CH4: Requirements Engineering

Birzeit University, CS Dept, Samer Zein (Ph.D). Updated by Saad Mansour, 2024

Objectives

- When you have read the chapter, you will:
  - understand the concepts of user and system requirements and why these requirements should be written in different ways;
  - understand the differences between functional and non-functional software requirements;
  - understand the main requirements engineering activities of elicitation, analysis, and validation, and the relationships between these activities;
  - understand why requirements management is necessary and how it supports other requirements engineering activities.

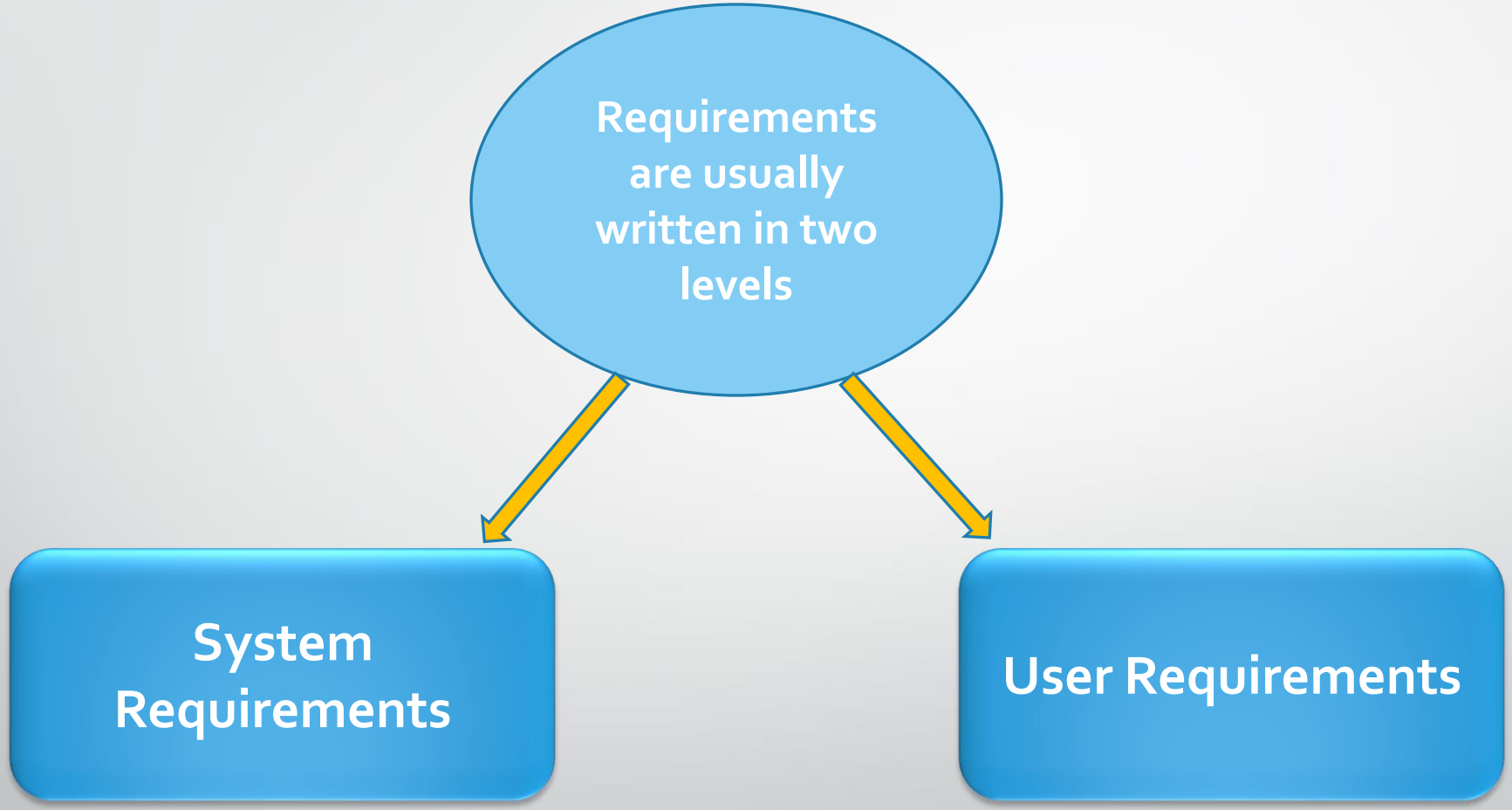
Introduction

- 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.
- The process of finding out, analyzing, documenting and checking these services and constraints is called requirements engineering (RE).

