Course: Software Requirements Engineering

Week 1: Introduction to Software Requirements

Lecturer: Yimer Amedie (MSc.)

Addis Ababa Science and Technology University

March, 2026

Course Description

- This course provides a comprehensive understanding of how to identify, analyze, document, validate, and manage software requirements.
- The course covers:
  - Requirement types
  - Elicitation techniques
  - Stakeholder analysis
  - Requirements modeling
  - Specification standards, and
  - validation and change management practices.

Course Goal

- To equip students with the knowledge and skills to effectively elicit, analyze, specify, validate, and manage quality software requirements
  - Explain software requirements engineering concepts.
  - Identify stakeholders and requirement sources.
  - Apply requirements elicitation techniques.
  - Analyze and refine requirements.
  - Model and document requirements (SRS).
  - Validate and manage requirements.

Course Goal

Requirement Types

Functional, Non-functional, and their impact.

Stakeholder Analysis

Identify and manage diverse interests.

Elicitation Techniques

Interviewing, workshops, prototyping.

SRS Documentation

Craft clear and comprehensive specifications.

Use Cases & User Stories

Model system behavior and user interactions.

Requirement Validation

Ensure completeness, consistency, and feasibility.

Change Management

Handle evolving requirements effectively.

- ✓ By the end, you will think like a consultant, ask smart questions, and avoid building the wrong systems

Week 1: Introduction to Software Requirements

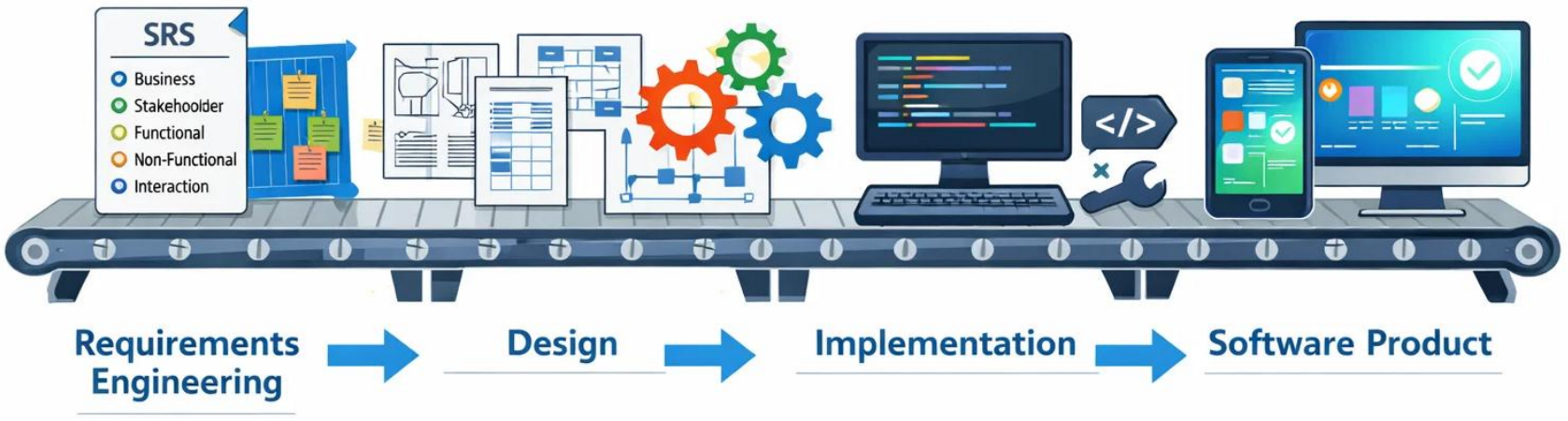


Figure 1. *The software engineering journey*

Note. Image generated using Sora by OpenAI (2026).

Contents

- Introduction
- Overview of Software Requirements Engineering (SRE)
- Requirements Lifecycle
- Importance and impact of requirements in software projects

Figure 2. *The software requirements*
Note. Image generated using Sora by OpenAI (2026).

Learning Outcomes

After completing this lesson, you will be able to:

- Define software requirements and requirement specification
- Explain the concept of Software Requirements Engineering (SRE).
- Describe the importance of requirements in software projects.
- Explain how requirements relate to the SDLC

.

Introduction … (1/3)

Now a days, software is a life blood of any organizations.

What is Software?

