Software Engineering

CSIT- 6th Semester

Prepared by: Er. Rupak Gyawali, 2024

Unit 4: Requirements Engineering

Concept of User and System Requirements; Functional and Non-Functional Requirements; Requirements Engineering Process; Requirements Elicitation; Requirements Specification; Requirements Validation; Requirements Change

Assignment:

Q.1 What are the various types of functional and non-functional requirements that are placed on the system? Explain with an example

System and Software Requirements:

What is a Requirement ?

- A requirement is a **detailed description** of what a system should do.
  - **Example:** Requirement: "The system should allow users to log in using their email address and a password."

    Detailed Description:
    - The login feature should validate the user's email address and password.
    - If the credentials are correct, the user should be granted access to their account.
    - If the credentials are incorrect, the system should display an error message and prompt the user to try again.
    - The system should enforce password rules (e.g., a minimum of 8 characters, including at least one number and one special character).
- Software Requirements Analysis is essential to ensure that the software meets the needs of the customer and does not fail in delivering what's expected.
- Data, functional, and behavioral requirements are gathered from the customer and refined to create a specification that can be used to design the system.
  - **Data Requirements:** The specific data the system needs to handle. Eg: The system must store user profiles, including name, email address, and date of birth.
  - **Functional Requirements:** What functions or tasks the system should perform. Eg: The system should allow users to reset their password through a 'Forgot Password' link.
  - **Behavioral Requirements:** How the system should behave or respond under certain conditions. Eg: If a user enters an incorrect password three times in a row, the system should temporarily lock the account for 15 minutes.
- It's important to review the requirements to make sure they are clear, complete, and consistent so that they can be used effectively to design the system.

Types of Requirements:

a) User Requirements: These are clear and simple descriptions, often written in natural language, that explain what the system should do and any limitations or rules it must follow. These requirements are created for the customer to understand how the system will work.

- Written for customer.
  - Eg: "The system should allow users to sign up, log in, and view their profile information. The system should be available 24/7 and work on both desktop and mobile devices."

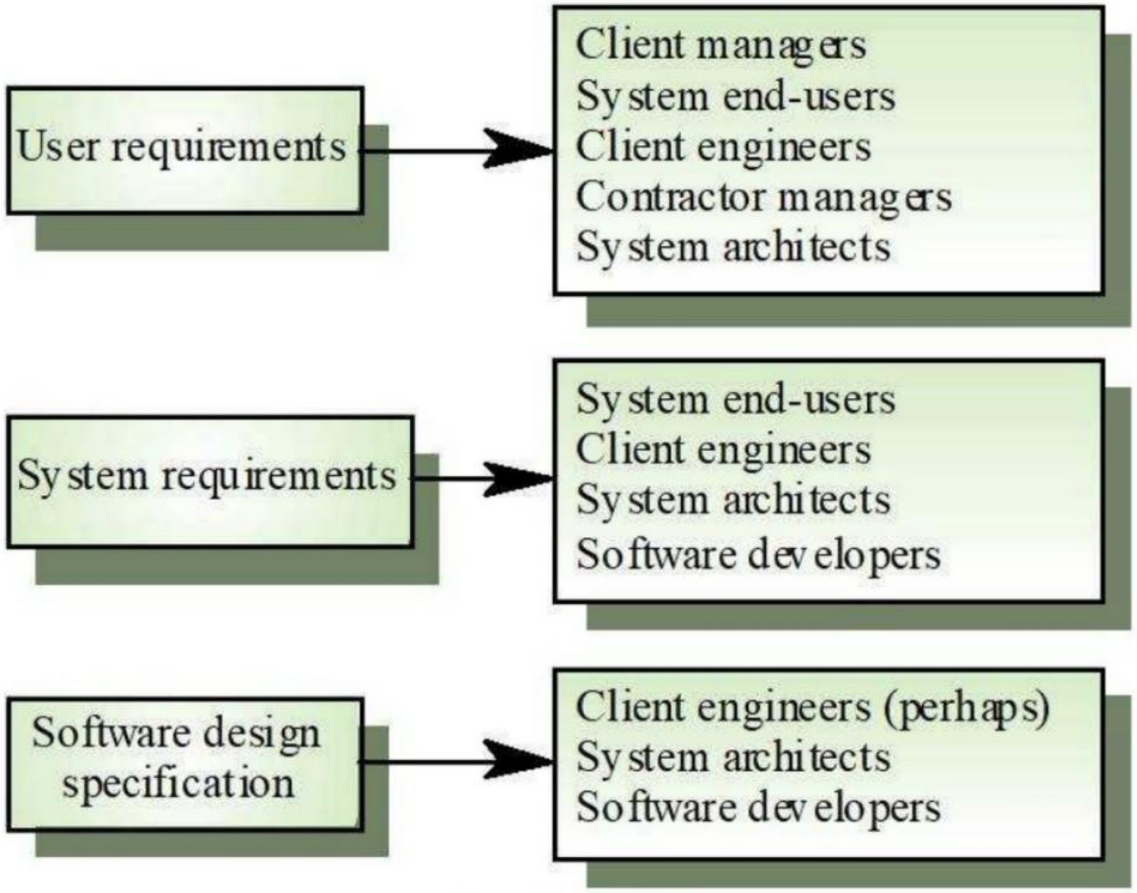
b) System Requirements: This is a detailed document that clearly describes what the system must do, including all the specific features and functions. It's like a formal agreement between the system buyer and the software developer, ensuring that everyone understands exactly what the system will deliver.

- Written as a contract between system buyer and system developer.
  - Eg: "The system must allow users to log in using their email and password, and it should encrypt all passwords using the SHA-256 algorithm. The system must handle up to 10,000 users at the same time and should generate a report of user activity every 24 hours."

c) Software Specification: This is a very detailed description of how the software should work, including technical details that developers need to design and build the system. It serves as a blueprint for the developers to follow during the implementation.

- Written for developers.
  - Eg: The login module must be implemented using a two-factor authentication system, with a secure API for password validation and a separate service for sending one-time passcodes.

Requirement Specification Readers:



User Requirements Specification

- **Readers**: System End Users, System Architect, Contractor etc.
- **Purpose**: To explain what the system will do in simple terms.

System Requirements Specification

- **Readers**: Customers, Software developer, System Architects etc.
- **Purpose**: To provide a detailed description of the system's features and functions.

Software Specification

- **Readers**: Developers, Tester, System Architect etc.
- **Purpose**: To give a technical guide on how to build the software.

Types of Requirements:

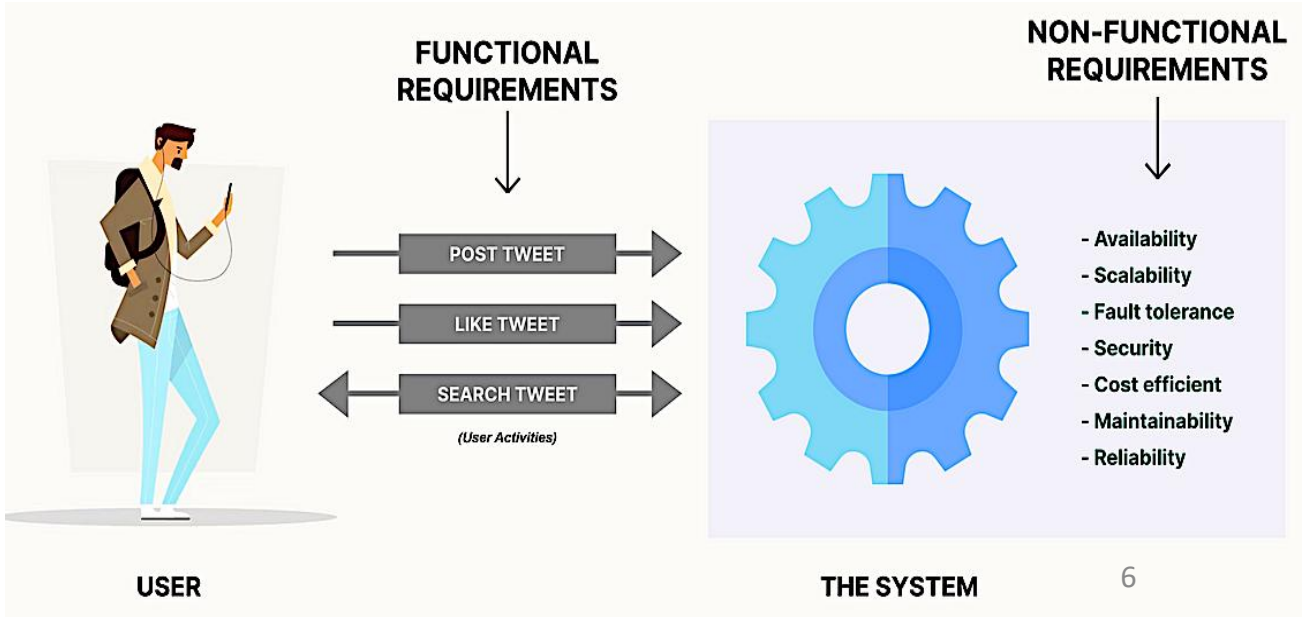
a) Functional and Non-Functional Requirements

Functional Requirements:

- Functional requirements describe what task the system does.
- **What They Are:** Descriptions of what the system should do, including how it should respond to inputs and behave in different situations.
- **Examples:** "The system should allow users to log in with their email and password" or "The system should send a confirmation email after a user registers."
- The functional requirements specification of a system should be both complete and consistent.
- Complete means covering every feature and service the user needs. Consistency means that the requirements should not contradict each other. However, for large and complex systems, achieving perfect completeness and consistency can be very challenging.

