BCSE301L SOFTWARE ENGINEERING

Dr.D.KAVITHA

SCOPE

VIT, CHENNAI

Modelling Requirements

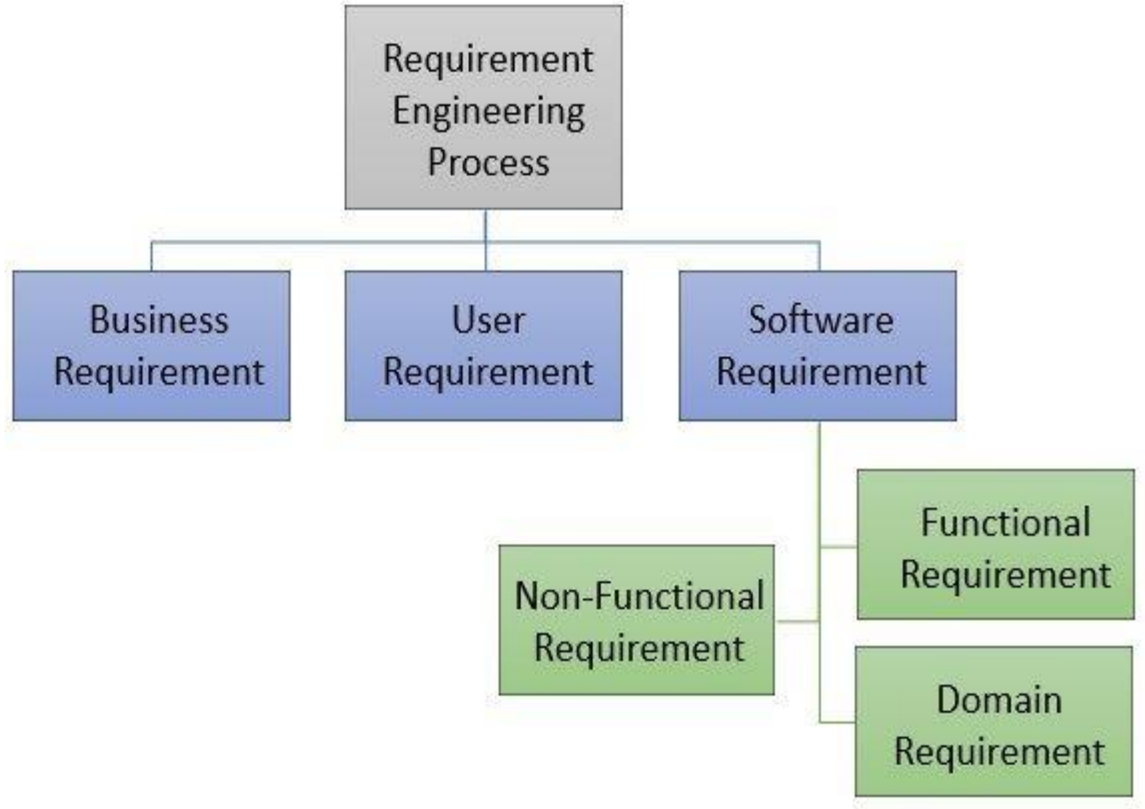
Software requirements and its types, Requirements Engineering process, Requirement Elicitation, System Modeling – Requirements Specification and Requirement Validation, Requirements Elicitation techniques, Requirements management in Agile.

Software requirements and its types

Types of Software Requirement

There are three types of software requirements as follows:

1. Functional requirements
2. Non-Functional requirements
3. Domain requirements



Functional Requirements

- Functional requirements are such software requirements that are demanded explicitly as basic facilities of the system by the end-users. So, these requirements for functionalities should be necessarily incorporated into the system as a part of the contract.
- They describe system behavior under specific conditions. In other words, they are the functions that one can see directly in the final product, and it was the requirements of the users as well. It describes a software system or its components.
- These are represented as inputs to the software system, its behavior, and its output. It can be a calculation, data manipulation, business process, user interaction, or any other specific functionality which defines what function a system is likely to perform.

Functional software requirements help us to capture the intended behavior of the system.

Functional requirements can be incorporated into the system in many ways as

1. Natural language
2. A structured or formatted language with no rigorous syntax and formal specification language with proper syntax.

Examples of functional requirements

1. Whenever a user logs into the system, their authentication is done.
2. In case of a cyber attack, the whole system is shut down
3. Whenever a user registers on some software system the first time, a verification email is sent to the user.

Non-functional Requirements(NFRs)

- These requirements are defined as the quality constraints that the system must satisfy to complete the project contract.

Non-functional requirements are more abstract. They deal with issues like-

- Performance
- Reusability
- Flexibility
- Reliability
- Maintainability
- Security
- Portability

Non-Functional Requirements are classified into many types. Some of them are as:

- Interface Constraints
- Economic Constraints
- Operating Constraints
- Performance constraints**:** storage space, response time, security, etc.
- Life Cycle constraints: portability, maintainability, etc.

Domain Requirements

- Domain requirements are the requirements related to a particular category like software, purpose or industry, or other domain of projects. Domain requirements can be functional or non-functional. These are essential functions that a system of specific domains must necessarily exhibit.
- The common factor for domain requirements is that they meet established standards or widely accepted feature sets for that category of the software project. Domain requirements typically arise in military, medical, and financial industry sectors. They are identified from that specific domain and are not user-specific.

Examples of domain requirements are- medical equipment or educational software.

Software in medical equipment

- In medical equipment, software must be developed per IEC 60601 regarding medical electrical equipment's basic safety and performance.
- The software can be functional and usable but not acceptable for production because it fails to meet domain requirements.

An Academic Software

- Such software must be developed to maintain records of an institute efficiently.
- Domain requirement of such software is the functionality of being able to access the list of faculty and list of students of each grade.

