Software Engineering

UNIT –II

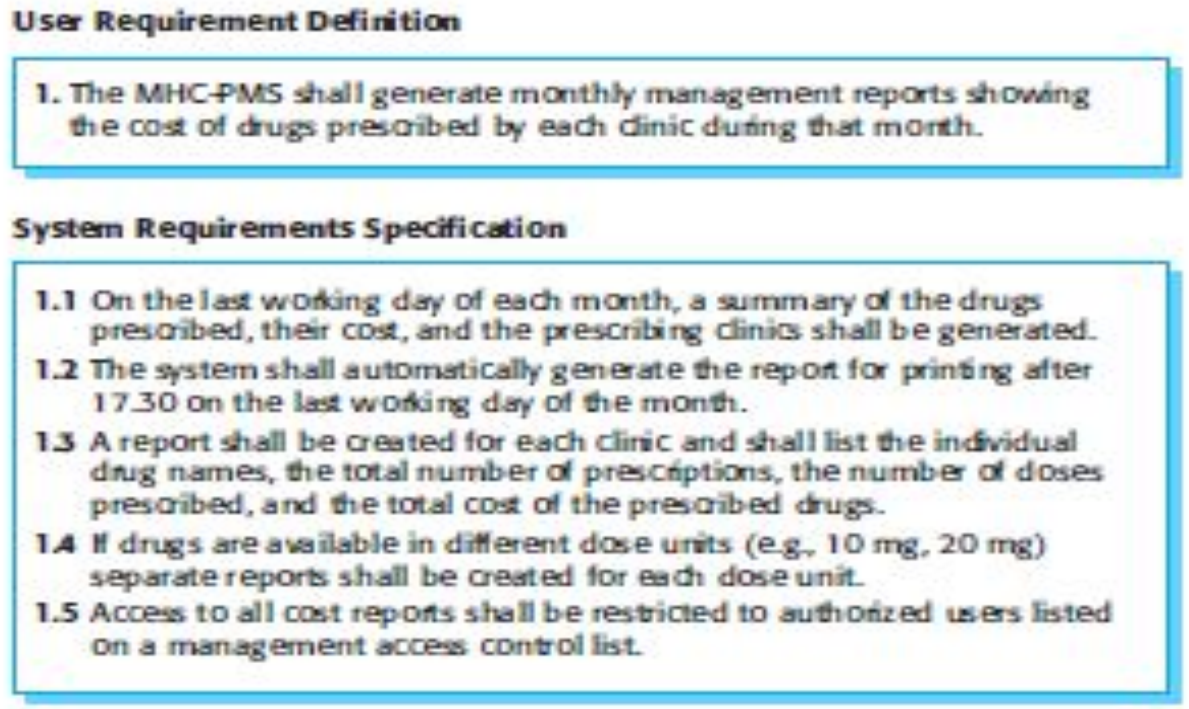
BY Tulasi Vemu
(Assistant Professor)

Software Requirements

- The term 'requirement' is not used consistently in the software industry. In some cases, a **requirement is simply a high-level, abstract statement of a service that a system should provide** or a constraint on a system. At the other extreme, it is a detailed, formal definition of a system function.

- User requirements are statements, in a natural language plus diagrams, of what services the system is expected to provide to system users and the constraints under which it must operate
- System requirements are more detailed

Mental Health Care Patient Management System (MHC-PMS)



User Requirement Definition

1. The MHC-PMS 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.

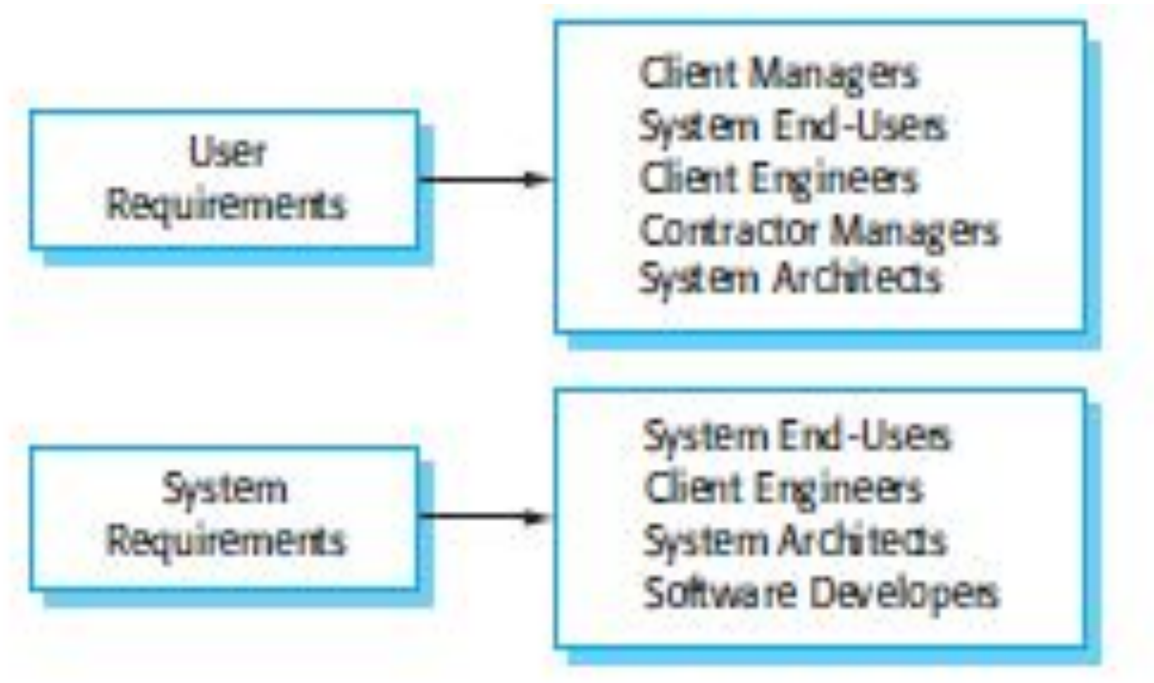
1.2 The system shall automatically generate the report for printing after 17.30 on the last working day of the month.

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., 10 mg, 20 mg) separate reports shall be created for each dose unit.

