Software Engineering

Grading Docus

2. Software Engineering

Quiz 1: No Quiz 1 Quiz 2: December 1st, 2024 End term: December 22nd, 2024
Above to be attended in person at designated centres.

Eligibility to write end term exam:
Average of the best 5 out of the first 7 weekly assignment scores >= 40/100 AND submission of Group project Milestone [1-3]

Eligibility to get final course grade: Attending the End term exam AND Submission of group project (All milestones) is mandatory for course grade AND score in group project > 0

Overall score for eligible students:
GAA = score in Best 9 out of first 10 graded assignments
Qz1 = NOT THERE IN THIS COURSE
Qz2 = score in Quiz II (0, if not attempted)
Group Project- Milestone 1-3 (After week 6) - GP1
Group project - Milestone 4-6 (After week 12) - GP2
Project Presentation - PP
Course participation activity - CP
F - score in End Term exam
T = 0.05GAA + 0.2Qz2 + 0.4F+ 0.1GP1 + 0.1GP2 + 0.1PP + 0.05CP
(More details about the Group project will be given in the course).

Graded Assignment	5 marks
Quiz 2	20 marks
End term	40 marks
Group project 1	10 marks
Group Project 2	10 marks
Project Presentation	10 marks
Course participation activity	5 marks

Last term Projects as a reference

1 click here
2
click here
3
click here

Week 1

Thinking of Software in terms of Components

Software can be divided into separately addressable components called **modules** that are integrated to satisfy requirements

Example

Amazon has several components.

- **Frontend Components**
    - User Interface (UI)
    - Search and Navigation
    - Product Display Pages
    - Cart Management
- **Backend Components**
    - Product Database
    - Order Management System (OMS)
    - Payment Gateway
    - Authentication Services
    - Recommendation Engine
- **Middleware and APIs**
    - Search APIs
    - Payment APIs
    - Notification Services
    - Delivery Tracking API
- **Supporting Services**
    - Customer Support
    - Feedback and Reviews Management
    - Analytics and Metrics:
- **Security and Compliance**
    - Data Encryption
    - Fraud Detection Systems:

Software Development Process

- **Requirement Specification** is the first step in the software development process.
    - Requirements are the **goals** the implemented system must meet. They should cater to the specific needs of the clients.
- It involves understanding what the software should do and identifying the goals it needs to achieve.

Who Are the Clients?

Clients can be classified into three types:

1. External Users (End-Users)

- Software serves people outside the company.
  Example: A **mobile banking app** is built for bank customers to check balances, transfer money, etc.

2. Internal Clients (Within the Company)

- Software is used by employees or departments inside the company.
  Example: An **employee resource portal** for internal HR tasks like leave applications or pay slips.

3. Other Software (System-to-System)

- The client is another software that interacts with yours.
  Example: A **payment gateway** (e.g., Razorpay) works with an **e-commerce system** to process payments.

Understanding the Clients

- Think about **who will use the software**, **how they will use it**, and **why they need it**.
  Example:
  - **Who**: College students.
  - **How**: Through a mobile app to track assignments.
  - **Why**: To improve academic performance by managing deadlines better.

Software Design and Development

It Involves collaboration among developers, designers, testers, and other stakeholders. Each team member plays a role in transforming requirements into a functional software system

Difficulties might arise if you start coding directly?

1. **Inconsistent Implementation**: Different developers may interpret requirements differently, leading to mismatched functionality.
   - **Example**: Two developers might design login pages differently without a clear structure.
2. **Integration Issues**: Combining code from multiple developers becomes challenging without a unified design.

3. **Difficulty Adding Features**: Without a big-picture view, adding new features can disrupt the system.
    - **Example**: Adding a payment feature to an e-commerce app without planning could interfere with existing cart and checkout functions.

Software Design

- Software design is the **big-picture view** that provides a structure for the system.
- **Purpose**:
    - Acts as a blueprint for developers.
    - Ensures the system meets client needs and can adapt to future changes.

Software development

Software development is the process of designing, coding, testing, and maintaining software applications to meet specific user needs. It involves translating requirements into a functional program.

Example: Building a food delivery app that allows users to browse restaurants, place orders, and track deliveries.