- Software consists of:
  - Programs
  - Data
  - Documentation
  - Procedures

Introduction … (2/3)

Why Software Projects Fail?

Common causes of failure:

Poor requirements

Communication gaps

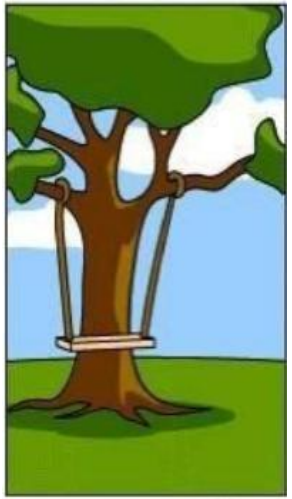
Scope creep

Unrealistic expectations

Introduction … (3/3)

Need

What the customer wanted

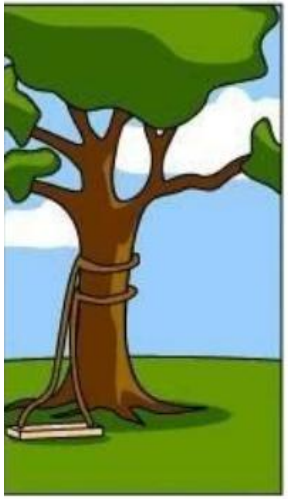


Analysis

What the analyst understand

Design

What the architect designed



Deployed system

What the programmers implemented

Figure 3: *Communication Problem (Glinz, 2013)*

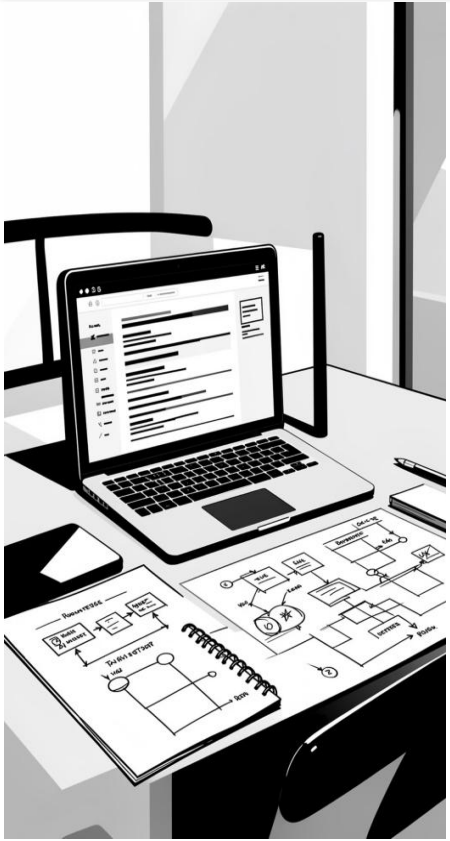
We need to know the requirements

Problem ➔ Requirements ➔ Solution

Having a problem, we need requirements for a system that solves the problem!!

Figure 4. *The problem, requirements and solution concepts*
Note. Image generated using Sora by OpenAI (2026).

Overview of Requirement … (1/2)

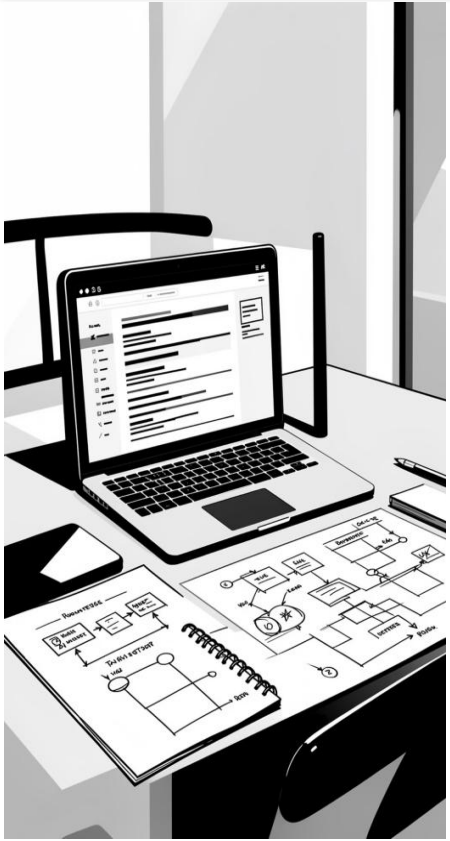


What is Requirement?