1.5 Access to all cost reports shall be restricted to authorized users listed on a management access control list.

Readers of different types of requirements specification



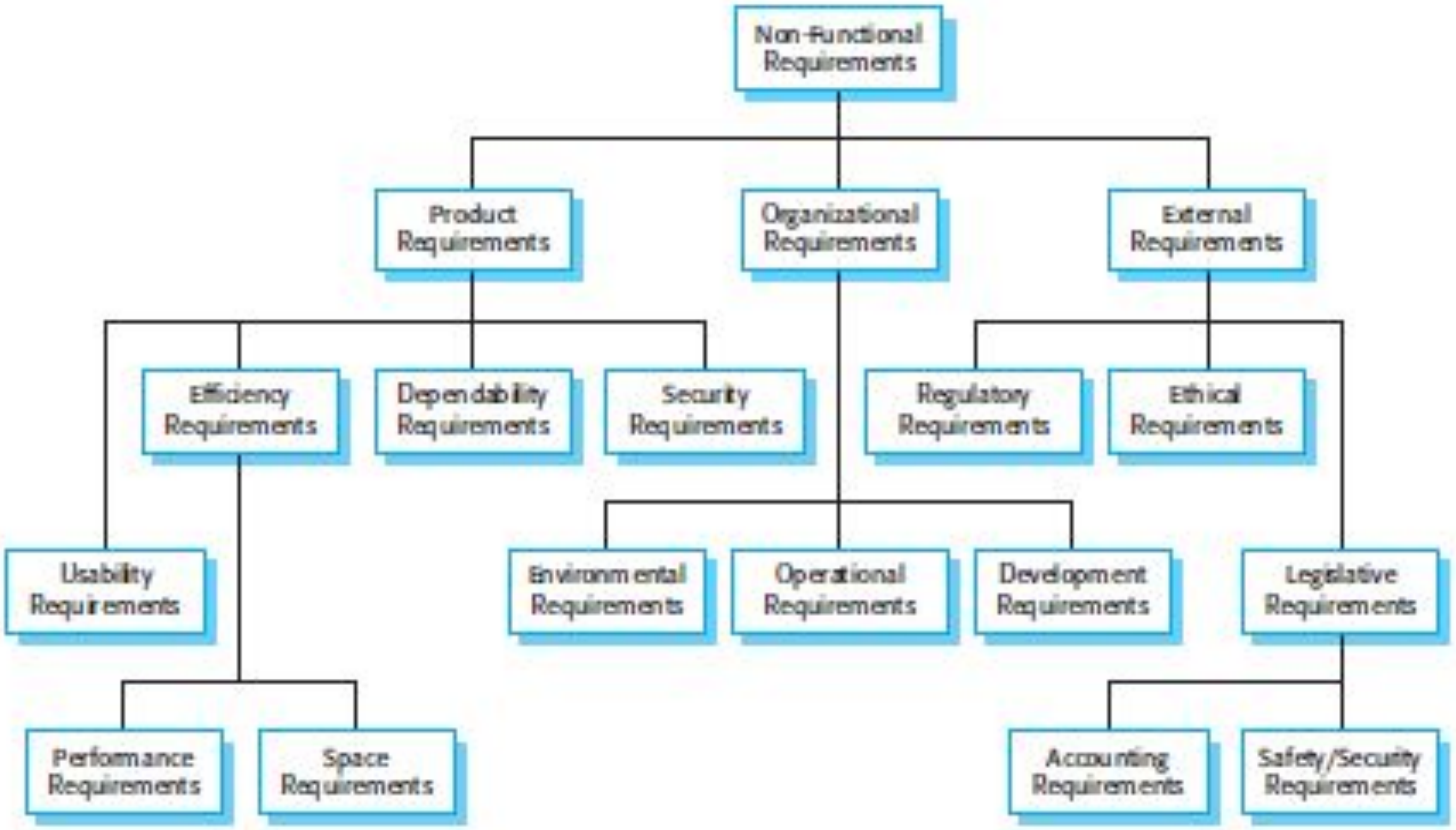
Functional & Non-Functional Requirements

- *Functional requirements These are statements of services the system should* provide, how the system should react to particular inputs, and how the systems should behave in particular situations. In some cases, the functional requirements may also explicitly state what the system should not do.
- *Non-functional requirements These are constraints on the services or functions* offered by the system. They include timing constraints, constraints on the development process, and constraints

Non-functional requirements

- 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 new system services that are required. In addition, it may also generate requirements that restrict existing

Non-functional requirements



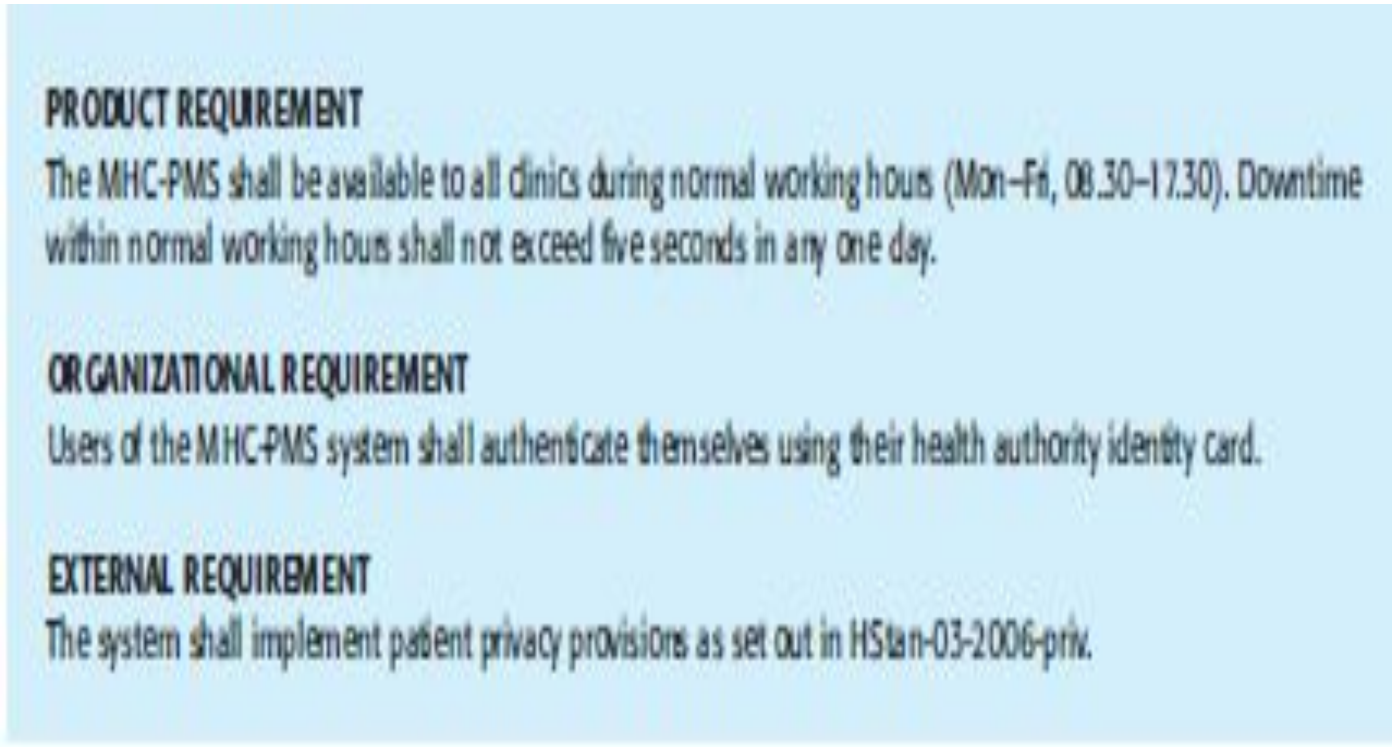
Non-functional requirements

- *Product requirements These requirements specify or constrain the behavior of the* software. Examples include performance requirements on how fast the system must execute and how much memory it requires, reliability requirements that set out the acceptable failure rate, security requirements, and usability requirements.
- *Organizational requirements These requirements are broad system requirements* derived from policies and procedures in the customer's and developer's organization. Examples include operational process requirements that define how the system will be used, development process requirements that specify the

Non-functional requirements

- *External requirements This broad heading covers all requirements that are* derived from factors external to the system and its development process. These may include regulatory requirements that set out what must be done for the system to be approved for use by a regulator, such as a central bank; legislative requirements that must be followed to ensure that the system operates within the law; and ethical requirements that ensure that the system will be acceptable to its users and the

(MHC-PMS) Non-Functional Requirements



PRODUCT REQUIREMENT

The MHC-PMS 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.

