Introduction to Software Engineering

Submitted by:
Ms. Anshika
Assistant Professor
Computer Science and Engineering ,MAIT
Maharaja Agrasen University

Outline

- Software Engineering Definition
- Software Characteristics
- Role of Software
- Principles Of Software Engineering
- Software Process
- Software Process Models

Software Engineering Definition

Software engineering involves the systematic process of designing, developing, testing, and maintaining software to ensure high quality, reliability, and maintainability. It encompasses various techniques, tools, and methodologies such as requirements analysis, design, testing, and maintenance. As a rapidly evolving field, new tools and technologies are continuously emerging to enhance the software development process.

Software Characteristics

- Software is intangible, meaning it cannot be seen or physically touched.
- It is non-perishable, as it does not degrade over time.
- It is easily replicable, allowing for effortless copying and distribution.
- It can be highly complex, consisting of numerous interrelated components and features.
- Understanding and modifying software can be challenging, particularly in large and intricate systems.
- It is influenced by changing requirements, often requiring updates or modifications to meet evolving user needs.
- Bugs and other issues can impact software, necessitating thorough testing and debugging to ensure proper functionality.

Role of Software

- Core Component : Software is the foundation of applications, systems, and digital solutions.
- Automation & Efficiency: It automates processes, improving productivity and reducing manual effort.
- Problem-Solving: Enables users to perform tasks and solve complex problems across various domains.
- Development & Maintenance: Supports designing, coding, testing, and maintaining software solutions.
- Scalability & Security: Ensures systems remain reliable, secure, and adaptable to changing needs.
- Innovation & Evolution: Continuously evolves with new technologies to enhance capabilities.

Principles Of Software Engineering

Principles of Software Engineering deal with both process of software Engineering. These principles are helping software engineers make decisions, solve problems, and manage projects. that provide structure and direction for developing high-quality software systems.

Here are some of the key principles of software engineering:

