Requirements Engineering

Chapter 4

Introduction

- The requirements for a system are the **descriptions** of what the system should do, the **services** that it provides and the **constraints** on its operation.

- These requirements reflect the needs of customers for a system that serves a certain purpose such as controlling a device, placing an order, or finding information

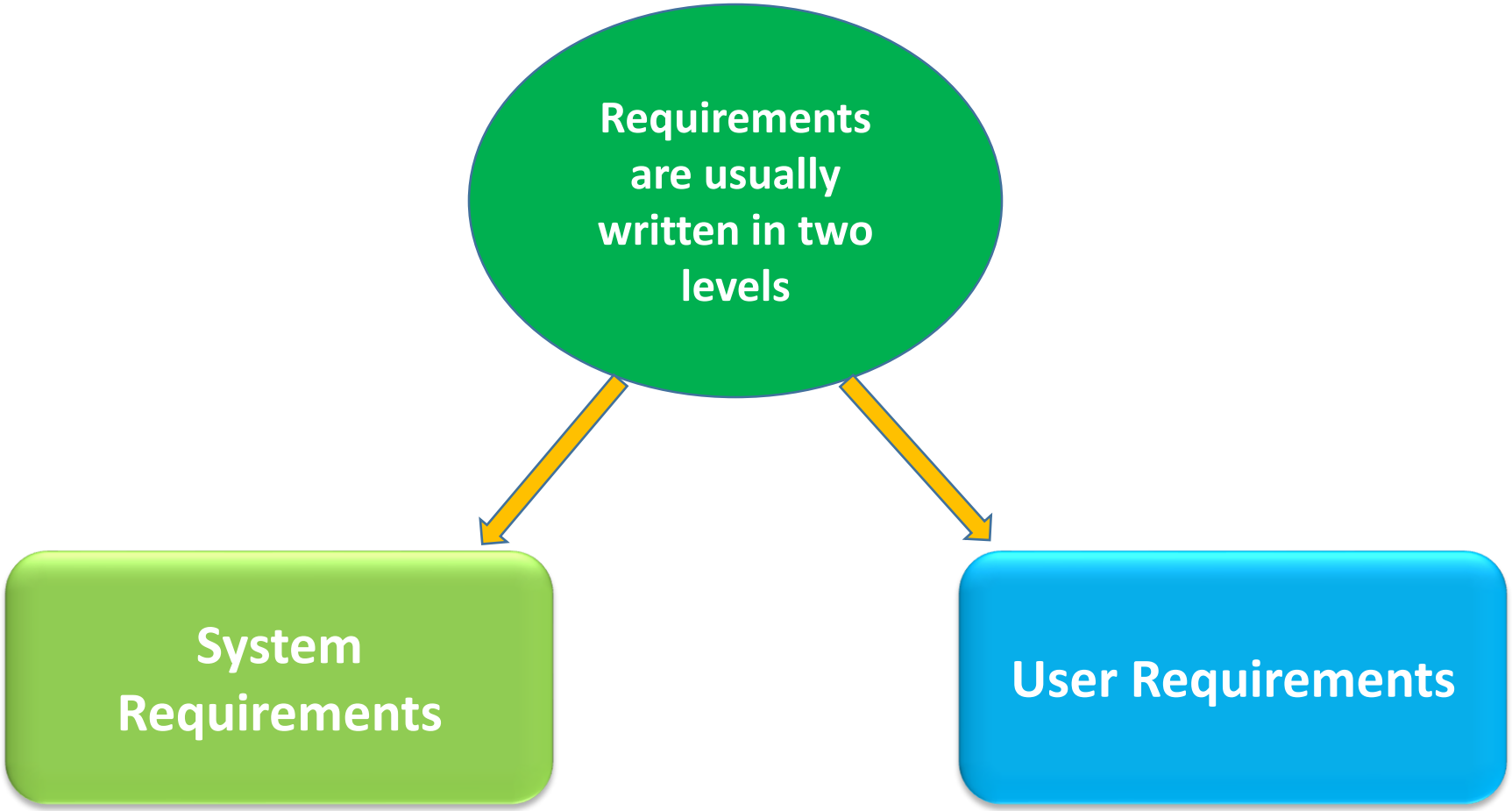
Requirements During Initial Phase

- The requirements are first written in **high-level**.
- This provides a chance for other software companies to provide detailed solutions for the customer organization.