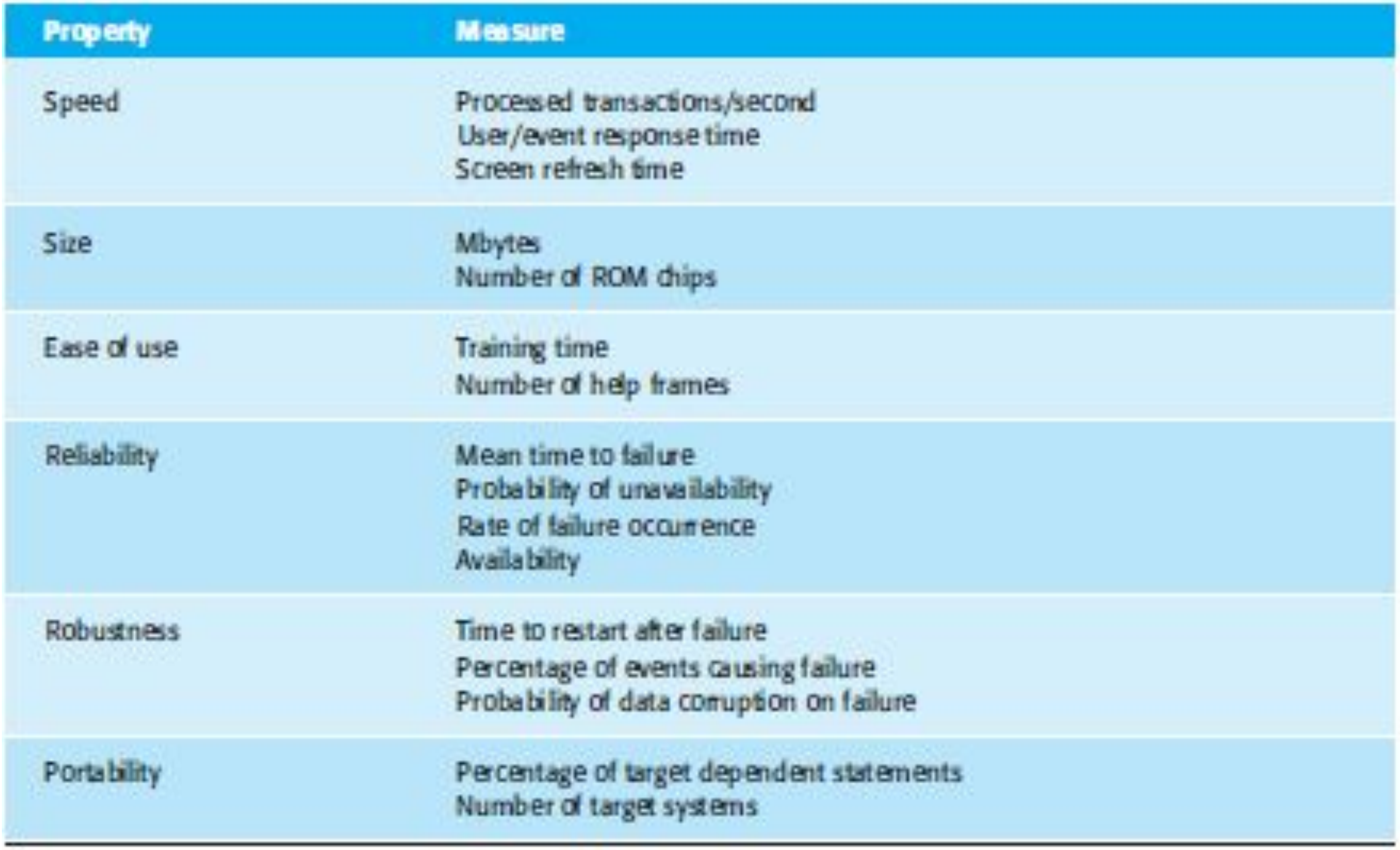
ORGANIZATIONAL REQUIREMENT

Users of the MHC-PMS 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.

Metrics For Specifying Non-functional Requirements



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

Non-functional Requirements

- Non-functional requirements often conflict and interact with other functional or non-functional requirements.

- For example, the authentication requirement obviously requires a card reader to be installed with each computer attached to the system.
- However, there may be another requirement that requests mobile access to the system from doctors' or nurses' laptops. These are not normally equipped with card readers so, in these circumstances, some alternative authentication method may have to be allowed

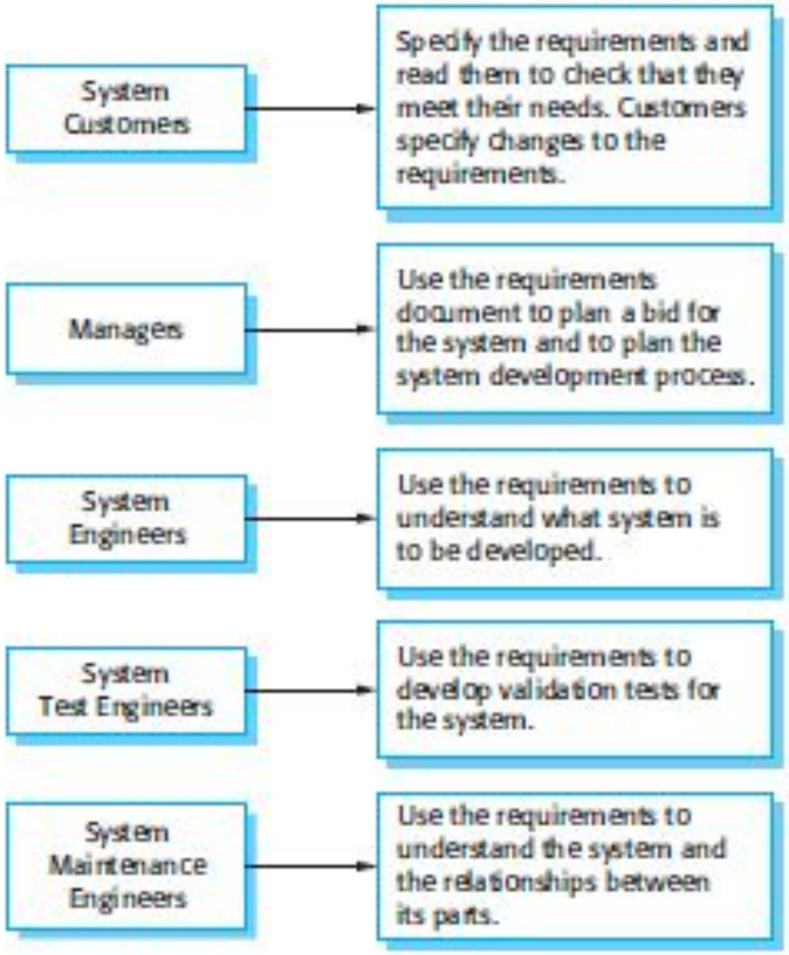
Non-functional Requirements

- It is difficult to separate functional and non-functional requirements in the requirements document.
- If the non-functional requirements are stated separately from the functional requirements, the relationships between them may be hard to understand.
- However, you should explicitly highlight requirements that are clearly related to emergent system properties, such as performance or reliability.
- You can do this by putting them in a separate section of the requirements document or by distinguishing them from other system requirements