Requirements During Initial Phase

- The requirements are first written in **high-level**. During the RFP (request for proposal), the customer may write the requirements in high level format. This is to allow contractors to specify different solutions.
- This provides a chance for other software companies to provide detailed solutions for the customer organization.
- Later, **a detailed description** of the system requirements should be generated.
- These **detailed requirements** serve as a contract between development company and customer.

Requirements' Levels



Requirements' Levels

- **User requirements** are statements, in a natural language (general) 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 (more specific) descriptions of the software system's functions, services, and operational constraints. The system requirements document (sometimes called a functional specification) should define exactly what is to be implemented. It may be part of the contract between the system buyer and the software developers.

Example: User vs. System Requirements

Figure 4.1 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.

UR 3 : The library system should allow users to search for books online and see their availability and status.

- SR 3.1: System users namely librarians, members, and managers, shall be able to search for and view books information.
- SR 3.2: System users can search for books using any combination of book title, author, category, and year.
- SR 3.3: The search results should contain the following book information: title, year, authors, number of pages, and book status.
- SR 3.4: System shall show results in no more than 3 seconds
- SR 3.5: Book status can be one of the following: available, rented, reserved, or pending
- SR 3.6: Clicking on book title, the system shall display complete book information along with book cover picture.

Requirements' Readers

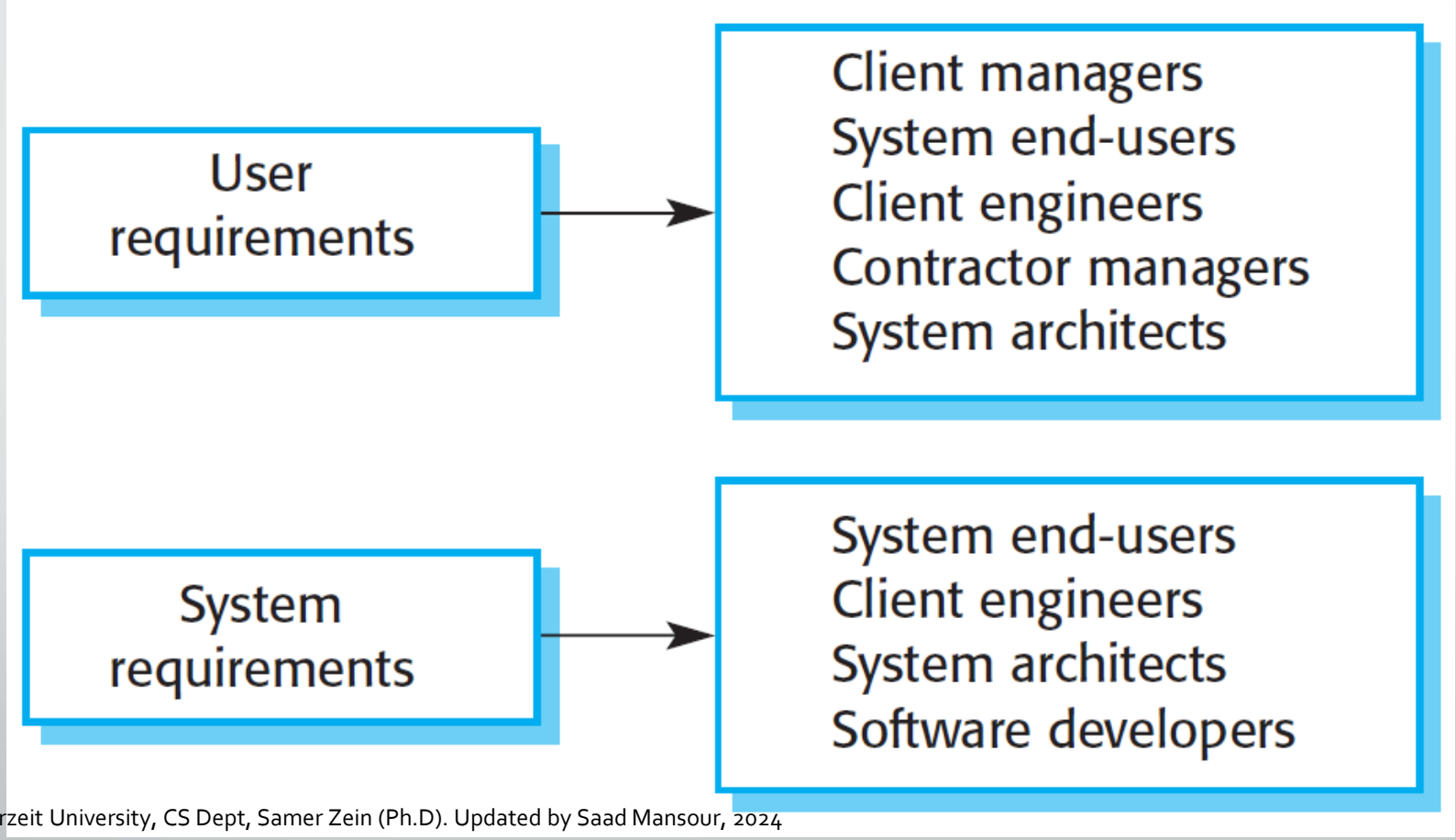


Figure 4.2 Readers of different types of requirements specification

4.1 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. They are often applied to the system as a whole, rather than individual system features or services.

4.1 Functional and non-functional requirements Cont.

- In reality, the distinction between different types of requirement is **not as clear-cut** as these simple definitions suggest.
- Requirements are not independent and that one requirement often generates or constrains other requirements.
- Sometimes, a non-functional requirement can **generate** several functional requirements.

4.1.1 Functional requirements

- When expressed as user requirements, functional requirements should be written in natural language so that system users and managers can understand them. Functional system requirements expand the user requirements and are written for system developers. They should describe the system functions, their inputs and outputs, and exceptions in detail.
- Examples of functional requirements to maintain information about patients receiving treatment for mental health problems:
  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.
- The requirements show that functional requirements may be written at different levels of detail (contrast requirements 1 and 3).

Requirements Properties/Characteristics

- In general, the system requirements should be:
  - **Correct**: a requirement is correct when it is part of the actual needs of the system. Should not be beyond the scope of the actual needs of system.
    - Example in requirement 1 “A user shall be able to search the appointments lists for all clinics”:
      - User requirement/need: search for a patient name across all appointments in all clinics
      - Implied requirements: search for a patient name, Date of Birth, Address in a clinic.
  - **Consistent:** means that requirements should not be contradictory.
    - Conflicts between different non-functional requirements are common in complex systems, such as Usability vs. Security.
  - **Complete:** means that all services and information required by the user should be defined
    - This however will not be easy in large and complex systems with many **stakeholders**