- Later on, **a detailed description** of the system requirements should be generated.
- These **detailed requirements** serve as a contract between development company and customer.

- During the RFP, the customer may write the requirements in high level format. This is to allow contractors to specify different solutions.
- Later on, when the customer contracts with software vendor, the requirements are written in more detailed format.

Requirements' Levels



Example: User vs. System Requirements

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

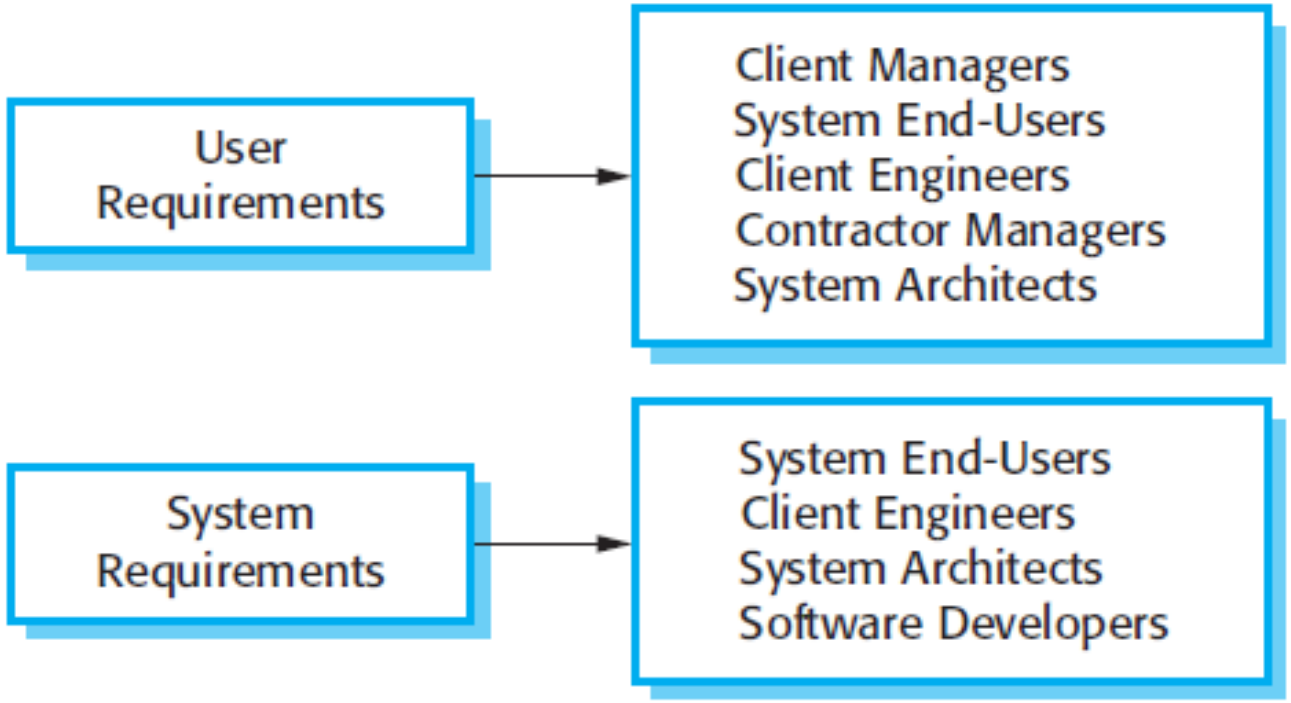
1.1 On the last working day of each month, a summary of the drugs prescribed, their cost, and the prescribing clinics shall be generated.

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.



Functional and 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 system should behave in particular situations.

- **Non-Functional Requirements**: These are constraints on the services or functions offered by the system.

Functional and Non-Functional Requirements..2

- In reality, the distinction between different types of requirement is **not as clear-cut** as these simple definitions suggest.
- Sometimes, a non-functional requirement can **generate** several functional requirements.