Non-Functional Requirements:

- **What They Are:** Constraints on how the system performs or operates, such as speed, reliability, and standards.
- **Examples:** "The system must load within 2 seconds" or "The system should be able to handle up to 10,000 users at the same time."
- Non-functional requirements describe how well system does the task.
- Failing to meet a non-functional requirement can mean that the whole system is unusable. For example, if an aircraft system does not meet its reliability requirements, it will not be certified as safe for operation.



Types of Requirements:

a) Functional and Non-Functional Requirements

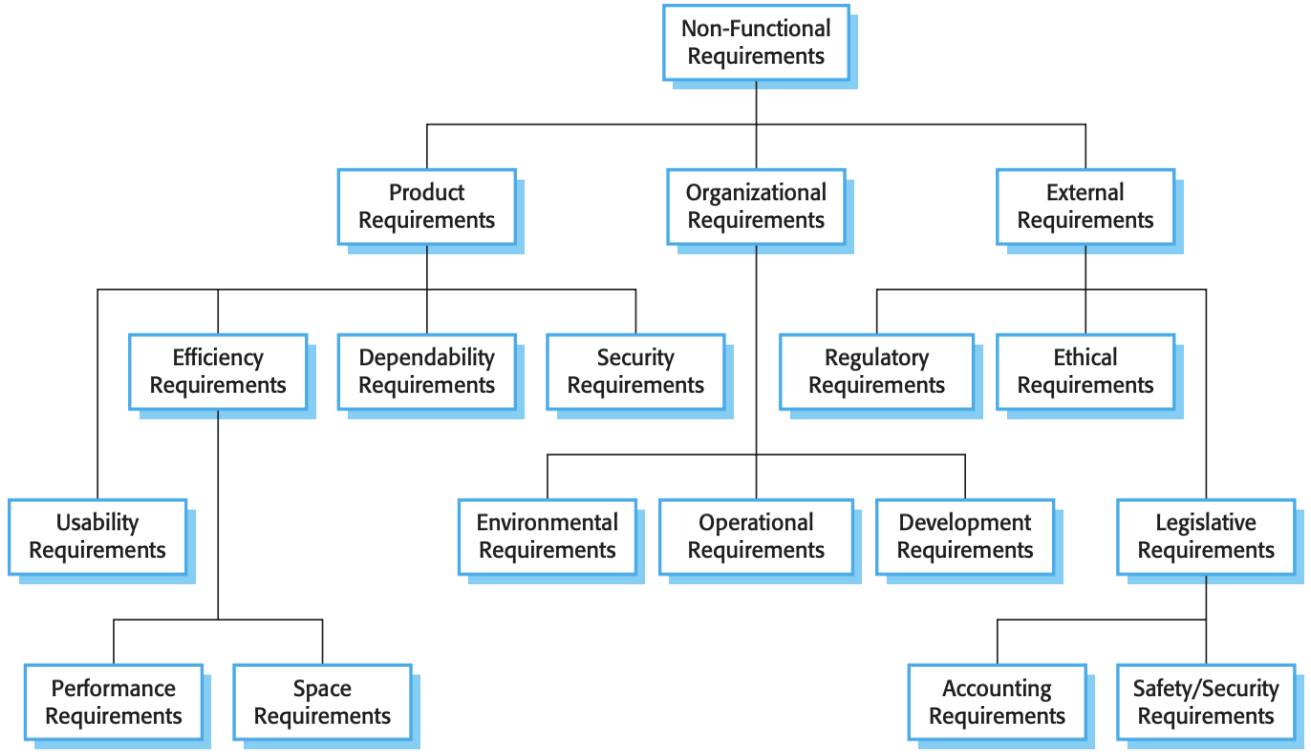
Aspect	Functional Requirements	Non-Functional Requirements
Definition	Describes what the system should do, i.e., specific functionality or tasks.	Describes how the system should perform, i.e., system attributes or quality.
Purpose	Focuses on the behavior and features of the system.	Focuses on the performance, usability, and other quality attributes.
Deals with	User authentication, data input/output, transaction processing.	Scalability, security, response time, reliability, maintainability.
Focus	Directly related to user and business requirements.	Focuses on user experience and system performance.
Evaluation	Can be tested through functional testing (e.g., unit or integration tests).	Evaluated through performance testing, security testing, and usability testing.
Example	"The system should allow users to log in."	"The system should respond within 2 seconds."
User Perspective	Directly visible to users (e.g., a search bar).	Indirectly affects users (e.g., speed, reliability).
Specified by	User Specifies Functional Requirements	Technical people (Technical leader, S/W developer) specifies Non-functional Requirements.
Essentiality	They are mandatory because they define the core functionalities the system must perform. Mandatory → The system must allow users to reset their passwords.	They are desirable because they enhance the quality of the system (e.g., speed, security, usability). Desirable → The system should reset the password within 2 seconds of the request.

Types of Requirements:

a) Functional and Non-Functional Requirements

Classification/ Types of Non-Functional Requirements:

- **Product Requirements:** These define how the software should behave, such as how fast it should run, how reliable it should be, how secure it needs to be, and how easy it is to use.
  - Example: "The system must process transactions in under 2 seconds."
- **Organizational Requirements:** These are based on the rules and practices of the customer's or developer's organization, such as which programming language to use, how the system will be operated, or what standards to follow.
  - Example: "The software must be developed using Python and follow company coding standards."
- **External Requirements:** These come from outside the system, like legal, regulatory, or ethical obligations that the system must meet.
  - Example: "The system must comply with data protection laws to be approved for use."



Non-functional requirements are more critical than functional requirements. Users can usually adapt if a specific function isn't perfect *(If a messaging app's emoji feature is missing some emojis, users can still communicate using text.*
), but failing to meet a non-functional requirement can make the system unusable. For example:

- If an aircraft system isn't reliable, it won't be certified as safe.

Types of Requirements:

b) Domain Requirement:

- Domain Requirements are specific needs that come from the industry or field where the software will be used.
- These can include both functional and non-functional requirements and are often related to standards or practices specific to that industry.
- If domain requirements are not satisfied, the system may be unworkable.
- Domain requirements typically arise in military, medical and financial industry.
- One example of a domain requirement is for software in medical equipment: The software must be developed in accordance with IEC 60601 regarding the basic safety and performance for medical electrical equipment.
- Another example: The landing gear of an aircraft must work safely on different surfaces like dirt, asphalt, water, and snow. The software should handle these conditions to ensure safe landings.

<u>Two problems with domain requirements:</u>

- **Understandability:** Requirements are written in the language specific to the field, which software engineers might not fully understand.
  - Example: In a medical software project, a requirement might state, "The system must calculate the APGAR score." If the software engineers are unfamiliar with medical terms, they might not understand what the APGAR score is or how it's calculated.
- **Implicitness:** Experts in the field are so familiar with the requirements that they may forget to clearly explain them, assuming everyone knows what they mean.
  - Example: In aviation, experts might expect the software to handle different surfaces like dirt or asphalt automatically, but they don’t mention it because they assume everyone already knows this. This could lead to the software not being properly designed for different surfaces.

Types of Requirements:

c) User Requirements:

- These are clear and simple descriptions, often written in natural language, that explain what the system should do and any limitations or rules it must follow. These requirements are created for the customer to understand how the system will work.
- User Requirement documents may include diagrams or tables.
- User Requirements should describe Functional and Non-functional Requirements.
- User Requirements should be understandable by system users who don't have detail technical knowledge.
- User Requirements is written for customer.

Example: User Requirements for Library Management System

1. Functional Requirements

- **User Login**: Users log in with their library ID and password (min. 6 characters).
- **Search for Books**: Users can search by title, author, or ISBN, with results in 2 seconds.
- **Book Borrowing**: Users can borrow up to 5 books at a time.
- **Return Books**: Users return books, and the system updates their borrowed book count.

2. Non-Functional Requirements

- **Performance**: Supports 100 simultaneous users without slowdown.
- **Security**: Encrypts all user passwords.
- **Usability**: Simple interface, max 5 clicks to borrow a book.
- **Availability**: System is available 24/7 with less than 1% downtime.