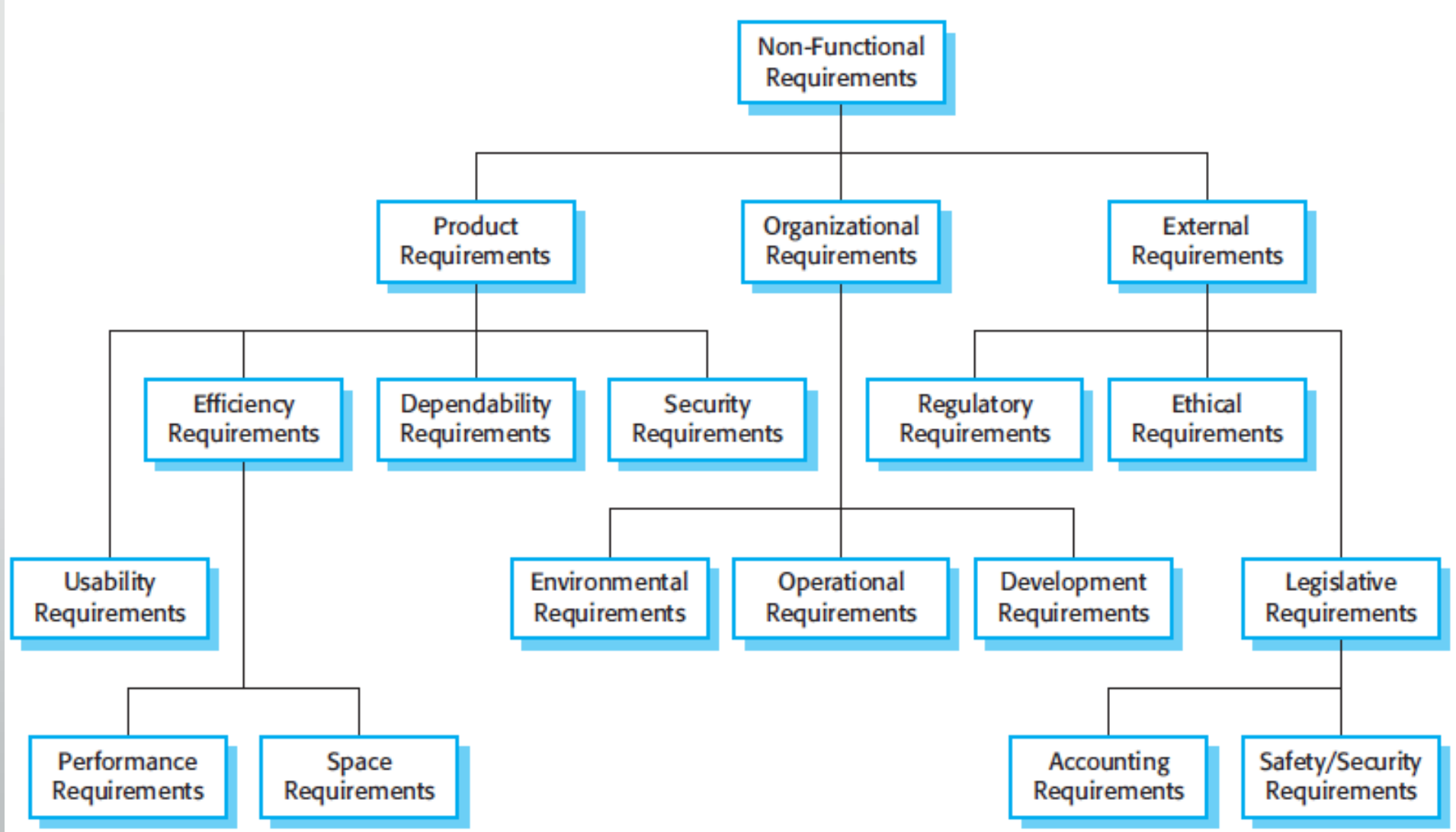
Requirements Properties/Characteristics

- **Unambiguous/precision, clear, easy to understand**: ambiguous/imprecise requirements may be interpreted in different ways by developers and users or customers.
  - It is natural for a system developer to interpret an ambiguous requirement in a way that simplifies its implementation
  - Example in requirement 1: "A user shall be able to search the appointments lists for all clinics"
    - User expect "search" to mean that, given a patient name, the system looks for that name in all appointments at all clinics.
    - System developers may interpret search function to require the user to choose a clinic and then carry out the search of the patients who attended that clinic.
- **Traceable**: requirements are often uniquely identified by a unique number to be traced/referenced in validation and testing phases.
  - Example: requirement has its own unique ID/Number

4.1.2 Non-functional requirements

- Non-functional requirements usually specify or constrain characteristics of the system as a whole.
- While it is often possible to identify which system components implement specific functional requirements, this is often more difficult with non-functional requirements.
- The implementation of these requirements may be spread throughout the system.

Types of non-functional requirements



Non-functional requirements

PRODUCT REQUIREMENT
The Mentcare system shall be available to all clinics during normal working hours (Mon–Fri, 08:30–17:30). Downtime within normal working hours shall not exceed 5 seconds in any one day.

ORGANIZATIONAL REQUIREMENT
Users of the Mentcare system shall identify themselves using their health authority identity card.

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

Figure 4.4 Examples of possible non-functional requirements for the Mentcare system

- A common problem with non-functional requirements is that stakeholders propose requirements as general goals, such as ease of use, the ability of the system to recover from failure, or rapid user response.

Non-Functional Requirements Should be Quantifiable

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

- Rewritten as a "testable" nonfunctional 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.

Metrics for specifying nonfunctional requirements

- Whenever possible, you should write non-functional requirements (NFR) quantitatively so that they can be objectively tested.
- You can measure these characteristics when the system is being tested to check whether or not the system has met its nonfunctional requirements.
- Keep in mind that some NFR cannot or difficult to be quantified, ex. maintainability.
- Non-functional requirements often conflict and interact with other functional or 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

- The software requirements document (sometimes called the software requirements specification or 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.
- Sometimes the user and system requirements are integrated into a single description.
- **Agile methods** do not use official SRS, instead they use **User Stories.**
- **Critical and Outsourcing Projects** needs detailed SRS document
- The requirements document has a diverse set of users with different ways of use. Ex: system customers use requirements to check that they meet their needs, while system engineers use requirements to understand what system is to be developed.

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** (e.g., through outsourcing), 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.

What Should be Included in the SRS?

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.

What Should be Included in the SRS?

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.
- User requirements should be **simple** natural language, with simple tables, forms, and intuitive diagrams so that they are understandable by system users who don't have detailed technical knowledge.
- System requirements are expanded versions of the user requirements that software engineers use as the starting point for the system design.
- At this stage, **focus on what the system should do (external behavior), not how the system will do it (design).**
- User requirements can have some **intuitive diagrams** such as use case diagram.