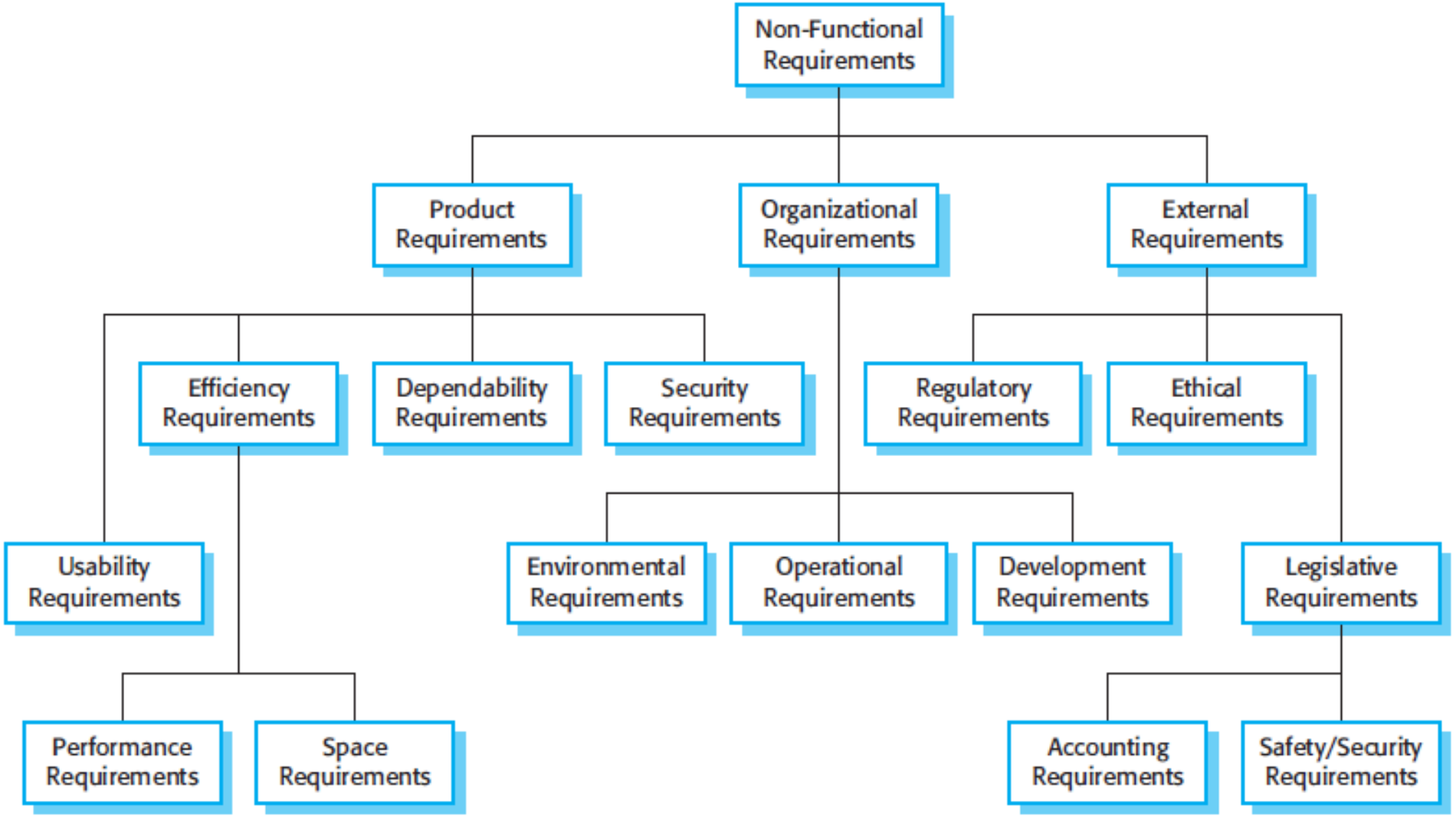
Examples on Functional Requirements:

1. A user shall be able to search the appointments lists for all clinics.
2. The system shall generate each day, for each clinic, a list of patients who are expected to attend appointments that day.
3. Each staff member using the system shall be uniquely identified by his or her eight-digit employee number.

Requirements Properties

- In general, the system requirements should be
  - Correct
  - Consistent
  - Complete

- This however will not be easy in large and complex systems with many **stakeholders**



Domain Requirements

- Domain requirements in software analysis refer to the specific needs, constraints, and conditions that are inherent to the domain or industry in which the software will operate.

- These requirements are influenced by the characteristics, practices, regulations, and standards of the domain and must be considered during the analysis and design phases of software development.