According to the **IEEE**, a requirement is:

- A condition or capability needed by a user to solve a problem or achieve an objective.
- It describes **what the system should do** or **how it should behave**.

Overview of Requirement … (2/2)



- ✓ A **need** perceived by a stakeholder.
- ✓ A **capability** or property that a system shall have.
- ✓ A **documented representation** of a need, capability or property

Requirements Specification

- ➢ A systematically represented collection of requirements, typically for a system or component, that satisfies given criteria

Example of Problem

Problem:

- ✓ In a university with a formal curriculum, students register for courses each semester using paper forms that require advisor approval. This process causes delays, errors, and difficulty tracking course availability and prerequisites.

Technical Solution:

✓Build a **software solution** that allows students to select courses, check prerequisites and availability, and submit registrations for advisor approval online.

When Building such Solution …

1. How do we determine the requirements?

2. How can we analyze and document these requirements?

3. How do we make sure that we've got the right requirements?

4. How do we manage and evolve the requirements?

Software Requirements Engineering (SRE)

Figure 5. *Software Requirements Engineering*
Note. Image generated using Sora by OpenAI (2026).

SRE … (1/2)

- The systematic process of discovering, analyzing, documenting, validating, and managing the requirements of a software system.
- The **foundation** of successful software development.

✓ It ensures that the final system meets

- ✓ The expectations of its users,
- ✓ Operates correctly within its intended environment,
- ✓ Satisfies business objectives

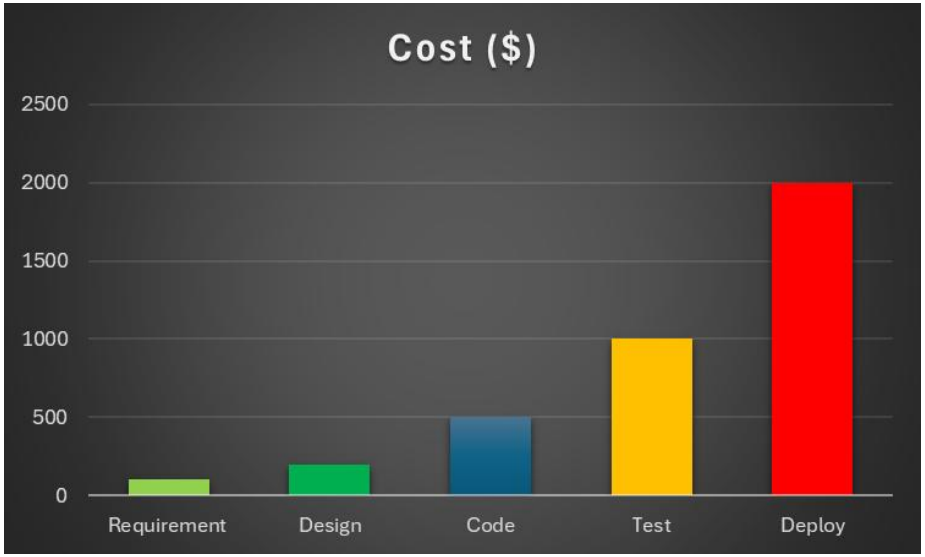
SRE … (2/2)

Focuses on:

- ✓ Identifying real problems
- ✓ Understanding stakeholder needs
- ✓ Defining clear, complete, and correct requirements
- ✓ Documenting and validating them

Why are Requirements important?

60–80% of software defects originate from poor requirements.



Fixing an error in:

- Requirements phase
  = Cheap
- After deployment
  = Extremely expensive

✓ Suppliers makes profit
✓ Stakeholders becomes satisfied

Requirement Error → Design → Code → Test → Deploy

Poor Requirements

- Vague, incomplete, ambiguous, or untestable statements.

Example:
- **The system should be fast.**

How fast? For whom? Under what conditions?

Poor requirements lead to:

- Misunderstandings
- Scope creep
- Budget overrun
- User dissatisfaction
- Rework
- Delays
- System rejection

Good Requirements

Example: The system shall process a customer transaction within 2 seconds under normal load

- ✓ Clear, specific, complete, measurable, consistent, feasible and testable statements.
- ✓ They provide unambiguous guidance for development and validation.