Key Phases in the Software Development Process

1. Requirement Specification:

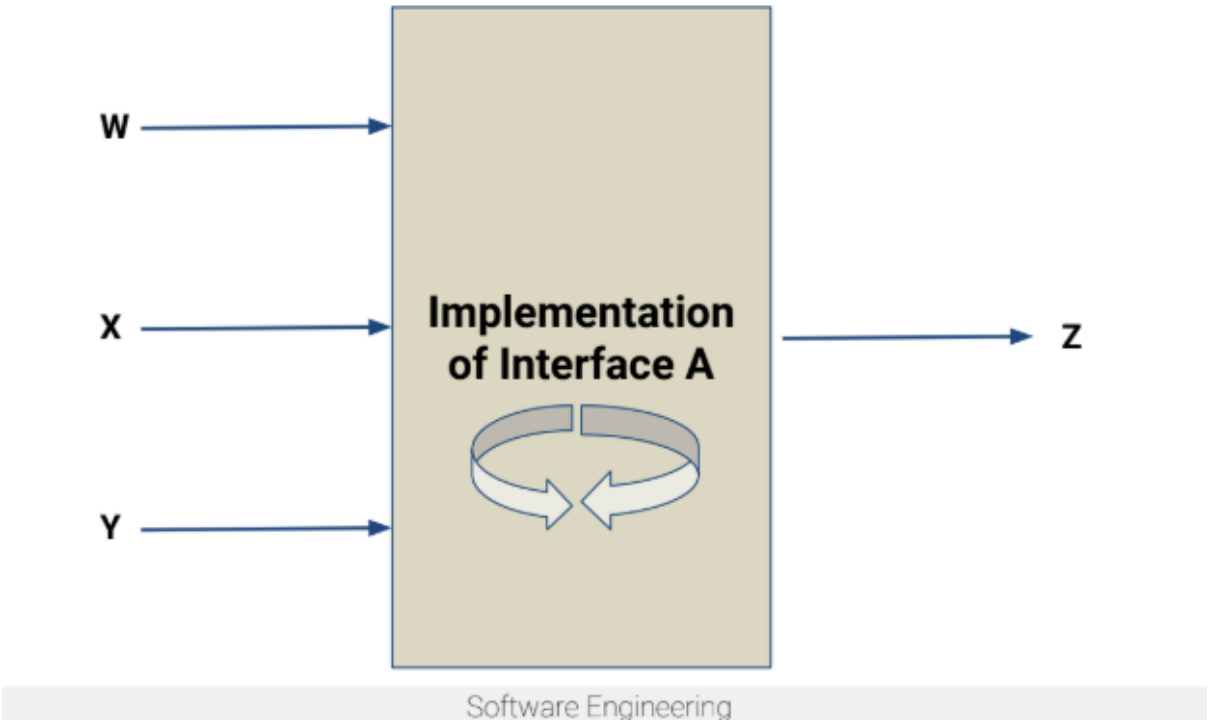
- Focuses on understanding **what the software should achieve**.
- Ensures client needs are clearly defined.
  Example: A banking app must allow users to transfer money, check balances, and pay bills.

2. Design:

- Provides a **big-picture view** of how the system will function and interact.
- Outlines the **structure** of the software.
  Example: Defining modules for a shopping app like **Product Catalog**, **Cart**, and **Payment**.

3. Development:

- Developers write code based on requirements and design.
- Tasks are **distributed** among team members.
- Clear documentation and interface definitions are crucial for consistency.
  Example:
    - **Developer W** works on the search functionality.
    - **Developer X** builds the checkout system.



Challenges During Development

1. **Coordination Issues**: Without clear communication, overlapping or missing functionality might occur.
2. **Lack of Documentation**: Leads to confusion during implementation.
3. **Adding New Features**: Requires understanding how the new feature fits into the system.

Development Phase in Action

- **Implementation of Interfaces**: Code is written to fulfill the design.
  Example:
  - **Interface A**: Payment Gateway (handles transactions).
  - **Developer W**: Adds credit card payments.
  - **Developer X**: Adds wallet payments.

Software Testing and Maintenance

Software testing

Software testing is the **process of evaluating a software application** to ensure it meets the specified requirements and is free of bugs. It helps verify that the software functions as intended in various scenarios.

Why is Testing Necessary?

- Ensures the software **behaves as per requirements.**
- Prevents bugs that could lead to financial or functional failures.
    - **Example**: In 2002, bugs caused $59.5 billion in losses; by 2016, this rose to $1.1 trillion.

Types of Testing

1. **Unit Testing**:
    - Tests individual components or functions of the software.
    - **Example**: Testing a single "login" function in an app.
2. **Integration Testing**:
    - Ensures that different modules work together.
    - **Example**: Checking how the "cart" and "payment" modules interact in an e-commerce app.
3. **Acceptance Testing**:
    - Conducted to ensure the software meets client requirements.
    - **Example**: A client testing if the payroll system calculates salaries correctly.