Example Domain Requirements for E-Commers web App

- **Payment Gateway Integration**: In the domain of ecommerce, one of the primary requirements is to integrate with various payment gateways to facilitate secure online transactions. The software needs to support multiple payment methods such as credit/debit cards, digital wallets, and bank transfers. Additionally, it must comply with relevant payment card industry (PCI) standards to ensure the security of sensitive financial information.
- **Product Categorization and Filtering**: Ecommerce platforms often require sophisticated product categorization and filtering functionalities to help customers easily find and navigate through a large inventory of products. Domain requirements might specify the need for features such as hierarchical category structures, product attribute filtering, and sorting options based on relevance, price, or popularity.

Example cont…….

- **Shipping and Logistics Integration**: Another important domain requirement is the integration with shipping and logistics services to facilitate order fulfillment and delivery. The software should support real-time calculation of shipping costs based on factors such as package weight, destination, and preferred shipping method. It should also generate shipping labels and tracking information to keep customers informed about the status of their orders.
- **Tax Calculation and Compliance**: Ecommerce websites operating in different regions or countries must comply with local tax regulations and calculate appropriate taxes for each transaction. Domain requirements may include the ability to automatically calculate taxes based on the customer's location, product type, and applicable tax rates. The software should also generate accurate tax invoices and reports for accounting and tax compliance purposes.
- **User Authentication and Authorization**: Security is paramount in the ecommerce domain, so domain requirements often include robust user authentication and authorization mechanisms. The software should support secure user registration, login, and password management features. Additionally, it should enforce access controls to restrict certain functionalities or data based on user roles and permissions.

Example..3

- Example1: a train control system has to take into account the braking characteristics in different weather conditions.
- Example2: a PMS has to enforce all confidentiality rules in accordance with national medical domain practices

May generate new functional requirements, and new constraints on existing requirements or define specific computations

If domain requirements are not satisfied, the system may be unworkable

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 Should be Quantifiable

- **A system goal**

  G.8.1 The PMS system should be easy to use by medical staff and should be organized in such a way that user errors are minimized. (**Goal**)

- **A verifiable non-functional requirement**

  8.4.3 Medical staff shall be able to use all the PMS 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. (**Testable non-functional requirement**)

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

The Software Requirements Document (SRS)

- It 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
- **Agile methods** do not use official SRS, instead they use **User Stories**
- **Critical and Outsourcing Projects** needs detailed SRS document

Level of Details When Writing Requirements

- The level of detail depends on the type of system that is being developed and the development process used.
- **Critical systems** need to have detailed requirements because safety and security have to be analyzed in detail.
- When the system is to be developed by a **separate company** the system specifications need to be detailed and precise.
- If an **in-house**, iterative development process is used, the requirements document can be much less detailed and any ambiguities can be resolved during development of the system.

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.

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 indude 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.

- At this stage, **focus on what the system should do, not how the system will do it (design)**
- User requirements should be **simple** natural language who don't have technical background.
- User requirements can have some **intuitive diagrams** such as use case diagram.

Requirements Specification: Natural language

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: Structured Format

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.

Guidelines when writing requirements specification:

1. You should have standard format for requirements specification.
2. Use language consistently to distinguish between mandatory (shall) and desirable (should) requirements.
3. Use text highlights (bold, italic) to highlight parts of req.
4. Do not use technical language.
5. It can be excellent to include a rationale with each user requirement (why this req. has been included?)