1. Separation of Concerns

Time Schedule

Qualities

Views

Size

2.Modularity Principle

Increases maintainability

Reusability

Understandability

Debugged separately

3. Anticipation of Change

Software meet evolve to meet
Changing customer requirement

4. Rigor and Formality

Rigor is defined as accuracy and exactness

Requires the software process to be driven

5. Abstraction Principle

Identifying the important aspects

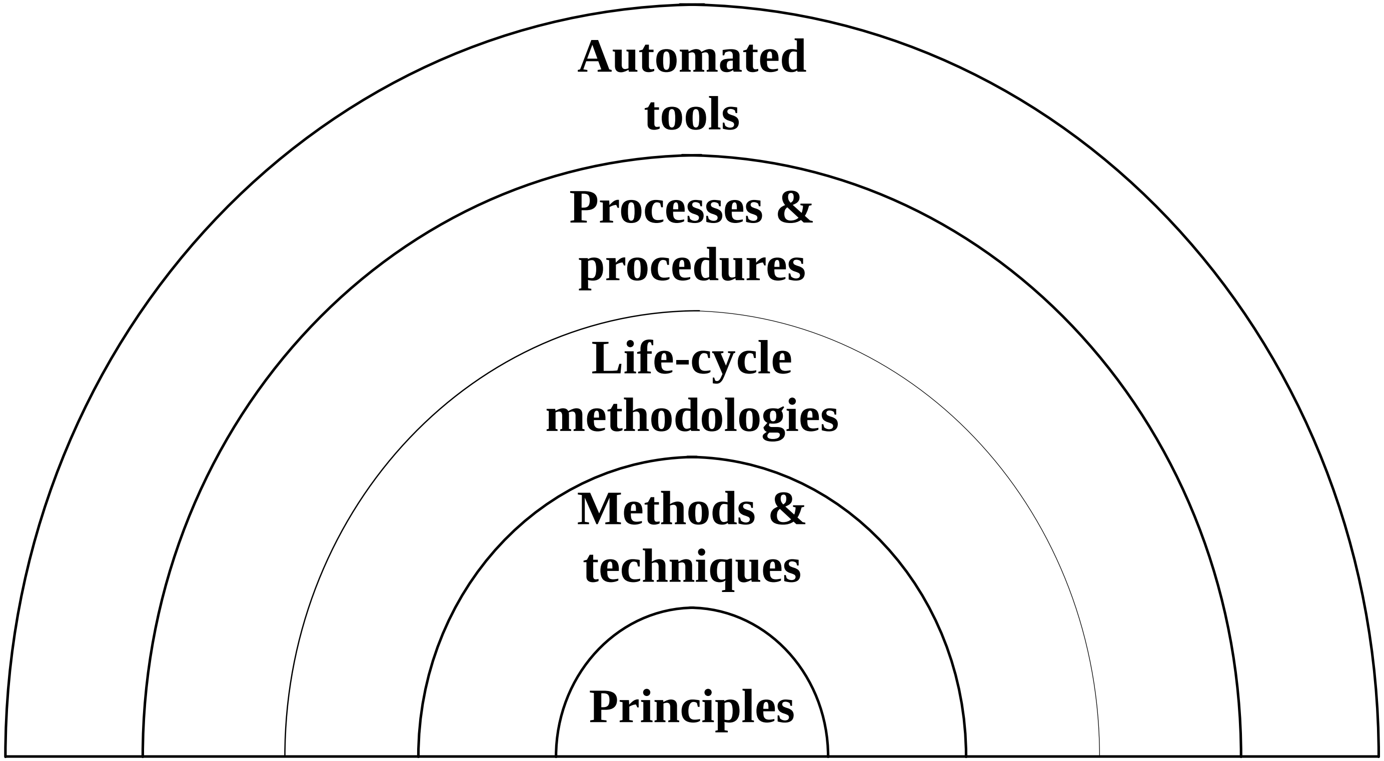
Concentrate on Solution

6. Principle of Generality

Trying to focus at more general problem

Instead of designing

Methods & Techniques



Automated tools

Processes & procedures

Life-cycle methodologies

Methods & techniques

Principles

Software Process

The software process refers to a structured set of activities involved in the development, maintenance, and management of software. It ensures the systematic production of high-quality, reliable, and maintainable software.

Key Phases of the Software Process:

- Requirement Analysis – Identifying and documenting user needs.
- Design – Creating the architecture and blueprint for software development.
- Implementation (Coding) – Writing the actual program code.
- Testing – Verifying and validating the software to ensure it meets requirements.
- Deployment – Releasing the software for use in a real-world environment.
- Maintenance – Updating and improving the software over time to fix bugs and adapt to new requirements.

Software Process Model

A software process model is a structured framework that defines the approach and sequence of activities involved in software development. It provides guidelines for planning, designing, implementing, testing, and maintaining software. Different models are used based on project requirements, complexity, and development methodology.

Software Process Models:

Waterfall Model – A linear and sequential approach with distinct phases.