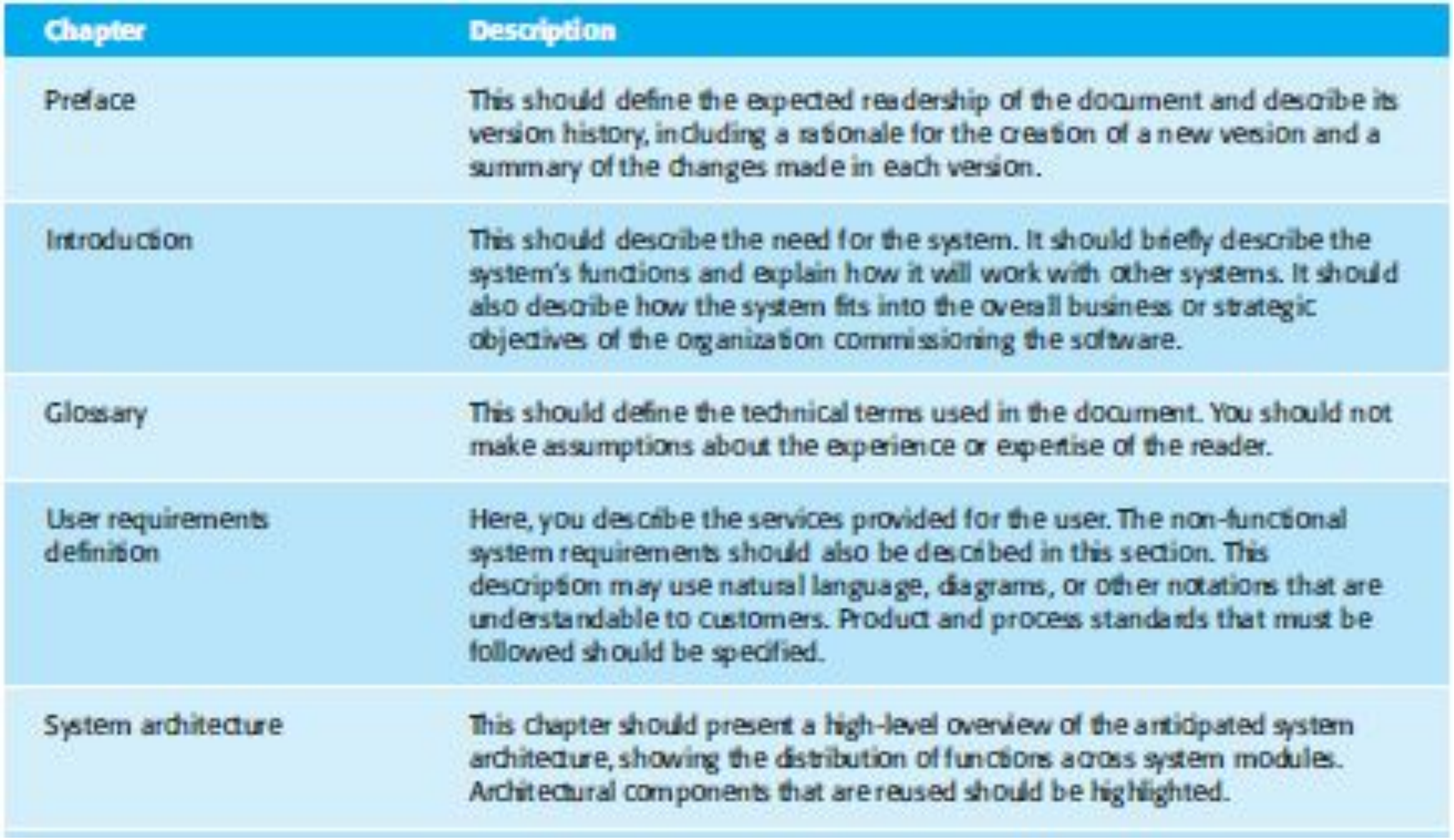
The software requirements document

- The software requirements document (sometimes called the software requirements specification or SRS) is an official statement of what the system developers should implement.
- It should include both the user requirements for a system and a detailed specification of the system requirements.
- Sometimes, the user and system requirements are integrated into a single description.
- In other cases, the user requirements are defined in an introduction to the system requirements specification. If there are a large number of requirements, the detailed system requirements may be presented in a separate document

Users of SRS Document

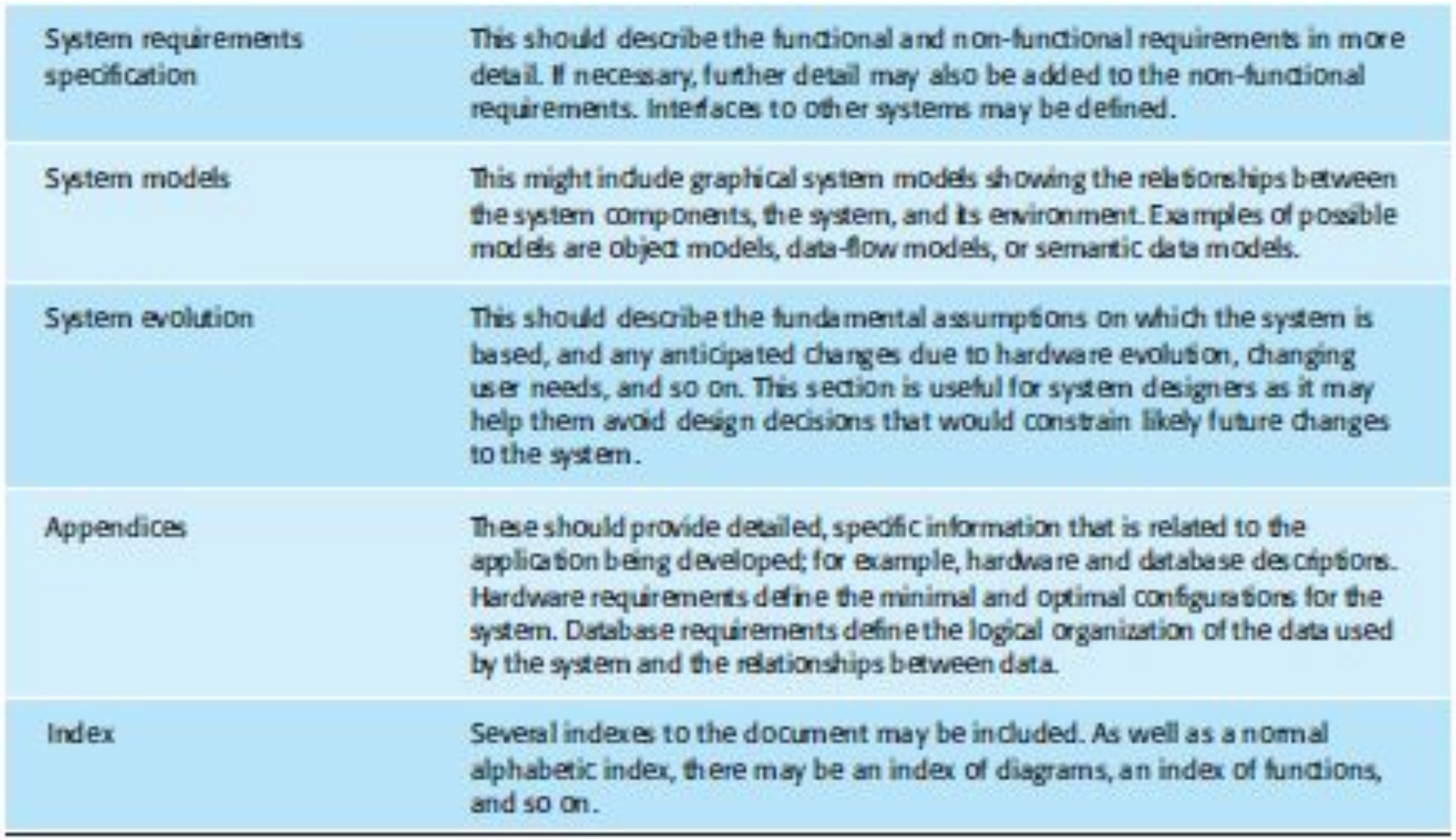

System Customers
Specify the requirements and read them to check that they meet their needs. Customers specify changes to the requirements.
Managers
Use the requirements document to plan a bid for the system and to plan the system development process.
System Engineers
Use the requirements to understand 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.

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.
User requirements definition	Here, you describe the services provided for the user. The non-functional 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