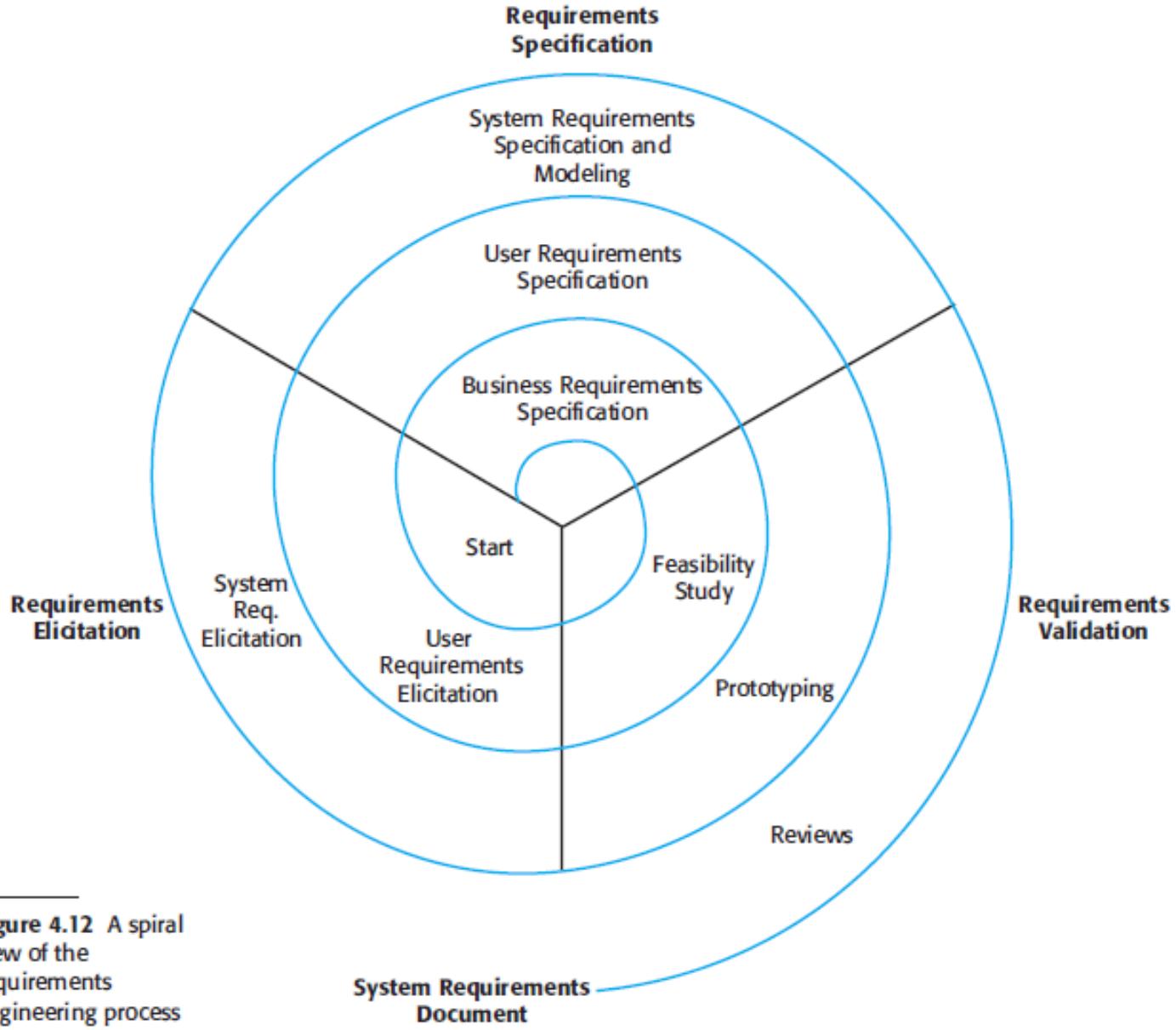