- ✓ Better project planning
- ✓ Reduced development cost
- ✓ Improved product quality
- ✓ Higher user satisfaction

Why Developers Hate SRE (But Shouldn’t)?

Common Misconceptions:

- Requirements is just documentation.
- We just want to code; requirements are boring.
- Agile means we don't need formal requirements.

The Reality:

- ✓ Requirements engineering is about truly **understanding the problem**.
- ✓ Coding without clear requirements is like **guessing** your way through a confusion.
- ✓ Even in Agile, understanding user needs is paramount.

A developer who deeply understands requirements becomes a leader, building solutions that truly matter.

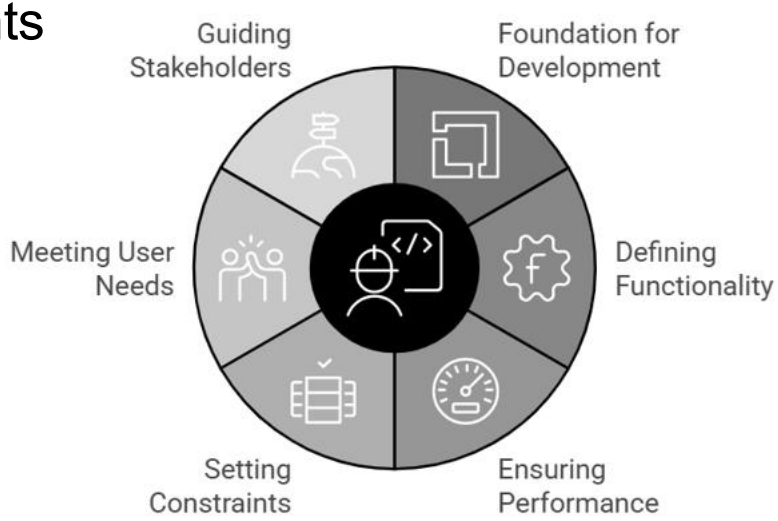
Why are Requirements become poor?

The reasons, among others are (Beatty, 2013)

- Some software teams aren't proficient at eliciting requirements from customers and other sources.
- Customers often don't have the time or patience to participate in requirements activities.
- Many undergraduate curricula in software engineering underemphasize the importance of RE.
- Most software engineers tend to be impressed with technical and process solutions.

Advantages of Requirement …(1/2)

- Therefore, software requirements are serving as a foundation for building effective and efficient software systems.
- They define
  - what the software should do?
  - how it should perform?
  - the constraints under which it must operate.



Advantages of Requirement … (2/2)

Requirements help to

Define Project Scope
Establish boundaries and objectives for project execution.

Avoid Misunderstandings
Ensure all stakeholders have a shared understanding.

Reduce Development Cost
Minimize expenses through efficient resource allocation.

Prevent Rework
Eliminate errors and unnecessary modifications.

Support Testing and Validation
Facilitate thorough testing and validation processes.

Serve as a Legal Reference
Provide a basis for legal agreements and disputes.

Requirements act as a **contract** between stakeholders and developers.