Diagram: Borrowing a Book Workflow

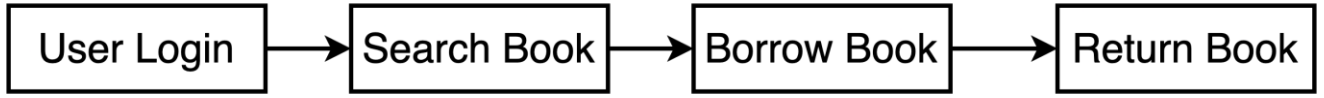


Table: User Action and System Response

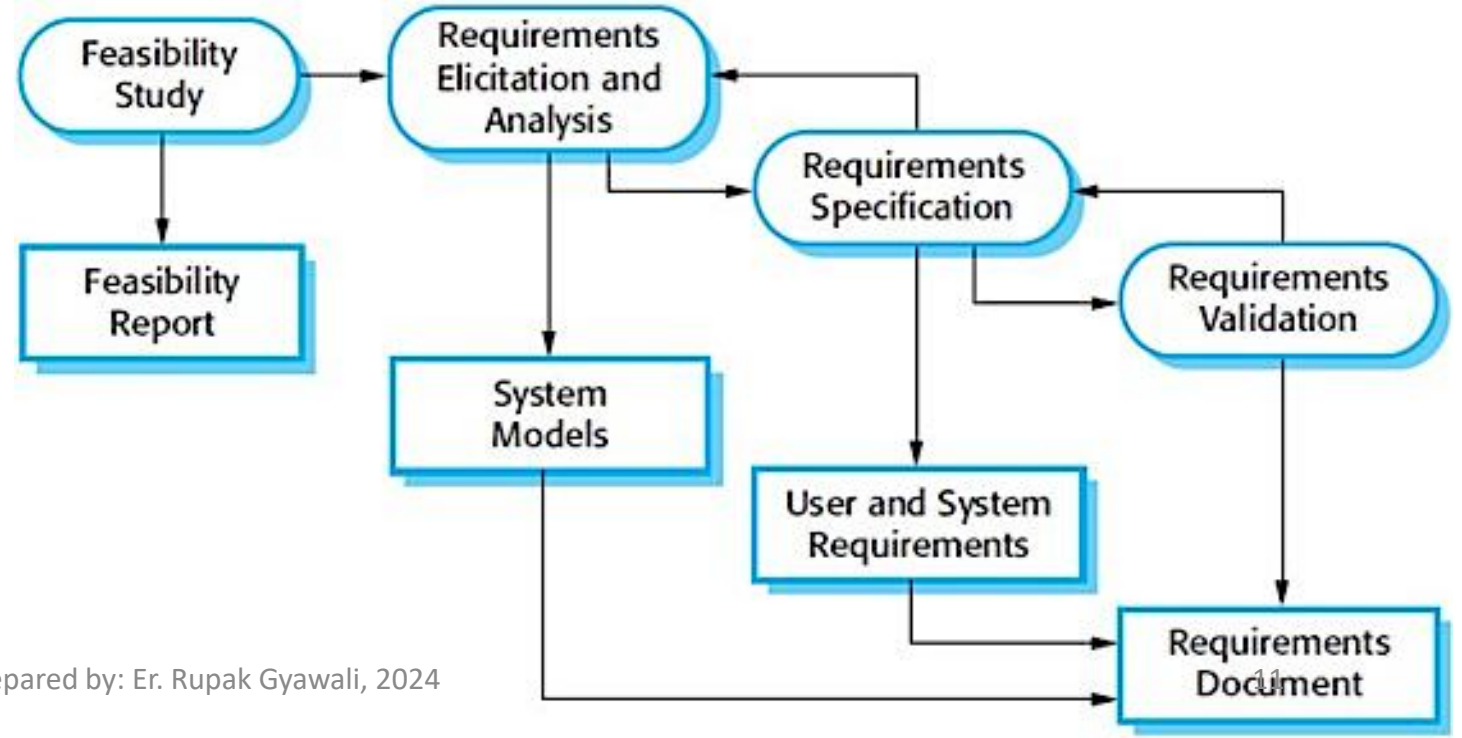
User Action	System Response
User logs in	Verifies credentials.
Searches for a book	Displays results in 2 seconds.
Borrows a book	Checks availability, updates count.
Returns a book	Updates account, book availability.

Requirement Engineering:

- Requirement Engineering is the process of gathering, documenting, and managing the needs and requirements for a system.
- It helps in understanding what the customer wants, analyzing whether it's possible, agreeing on a solution, and clearly specifying what the system should do.
- This process ensures that the system is built according to these well-defined requirements and any constraints.
- It's a systematic approach that uses proven methods and tools to describe how the system should behave and what limitations it must follow.
    - Proven methods and tools like: Interviewing, Surveying, Brainstorming, Prototyping, UseCases, UML Diagrams etc.

Requirement Engineering Processes/ Activities in Requirement Engineering:

1. Feasibility Study
2. Requirement Elicitation
3. Requirement Analysis
4. Requirement Documentation
5. Requirement Validation
6. Requirement Management



Requirement Engineering Processes/ Activities in Requirement Engineering:

1) Feasibility Study:

- To determine if developing the software is worthwhile, meets user needs, can adapt to changes, and follows standards.
- A feasibility study is a short, focused study that should take place early in the RE process. It should answer three key questions: a) does the system contribute to the overall objectives of the organization? b) can the system be implemented within schedule and budget using current technology? and c) can the system be integrated with other systems that are used? If the answer to any of these questions is no, you should probably not go ahead with the project.

Types of Feasibility:

- **Technical Feasibility:** Checks whether the current technology can meet customer needs within the time and budget available.
  - Example: A company wants to develop a mobile app with real-time video streaming. The Technical feasibility study checks if their current servers and internet speed can support this feature without needing expensive upgrades.
- **Operational Feasibility:** Checks If the software can effectively solve business problems and meet customer needs.
  - Example: A hospital plans to implement an electronic health records (EHR) system. The Operational feasibility study examines if a new electronic health records (EHR) system will fit with their current processes, improve record management, and be easy for staff to use.
- **Economic Feasibility:** Checks If the software will be financially beneficial for the organization.
  - Example: A startup is considering developing an online shopping platform. The Economic feasibility study calculates whether the revenue from the platform will cover the development and operational costs, and eventually lead to profits.

Requirement Engineering Processes/ Activities in Requirement Engineering:

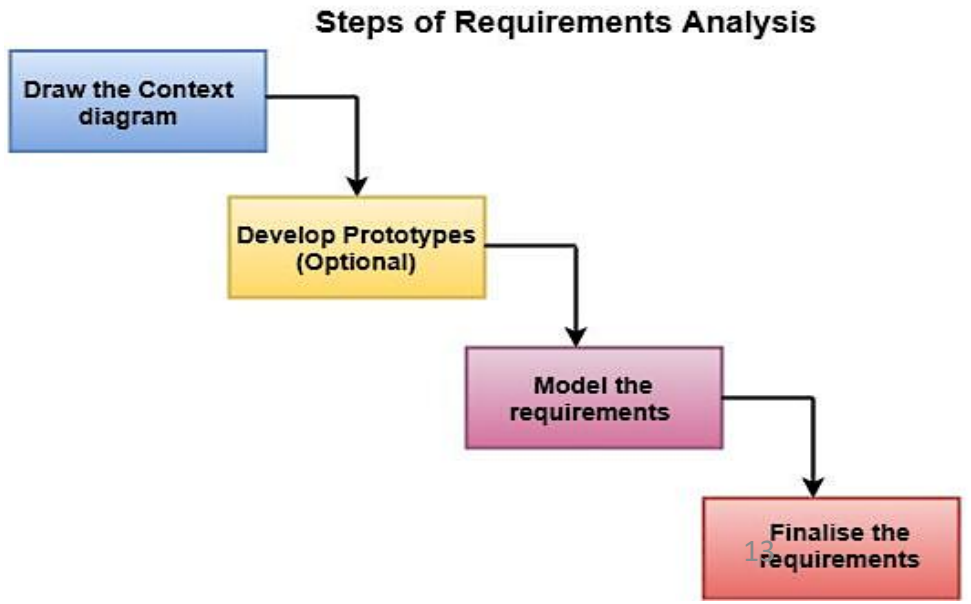
2) Requirement Elicitation and Analysis:

Requirement Analysis: Requirement analysis is figuring out what a project needs to do and what features it must have. It involves understanding and documenting what the users want and need from the system or product.

- Example: Imagine you're creating a new online shopping website. Requirement analysis would involve finding out what users need, such as easy navigation, a search function, secure payment options, and a way to track their orders. You'd then use these insights to design and build the website to meet those needs.