- **Difference between Functional Requirement and Non-Functional Requirement**

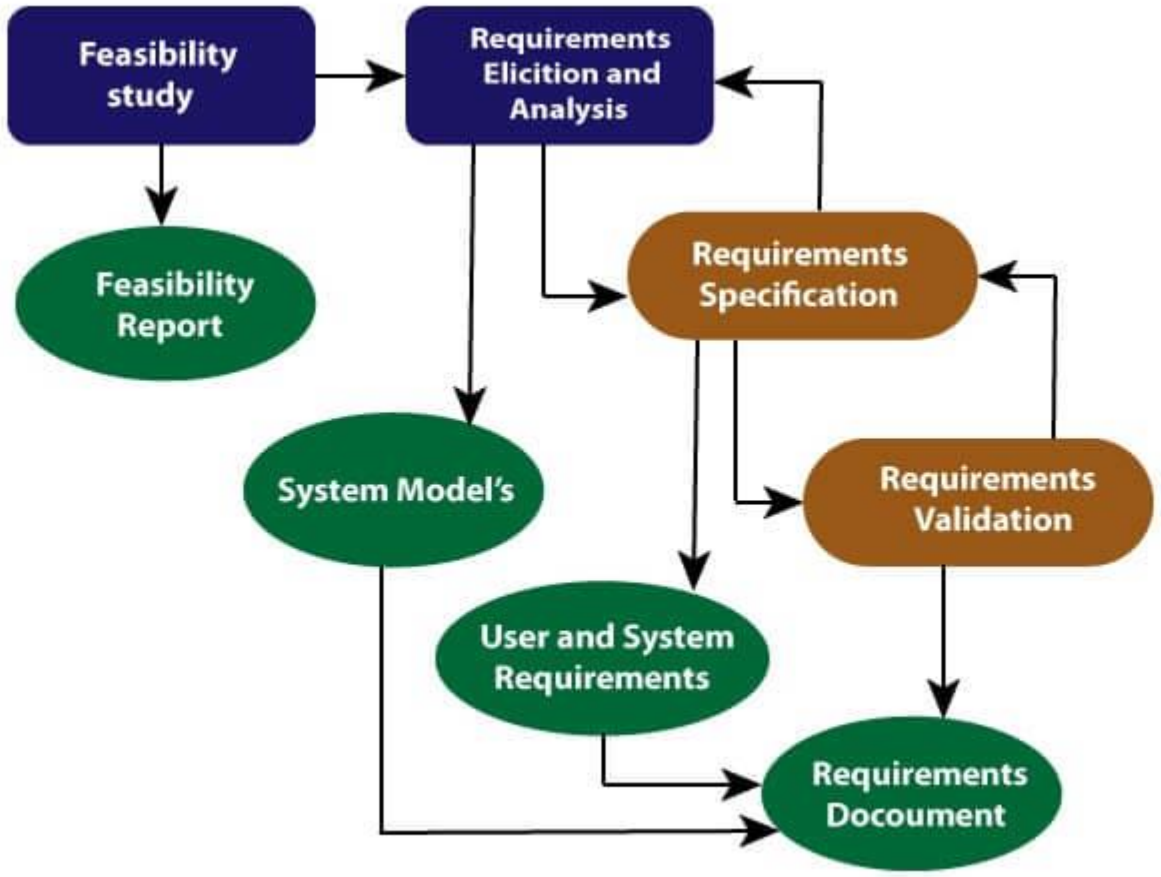
Functional Requirement	Non-Functional Requirement
It is used for defining a system and its components.	It is used for defining the quality attribute of a software system.
It focuses on what software will be doing.	It fixes the constraint on which software should fulfill the functional requirement.
The user specifies it.	Techies like architects or software developers specify it.
It is compulsory.	It is not compulsory.
It is easy to define.	It is comparatively tough to define.
It verifies the functionality of the system.	It verifies the performance of the system.
It is defined as a system at the component level.	It is defined as a system as a whole.
Example-System should be shut down if a cyber attack happens.	Example-Within 10 seconds, the processing should be done of each request.

Requirements Engineering process

- **Requirements engineering (RE)** refers to the process of defining, documenting, and maintaining requirements in the engineering design process. Requirement engineering provides the appropriate mechanism to understand what the customer desires, analyzing the need, and assessing feasibility, negotiating a reasonable solution, specifying the solution clearly, validating the specifications and managing the requirements as they are transformed into a working system.

Requirement Engineering Process is a five-step process, which includes -

1. Feasibility Study
2. Requirement Elicitation and Analysis
3. Software Requirement Specification
4. Software Requirement Validation
5. Software Requirement Management



Requirement Engineering Process

1. Feasibility Study:

- The objective behind the feasibility study is to create the reasons for developing the software that is acceptable to users, flexible to change and conformable to established standards.

Types of Feasibility:

1. **Technical Feasibility** - Technical feasibility evaluates the current technologies, which are needed to accomplish customer requirements within the time and budget.
2. **Operational Feasibility** - Operational feasibility assesses the range in which the required software performs a series of levels to solve business problems and customer requirements.
3. **Economic Feasibility** - Economic feasibility decides whether the necessary software can generate financial profits for an organization.

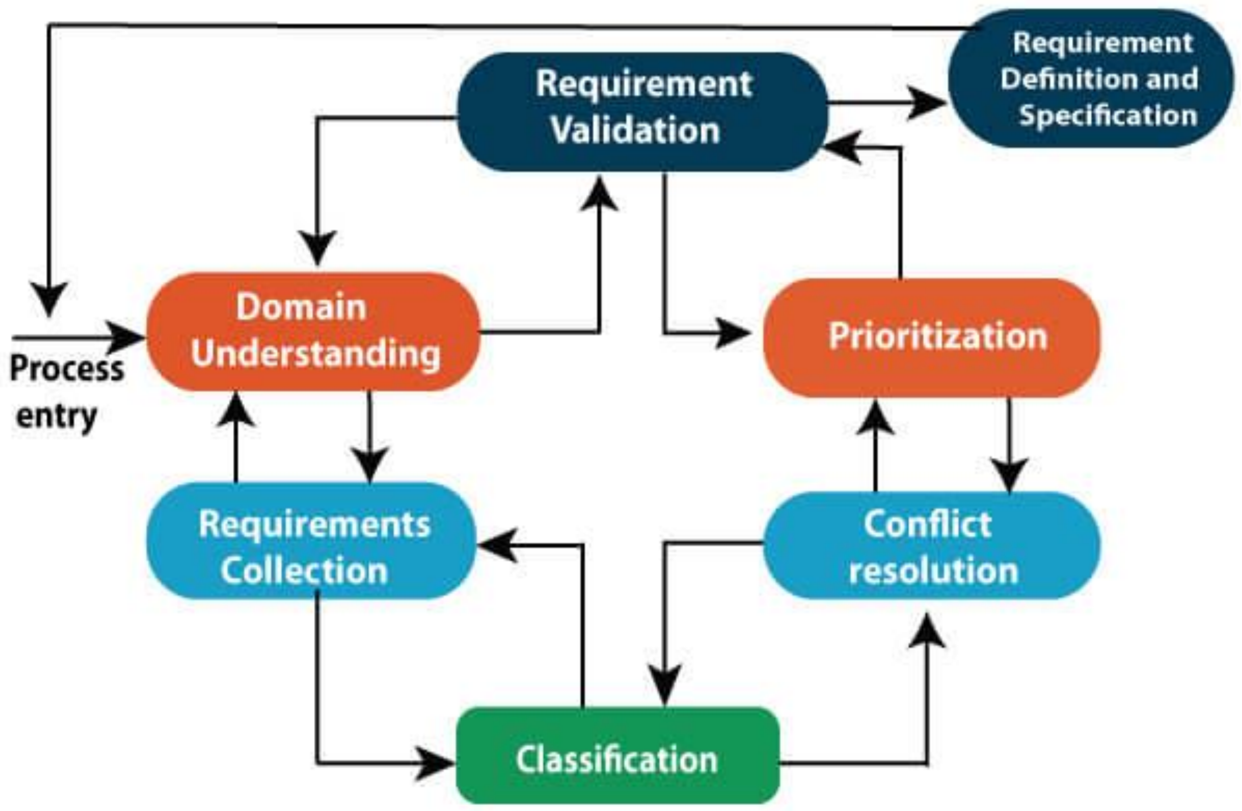
2.Requirement Elicitation and Analysis:

- This is also known as the **gathering of requirements**. Here, requirements are identified with the help of customers and existing systems processes, if available.
- Analysis of requirements starts with requirement elicitation. The requirements are analyzed to identify inconsistencies, defects, omission, etc. We describe requirements in terms of relationships and also resolve conflicts if any.