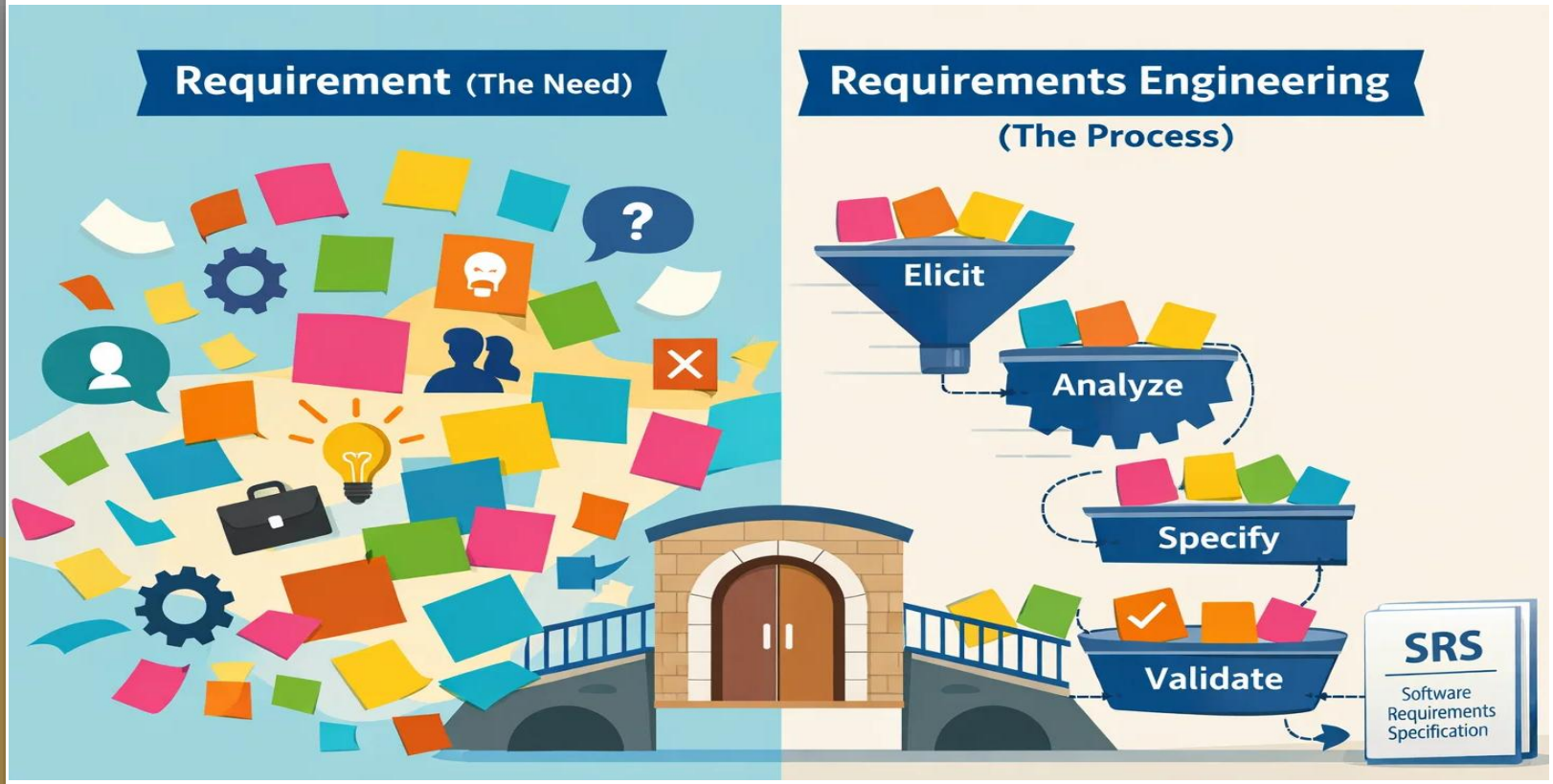


Figure 6. *Requirement Vs Requirements Engineering*
Note. Image generated using Sora by OpenAI (2026).

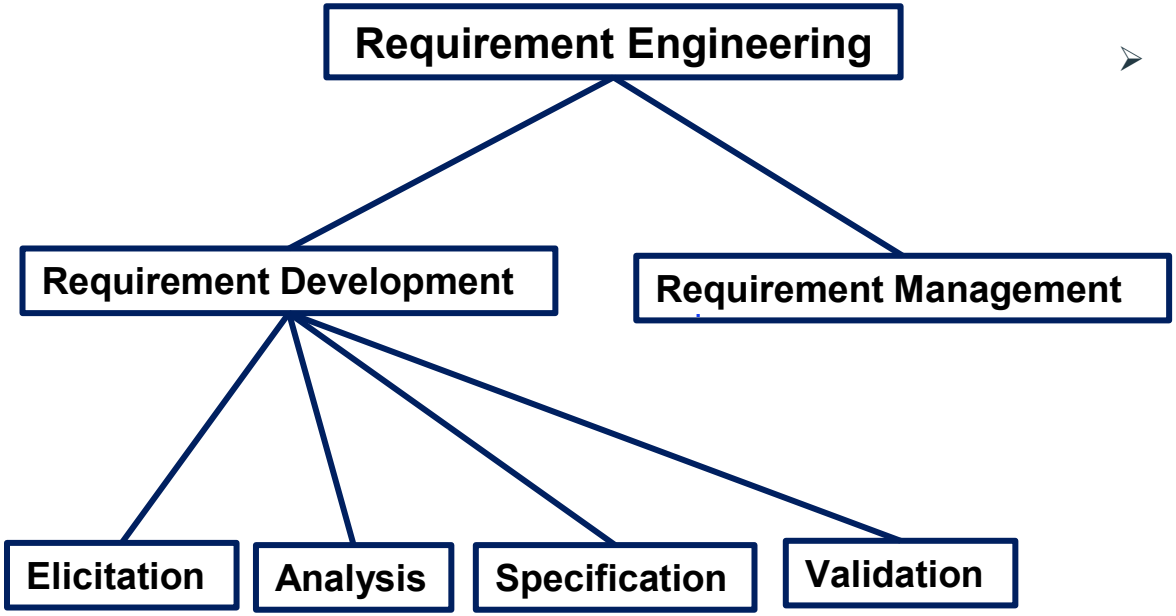
Principles of RE … (1/2)