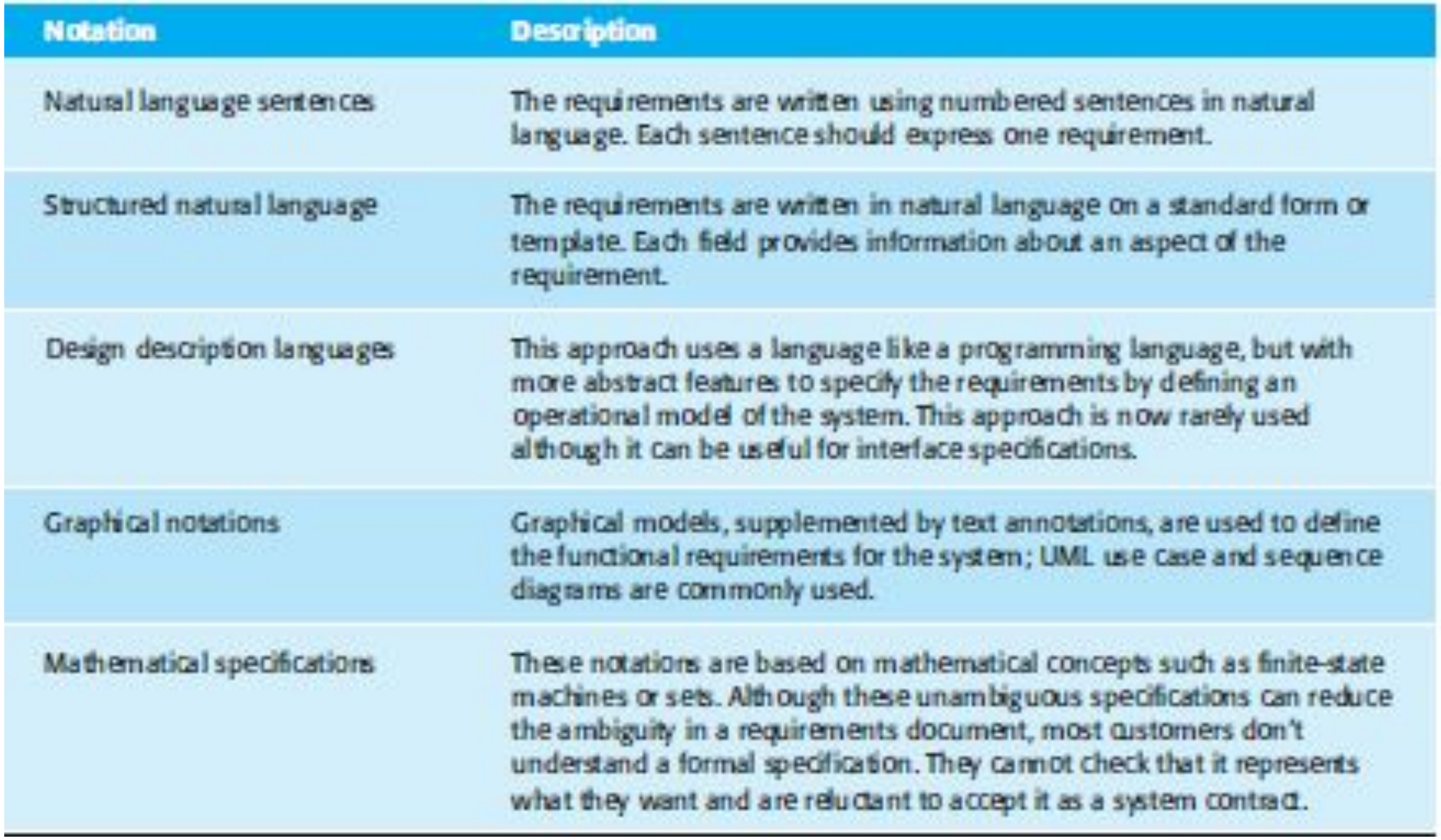
System requirements specification	This should describe the functional and non-functional requirements in more detail. If necessary, further detail may also be added to the non-functional requirements. Interfaces to other systems may be defined.
System models	This might include graphical system models showing the relationships between the system components, the system, and its environment. Examples of possible models are object models, data-flow models, or semantic data models.
System evolution	This should describe the fundamental assumptions on which the system is based, and any anticipated changes due to hardware evolution, changing user needs, and so on. This section is useful for system designers as it may help them avoid design decisions that would constrain likely future changes to the system.
Appendices	These should provide detailed, specific information that is related to the application being developed; for example, hardware and database descriptions. Hardware requirements define the minimal and optimal configurations for the system. Database requirements define the logical organization of the data used by the system and the relationships between data.
Index	Several indexes to the document may be included. As well as a normal alphabetic index, there may be an index of diagrams, an index of functions, and so on.

Requirements specification

- Requirements specification is the process of writing down the user and system requirements in a requirements document.

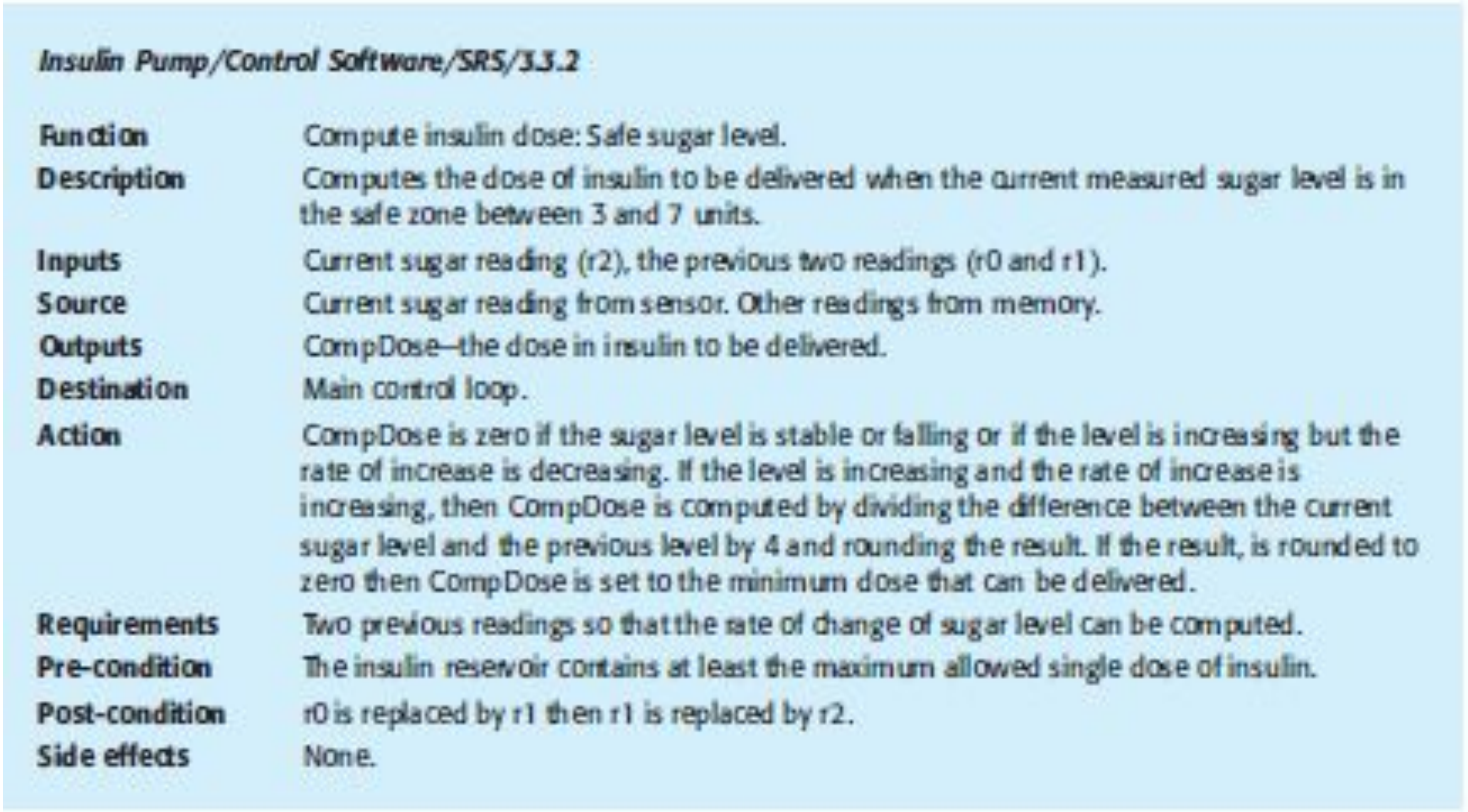
- Ideally, the user and system requirements should be clear, unambiguous, easy to understand, complete, and consistent.

Requirements specification



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.