Steps in Requirement Analysis:

- **Draw the Context Diagram:** Create a visual map showing how the system interacts with external entities (e.g., users, other systems).
  - Example: For an online shopping app, the context diagram might show interactions between the app (system), users (customers), a payment gateway, and a shipping service.
- **Develop Prototype:** Build a basic version of the system to gather feedback and make improvements.
  - Example: Create a simple, working version of the shopping app with basic features to let users try it out and provide feedback.
- **Model the Requirement:** Create detailed diagrams showing how the system will work and handle data.
  - Example: Use DFDs/ flowcharts to illustrate how users will navigate through the shopping app, from browsing products to checking out.
- **Finalize the Requirement:** Confirm and finalize the list of features and functions based on feedback.
  - Example: After testing the prototype and reviewing the models, finalize the list of features and functions for the shopping app, making sure all stakeholders agree with them.



Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation/ Requirement Discovery:

- Requirement Elicitation is the process of finding out what a system needs to do by asking users, customers, and other stakeholders.

Requirements Elicitation Activities:

- **Understand the context:** Learn about the general area where the system will be used.
  - Example: If you're designing a library management system, first learn how libraries operate, including how books are checked out, returned, and cataloged.
- **Identify specific problems:** Know the exact issues the system should solve.
  - Example: Discover that the current library system struggles with tracking overdue books, causing confusion among staff and patrons.
- **Understand interactions:** Figure out how the system will work with other systems or requirements.
  - Example: Determine that the new system needs to integrate with the library's existing barcode scanner and email notification system.
- **Investigate user needs:** Look into what users really need from the system.
  - Example: Ask librarians and patrons what they need, such as easy book search features for patrons and a quick way for librarians to update the catalog.
- **Define constraints:** Identify any limits or restrictions on how the system can be developed.
  - Example: Identify that the system must be developed within a tight budget and should be compatible with older computers already in use at the library.

Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation:

Stages in Requirement Elicitation and Analysis:

- **Requirements Discovery:** Also called Requirement Elicitation. Talk to stakeholders to find out what they need and discover any domain-specific requirements.
  - Example: For a school management system, talk to teachers, students, and administrators to find out they need features like grade tracking, attendance monitoring, and communication tools.
- **Requirements Classification and Organization:** Group similar requirements together and organize them into clear sections.
  - Example: Organize the requirements into categories like "Student Management," "Teacher Tools," and "Communication Features."
- **Prioritization and Negotiation:** Decide which requirements are most important and resolve any conflicts between them.
  - Example: Determine that grade tracking is the most critical feature, while attendance monitoring can be added later. If some teachers want detailed reports and others want simple summaries, agree to include both options.
- **Requirements Specification:** Write down the final list of requirements to use in the next stages of development.
  - Example: Document the features, like "The system will allow teachers to enter grades, monitor attendance, and send messages to students," to ensure everyone knows what will be developed.

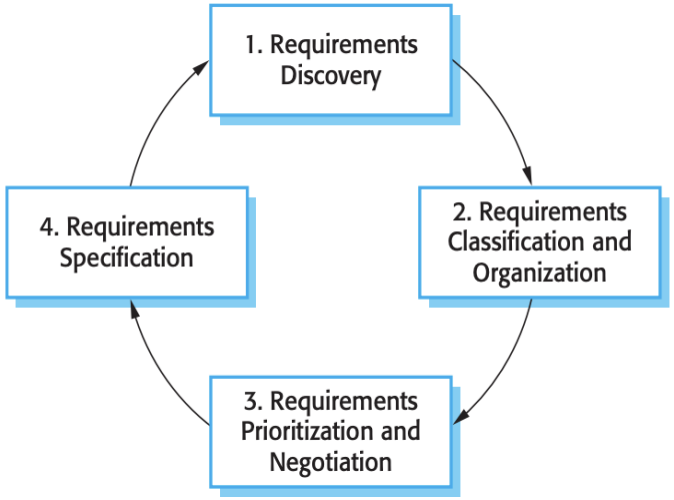


Figure: The requirements elicitation and analysis process

Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation:

Problems of Requirement Elicitation/ Eliciting and understanding requirements from system stakeholders is a difficult process :

- Stakeholders don't know what they really want, making it hard to pinpoint requirements.
- Stakeholders express requirements in their own terms, leading to potential misunderstandings.
- Different stakeholders may have conflicting requirements.
- Organizational and political factors may influence the system requirements.
- As the analysis progresses, the understanding of requirements may evolve, leading to changes in the initial requirements.
- New stakeholders may emerge and the business environment change and can introduce new needs or alter existing ones.

Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation Methods:

a) View Point:

- A viewpoint is a way to gather and organize requirements from a specific group of people who share similar interests or roles, like end-users or managers.
- Each viewpoint includes the requirements that group needs.
- Requirements can come from different sources, like the people who will use the system (stakeholders), the industry or field where the system will be used (application domain), and other systems that the new system will need to interact with. All of these need to be considered when gathering requirements.
- These different sources can be viewed as different "viewpoints," each providing a piece of the overall requirements. Each viewpoint sees the problem from its own angle, but often they share some common requirements. You can use these viewpoints to help organize both the process of discovering requirements and the way you document them.

Example of Viewpoint:

Imagine you're developing a new software system for a hospital.

- **End-User Viewpoint (Doctors and Nurses):**
  - **Example:** Doctors and nurses need the system to quickly access patient records, update treatment plans, and schedule appointments.
- **Management Viewpoint (Hospital Administrators):**
  - **Example:** Hospital administrators need the system to generate reports on patient care, monitor staff performance, and manage billing.
- **Technical Viewpoint (IT Department):**
  - **Example:** The IT department needs the system to be secure, easy to integrate with existing hospital software, and simple to maintain.

Each of these viewpoints provides different requirements based on the unique needs and perspectives of the group they represent.
However, they might all agree on common requirements, like the need for the system to be user-friendly and reliable.

Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation Methods:

b) Interviewing

- Interviewing is a primary method for requirement engineering team to gather information about the system that they currently use and the system to be developed.
- During interviewing you will gather facts, opinions, and assumptions and observe body language, emotions, and other signs of what people want and how they assess current systems.
- Preparation before the interview is essential, including setting up appointments, explaining the interview's purpose to the interviewee, and preparing questions.
- Making a list of questions and a plan for the interview helps to stay organized and cover all the important topics.
- Flexibility is necessary during the interview, as unexpected information may arise, requiring adjustments to the planned sequence of questions.

Choosing Interviewing Questions:

Open-ended questions: Questions in interviews that have no prespecified answers.

- Open-ended questions let people share their thoughts freely. They're good for finding new information and making the interviewee feel involved. But they can take a while to answer and may be tricky to summarize.
- An example is, “What would you say is the best thing about the information system you currently use to do your job?” or “List the three most frequently used menu options.”

Closed-ended questions: Those types of questions that we ask people in the interview to choose from a set of specific responses.

- Closed-ended questions work well when the main answers are already known.
- A benefit of closed-ended questions is that they save time, allowing more topics to be covered in the interview.
- Closed-ended questions, like objective questions on an examination, can follow several forms, including the following choices:
  - True or false.
  - Multiple choice (one response or selecting all relevant choices).
  - Rating a response or idea on a scale, say from bad to good or strongly agree to strongly disagree.
  - Ranking items in order of importance.

Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation Methods:

b) Interviewing

Eliciting domain knowledge through interviews can be challenging because:

- Experts often use specialized jargon that can be hard to understand or easy to misinterpret by those outside the field.
- Some knowledge is so familiar to experts that they either struggle to explain it or assume it's too basic to mention, leading to important details being not seen.

Good interviewers have two key qualities:

- They keep an open mind, listen carefully to what stakeholders say, and are willing to change their views if new ideas or surprising requirements come up.
- They ask specific questions or show examples, like a prototype, to help the conversation flow. Simply asking "What do you want?" usually doesn't work well; it's easier for people to discuss ideas when they have something concrete to focus on.

Interview Guidelines:

1st: Avoid asking questions that suggest a right or wrong answer. Let people share their opinions freely without feeling pressured.

- Instead of asking, "Do you think our new website design is better than our old one?" (which might lead the respondent to feel pressured to say yes),
- You could ask, "What do you like or dislike about the new website design compared to the old one?" This way, you're inviting the respondent to share their genuine thoughts without suggesting a preferred answer.

2nd: Listen carefully during the interview and take detailed notes or record with permission. This information may be crucial for the project, and it might be your only chance to gather it from this person. If needed, schedule a follow-up interview.

3rd: Once the interview is over, go back to your office and type up your notes within 48 hours and check them against any recordings you made. Organize your notes and ask for clarification if needed.

4th: Don't promise features for the new system unless you're sure they'll be included. Explain that decisions are still being made.

5th: Gather input from various perspectives, including potential users, managers, IT staff, and others. Understanding all viewpoints will help in making decisions that everyone can agree on later in the project.

Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation Methods:

c) Scenarios

- Scenarios are like stories that show how people will use a software system.
- Scenarios can be written out in text, with diagrams or screenshots, or more formally as use cases
- Scenarios help users understand the system better and give useful feedback, which helps define what the system needs to do.
- Scenarios are often created by requirements engineers, but they are usually developed with input from users.

Steps in creating Scenarios:

- **Collect Info:** Engineers gather basic information about what the system should do.
- **Identify Key Actions:** They figure out the main tasks users will perform with the system.
- **Create Draft Scenarios:** Engineers write up these tasks as example stories or scenarios.
- **Get Feedback:** They review these scenarios with users to ensure they are accurate and make changes based on user feedback.
- **Use Scenarios:** Finalized scenarios help users understand the system and provide useful feedback.

A typical scenario includes:

Example include scenario for opening food ordering app & ordering food

- What the system and users expect at the start of scenario.
  - **Example:** A user opens a food delivery app and expects to see a menu of local restaurants.
- The normal steps or flow of events in the scenario.
  - **Example:** The user selects a restaurant, chooses items from the menu, adds them to the cart, and proceeds to checkout.
- What might go wrong and how it should be handled.
  - **Example:** If a user's payment fails, the app shows an error message and prompts the user to try a different payment method.
- Any other activities happening at the same time.
  - **Example:** The user might be browsing another app for reviews while placing their order.
- The state of the system when the scenario ends.
  - **Example:** The order is successfully placed, and the user sees a confirmation screen with estimated delivery time.

Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation Methods:

d) Use Cases

- Use cases are a way to figure out what a system should do by showing how different users or systems will interact with it. They were introduced in the Objectory method and are now a key part of the Unified Modeling Language (UML).
- A use case describes an interaction between a user (or another system) and the system. It identifies who is involved (called "actors") and what the interaction is about.
- Use cases can include a written description or diagrams (like UML sequence diagrams) showing how the interaction happens.
- Use cases are often shown in a high-level diagram. Actors are shown as stick figures, and interactions are represented as named ovals. Lines connect actors to these interactions, and sometimes arrows show how the interaction starts.
- Some people think each use case is a single scenario, while others think a use case can include multiple scenarios.
- Use cases are great for showing how the system interacts with users or other systems but are less useful for finding out constraints, overall business needs, or detailed domain requirements.

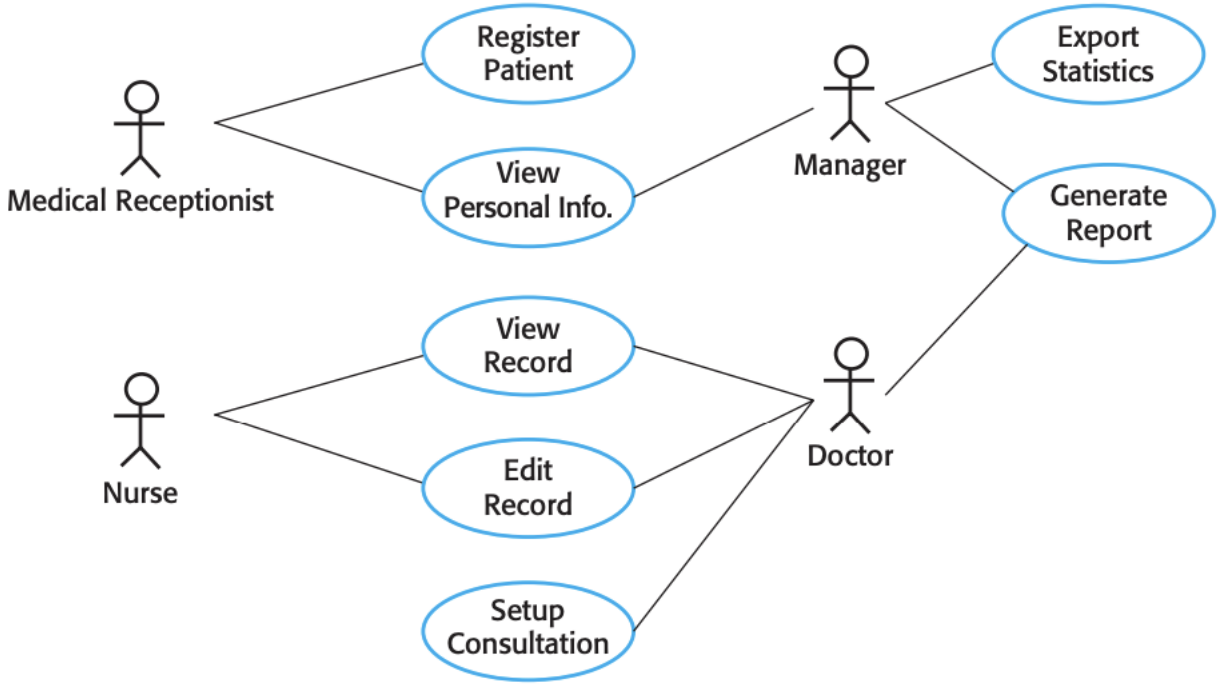


Figure: Simple Use cases for a Hospital

Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation Methods:

d) Use Cases

Benefits of Use Case Modeling:

- **Captures Functional Requirements**: Helps identify what the system should do.
- **Breaks Down Scope**: Makes it easier to manage and understand the overall system by dividing it into smaller parts.
- **Communicates Clearly:** Provides a means of communicating with users and other stakeholders concerning system functionality
- **Estimates Work**: Assists in predicting how much work, time, and resources are needed.
- **Supports Documentation**: Helps create user guides, help systems, and other documentation.
- **Designs Interfaces**: Assists in designing how users and systems will interact.

Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation Methods:

e) Ethnography/ Directly observing user

- Ethnography is a technique of requirement elicitation that helps understand the actual work processes by observing client directly.
- An analyst spends time in the environment where the system will be used, watching how people really do their jobs and taking notes.
- This helps uncover hidden requirements based on how people actually work.
- Interviewing methods for gathering information rely on people's ability to recall and describe their work and the systems they use.
- But people aren't always reliable in remembering or explaining things accurately. Sometimes, they don't have a clear understanding of their own actions, especially for rare events.
- To overcome this, we can watch what people do or collect objective data on their behavior.
    - One way to describe how a hypothetical manager does her job is that she plans her activities carefully, spends long periods consistently working on solving problems, and controls the pace of her work.
    - For example, a manager might say she plans her day carefully, but when observed, it's found she's often interrupted by phone calls or visits from their subordinates and other managers.

Consider another example:

- An employee says they have too much email to handle and feel stressed.
- However, upon checking email records, it's found that the employee receives an average of only 3 emails per day.
- The highest number of emails received in one eight-hour period is 10.
- This objective data contradicts what the employee believes, showing a big difference between what they say and what their email activity actually is.

Requirement Engineering Processes/ Activities in Requirement Engineering:

2) Requirement Analysis and Elicitation:

Requirement Elicitation Methods:

e) Ethnography/ Directly observing user

- The goal of directly observation is to get a more accurate understanding of how employees interact with systems.
- Sometimes, these behavioral measures reveal more than what people say directly. However, these methods have limitations too. Observation can change people's behavior, Employees who know they are being observed may be nervous and make more mistakes than normal and it's impractical to observe everyone all the time. So, we have to be careful in choosing whom and what to observe to get the most useful information.
- Observation is very time consuming, you will not only observe for a limited time, but also a limited number of people and a limited number of sites.

Ethnography is especially useful for finding two types of requirements:

i) **Real Work Practices**: People sometimes do their jobs differently than the official rules say. For example, air traffic controllers might turn off a warning system because it distracts them, even though they're supposed to use it. They have their own way of keeping planes safe.

ii) **Cooperation and Awareness:** People pay attention to what others around them are doing to work better together. For example, air traffic controllers might watch what nearby controllers are doing to guess how many planes will come into their area. A new system should help them see what's happening nearby.

Requirement Engineering Processes/ Activities in Requirement Engineering:

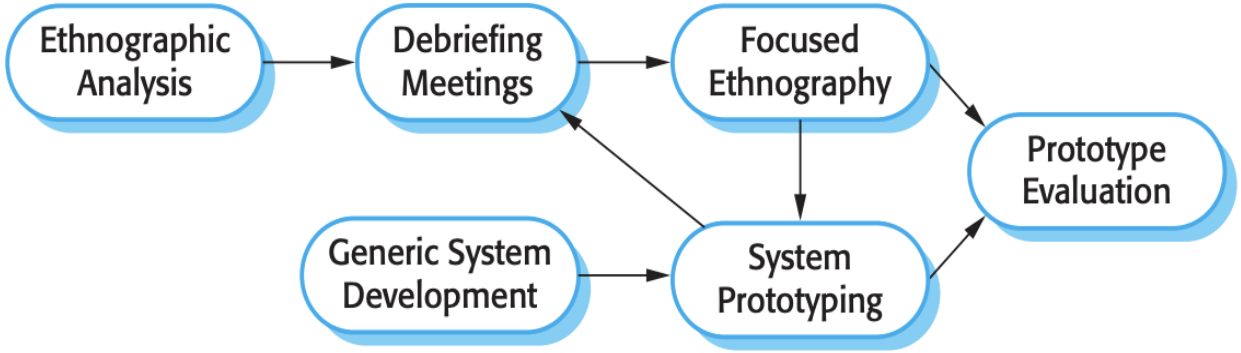
2) Requirement Analysis and Elicitation:

Requirement Elicitation Methods:

e) Ethnography/ Directly observing user

Pairing Ethnography with Prototyping:

Ethnography can be paired with prototyping to build better systems faster. Ethnography helps create a more accurate prototype by showing how people really work, so fewer changes are needed later. The prototype, in turn, points out specific issues that the ethnographer can focus on next.