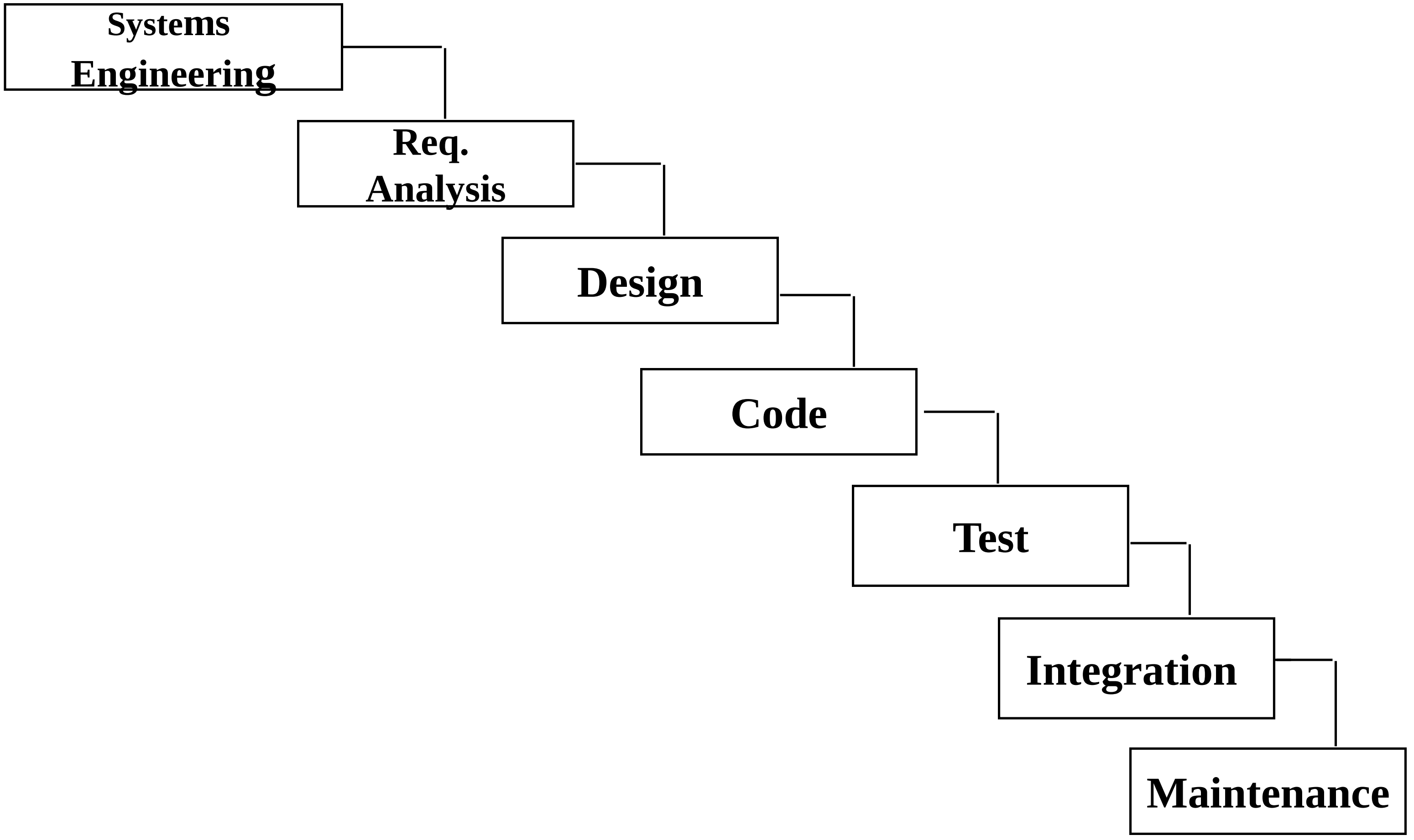
Incremental Model – An iterative and incremental approach that emphasizes flexibility and collaboration.

Spiral Model – A risk-driven model that combines iterative development with risk assessment.

Waterfall Model

- Requirement Analysis – Collect and document system and user requirements.
- System Design – Develop architecture and design specifications.
- Implementation (Coding) – Write and develop the actual software.
- Testing – Verify and validate that the software meets requirements and is free of defects.
- Deployment – Release the software for real-world use.
- Maintenance – Provide updates, bug fixes, and improvements as needed.