Problems of Elicitation and Analysis

- Getting all, and only, the right people involved.
- Stakeholders often don't know what they want
- Stakeholders express requirements in their terms.
- Stakeholders may have conflicting requirements.
- Requirement change during the analysis process.
- Organizational and political factors may influence system requirements.

Elicitation and Analysis Process



3. Software Requirement Specification:

- Software requirement specification is a kind of document which is created by a software analyst after the requirements collected from the various sources - the requirement received by the customer written in ordinary language. It is the job of the analyst to write the requirement in technical language so that they can be understood and beneficial by the development team.
- The models used at this stage include ER diagrams, data flow diagrams (DFDs), function decomposition diagrams (FDDs), data dictionaries, etc.
- **Data Flow Diagrams:** Data Flow Diagrams (DFDs) are used widely for modeling the requirements. DFD shows the flow of data through a system. The system may be a company, an organization, a set of procedures, a computer hardware system, a software system, or any combination of the preceding. The DFD is also known as a data flow graph or bubble chart.
- **Data Dictionaries:** Data Dictionaries are simply repositories to store information about all data items defined in DFDs. At the requirements stage, the data dictionary should at least define customer data items, to ensure that the customer and developers use the same definition and terminologies.
- **Entity-Relationship Diagrams:** Another tool for requirement specification is the entity-relationship diagram, often called an "***E-R diagram***." It is a detailed logical representation of the data for the organization and uses three main constructs i.e. data entities, relationships, and their associated attributes.

4. Software Requirement Validation:

- After requirement specifications developed, the requirements discussed in this document are validated. The user might demand illegal, impossible solution or experts may misinterpret the needs. Requirements can be the check against the following conditions -
- If they can practically implement
- If they are correct and as per the functionality and specially of software
- If there are any ambiguities
- If they are full
- If they can describe

Requirements Validation Techniques

- **Requirements reviews/inspections:** systematic manual analysis of the requirements.
- **Prototyping:** Using an executable model of the system to check requirements.
- **Test-case generation:** Developing tests for requirements to check testability.
- **Automated consistency analysis:** checking for the consistency of structured requirements descriptions.

5. Software Requirement management

A typical requirements management process complements the through these steps:

- Collect initial requirements from stakeholders
- Analyze requirements
- Define and record requirements
- Prioritize requirements
- Agree on and approve requirements
- Trace requirements to work items
- Query stakeholders after implementation on needed changes to requirements
- Utilize test management to verify and validate system requirements
- Assess impact of changes
- Revise requirements
- Document changes

Some of the benefits of requirements management include:

- Lower cost of development across the lifecycle
- Fewer defects
- Minimized risk for safety-critical products
- Faster delivery
- Reusability
- Traceability
- Requirements being tied to test cases
- Global configuration management

Requirements Elicitation

- Requirement elicitation can be done by communicating with stakeholders directly or by doing some research, experiments. The activities can be planned, unplanned, or both.
- **Planned activities** include workshops, experiments.
- **Unplanned activities** happen randomly. Prior notice is not required for such activities. **For example**, you directly go to the client site and start discussing the requirements however there was no specific agenda published in advance.

Following tasks are the part of elicitation:

- **Prepare for Elicitation:** The purpose here is to understand the elicitation activity scope, select the right techniques, and plan for appropriate resources.
- **Conduct Elicitation:** The purpose here is to explore and identify information related to change.
- **Confirm Elicitation Results:** In this step, the information gathered in the elicitation session is checked for accuracy.

Requirements Elicitation Techniques

- **There are several techniques available for elicitation, however, the commonly used techniques are explained below:**

1) Stakeholder Analysis

- Stakeholders can include team members, customers, any individual who is impacted by the project or it can be a supplier. Stakeholder analysis is done to identify the stakeholders who will be impacted by the system.

2) Brainstorming

- This technique is used to generate new ideas and find a solution for a specific issue. The members included for brainstorming can be domain experts, subject matter experts.

- **Brainstorming technique is used to answer the below questions:**
- What is the expectation of a system?
- What are the risk factors that affect the proposed system development and what to do to avoid that?
- What are the business and organization rules required to follow?
- What are the options available to resolve the current issues?
- What should we do so that this particular issue does not happen in the future?

3) Interview

Interview techniques should be used for building strong relationships between business analysts and stakeholders.

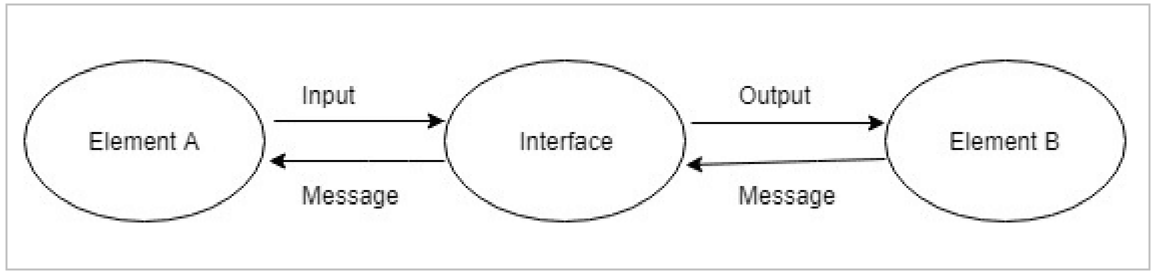
- If the interviewer has a predefined set of questions then it's called **a structured interview.**
- If the interviewer is not having any particular format or any specific questions then it's called an **unstructured interview**.

4) Document Analysis/Review

This technique is used to gather business information by reviewing/examining the available materials that describe the business environment. This analysis is helpful to validate the implementation of current solutions and is also helpful in understanding the business need.

- Document analysis includes reviewing the business plans, technical documents, problem reports, existing requirement documents, etc. This is useful when the plan is to update an existing system. This technique is useful for migration projects.
- 5) Focus Group
- The Focus group includes subject matter experts. The objective of this group is to discuss the topic and provide information. A moderator manages this session.
- The moderator should work with business analysts to analyze the results and provide findings to the stakeholders.
- If a product is under development and the discussion is required on that product then the result will be to update the existing requirement or you might get new requirements. If a product is ready to ship then the discussion will be on releasing the product.

- 6) Interface Analysis
- Interface analysis is used to review the system, people, and processes. This analysis is used to identify how the information is exchanged between the components. An Interface can be described as a connection between two components. **This is described in the below image:**



The interface analysis focus on the below questions:

1. Who will be using the interface?
2. What kind of data will be exchanged?
3. When will the data be exchanged?
4. How to implement the interface?
5. Why we need the interface? Can’t the task be completed without using the interface?