A structured specification of a requirement for an insulin pump



Insulin Pump/Control Software/SRS/3.3.2

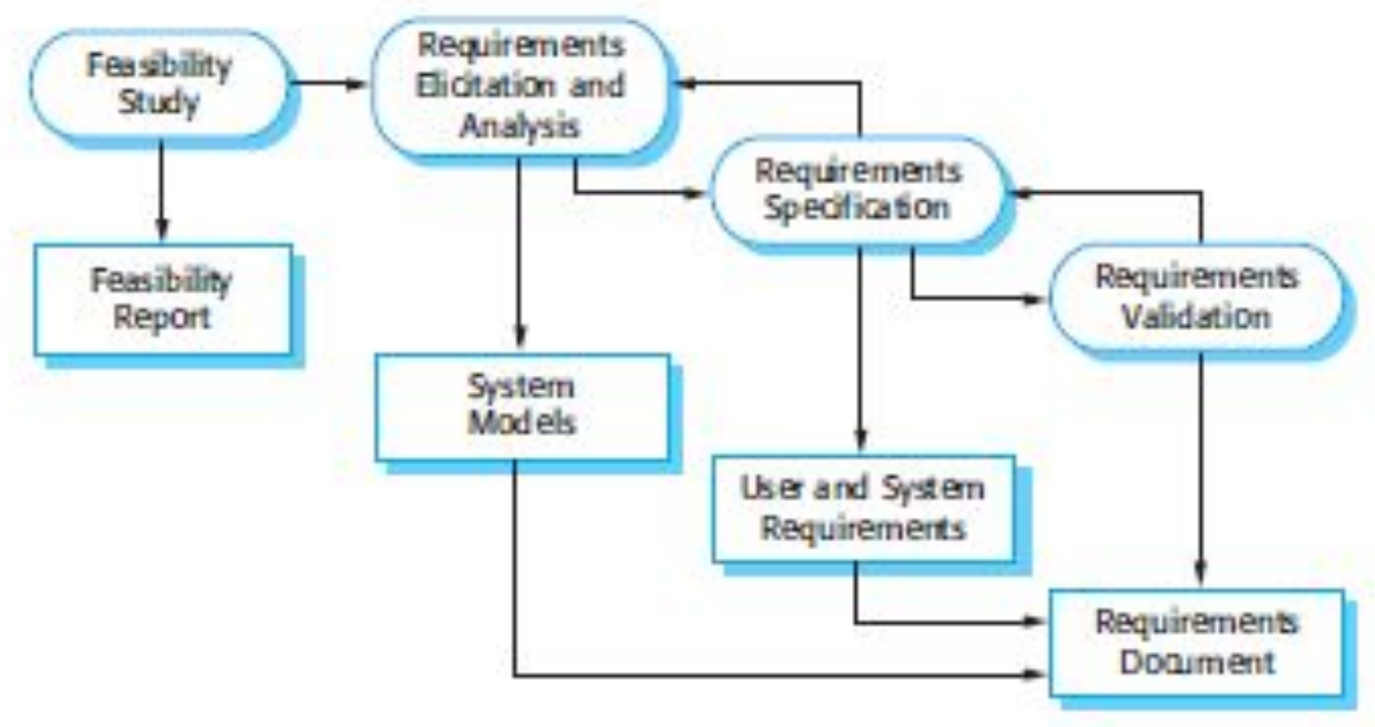
Function	Compute insulin dose: Safe sugar level.
Description	Computes the dose of insulin to be delivered when the current measured sugar level is in the safe zone between 3 and 7 units.
Inputs	Current sugar reading (r2), the previous two readings (r0 and r1).
Source	Current sugar reading from sensor. Other readings from memory.
Outputs	CompDose—the dose in insulin to be delivered.
Destination	Main control loop.
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.

The requirements engineering process

Feasibility Study:

- An estimate is made of whether the identified user needs may be satisfied using current software and hardware technologies.
- The study considers whether the proposed system will be cost-effective from a business point of view and if it can be developed within existing budgetary constraints.
- A feasibility study should be relatively cheap and quick. The result should inform the decision of whether or not to go ahead

The requirements engineering process



A spiral view of the requirements engineering process

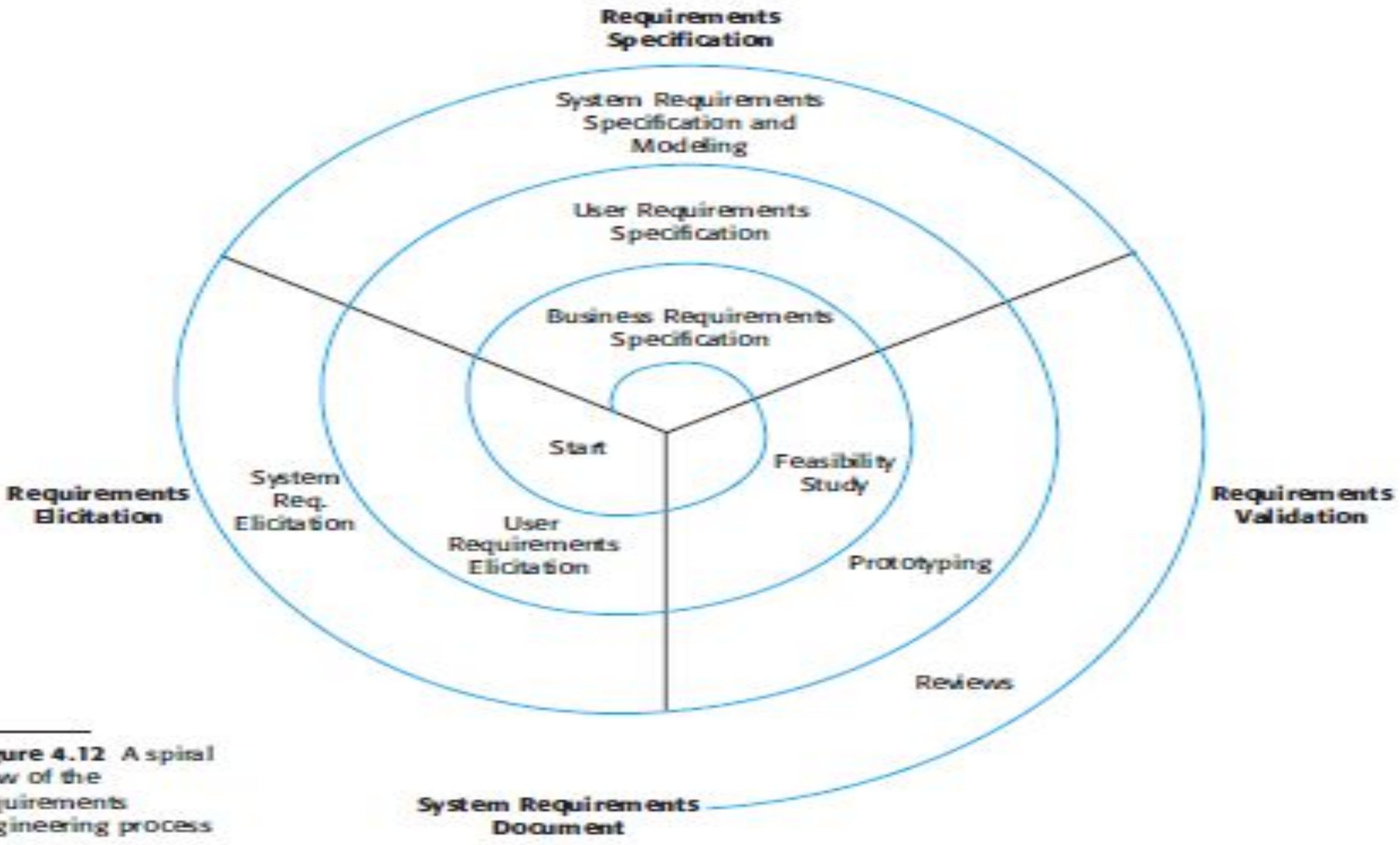
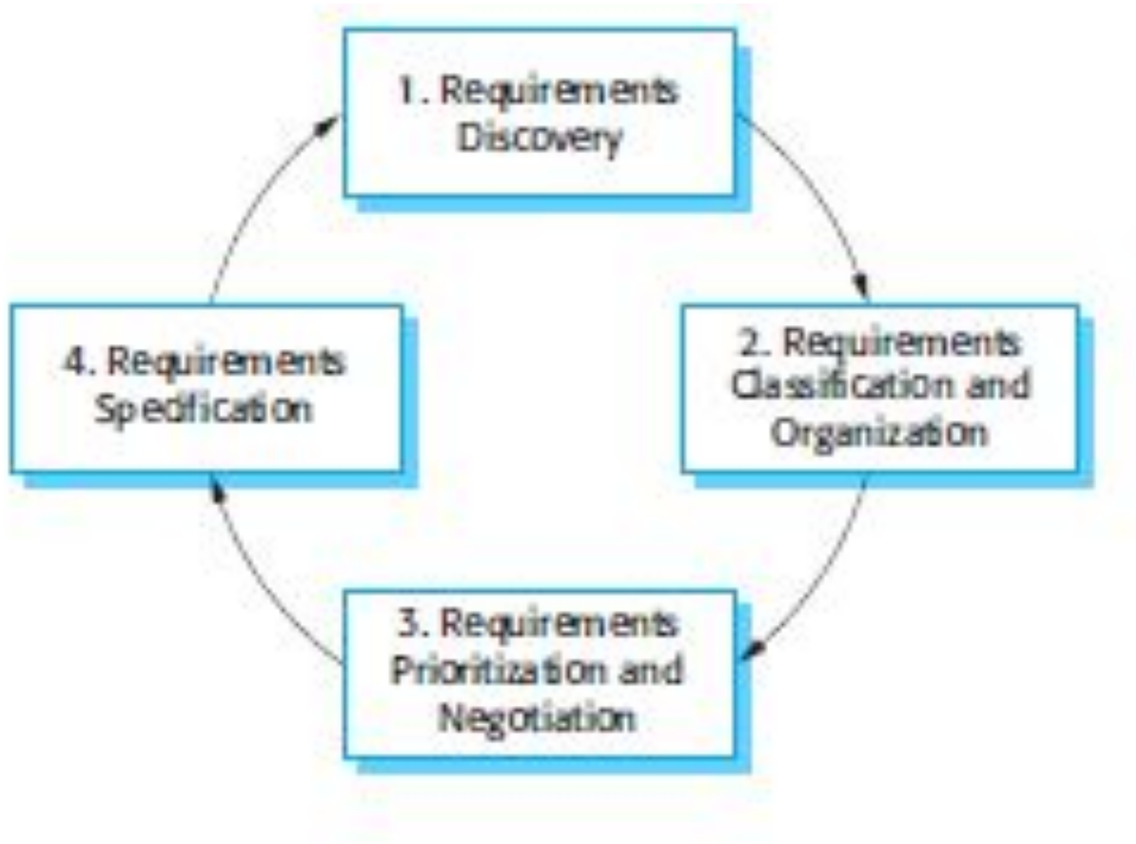