The Waterfall Life-cycle



Systems Engineering

Req. Analysis

Design

Code

Test

Integration

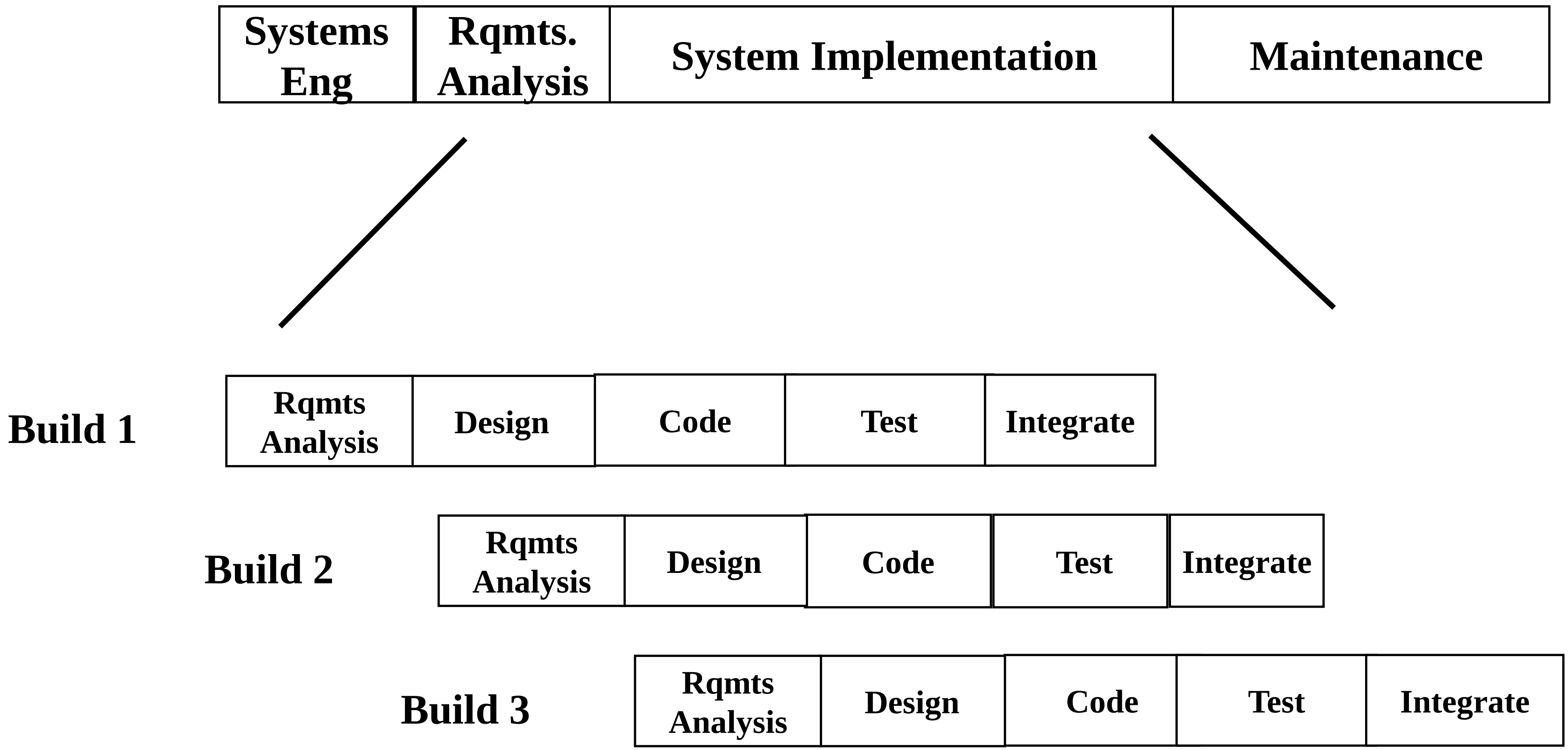
Maintenance

Incremental Model

The Incremental Model is a software development approach where the system is built and delivered in small, manageable increments rather than as a whole. Each increment adds functionality, making the software progressively more complete.

- Phased Development
- Early and Continuous Delivery
- Customer Feedback
- Flexible and Adaptive

Incremental Life-Cycle



Systems Eng | Rqmts. Analysis | System Implementation | Maintenance

Build 1: Rqmts Analysis | Design | Code | Test | Integrate

Build 2: Rqmts Analysis | Design | Code | Test | Integrate

Build 3: Rqmts Analysis | Design | Code | Test | Integrate

Spiral Model

The Spiral Model is a risk-driven software development process that combines iterative development with systematic risk management. It focuses on continuous refinement through repeated cycles, allowing for flexibility and adaptability. Each iteration or phase of the spiral involves four key activities:

- Risk-Driven
- Iterative and Incremental
- Flexibility and Adaptability
- Customer Feedback

1. Planning

Identify objectives, gather requirements, and plan the next phase.

2. Risk Analysis

Assess potential risks, develop mitigation strategies, and make decisions.

