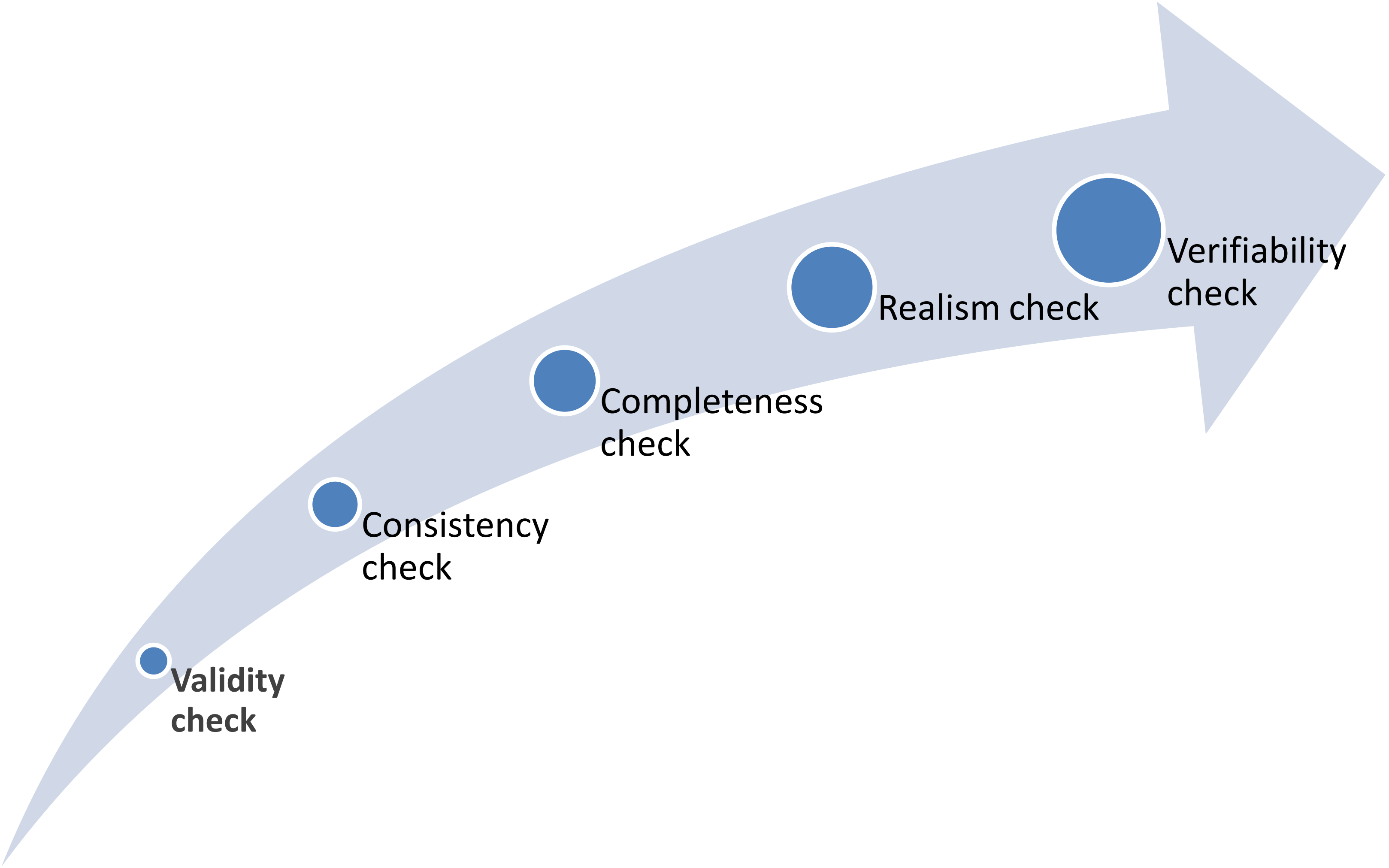
Topics covered

- ✧ Define Requirement
- ✧ Functional and non-functional requirements
- ✧ Requirement specification
- ✧ Requirements engineering processes:
  - ▪ Requirements elicitation and analysis
  - ▪ Develop SRS (The software requirements document)
  - ▪ Requirements validation
  - ▪ Requirements management