- 7) Observation
- The main objective of the observation session is to understand the activity, task, tools used, and events performed by others.
- The plan for observation ensures that all stakeholders are aware of the purpose of the observation session, they agree on the expected outcomes, and that the session meets their expectations
- During the session, the observer should record all the activities and the time taken to perform the work by others so that he/she can simulate the same. Observation can be either active or passive.
- **Active observation** is to ask questions and try to attempt the work that other persons are doing.
- **Passive observation** is silent observation i.e. you sit with others and just observe how they are doing their work without interpreting them.

- 8) Prototyping
- Prototyping is used to identify missing or unspecified requirements. In this technique, frequent demos are given to the client by creating the prototypes so that client can get an idea of how the product will look like.
- 9) Joint Application Development (JAD)/ Requirement Workshops
- This technique is more process-oriented and formal as compared to other techniques. These are structured meetings involving end-users. This is used to define, clarify, and complete requirements.
- **This technique can be divided into the following categories:**
- **Formal Workshops:** These workshops are highly structured and are usually conducted with the selected group of stakeholders. The main focus of this workshop is to define, create, refine, and reach closure on business requirements.
- **Business Process Improvement Workshops:** These are less formal as compared to the above one. Here, existing business processes are analyzed and process improvements are identified.

- 10) Survey/Questionnaire
- For Survey/Questionnaire, a set of questions is given to stakeholders to quantify their thoughts. After collecting the responses from stakeholders, data is analyzed to identify the area of interest of stakeholders.
- Questions should be based on high priority risks. Questions should be direct and unambiguous. Once the survey is ready, notify the participants and remind them to participate.
- **Two types of questions can be used here:**
- **Open-Ended:** Respondent is given the freedom to provide answers in their own words rather than selecting from predefined responses. This is useful but at the same time, this is time-consuming as interpreting the responses is difficult.
- **Close Ended:** It includes a predefined set of answers for all the questions and the respondent has to choose from those answers. Questions can be multiple choice or can be ranked from not important to very important.

System Modeling

- System modeling is the process of developing abstract models of a system, with each model presenting a different view or perspective of that system.
- most models use some kind of graphical notation representing a system, which is known as Unified Modeling Language (UML)

Benefits of System Modeling

1. Ease project management tasks.

2. Can provide complete views of a system, as well as detailed views of subsystems.

3. Clarify structures and relationships.

4. Offer a communication framework for ideas within and between teams. Can generate new ideas and possibilities.

5. Allow quality assurance and testing scenarios to be generated.

6. Platform independent.

UML Diagram Types

- The System Perspectives (last slide) are modeled with diagrams
- Activity diagrams, which show the activities involved in a process or in data processing .
- Use case diagrams, which show the interactions between a system and its environment.
- Sequence diagrams, which show interactions between actors and the system and between system components.
- Class diagrams, which show the object classes in the system and the associations between these classes.
- State diagrams, which show how the system reacts to internal and external events.

Types of Models

1. Context models
2. Interaction models
3. Structural models
4. Behavioral models
5. Model-driven engineering

1. Context Models

❿ Context models are used to illustrate the operational context of a system -

❿ They show what lies outside the system boundaries.

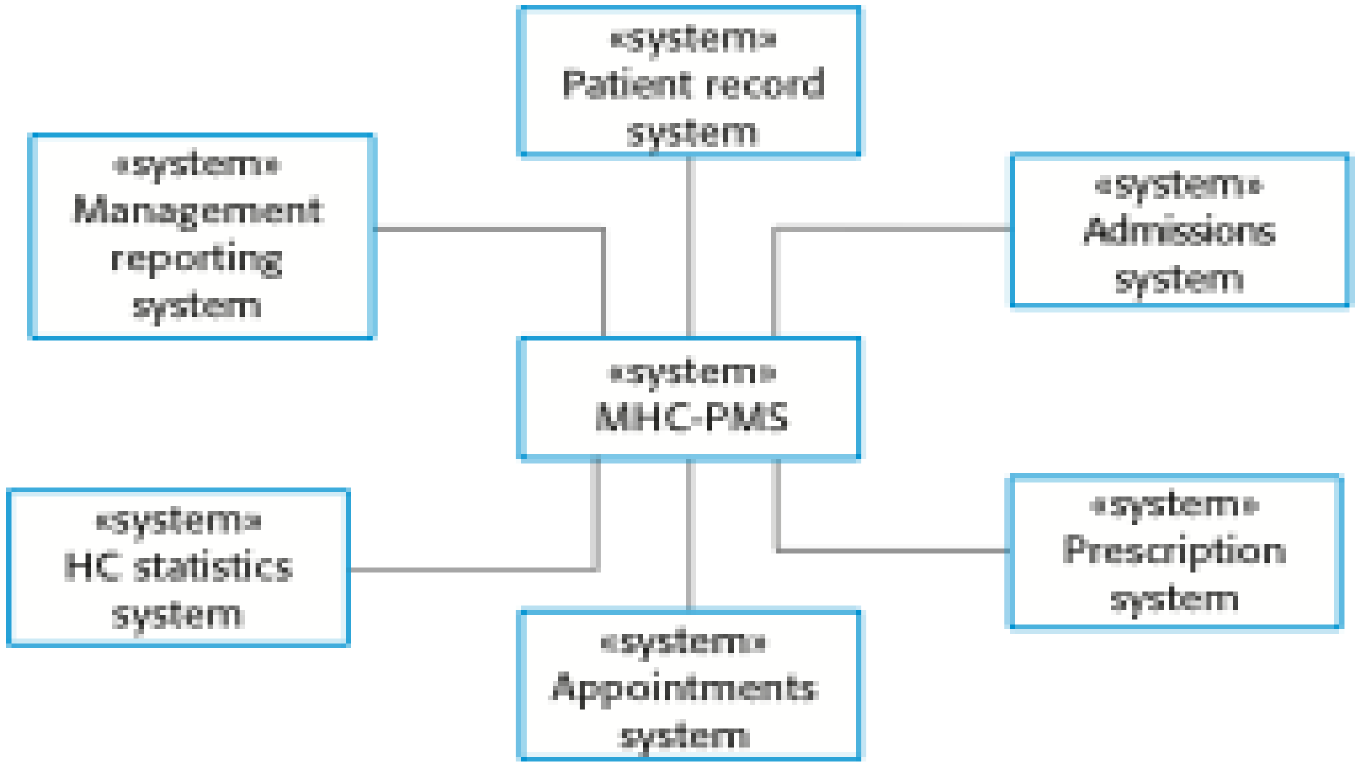
❿ Social and organizational concerns may affect the decision on where to position system boundaries.

❿ Architectural models show the system and its relationship with other systems.

System Boundaries

- System boundaries are established to define what is inside and what is outside the system.
  - They show other systems that are used or depend on the system being developed.
- The position of the system boundary has a profound effect on the system requirements.
- Defining a system boundary is a political judgment
  - There may be pressures to develop system boundaries that increase / decrease the influence or workload of different parts of an organization.