Requirements specification: Natural language specification

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.*)

Figure 4.9 Example requirements for the insulin pump software system

Requirements specification: Structured specifications

- Structured natural language is a way of writing system requirements where requirements are written in a standard way or format rather than as free-form text.

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. (see Figure 4.14)
Requires	Two previous readings so that the rate of change of sugar level can be computed.
Precondition	The insulin reservoir contains at least the maximum allowed single dose of insulin.
Postcondition	r0 is replaced by r1 then r1 is replaced by r2.
Side effects	None.

Figure 4.10 A structured specification of a requirement for an insulin pump

Requirements specification: Structured specifications

- Tables are particularly useful when there are a number of possible alternative situations and you need to describe the actions to be taken for each of these.

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 & ((r2 − r1) ≥ (r1 − r0))	CompDose = round ((r2 − r1)/4) If rounded result = 0 then CompDose = MinimumDose

Figure 4.14 The tabular specification of computation in an insulin pump

Guidelines when writing requirements specification

- To minimize misunderstandings when writing natural language requirements, you follow guidelines such as:
  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, or color) to highlight parts of the requirement..
  4. Do not use technical language (do not assume that readers understand technical).
  5. It can be excellent to include a rationale with each user requirement (why this req. has been included? and source)

Requirements engineering processes

- In practice, requirements engineering is an <u>iterative process</u> in which the activities are interleaved.