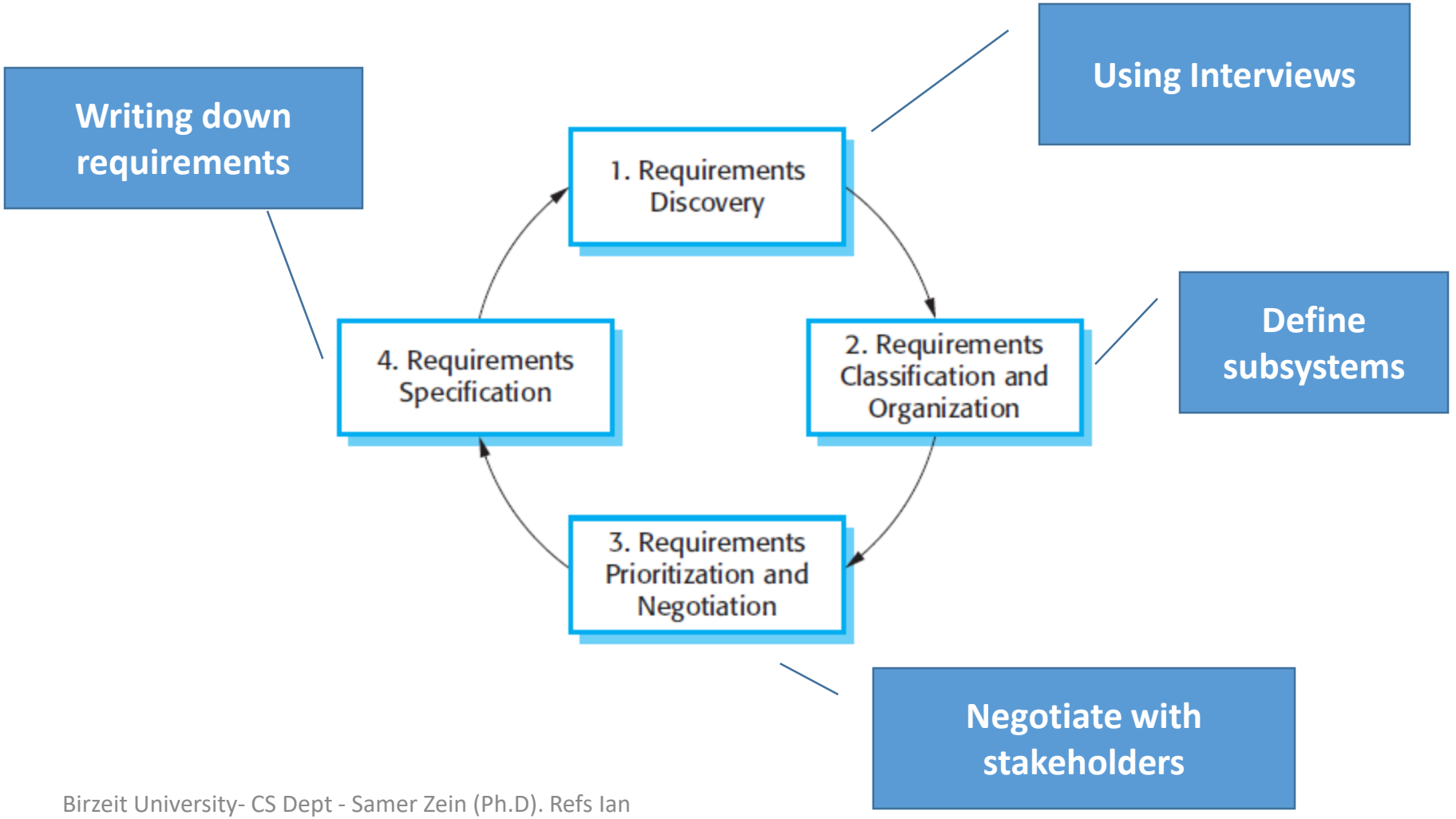