- **Value-Oriented:** Focus on business and user goals.
- **Stakeholder-Centered:**
  - Satisfy stakeholders' needs and expectations.
- **Shared Understanding:**
  - Ensure all parties have a common view.
- **Context-Aware:** Consider the system within its environment.

Principles of RE … (2/2)

- **Problem → Requirement →Solution:**
  - Treat them as interconnected.
- **Validation:** Confirm requirements are correct and complete.
- **Evolution:** Accept and manage changing requirements.
- **Innovation:** Enable new and improved solutions.
- **Systematic Approach:** Use structured, disciplined processes.

Components of RE

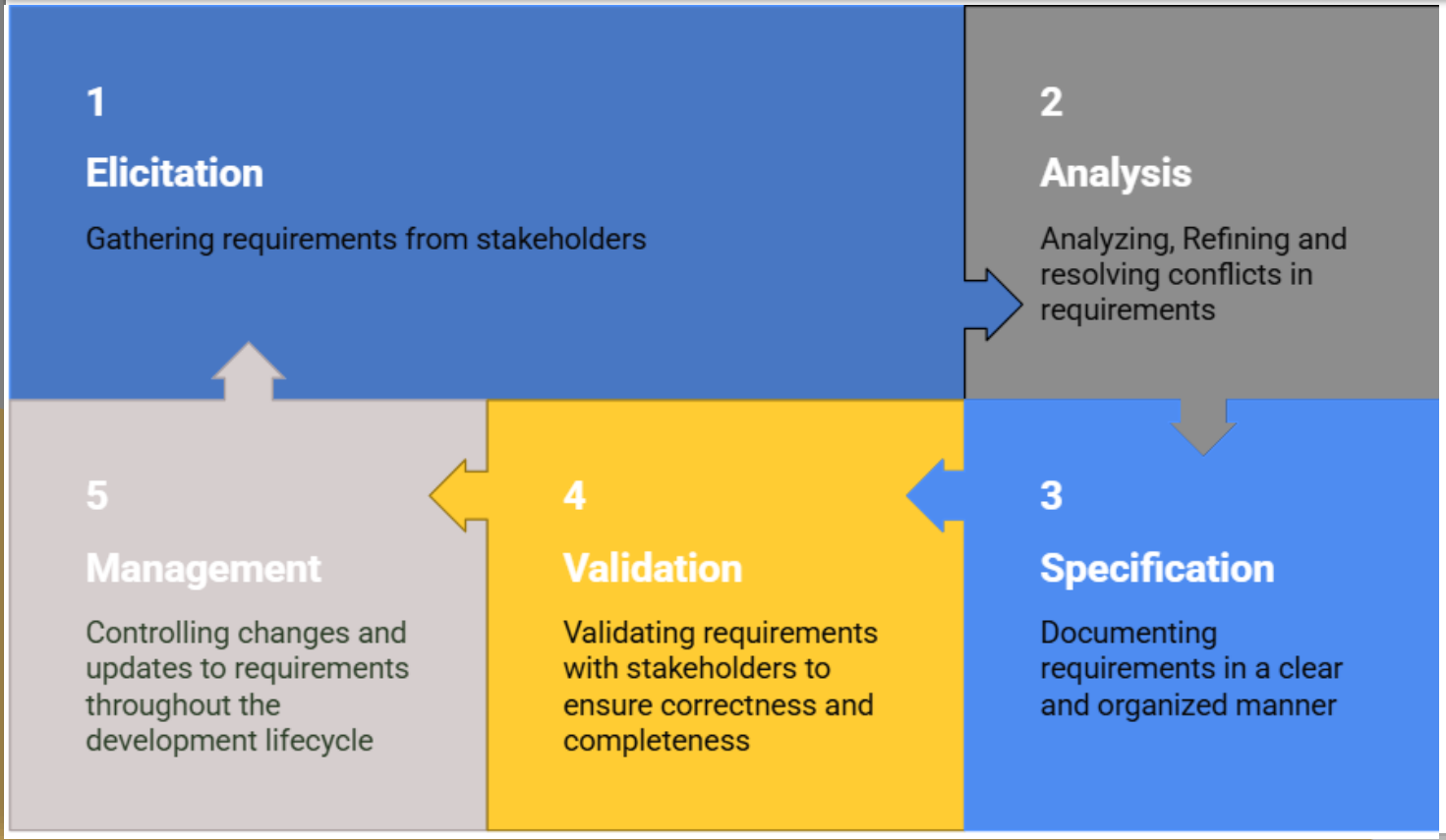


- Whether the approach is:
  - ✓ Pure waterfall
  - ✓ Phased
  - ✓ Iterative/Incremental
  - ✓ Agile or some hybrid

Requirements Lifecycle … (1/2)

Requirements Engineering: Process