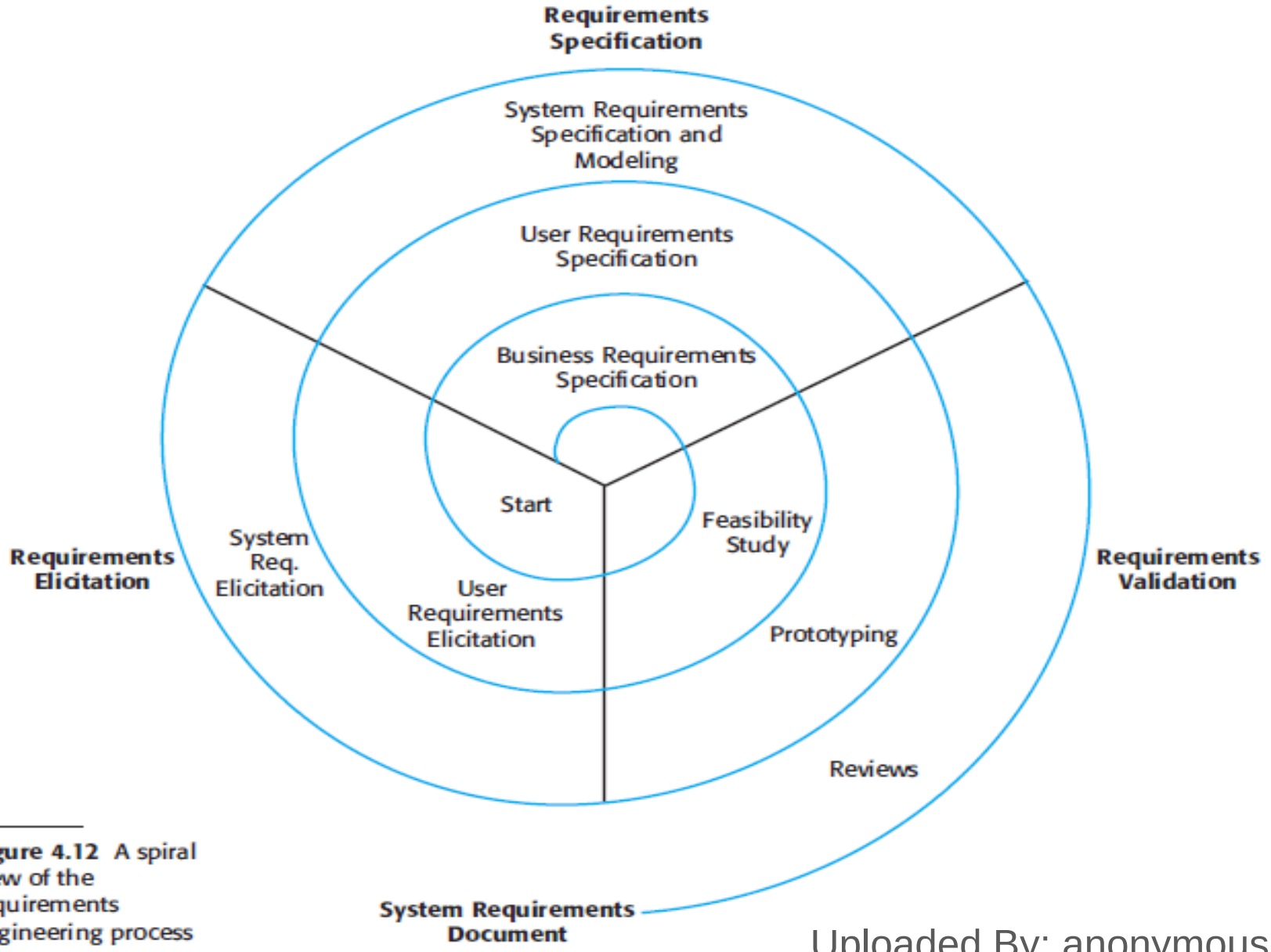
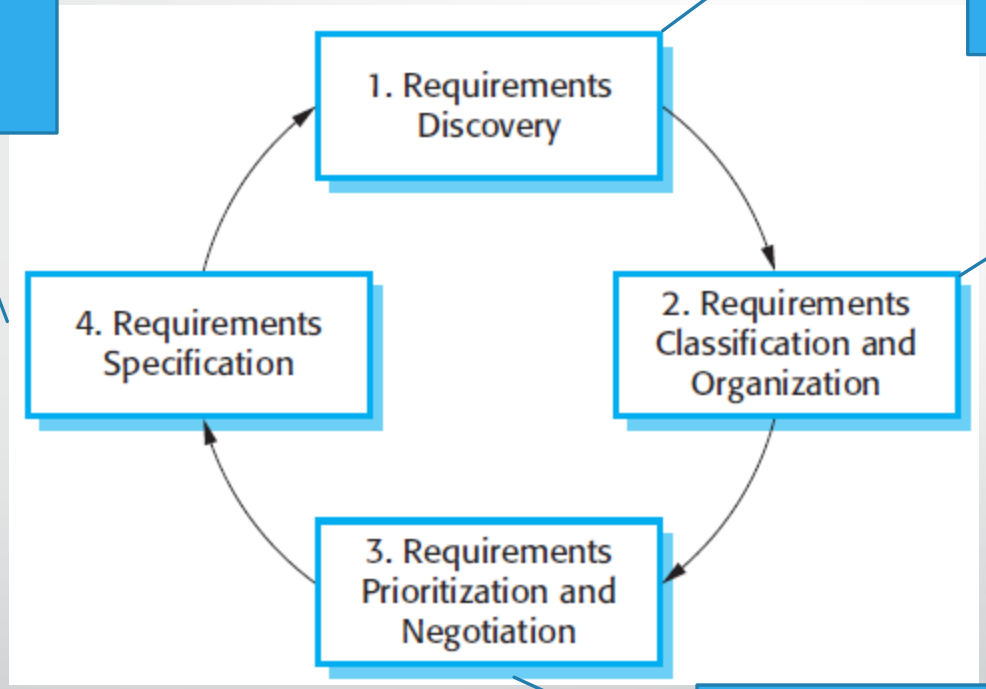


Figure 4.12 A spiral view of the requirements engineering process

Requirements elicitation and analysis

- The aims of the requirements elicitation process are to understand the work that stakeholders (ex: customers, end-users) do and how they might use a new system to help support that work.

Requirements Elicitation & Analysis Process



interacting with stakeholders to discover their requirements, ex: using Interviews

Define subsystems, groups related requirements

Negotiate with stakeholders, prioritizing requirements and resolving requirements conflicts

Requirements documentation

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 Eliciting requirements and analysis

- Eliciting and understanding requirements from system stakeholders is a difficult process for several reasons:
  - Stakeholders often don’t know what they want.
  - Stakeholders express requirements in their own terms and with implicit knowledge of their own work.
  - Different stakeholders may have conflicting requirements and express their requirements in different ways.
  - The requirements change during the analysis process.
  - Political factors.

System Stakeholders

- Stakeholders range from end-users of a system through managers to external stakeholders such as regulators, who certify the acceptability of the system. For example, system stakeholders for the mental healthcare patient information system include:
  - **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

Requirements elicitation 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
- You could use a mix of techniques to collect information and, from that, you derive the requirements.