Requirements Specification
System Requirements Specification and Modeling
User Requirements Specification
Business Requirements Specification
Start
Feasibility Study
Requirements Elicitation
System Req. Elicitation
User Requirements Elicitation
Prototyping
Requirements Validation
Reviews
System Requirements Document

Figure 4.12 A spiral view of the requirements engineering process

Requirements elicitation and analysis

- This is the process of deriving the system requirements through observation of existing systems, discussions with potential users and procurers, task analysis, and so on.
- After an initial feasibility study, the next stage of the requirements engineering process is requirements elicitation and analysis.
- In this activity, software engineers work with customers and system end-users to find out about the application domain, what services the system should provide, the required performance of the system, hardware constraints, and so on.

The requirements elicitation and analysis process


1. Requirements Discovery
2. Requirements Classification and Organization
3. Requirements Prioritization and Negotiation
4. Requirements Specification

The requirements elicitation and analysis process

- *Requirements discovery:*
  - This is the process of interacting with stakeholders of the system to discover their requirements.
- *Requirements classification and organization:*
  - This activity takes the unstructured collection of requirements, groups related requirements, and organizes them into coherent clusters.
- *Requirements prioritization and negotiation:*
  - This activity is concerned with prioritizing requirements and finding and resolving requirements conflicts through negotiation.
- *Requirements specification:*
  - The requirements are documented and input into the next round of the spiral.

The requirements elicitation and analysis process

- Requirements discovery
  - *Interviewing*
  - *Scenarios*
  - *Use cases*
  - *Ethnography*

Requirements discovery

- Requirements discovery (sometime called requirements elicitation) is the process of gathering information about the required system and existing systems, and getting the user and system requirements from this information.
- Sources of information during the requirements discovery phase include documentation, system stakeholders, and specifications of similar systems.
- You interact with stakeholders through interviews and observation and you may use scenarios and prototypes to help stakeholders understand what the system will be like.

Stakeholders of MHC-PMS System

- Patients whose information is recorded in the system.
- Doctors who are responsible for assessing and treating patients.
- Nurses who coordinate the consultations with doctors and administer some treatments.
- Medical receptionists who manage patients' appointments
- IT staff who are responsible for installing and maintaining the system.
- A medical ethics manager who must ensure that the system meets current ethical guidelines for patient care.
- Healthcare managers who obtain management information from the system.
- Medical records staff who are responsible for

Interviewing

- Closed Interviews:
  Closed interviews, where the stakeholder answers a pre-defined set of questions.

- Open Interviews
  Open interviews, in which there is no pre-defined agenda. The requirements engineering team explores a range of issues with system stakeholders and hence develop a better understanding of their needs.

Date: 28-08-2020

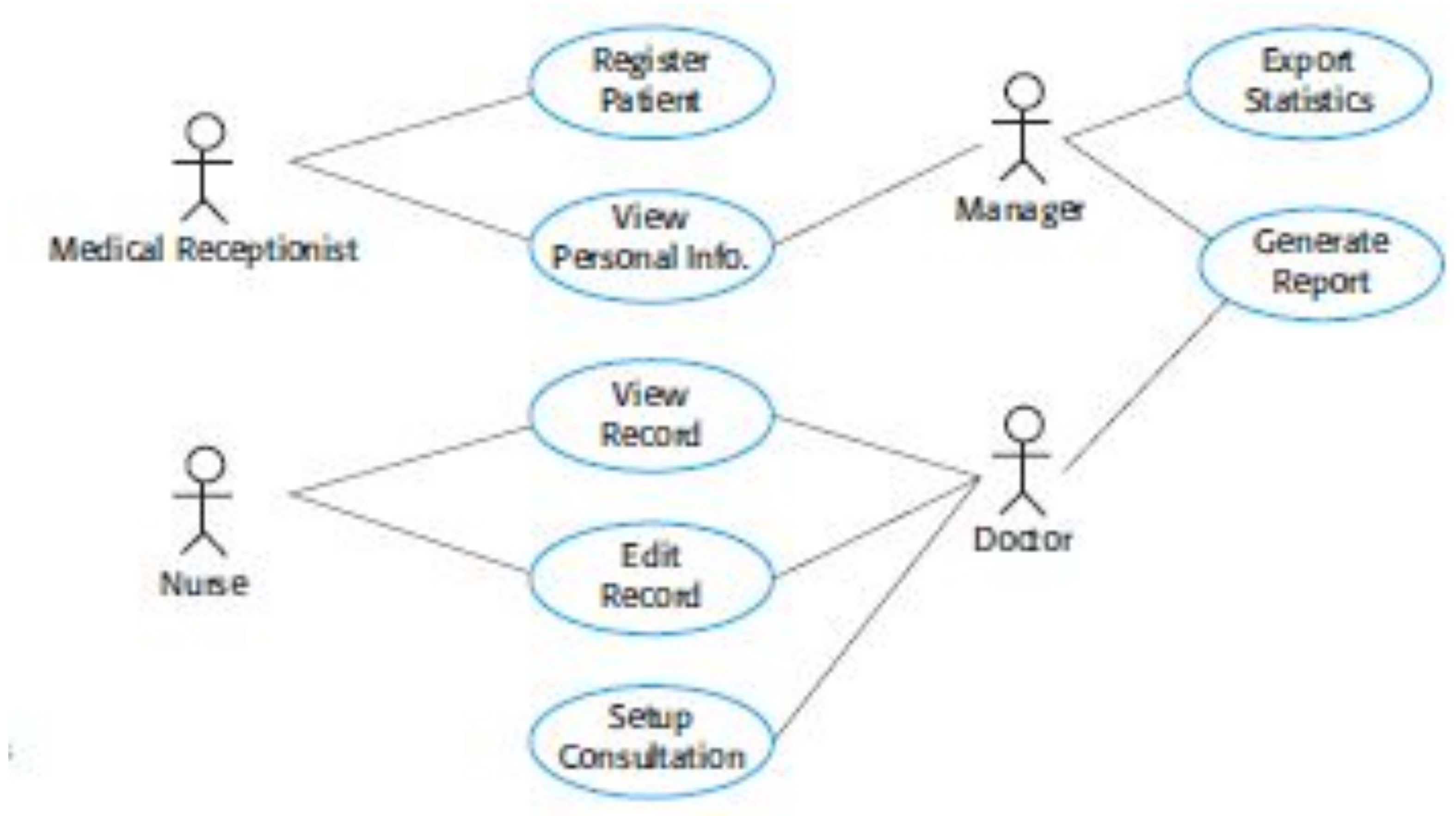
Scenarios

- 1. A description of what the system and users expects when the scenario starts.
- 2. A description of the normal flow of events in the scenario.
- 3. A description of what can go wrong and how this is handled.
- 4. Information about other activities that might be going on at the same time.
- 5. A description of the system state when the scenario finishes.