Requirement Process
Validate Requirements – Purpose

- ✧ Make sure 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

Requirement Process
Validate Requirements – Process



Validity check

Consistency check

Completeness check

Realism check

Verifiability check

Topics covered

- ✧ Define Requirement
- ✧ Functional and non-functional requirements
- ✧ Requirement specification
- ✧ Requirements engineering processes:
  - ▪ Requirements elicitation and analysis
  - ▪ Develop SRS (The software requirements document)
  - ▪ Requirements validation
  - ▪ Requirements management **(Self Study)**

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.

Changing requirements (Self Study)

- ✧ 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.

Requirement Process
Requirements management

✧ Manage requirement

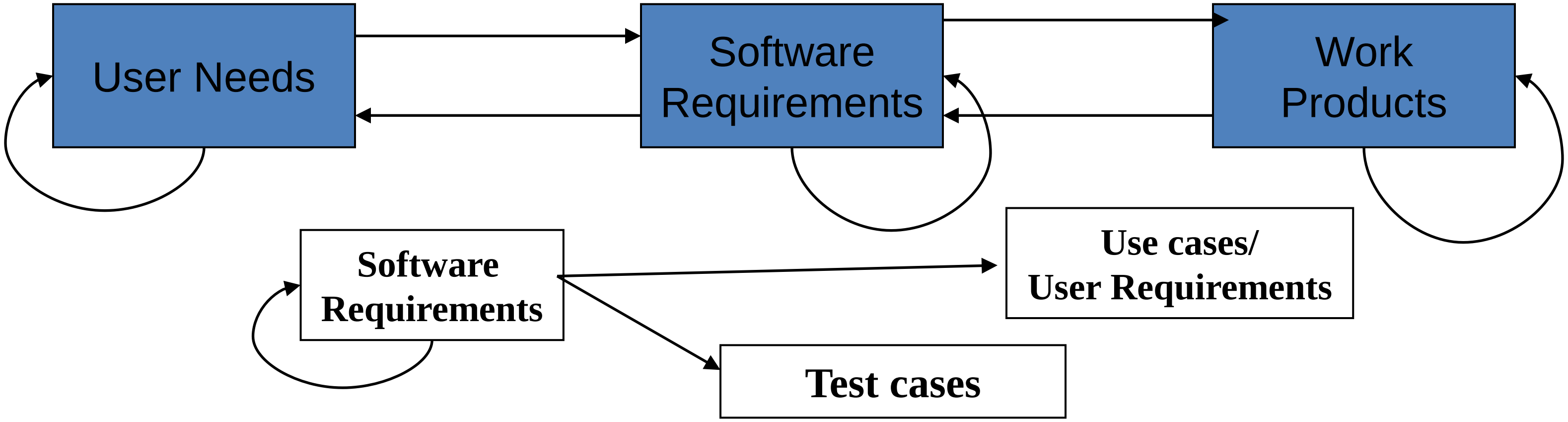
- Requirement Management Sheet, Excel sheet, used to track the status, relationship and change of requirements during the whole project.
- A mandatory document (dynamic version of SRS)
  - Classify requirement to functional/non-functional requirement
  - To maintain the common reference for all related parties (traceability of requirement and software product)
  - To track the project progress (status of requirement)
  - To track the change (including change request)
  - To collect requirement related metrics for reporting
- The sheet is created the first time client requirement come