Interviewing

- Formal or informal interviews with system stakeholders are part of most requirements engineering processes.
- In these interviews, the requirements engineering team puts questions to stakeholders about the system that they currently use and the system to be developed.
- Types of interview:
  - Closed interviews: where the stakeholder answers a predefined set of questions.
  - Open interviews: no predefined agenda, where range of issues are explored with system stakeholders.
  - Focused interviews: with clusters of stakeholders, for resolving conflicts and agree on compromise requirements
- Effective interviewers:
  - 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 requirements proposal, or by working together on a prototype system. Saying to people "tell me what you want" is unlikely to result in useful information.

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 (chairperson, organizer) 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.
- A scenario should include:
  - A description of what the system and users expects when the scenario starts.
  - A description of the normal flow of events in the scenario.
  - A description of what can go wrong and how this is handled.
  - Information about other activities that might be going on at the same time (concurrent activities).
  - A description of the system state when the scenario finishes.

Scenario for collecting medical history

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?

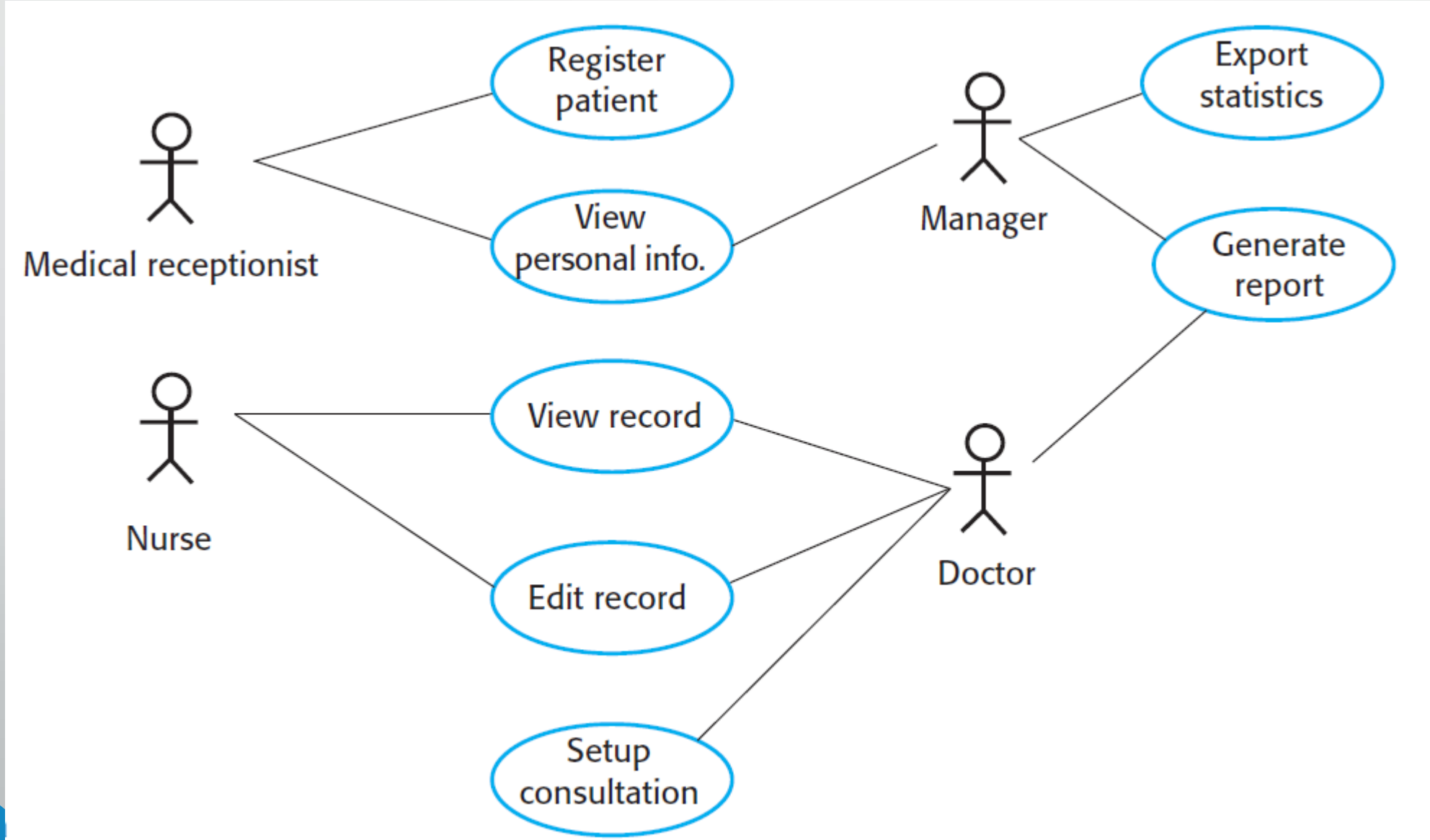
Other activities: Record may be consulted but not edited by other staff while information is being entered.