- **Ethnographic Analysis**: Watching how people work to understand what they really need in a system.
- **Debriefing Meetings**: Talking about what was observed to decide on the next steps.
- **Focused Ethnography**: Zooming in on specific parts of the work to gather detailed information.
- **Prototype Evaluation**: Testing a simple version of the system to see if it works well for users.
- **Generic System Development**: Creating a basic system that can be adjusted based on what was learned.
- **System Prototyping**: Building a trial version of the system to try out ideas and get user feedback before making the final product.

Requirement Engineering Processes/ Activities in Requirement Engineering:

3) Requirement Specification

- Requirements Specification is the process of documenting what the system should do.
- User Requirements: These are written in a way that is easy for non-technical users and customers to understand.
- System Requirements: These are more detailed and may include technical information.
- These requirements are often part of a contract for developing the system, so it's important that they are as clear and complete as possible.

Ways of Writing Requirement Specification:

Notation	Description
Natural Language	Requirements are written in plain sentences, each stating one requirement.
Structured Natural Language	Requirements are written in a template, with specific fields for different parts of the requirement.
Design Description Languages	Uses a special language to describe system requirements in an abstract way, often for interfaces.
Graphical Notations	Uses diagrams and charts, like UML use case diagrams, to show how the system should work.
Mathematical Specifications	Uses mathematical concepts to define requirements. This method is precise but hard for most people to understand.

Requirement Engineering Processes/ Activities in Requirement Engineering:

3) Requirement Specification

a) Natural Language Specification:

- **What It Is**: Requirements are written in plain sentences, often with diagrams and tables to help explain things.
- **Why Use It**: It's easy to understand and communicate because it uses everyday language that users and customers can grasp.

Guidelines for writing SRS:

1. **Use a Standard Format**: Write each requirement in a single sentence and include a reason why it's needed. This makes it clear and easier to check.
2. **Be Clear About Necessity**: Use "shall" for things that must be done and "should" for things that are nice to have.
3. **Highlight Important Parts**: Use bold or italics to make key points stand out.
4. **Avoid Jargon**: Don't use technical terms or abbreviations that people might not understand.
5. **Explain Why**: Add a reason for each requirement so people know why it's needed and it helps if changes are needed later.

Problems with Natural Language

- **Lack of Clarity**: It can be hard to be very precise, making the document harder to read.
- **Requirements Confusion**: It's easy to mix up functional (what the system should do) and non-functional (how the system should perform) requirements.
- **Requirements Amalgamation**: Sometimes, multiple requirements are combined into one sentence, making them unclear.

b) Structured Specifications

- **Written in Standard Format**: Requirements are written in a fixed, standard way, freedom of the requirements writer is limited.
- **Good for Some Systems**: Works well for technical systems like Embedded control systems. Because, Technical jargon is acceptable here
- **Too Rigid for Others**: Can be too strict for business system requirements. Because, Business requirements often need to be expressed in simpler language which should be understood by stakeholders including non-technical users

Requirement Engineering Processes/ Activities in Requirement Engineering:

3) Requirement Specification

Significance of Requirement Specifications:

- A well-specified set of requirements ensures that all parties involved (stakeholders, developers, testers) have a common understanding of the system. It reduces miscommunication and serves as a contract between the client and the developers.

Types of Requirements Specifications:

- **Functional Requirements**: Describe the functions the system must perform (e.g., "The system shall allow users to withdraw funds from their account").
- **Non-functional Requirements**: Specify the system's quality attributes such as performance, reliability, security, etc. (e.g., "The system shall support up to 1,000 concurrent users").
- **Interface Requirements**: Define how the system will interact with other systems or components (e.g., "The system shall provide an API for mobile banking apps").

Challenges of Requirement Specifications:

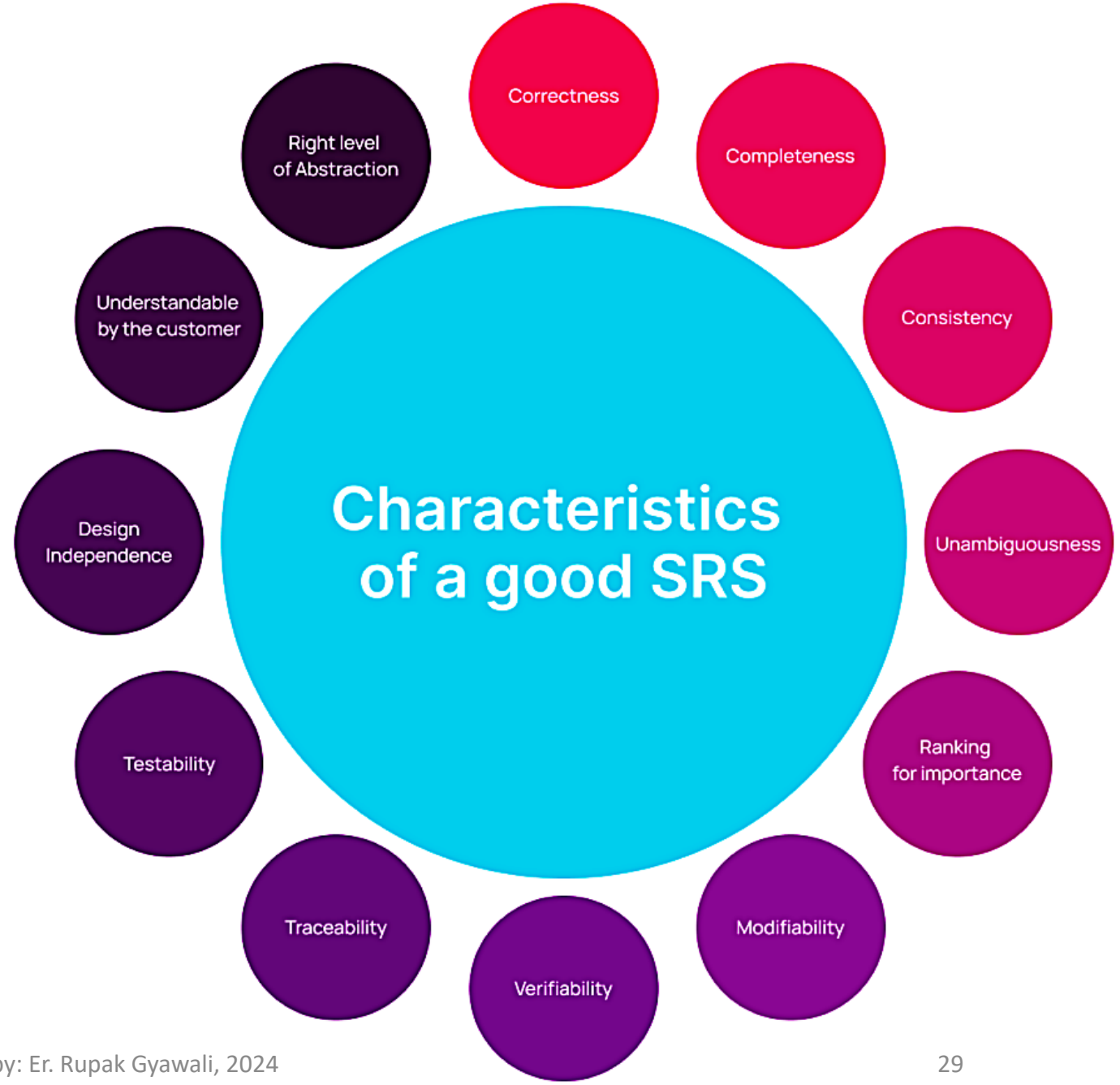
- **Incomplete or Evolving Requirements**: Capturing all requirements can be challenging, especially in projects with evolving needs.
- **Complexity**: For large systems, managing the complexity of requirements can be difficult.
- **Lack of Stakeholder Involvement**: Incomplete involvement from stakeholders can lead to gaps in requirements.

Example of Requirement Specifications:

In a library management system, the functional specification might include details about the features such as borrowing books, managing book inventory, and processing user fines. Non-functional specifications would detail system availability, response times, and security measures like user authentication.