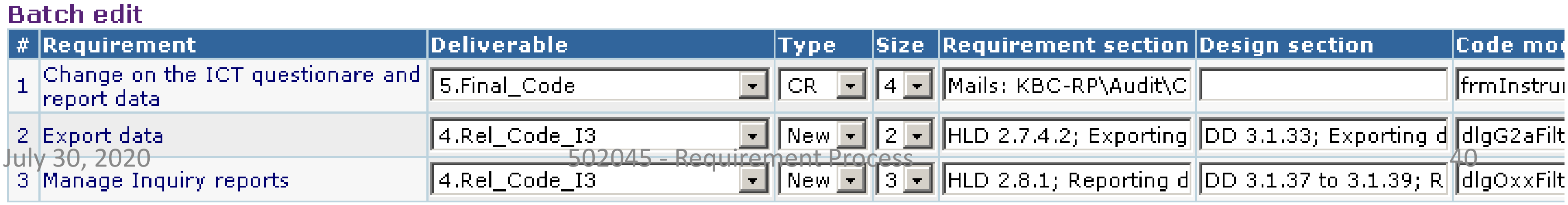
Refer: Template_Requirement Management Sheet.xls

Requirement Process
Manage Traceability & Requirement Status

✧ Why is traceability necessary?

- The requirements can change at any stage during the product's life.
- If the requirements are traceable, then when changes happen, it is far easier to find the impacted parts of the product





Batch edit

#	Requirement	Deliverable	Type	Size	Requirement section	Design section	Code mo
1	Change on the ICT questionare and report data	5.Final_Code	CR	4	Mails: KBC-RP\Audit\C		frmInstru
2	Export data	4.Rel_Code_I3	New	2	HLD 2.7.4.2; Exporting	DD 3.1.33; Exporting d	dlgG2aFilt
3	Manage Inquiry reports	4.Rel_Code_I3	New	3	HLD 2.8.1; Reporting d	DD 3.1.37 to 3.1.39; R	dlgOxxFilt

Requirements management planning

- ✧ Establishes the level of requirements management detail that is required.
- ✧ Requirements management decisions:
  - ▪ *Requirements identification*Each requirement must be uniquely identified so that it can be cross-referenced with other requirements.
  - ▪ *A change management process*This is the set of activities that assess the impact and cost of changes. I discuss this process in more detail in the following section.
  - ▪ *Traceability policies*These policies define the relationships between each requirement and between the requirements and the system design that should be recorded.
  - ▪ *Tool support*Tools that may be used range from specialist requirements management systems to spreadsheets and simple database systems.

Requirements management planning (Self Study)

✧ You need tool support for

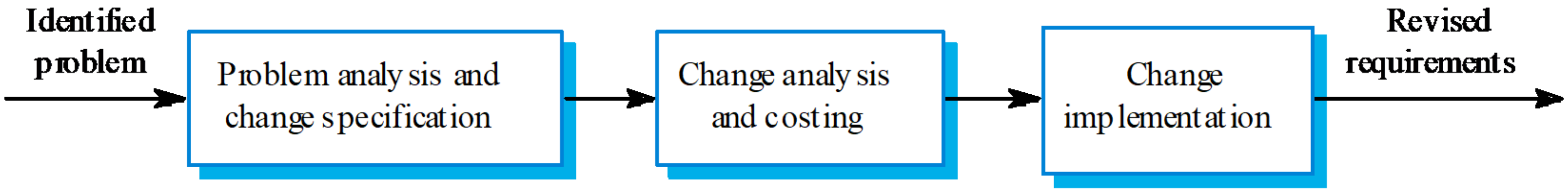
- **Requirements storage** The requirements should be maintained in a secure, managed data store that is accessible to everyone involved in the requirements engineering process
- **Change management** The process of change management is simplified if active tool support is available
- **Traceability management** As discussed above, tool support for traceability allows related requirements to be discovered. Some tools are available which use natural language processing techniques to help discover possible relationships between requirements.

Requirement Process
Manage Requirements Changes & Status (Self Study)

✧ Requirements change (CR – Change request)

- The priority of requirements from different viewpoints changes during the development process
- Customers may specify requirements from a business perspective that conflict with end-user requirements
- The business and technical environment of the system changes during its development

✧ Requirements change process


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

Requirements change management (Self study)

✧ 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.