Requirements Lifecycle … (2/2)



Activity: Fill the correct terms

Requirement vs. Requirements Engineering

Requirement _____

SRS _____

Requirements Eng. _____

Validation _____

Elicitation _____

A) A statement about what a system must do or how it should behave.

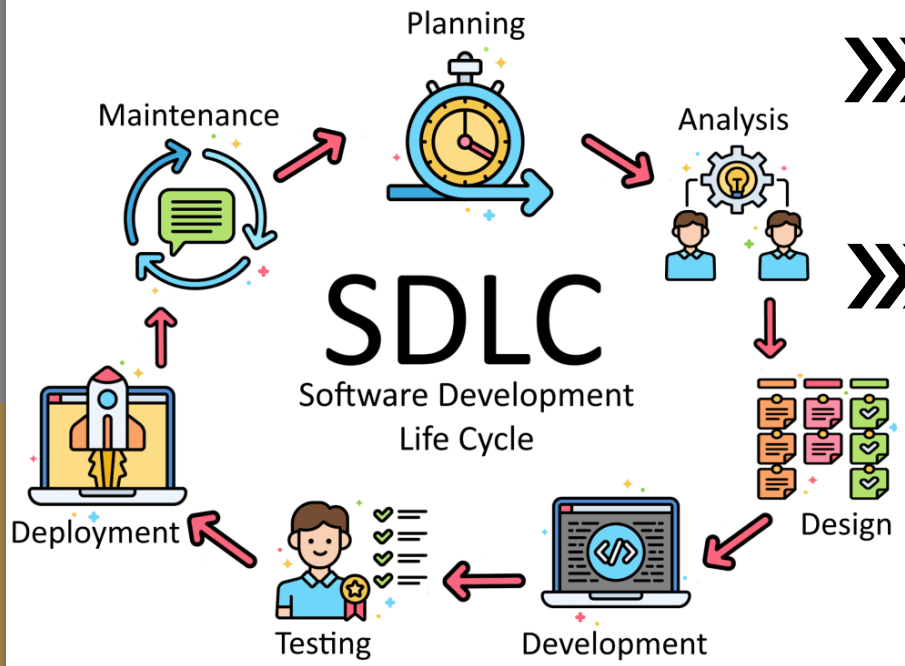
B) The complete collection of all requirements in one place (the blueprint).

C) The overall discipline of gathering, analyzing, documenting, and managing needs.

D) Checking if we understood and documented the requirement correctly.

E) The process of discovering hidden needs by talking to users and stakeholders.

Requirements in SDLC … (1/3)



- In the **Waterfall model**,
  - ☑ requirements are defined at the beginning.
- In **Agile software development**,
  - ☑ requirements evolve continuously through user stories and feedback.