Use cases

- In their simplest form, a use case identifies the actors involved in an interaction and names the type of interaction.
- This is then supplemented by additional information describing the interaction with the system.
- The additional information may be a textual description or one or more graphical models such as UML sequence or state charts.
- Use cases are documented using a high-level use case diagram.
- The set of use cases represents all of the possible interactions that will be described in the system requirements.
- Actors in the process, who may be human or other systems, are represented as stick figures.
- Each class of interaction is represented as a named ellipse.
- Lines link the actors with the interaction.

Use cases for the MHC-PMS

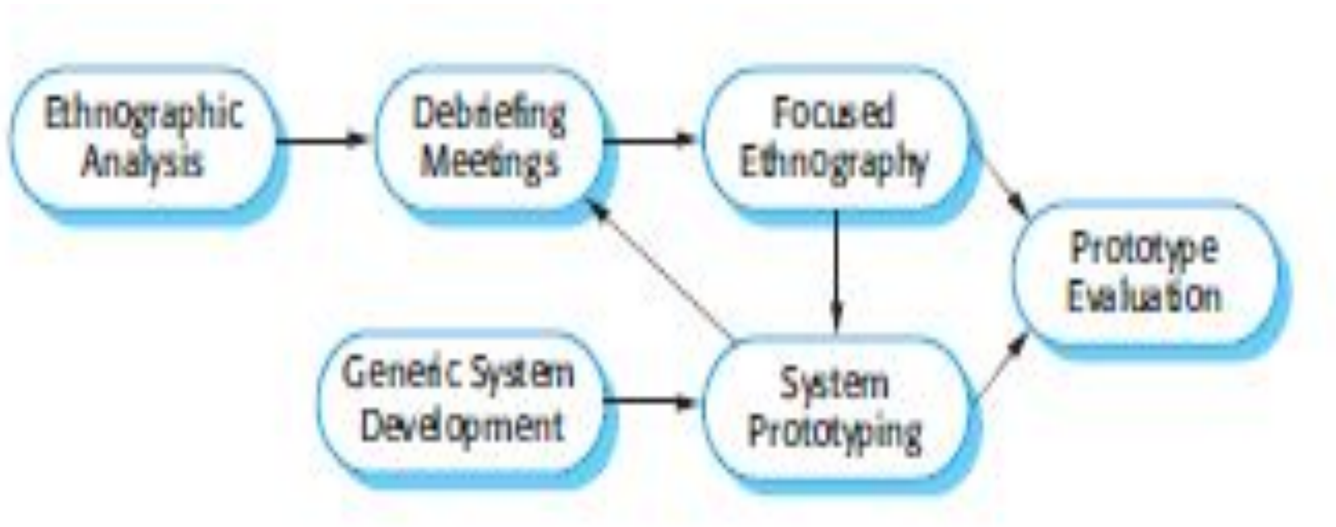


Ethnography

- Ethnography is an observational technique that can be used to understand operational processes and help derive support requirements for these processes.
- The value of ethnography is that it helps discover implicit system requirements that reflect the actual ways that people work, rather than the formal processes defined by the organization.
- Ethnography is particularly effective for discovering two types of requirements:

- 1. Requirements that are derived from the way in which people actually work, rather than the way in which process definitions say they ought to work.
- 2. Requirements that are derived from cooperation and awareness of other people's activities.

Ethnography



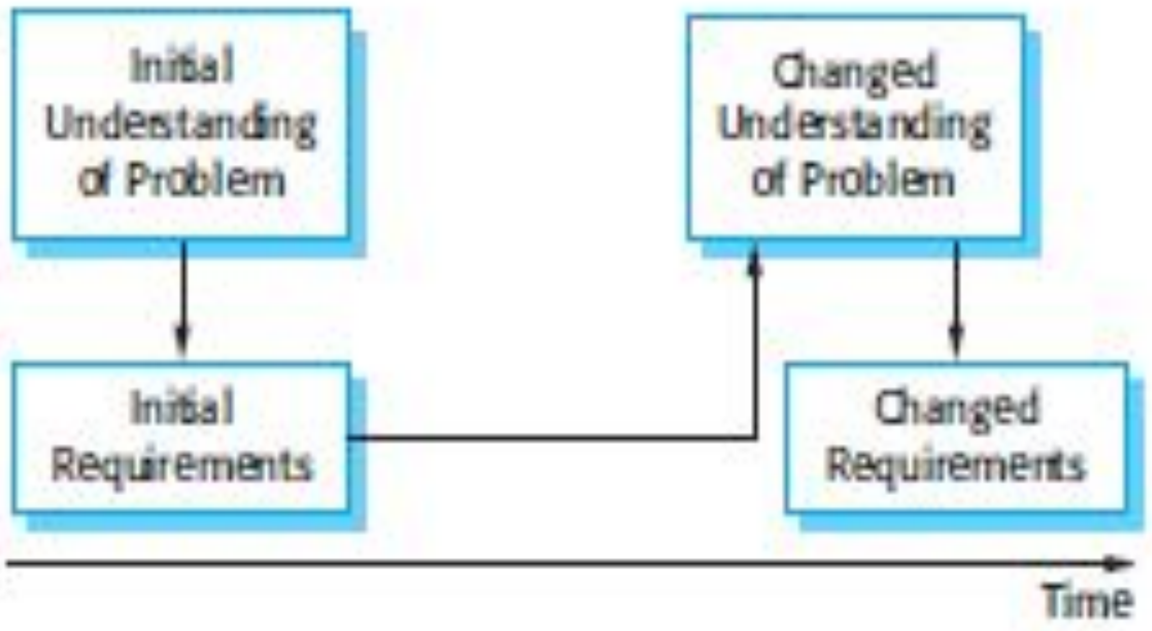
Requirements validation

During the requirements validation process, different types of checks should be carried out on the requirements in the requirements document. These checks include:

1. *Validity checks*
2. *Consistency checks*
3. *Completeness checks*
4. *Realism checks*
5. *Verifiability*

Date: 30-09-2020

Requirements evolution

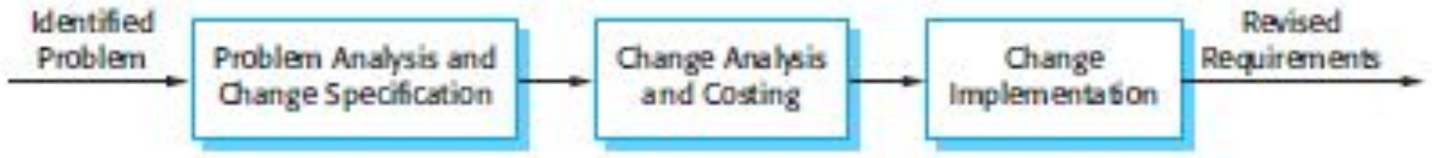

Initial Understanding of Problem
Initial Requirements
Changed Understanding of Problem
Changed Requirements
Time

Reasons for requirements change

- The business and technical environment of the system always changes after installation.
- 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.
- Large systems usually have a diverse user community, with many users having different requirements and priorities that may be conflicting or contradictory.

Requirements management planning

- *Requirements identification*
- *A change management process*
- *Traceability policies*
- *Tool support*
  - *Requirements storage*
  - *Change management*
  - *Traceability management*


Identified Problem
Problem Analysis and Change Specification
Change Analysis and Costing
Change Implementation
Revised Requirements

Requirements Change Management

- Problem Analysis and Change Specification
- Change Analysis and Costing
- Change Implementation