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.

Requirements specification: Use cases

- Use-cases are a scenario based technique, in the UML, which identifies the actors in an interaction and describes the interaction itself.
- The Unified Modeling Language (UML) is a standard for object-oriented modeling.
- Use cases are a way of describing interactions between users and a system using a graphical model and structured text.
- **A *Use Case diagram*** is a high-level diagram, that 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. A use case describes a function provided by the system that yields a visible result for an actor.
- **An actor** is anyone or anything that interacts with the system being built. Actors in the process, who may be human or other systems, are represented as stick figures.
- A use case identifies the actors involved in an interaction and names the type of interaction. You then add additional information describing the interaction with the system. The additional information may be a textual description or one or more graphical models such as the UML sequence or state charts.
- The set of use cases represents all of the possible interactions that will be described in the system requirements

Example: Use cases consultation for the Mentcare system



Requirements specification: Use cases

- Used during inception, requirements elicitation and requirements analysis.
- Use cases focus on the behavior of the system from an external point of view.
- High-level graphical model supplemented by more detailed structured or tabular description.
- Use cases identify the individual interactions between the system and its users or other systems. Each use case should be documented with a textual description.
- For example, a brief description of the Setup Consultation use case:
  - *Setup consultation allows two or more doctors, working in different offices, to view the same patient record at the same time. One doctor initiates the consultation by choosing the people involved from a dropdown menu of doctors who are online. The patient record is then displayed on their screens, but only the initiating doctor can edit the record. In addition, a text chat window is created to help coordinate actions. It is assumed that a phone call for voice communication can be separately arranged.*

Case Study

- In an HR management system, Rami, can send a vacation request specifying start and end dates, notes, etc. Mariam on the other hand, the HR manager, can view, and either approve or reject the request. If she rejects the request, she has to specify why. Upon approval or rejection, a notification appears for Rami.

Case Study

- Dialy FitPro is a mobile app for tracking user activity, calories, and overall fitness status. After installing the app, the user needs to create a profile (name, weight, gender, height, etc). After successful registration, the app send email notification. Later on, app can monitor user daily activities and movements using accelometer sensor. The app can also show daily routes taken by user through integration with Google Maps. Further, the app allows the user to specify a fitness plan, for example, the user can specify that she/he wants to lose certain weight in specific time and so on. Finally, the user can allow the app to share activity status on Facebook account.

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 that affect the use of the system may be observed.
- Ethnography is an observational technique that can be used to understand operational processes and help derive requirements for software to support these processes.
- Ethnographic studies have shown that work is usually richer and more complex than suggested by simple system models.
- Ethnography is particularly effective for discovering two types of requirements:
  - Requirements derived from the way in which people actually work, rather than the way in which business process definitions say they ought to work.
  - Requirements derived from cooperation and awareness of other people's activities.

Requirements validation

- Requirements validation is the process of checking that requirements define the system that the customer really wants.
- It overlaps with elicitation and analysis.
- Requirements validation is critically important because errors in a requirements document can lead to extensive rework costs when these problems are discovered during development or after the system is in service.
- The cost of fixing a requirements problem by making a system change is usually much greater than repairing design or coding errors. A change to the requirements usually means that the system design and implementation must also be changed. Furthermore, the system must then be retested.
- 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**: check that the requirements reflect the real needs of system users. Sometimes additional requirements are needed.
- **Consistency checks**: requirements in the document should not conflict. That is, there should not be contradictory constraints or different descriptions of the same system function.
- **Completeness checks**: nothing is missing (all functions and constraints).
- **Realism checks**: can such requirements be implemented using current knowledge and technology? These checks should also take account of the budget and schedule for the system development.
- **Verifiability**: requirements should be verifiable. This means that you should be able to write a set of tests that can demonstrate that the delivered system meets each specified requirement.

Requirements Validation Techniques

- **Requirements reviews**: the requirements are analyzed systematically by a team of reviewers who check for errors and inconsistencies.
- **Prototyping**: executable model of system is developed and presented to stakeholders to see if it meets their needs and expectations.
- **Test-case generation**: requirements should be testable (developing tests from the user requirements before any code is written is an integral part of test-driven development).