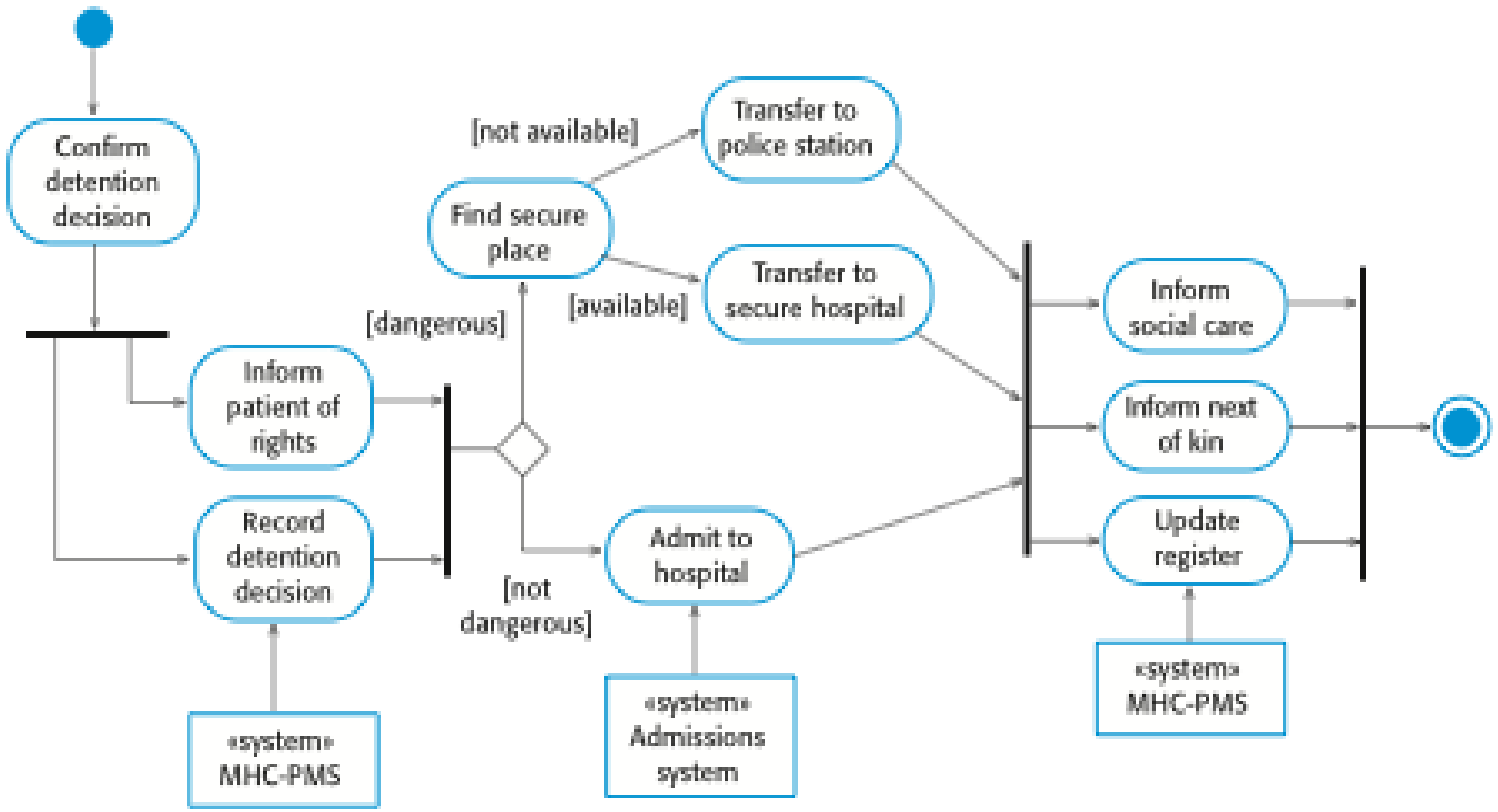
The Context of the MHC-PMS


«system» Patient record system
«system» Management reporting system
«system» Admissions system
«system» MHC-PMS
«system» HC statistics system
«system» Prescription system
«system» Appointments system

Process Perspective

- Context models simply show the other systems in the environment, not how the system being developed is used in that environment.
- Process models reveal how the system being developed is used in broader business processes.
- How it 'works' Detailed.
- UML activity diagrams may be used to define business process models.

Process Model of Involuntary Detention



2. Interaction Models

- ➓Modeling user interaction is used to identify user requirements.
- ➓We see structural connections and dynamic (behavioral) interactions.
- ➓We do this with graphical models.

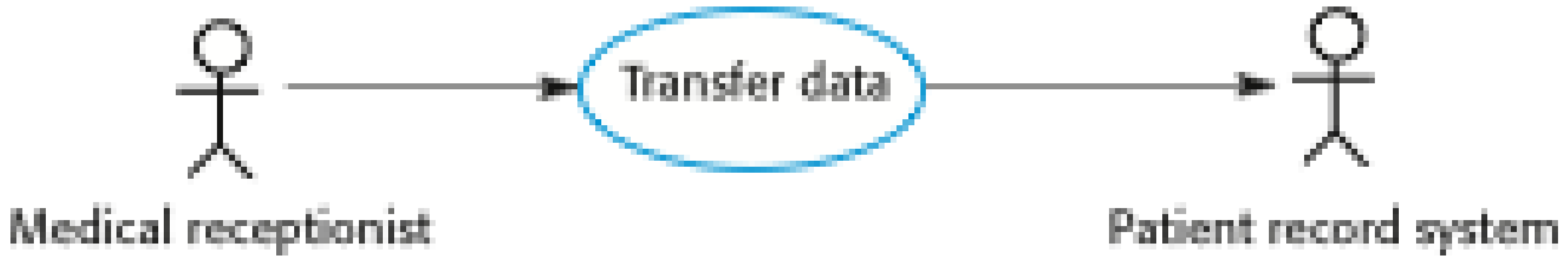
- ➓Use case diagrams and sequence diagrams may be used for interaction modeling.
- ➓These are the most popular modeling mechanisms

Use Case Modeling (Interaction Model)

- Use cases were developed originally to support requirements elicitation and now incorporated into the UML.
- Each use case represents a discrete task that involves external interaction with a system.
- Actors in a use case may be people, devices, or other systems.
- Represented diagramatically to provide an overview of the use case and in a more detailed textual form.