System Requirement Specification (SRS):

- SRS is the document through which the client and the user needs are accurately specified.
- The SRS describes what the system should do, not how it will be built.
- The basic goal of requirement phase is to produce the SRS, which describes the complete external behavior of the proposed software.
    - External behavior is how the software interacts with users and other systems, including inputs, outputs, and responses.
- A well-written SRS is crucial for creating good software and can help lower development costs by avoiding misunderstandings and errors.



Characteristics of Good SRS:

- **Correctness:** Accurately reflects what the client needs.

  Example: If the client needs a search function, the SRS correctly includes that requirement.

System Requirement Specification (SRS):

Characteristics of Good SRS:

- **Completeness:** Includes all necessary details and requirements.

  Example: The SRS covers user login, data storage, and reporting features as required.

- **Consistency:** No conflicting information or requirements.

  Example: The SRS consistently states that the system should support 100 users, without any conflicting limits.

- **Unambiguity:** Clear and precise without vague terms.

  Example: The SRS specifies that "user" means an individual with a registered account, not just anyone accessing the system.

- **Ranking for Importance:** Prioritizes requirements based on their significance.

  Example: High-priority requirements like security are listed before lower-priority features like color customization.

- **Modifiability:** Easy to update when requirements change.

  Example: The SRS allows changes in user role definitions without disrupting other sections.

- **Verifiability:** Requirements can be tested to ensure they're met.

  Example: The SRS includes a requirement to log all user actions, which can be tested with a logging function.

- **Testability:** Requirements can be tested with specific criteria.

  Example: The SRS specifies that the system must process transactions in under 2 seconds, which can be tested.

- **Design Independence:** Describes what the system should do without dictating how to build it.

  Example: The SRS specifies the need for data encryption without specifying the encryption algorithm.

- **Understandable by the Customer:** Written in a way that the client can easily understand.

  Example: The SRS uses simple language to explain that users need to be able to reset their passwords.

- **Right Level of Abstraction:** Provides enough detail without being too technical.

  Example: The SRS outlines that users can create accounts and log in, without going into the specifics of database management.

System Requirement Specification (SRS):

What SRS should address?

- **Functionality:** What the software should do.
  - Example: The software must allow users to upload and share photos.

- **External Interfaces:** How the software interacts with users, hardware, and other systems.
  - Example: The software must connect to a cloud storage service and have a user-friendly login page.

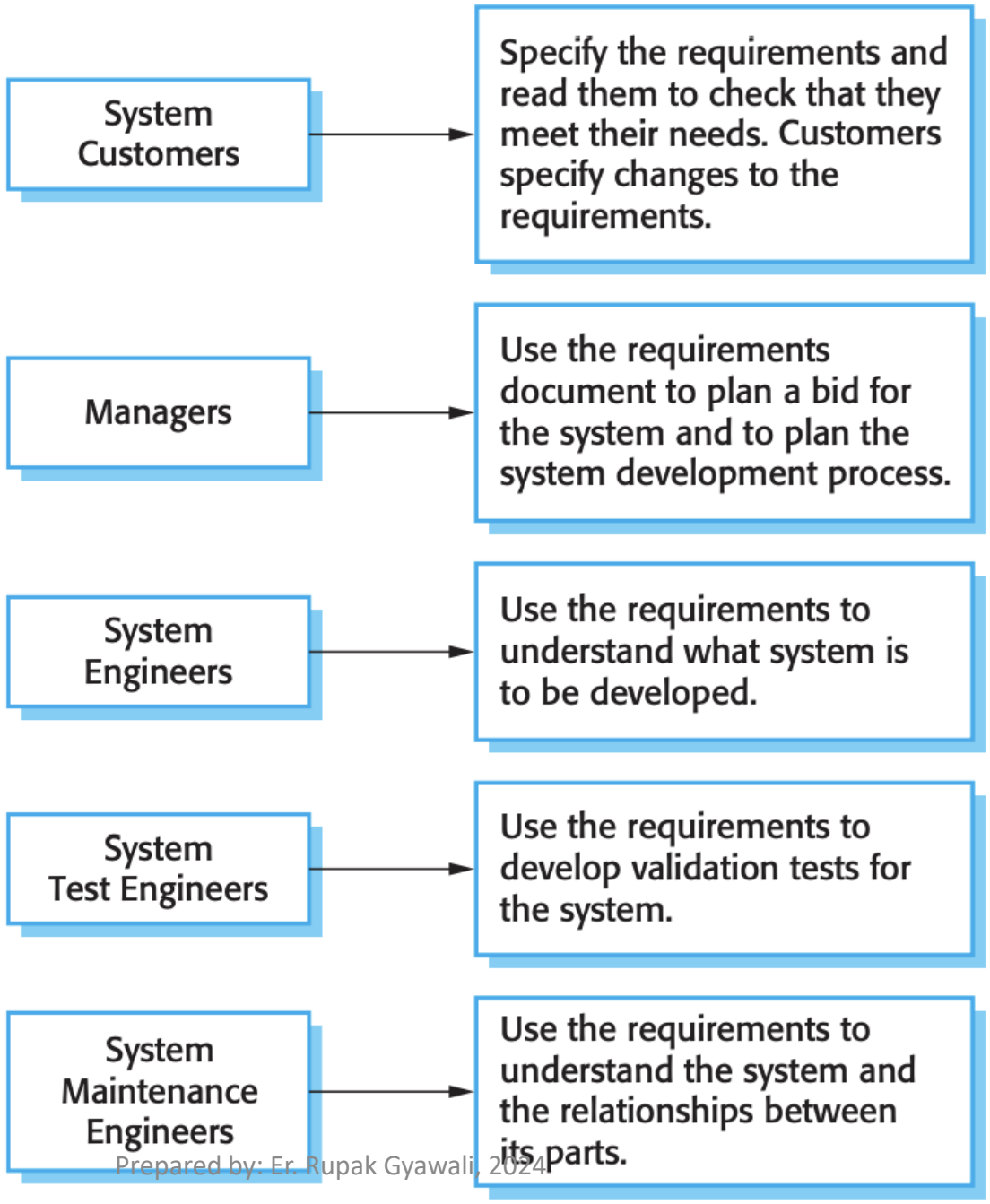
- **Performance:** How fast and reliable the software should be.
  - Example: The software should load pages within 3 seconds and be available 99.9% of the time.

- **Attributes:** Key qualities like portability, correctness, and security.
  - Example: The software should be easy to update, correct errors promptly, and protect user data.

- **Design Constraints:** Any limitations or standards that must be followed.
  - Example: The software must be built using Python and must comply with GDPR data protection standards.

System Requirement Specification (SRS):

Users of SRS document



Requirement Engineering Processes/ Activities in Requirement Engineering:

4) Requirement Validation

- Requirements validation is the process of making sure the listed requirements actually match what the customer wants.
- It involves checking for mistakes or issues in the requirements before development starts.
- It's crucial because fixing problems later in the development process or after the system is in use can be very costly.
- Fixing errors in requirements is often more expensive than fixing problems in design or coding because changes to requirements usually require revising the whole system design and implementation.

Techniques/ Methods for Requirement Validation:

1.Reviewing Requirements: Team members and stakeholders review the documented requirements to ensure they accurately reflect what the customer needs.

2.Prototyping: Building a preliminary version of the system to see if it meets the requirements and gathering feedback from users.

3.Testing: Checking if the requirements are feasible and can be tested with the system.

4.Interviews and Surveys: Asking customers and stakeholders to confirm that the requirements align with their expectations and needs.

5.Walkthroughs: Going through the requirements step-by-step with the team and stakeholders to identify any issues or misunderstandings.

6.Validation with Use Cases and Scenarios: Using detailed use cases or scenarios to ensure that the requirements cover all necessary interactions and use situations.

Requirement Engineering Processes/ Activities in Requirement Engineering:

4) Requirement Validation

Different types of checks during requirements validation:

- **Validity Checks:** Confirm the requirements actually cover what's needed and consider any additional needs that might come up.
  - *Example*: If someone asks for a system that can handle orders, a validity check might find that they also need it to handle returns, which wasn't mentioned before.
- **Consistency Checks:** Make sure there are no conflicting or contradictory requirements in the document.
  - *Example*: If one requirement says the system should only work during business hours and another says it should work 24/7, these need to be aligned.
- **Completeness Checks:** Ensure the document includes all the necessary functions and rules for the system.
  - *Example*: If the document says the system should let users log in but forgets to include a way for users to sign up, it's missing a part.
- **Realism Checks:** Verify that the requirements can be achieved with current technology and fit within the budget and schedule.
  - *Example*: If the requirements ask for a feature that needs special equipment that isn't available or is too costly, it should be adjusted to use what's available.
- **Verifiability:** Ensure that each requirement can be tested to confirm that the final system meets it. This helps avoid disputes between the customer and the developer.
  - *Example*: If a requirement says the system should load a webpage in under 3 seconds, you should be able to test this by measuring the load time to confirm it meets the requirement.