Figure 4.12 A spiral view of the requirements engineering process

Requirements Elicitation & Analysis Process



Requirement Analysis

The process of understanding customer requirements and their implications. It comes after requirements discovery

It involves technical staff working on the discovered requirements, iteratively with customers, to understand their technical implication and importance.

It includes the processes of classifying and organising requirements, negotiating (with customers) and prioritizing them in the order of their importance to the customers.

The output of this process is the requirement specification document (SRS) negotiated and accepted with customers.

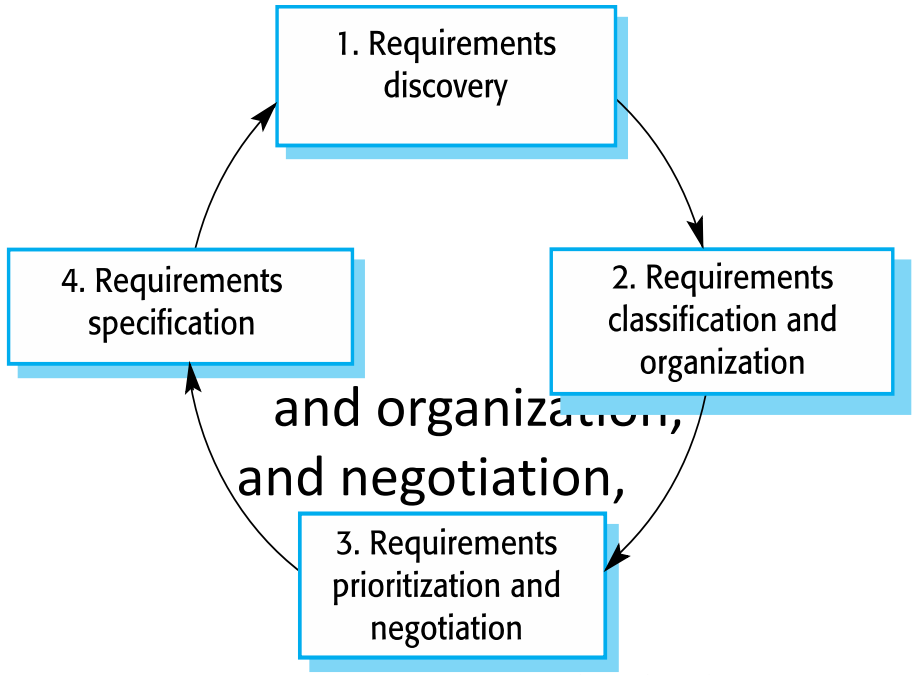
Problems of requirements analysis

- Stakeholders don’t know what they really want.
- Stakeholders express requirements in their own terms.
- Different stakeholders may have conflicting requirements.
- The requirements change during

the analysis process

- Political factors

- Stages include:
  - Requirements discovery,
  - Requirements classification
  - Requirements prioritization
  - Requirements specification.



Stakeholders in the MHC-PMS

- **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.
- etc

Techniques

There are many requirement engineering techniques for requirement elicitation and analysis, some of the often used ones:

- Interviewing
- Scenario generation
- Use case analysis
- Ethnography

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.
- Focused interviews, with clusters of 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.

Interviews

Meeting introductory protocol

- Ensure cultural introduction protocols are followed

First meeting

- Aim: to understand the business and its context with a clear aim to understand business processes and services.

Effective meetings:

- Ensure a chair is assigned at the beginning, to keep time-controlled progress
- Ensure an agenda is defined with clear objectives of the target outcome of the meeting
- Ensure a timescale is set for each agenda item and is kept/controlled by the chair
- Ensure clear actions and decisions (and who is responsible for and by when) are identified and reached by the end of the meeting
- Ensure the actions and decisions are summarised at the end of the meeting

Scenarios

Scenarios are real-life examples of how a system can be used.

They 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.

Scenario for collecting medical history: Example

- **Initial assumption**: The patient has seen a medical receptionist who has created a record in the system and collected the patient's personal information (name, address, age, etc.). A nurse is logged on to the system and is collecting medical history.
- **Normal**: The nurse searches for the patient by family name. If there is more than one patient with the same surname, the given name (first name in English) and date of birth are used to identify the patient.
- The nurse chooses the menu option to add medical history.
- The nurse then follows a series of prompts from the system to enter information about consultations elsewhere on mental health problems (free text input), existing medical conditions (nurse selects conditions from menu), medication currently taken (selected from menu), allergies (free text), and home life (form).

Scenario for collecting medical history: Example

Successful output?

What can go wrong?

Alternative: The patient's record does not exist or cannot be found. The nurse should create a new record and record personal information. — Yes

Alternative: Patient conditions or medication are not entered in the menu. The nurse should choose the 'other' option and enter free text describing the condition/medication. — Yes

Error: Patient cannot/will not provide information on medical history. The nurse should enter free text recording the patient's inability/unwillingness to provide information. The system should print the standard exclusion form stating that the lack of information may mean that treatment will be limited or delayed. This should be signed and handed to the patient. — No?

Other activities: Record may be consulted but not edited by other staff while information is being entered.

System state on completion: User is logged on. The patient record including medical history is entered in the database, a record is added to the system log showing the start and end time of the session and the nurse involved.

Use cases

Use-cases are a scenario based technique, in the UML, which identifies the actors in an interaction and describes the interaction itself.

A set of use cases should describe all possible interactions with the system.

High-level graphical model supplemented by more detailed structured or tabular description.

Sequence diagrams may be used to add detail to use-cases by showing the sequence of event processing in the system.