Transfer-data Use Case Diagram (graphical model)

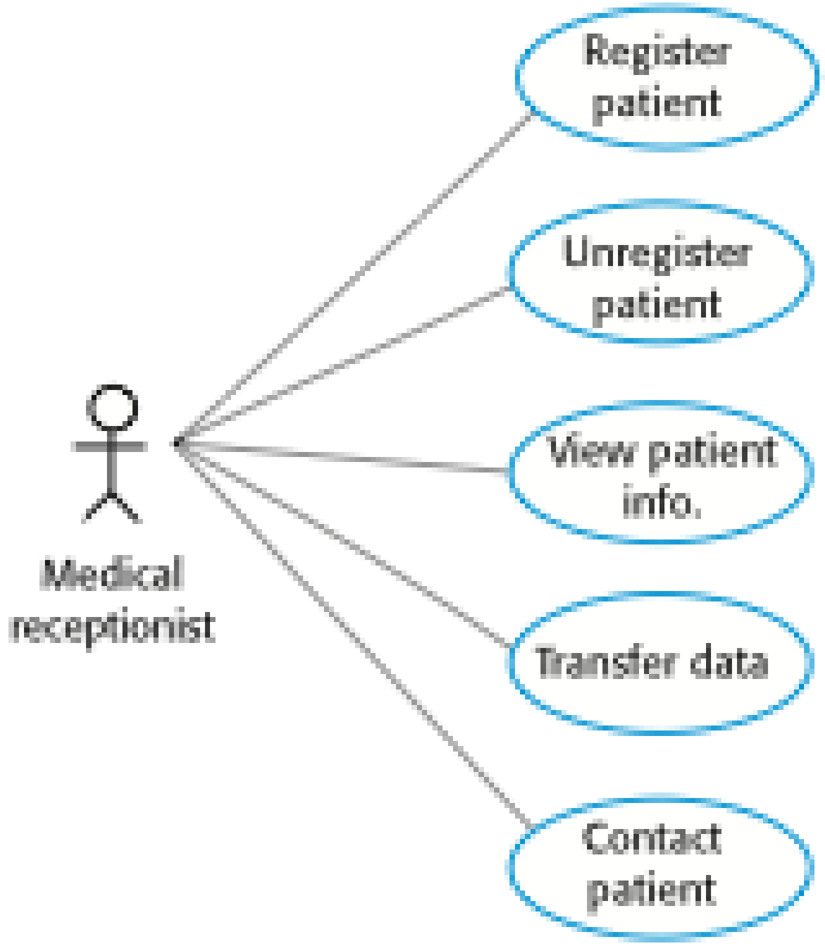
- A use case in the MHC-PMS



Tabular Description of the ‘Transfer data’ use-case

MHC-PMS: Transfer data	
Actors	Medical receptionist, patient records system (PRS)
Description	A receptionist may transfer data from the MHC-PMS to a general patient record database that is maintained by a health authority. The information transferred may either be updated personal information (address, phone number, etc.) or a summary of the patient’s diagnosis and treatment.
Data	Patient’s personal information, treatment summary
Stimulus	User command issued by medical receptionist
Response	Confirmation that PRS has been updated
Comments	The receptionist must have appropriate security permissions to access the patient information and the PRS.

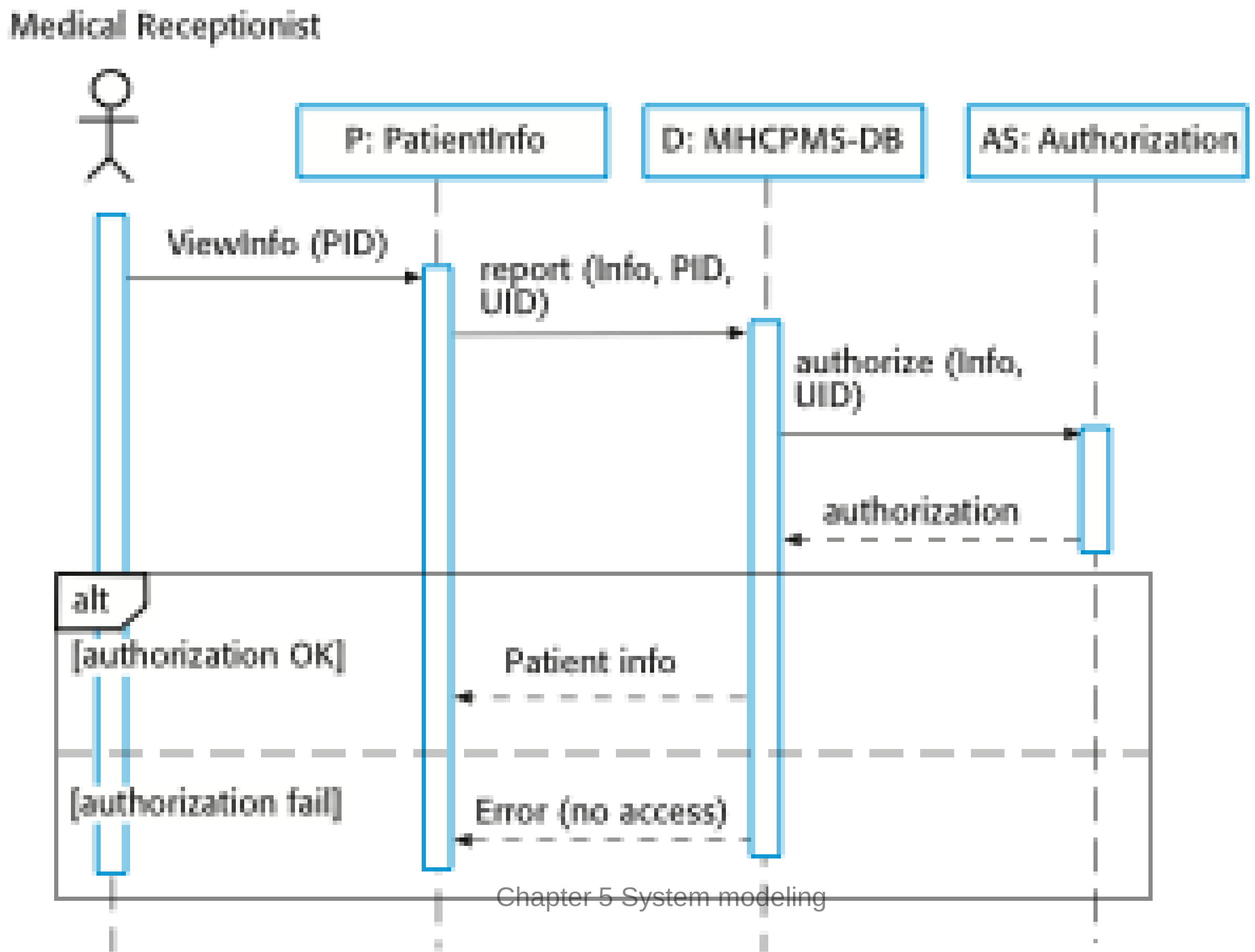
Use Cases in the MHC-PMS involving the role 'Medical Receptionist' (only showing one actor here)