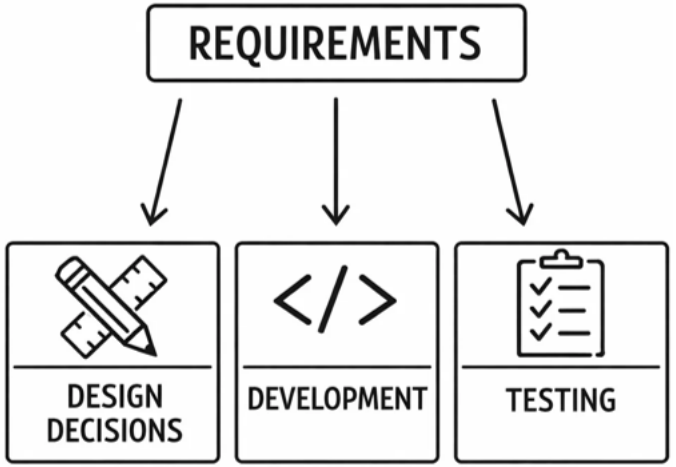
Figure 7. *Software Development Lifecycle*
Note. Image generated using Sora by OpenAI (2026).

Requirements in SDLC … (2/3)

- Agile projects frequently do not produce a comprehensive requirements specification (Glinz, 2013). Instead, they express requirements in
  - ✓ User stories, issues, storyboards, etc.
  - ✓ Acceptance criteria associated with user stories
  - ✓ A vision document
  - ✓ Implicit shared understanding among the people involved

Requirements in SDLC … (3/3)

- Requirements guide:
  - Design decisions
  - Development
  - Testing



Stakeholders in Requirement

Stakeholders

- Anyone who has an interest in the system.

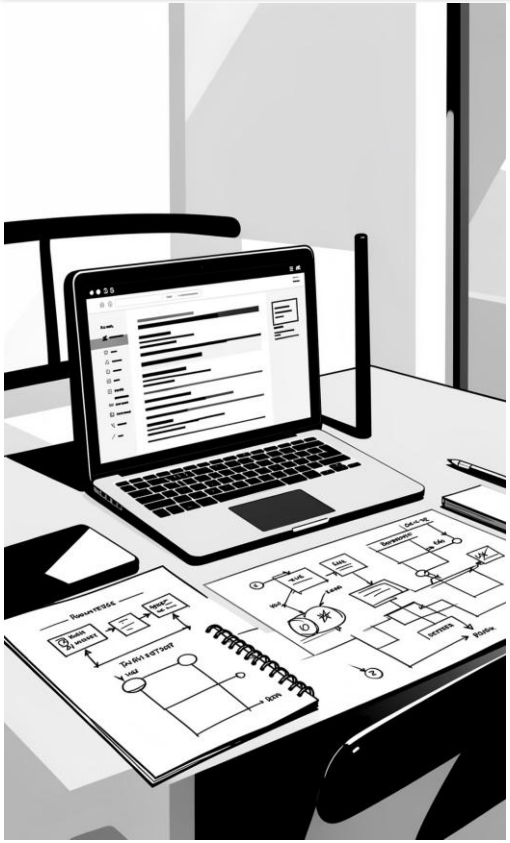
  - End users
  - Clients
  - Project sponsors
  - Developers
  - Regulators

Example

Example:
ERP system in banking

- ✓ Bank staff
- ✓ IT department
- ✓ Management
- ✓ Central bank regulators

Discussion Questions



1. What happens if requirements are misunderstood?
2. Who suffers most from poor requirements?
3. Why requirements become poor?
4. How can we avoid poor requirements

Discussion Questions - Answers

Q1A. Wrong product is built, Rework and delays, Budget overruns, Scope creep and confusion, Low quality and defects, Stakeholder dissatisfaction

Q2A. End users, the organization/client, Project Team (Developers, Testers, Analysts), Project Managers

Q3A. Because of weak communication, unclear goals, rushed processes, or lack of stakeholder involvement

Q4A. With strong analysis, regular validation, and clear documentation

Summary

- ✓ Requirements are the heartbeat of successful software.
- ✓ Requirements describe what a system should do and how it should behave.
- ✓ Software Requirements Engineering ensures the right system is built.
- ✓ Poor requirements often lead to project failures and rework.
- ✓ Clear and validated requirements guide design, development, and testing

References

1. Beatty, K. W. (2013). Software Requirements (3rd ed.). Washington: Microsoft Press.
2. Glinz, M. (2013). Department of Informatics. Retrieved March 7, 2026, from University of Zurich: https://www.ifi.uzh.ch/dam/jcr:386e83a9-f3c7-4d95-87e6-800841406473/RE_I_Slides_Part_I.pdf

Thank You!