Use case diagrams

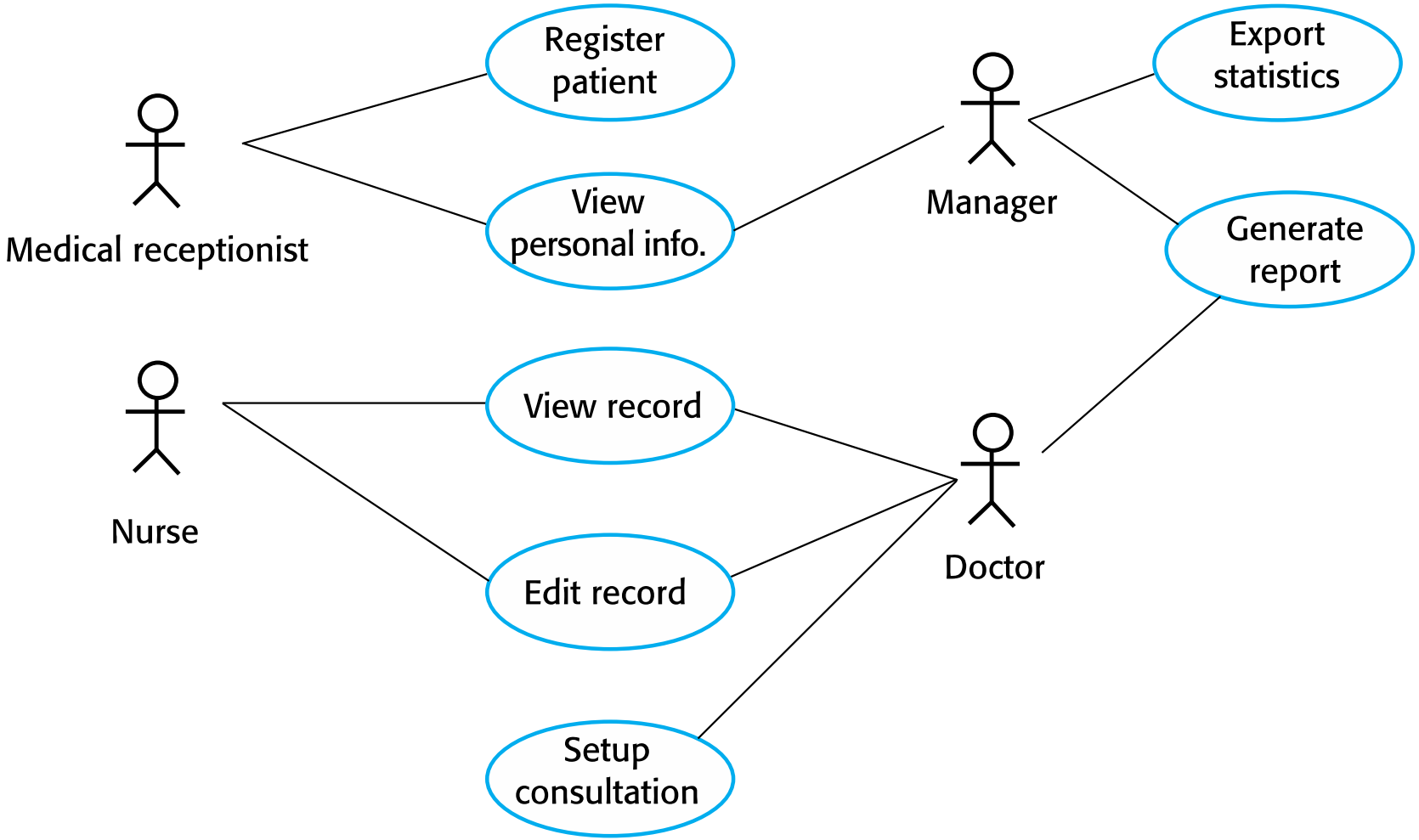
- Used during inception, requirements elicitation and requirements analysis
- Describe the functionality of the system
- Use cases focus on the behavior of the system from an external point of view.
- A use case describes a **function** provided by the system that **yields a visible result** for an actor

- **A *Use Case diagram*** shows you some of the use cases in your system, some of the actors in your system, and the relationships between them.

- ***use case*** is a high−level piece of functionality that the system will provide.

- **An actor** is anyone or anything that interacts with the system being built.

Use cases for the MHC-PMS



Ethnography

A social scientist spends a considerable time observing and analysing how people actually work.

People do not have to explain or articulate their work.

Social and organisational factors of importance may be observed.

Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.

Requirements validation

Concerned with demonstrating that the requirements define the system 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 Check list

- **Validity Checks**: sometimes additional requirements are needed.

- **Consistency Checks**: no conflicts.
- C**ompleteness Checks**: nothing is missing.

- **Realism Checks**: can such requirements be done using current knowledge and technology?
- **Verifiability**: requirements should be verifiable

Requirements Validation Techniques

- **Requirements reviews**: systematic analysis of requirements by a team of reviewers.
- **Prototyping**: executable model of system is developed and presented to stakeholders.
- **Test-case generation**: requirements should be testable