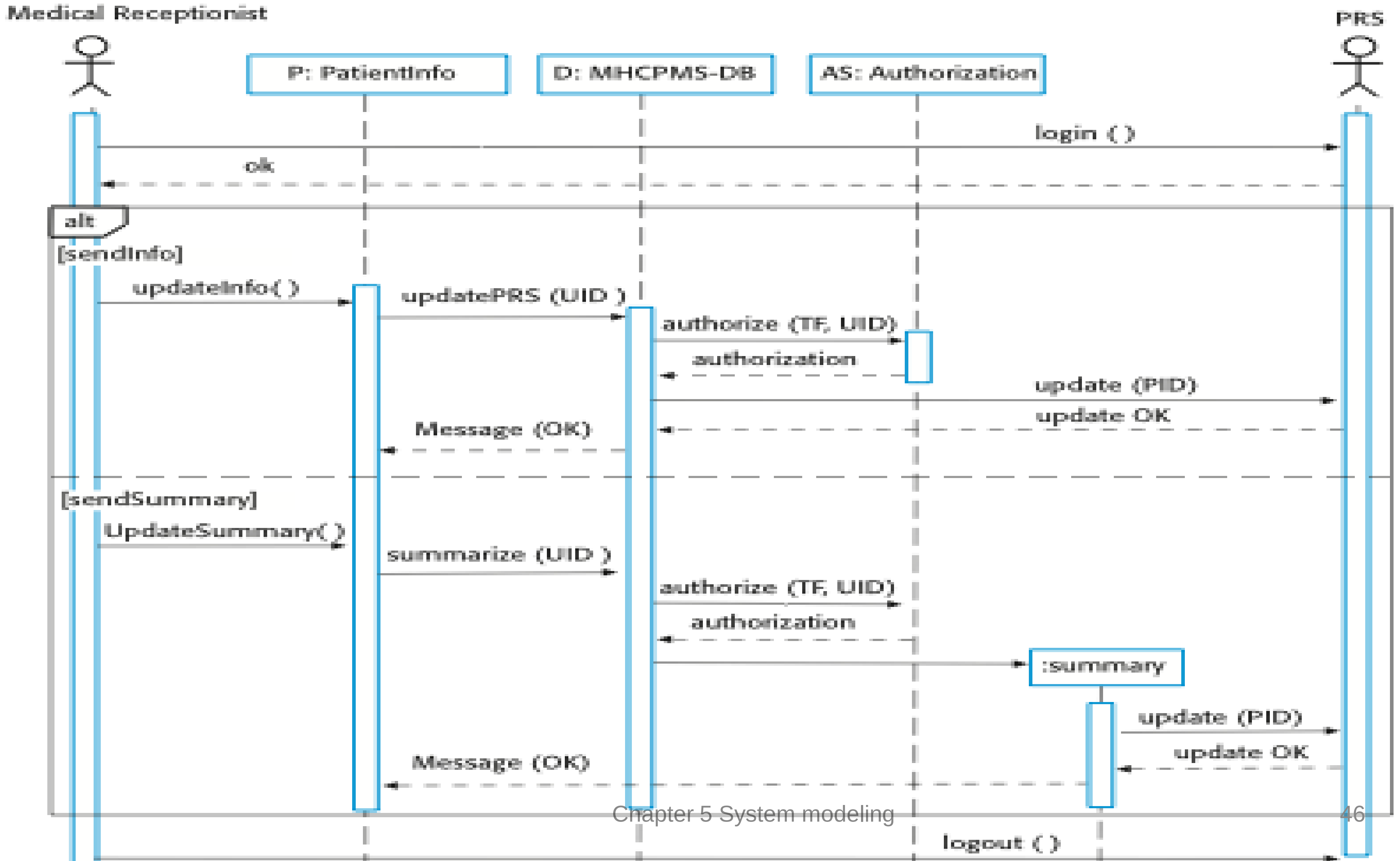
Sequence Diagrams (Interaction Model)

- Sequence diagrams are part of the UML and are used to model the interactions between the actors and the objects within a system.
- A sequence diagram shows the sequence of interactions that take place during a particular use case or use case instance.
- The objects and actors involved are listed along the top of the diagram, with a dotted line drawn vertically from these.
- Interactions between objects are indicated by annotated arrows.

Sequence diagram for View Patient Information



Sequence diagram for Transfer Data



Key points

- A model is an abstract view of a system that ignores system details. Complementary system models can be developed to show the system’s context, interactions, structure and behavior.

- Context models show how a system that is being modeled is positioned in an environment with other systems and processes.

- Use case diagrams and sequence diagrams are used to describe the interactions between users and systems in the system being designed. Use cases describe interactions between a system and external actors; sequence diagrams add more information to these by showing interactions between system objects.

Requirements Specification and Requirement Validation

- For requirement specification write(SRS)

Once the SRS completed validation begins

Requirement Validation

- The development of software begins once the requirements document is ‘ready’
- Ensure that the requirements specification (SRS)contains no errors and that it specifies the user’s requirements correctly.
- If errors present in the SRS will adversely affect the cost if they are detected later in the development process or when the software is delivered to the user.

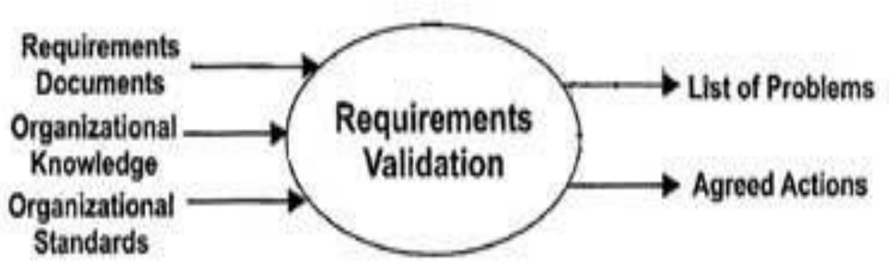
- So it is desirable to detect errors in the requirements before the design and development of the software begins.
- In the validation phase the requirements are examined for consistency, omissions, and ambiguity.
- The basic objective is to ensure that the SRS reflects the actual requirements accurately and clearly.
- objectives of the requirements document are given below.

1. To certify that the SRS contains an acceptable description of the system to be implemented
2. To ensure that the actual requirements of the system are reflected in the SRS
3. To check the requirements document for completeness, accuracy, consistency, requirement conflict‘, conformance to standards and technical errors.

- Requirements validation studies the ‘final draft’ of the requirements document while requirements analysis studies the ‘raw requirements’ from the system stakeholders (users).

- **Requirements validation:** Have we got the requirements right?
- **Requirements analysis:** Have we got the right requirements?

- The requirements document should be formulated and organized according to the standards of the organization.
- The **organizational standards are** specified standards followed by the organization according to which the system is to be developed.



Requirements Validation

The **agreed actions** is a list that displays the actions to be performed to resolve the problems depicted in the problem list.

Requirements Review

- Requirements validation determines whether the requirements are substantial to design the system.

The problems encountered during requirements validation are listed below.

1. Unclear stated requirements
2. Conflicting requirements are not detected during requirements analysis
3. Errors in the requirements elicitation and analysis
4. Lack of conformance to quality standards.

- To avoid the problems stated above, a **requirements review** is conducted, which consists of a review team that performs a systematic analysis of the requirements.
- The review team consists of software engineers, users, and other stakeholders who examine the specification to ensure that the problems associated with consistency, omissions, and errors are detected and corrected.