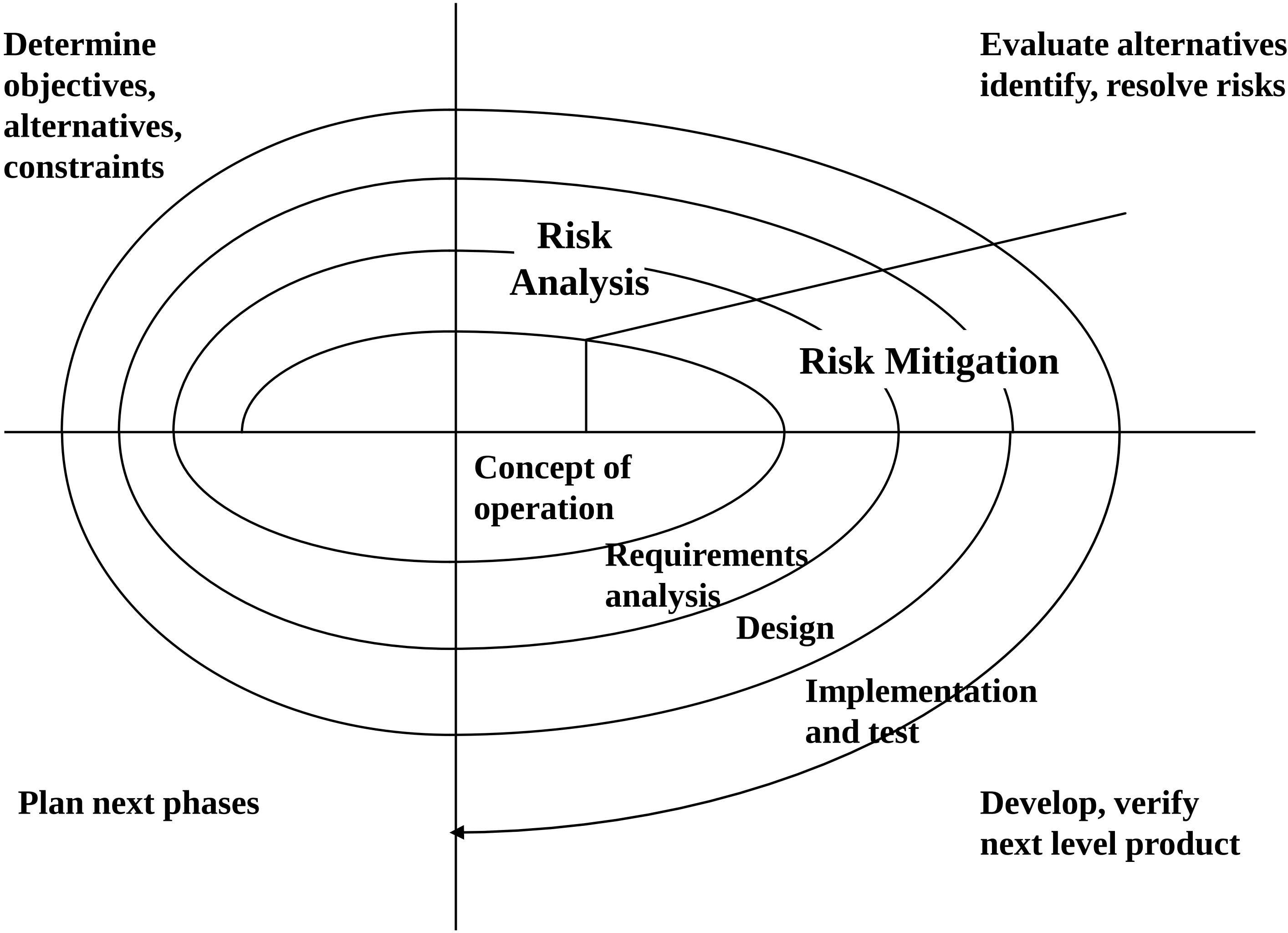
3. Engineering (Development & Testing)

Design, implement, and test the software incrementally.

4. Evaluation & Review

Gather user feedback, validate progress, and decide on the next iteration.

Spiral Life-cycle


Determine objectives, alternatives, constraints
Evaluate alternatives, identify, resolve risks
Risk Analysis
Risk Mitigation
Concept of operation
Requirements analysis
Design
Implementation and test
Plan next phases
Develop, verify next level product

Thank you