Requirement Engineering Processes/ Activities in Requirement Engineering:

4) Requirement Validation

Significance of Requirement Validation:

Requirements validation is critical because incorrect or incomplete requirements can lead to software that does not meet user needs, leading to expensive rework, project delays, or even project failure.

Challenges of Requirement Validation:

- **Ambiguity**: Vague or unclear requirements can be difficult to validate.
- **Conflicting Requirements**: Different stakeholders might have conflicting requirements, which makes it challenging to resolve differences.
- **Incomplete Information**: Not all stakeholders might fully articulate their needs or anticipate future changes.
- **Changing Requirements**: As the project progresses, stakeholders may change their requirements, which can complicate the validation process.

Example of Requirement Validation:

In the development of an online banking system, requirements validation would involve verifying that the login, fund transfer, and transaction history features are clearly defined and meet the security expectations of the bank and its customers. A prototype interface could be presented to stakeholders to validate the user experience and functionality.

Requirement Engineering Processes/ Activities in Requirement Engineering:

5) Requirement Management

- Requirements management is about handling the fact that software needs change over time. Here's why:
  - **Changing Problem**: As people work with the software, they often discover new problems. The requirements must be updated to match these changes.
  - **Emerging Need**: After using the system, users might find new needs or issues that weren't clear before. The requirements have to be adjusted to include these new needs.
- As time evolves, the requirements for the software need to be updated to stay useful.

<u>Several reasons why change is inevitable/ Change happens</u>

- **Changing Environment:** After a system is installed, new technology, different business needs, or new rules may require updates to the system.
  - *Example*: A company installs a new software system for tracking inventory. Later, they upgrade their hardware to faster computers, which means the software needs to be updated to work with the new equipment.
- **Different People, Different Needs:** The people who pay for the system and the people who use it often have different needs. This can lead to changes in the system after it's delivered to better meet user needs.
  - *Example*: A company buys a new project management tool. The managers wanted features to track budgets, but the employees need better task management tools. After seeing how the tool is used, the company adds features to support both needs.
- **Conflicting Requirements:** Large systems serve many users with different needs. Over time, it might become clear that the system needs to be adjusted to better meet everyone's needs.
  - *Example*: A large hospital system is used by doctors, nurses, and administrative staff. Initially, the system focuses on doctors' needs for patient records, but later it becomes clear that nurses need more tools for scheduling and patient care, so adjustments are made to better support everyone.

Requirement Engineering Processes/ Activities in Requirement Engineering:

5) Requirement Management

a) Requirement Management Planning:

- Planning is an essential first stage in the requirements management process.

Planning for requirements management process:

- **Requirements Identification**: Give each requirement a unique label so you can easily find and connect it with others.
- **Change Management Process**: Set up a method to evaluate how changes to requirements will affect the project and their cost.
- **Traceability Policies**: Create rules for tracking how requirements relate to each other and to the system design. Decide how to keep these records updated.
- **Tool Support**: Use tools to manage all the information about requirements. These tools can range from simple spreadsheets to specialized software like DBMS
  - **Requirements Storage**: Keep all requirements in a secure and accessible place.
  - **Change Management**: Use tools to help manage changes to requirements efficiently.
  - **Traceability Management**: Use tools to find and manage relationships between different requirements.

b) Requirement Change Management:

- After the requirements are approved, any proposed changes need to be managed carefully.
- This means deciding if the benefits of new requirements are worth the cost of making those changes.
- A formal change management process helps make sure all changes are handled the same way and keeps everything organized.

Three principal stages to a change management process:

- **Problem analysis and change specification:** Start by looking into the problem or proposed change. Check if it's valid and give feedback to the requester, who might provide more details or cancel the request.
- **Change analysis and costing:** Evaluate how the change will affect the system and estimate how much it will cost. Decide if the change is worth making based on this analysis.
- **Change Implementation:** Update the requirements document and the system design if needed. Keep the document organized so changes can be made easily without affecting other parts.

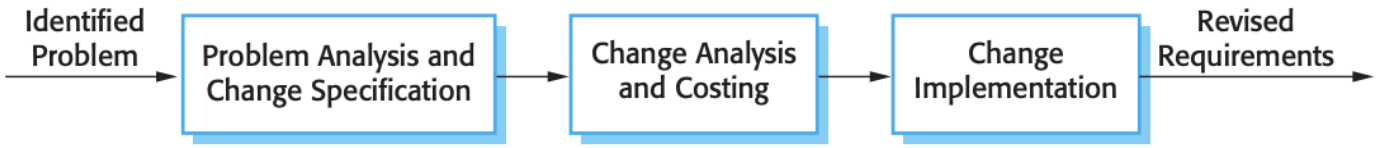


Figure: Requirements change management

Requirement Engineering Processes/ Activities in Requirement Engineering:

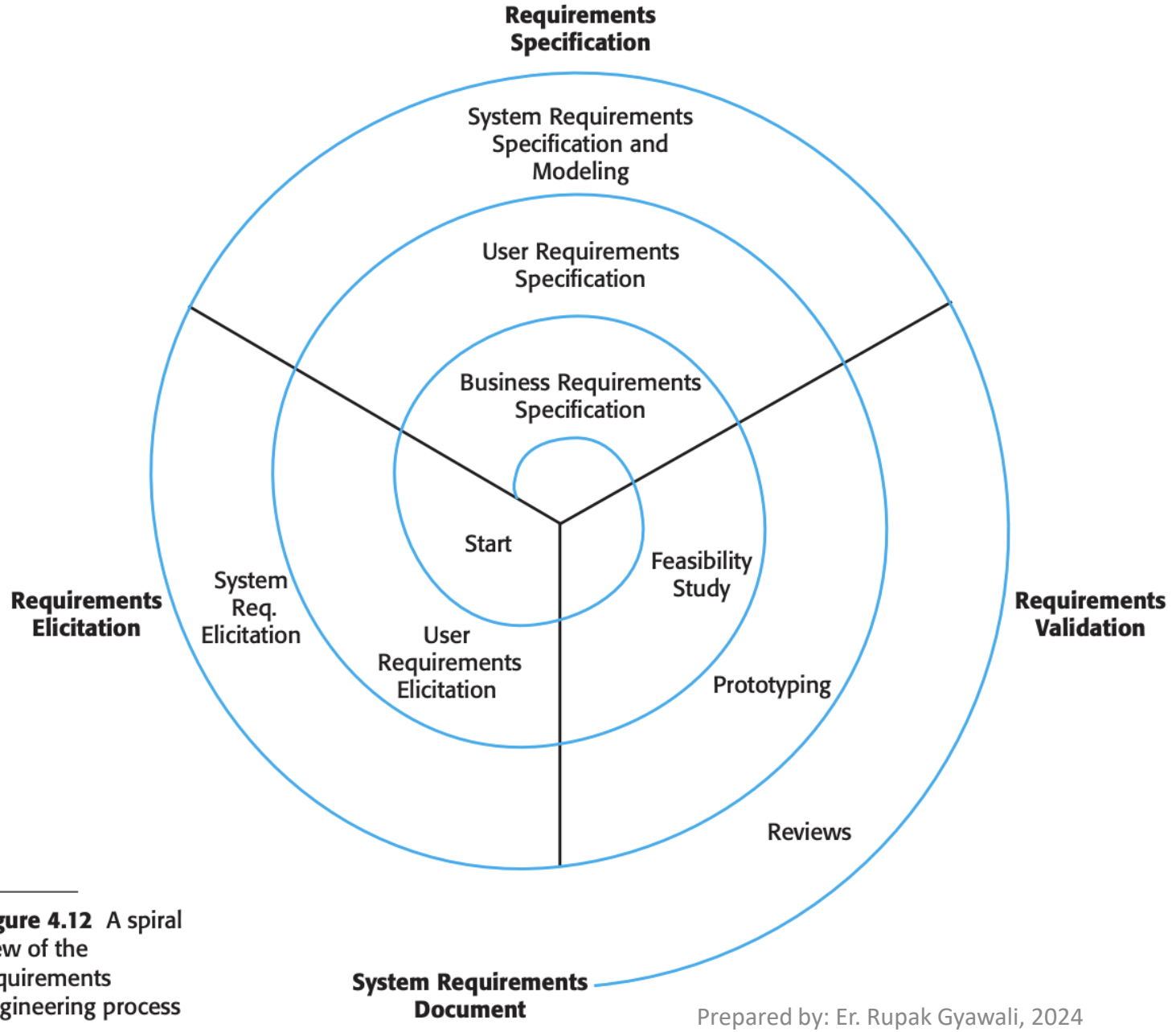


Figure 4.12 A spiral view of the requirements engineering process