- In the review meeting check list are:

1. Is the initial state of the system defined?
2. Is there a conflict between one requirement and the other?
3. Are all requirements specified at the appropriate level of abstraction?
4. Is the requirement necessary or does it represent an add-on feature that may not be essentially implemented?
5. Is the requirement bounded and has a clear defined meaning?
6. Is each requirement feasible in the technical environment where the product or system is to be used?
7. Is testing possible once the requirement is implemented?
8. Are requirements associated with performance, behavior, and operational characteristics clearly stated?
9. Are requirements patterns used to simplify the requirements model?
10. Are the requirements consistent with the overall objective specified for the system/product?
11. Have all hardware resources been defined?
12. Is the provision for possible future modifications specified?
13. Are functions included as desired by the user (and stakeholder)?
14. Can the requirements be implemented in the available budget and technology?
15. Are the resources of requirements or any system model (created) stated clearly?

Other Requirements Validation Techniques

- A number of other requirements validation techniques are used either individually or in conjunction with other techniques to check the entire system or parts of the system. The selection of the validation technique depends on the appropriateness and the size of the system to be developed. Some of these techniques are listed below.

1. **Test case generation:** The requirements specified in the SRS document should be testable. The test in the validation process can reveal problems in the requirement. In some cases test becomes difficult to design, which implies that the requirement is difficult to implement and requires improvement.
2. **Automated consistency analysis:** If the requirements are expressed in the form of structured or formal notations, then CASE tools can be used to check the consistency of the system. A requirements database is created using a CASE tool that checks the entire requirements in the database using rules of method or notation. The report of all inconsistencies is identified and managed.
3. **Prototyping:** Prototyping is normally used for validating and eliciting new requirements of the system. This helps to interpret assumptions and provide an appropriate feedback about the requirements to the user. For example, if users have approved a prototype, which consists of graphical user interface, then the user interface can be considered validated.

Requirements Elicitation techniques

https://www.softwaretestinghelp.com/requirements-elicitation-techniques/

Requirements management in Agile

- Agile requirements management is built on the principle of flexibility, so any potential challenges are identified and resolved much earlier in the process, minimizing expensive and costly rework. A few benefits of the agile approach include:
- **Improved product design and delivery.** A recent report suggests that best-in-class RM solutions can significantly improve product design and delivery for agile development teams. "In the face of increasing regulations, connected products for the internet of things (IoT), and scaling Agile practices, AD&D [application development and delivery] leaders long for something to bring traceability and auditability to their processes without sacrificing speed."
- **Improved traceability.** The right RM solution can enhance development transparency through traceability. Traceability empowers teams to perform impact analysis more readily, which is critical to product development.
- **Achieve quicker time to market.** Teams are facing more complexity and pressure to comply with industry regulations, and need to measure customer value to search, track and connect interdependent requirements. Achieving faster time to market requires that teams collaborate faster and more effectively, working to build traceability requirements and test case

Designing with greater flexibility through the agile requirements management lifecycle

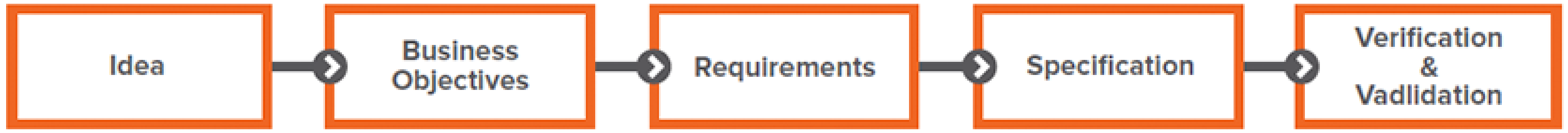
The agile requirements management lifecycle is focused on clearly defining a project's scope, so that you better understand what needs to happen to meet the desired end goal. It provides a high-level understanding of business goals, and outlines what is needed for the project to be a success.

Consider taking the following steps:

- **Understand user stories.** User stories give you powerful information about the problem you're trying to solve. A user story is a quick description of everyday use cases and might include a few sentences about how the user expects the product to perform. A template might be something like this: "As a [role], I need [product] to do [goal of the software] so that I can [benefit of the product]."
- **Outline the most important requirements.** Identify what requirements are most essential based on the high-level business strategy. These requirements may be supported by user stories, functional requirements and more based on the specific client goals.
- **Transform to product features.** This stage is about fine-tuning and translating the details that you've gathered into product features. The development team collaborates to ensure that any requirements are easily understood by anyone working to implement them. User stories are linked to features and tasks, so that developers understand what they need to do – but also why they are doing it.

- **Collaboration and Buy-In**
- It’s difficult to get a company to agree on requirements, particularly for large projects with many stakeholders. In practice, it’s not necessary to achieve consensus through compromise. It’s more important to have team buy-in (before or after management approves the project) so the development process can move forward. With buy-in, the team backs the best solution, makes a smart decision and does what is necessary to move forward with the requirements management process.
- Team collaboration is key to establishing good requirements. Collaborative teams work hard to make sure everyone has a stake in the project and provides feedback. When there is a commitment and understanding of project goals, team members tend to support other’s decisions. It’s when developers, testers or other stakeholders feel “out of the loop” that communication issues arise, people get frustrated and projects get delayed.

- **Traceability and Change Management**
- Requirements traceability is a way to keep everyone in the loop. It organizes, documents and keeps track of all requirements, from initial idea to testing. A simple metaphor for traceability is connecting the dots to identify the relationships between items within a project. The following figure shows an example of a common downstream flow.



- Companies should be able to trace each requirement back to its original business objective throughout the development process, not merely after it’s complete. By tracing requirements, companies can identify the ripple effect changes have, see if they have completed a requirement and if they tested it properly. With traceability, and by managing changes effectively, managers gain the visibility to anticipate issues and ensure continuous quality.
- Traceability also makes sure the product meets all the vital requirements that come from different stakeholders. By tracing requirements, all team members stay connected to each other and to all interdependencies. And by managing changes well, a company can avoid scope creep — unplanned changes that occur when requirements are not clearly captured, understood and communicated. The benefit of good requirements is a clear understanding of the product and the scope involved. This leads to a better development schedule and budget, which prevents delays and cost overruns.

- **Quality Assurance**
- Getting requirements right the first time means better quality, faster development cycles and higher customer satisfaction with the product. Concise, specific requirements can help companies detect and solve problems earlier rather than later, when they are much more expensive to fix.
- Research has shown that project teams can eliminate 50 percent to 80 percent of project defects by effectively managing requirements. In addition, according to Borland Software (now Micro Focus), it can cost up to 100 times more to correct a defect later in the development process, after it's been coded, than when it's still in written form.
- By integrating requirements management best practices into their quality assurance process, companies can help teams increase efficiency and eliminate rework. According to the Carnegie Mellon Software Engineering Institute, 60 to 80 percent of the cost of software development is in rework. In other words, development teams are wasting the majority of their budgets on efforts they didn't perform correctly the first time.
- Requirements management best practices can appear to be a complex topic, but at its core, it's a simple concept. It helps teams answer the question: Does everyone — from business leaders to product managers and project leaders to developers, QA managers and testers — understand what is being built and why?
- When everyone is collaborating and has full context and visibility into the discussions, decisions and changes involved in product development, they maintain high quality and almost always ensure success.