Testing Methodologies

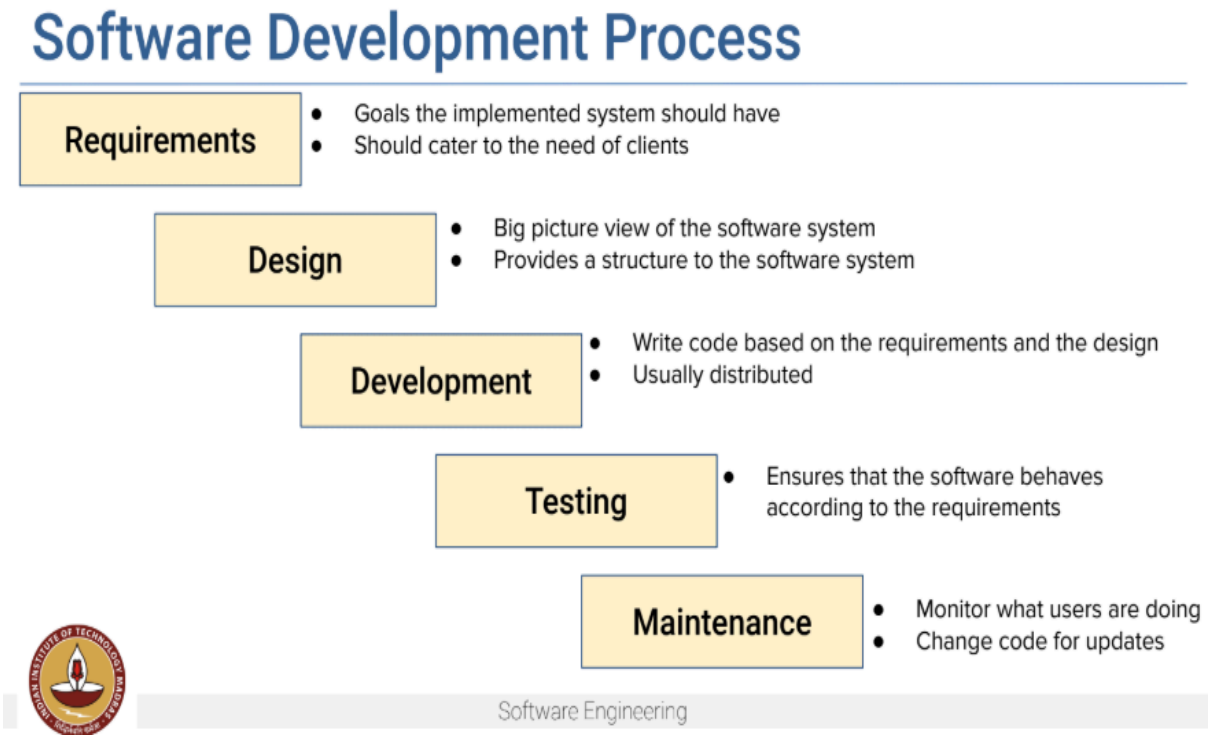
1. **Alpha Testing**:
    - Done by internal employees in a controlled environment.
    - **Goal**: Identify as many issues as possible before release.
2. **Beta Testing**:
    - Conducted by actual users in real-world settings.
    - **Example**: Releasing a gaming app to select users for feedback before a full launch.\

Software maintenance

It is the process of updating and improving software after its release to fix issues, enhance performance, add new features, or adapt it to changing user needs and environments

Maintenance Phase

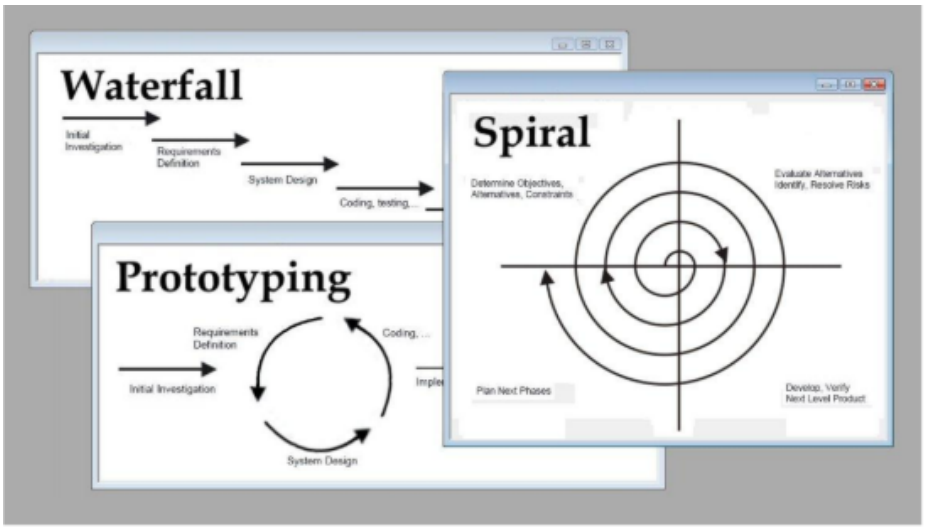
- Happens **after the software is released** to ensure smooth operation and user satisfaction.
- **Purposes**:
    - Monitor user behavior and software usage.
    - Update or upgrade features.
    - Fix bugs and ensure compatibility with new environments.
    - **Example**: Adding dark mode to an app post-release.



Software Development Lifecycle (SDLC)

1. **Requirements**: Define the software's goals based on client needs.
2. **Design**: Create a blueprint of the system.
3. **Development**: Write and integrate code.
4. **Testing**: Verify functionality and fix bugs.
5. **Maintenance**: Update and monitor the software.

Software Development Models



Waterfall Model

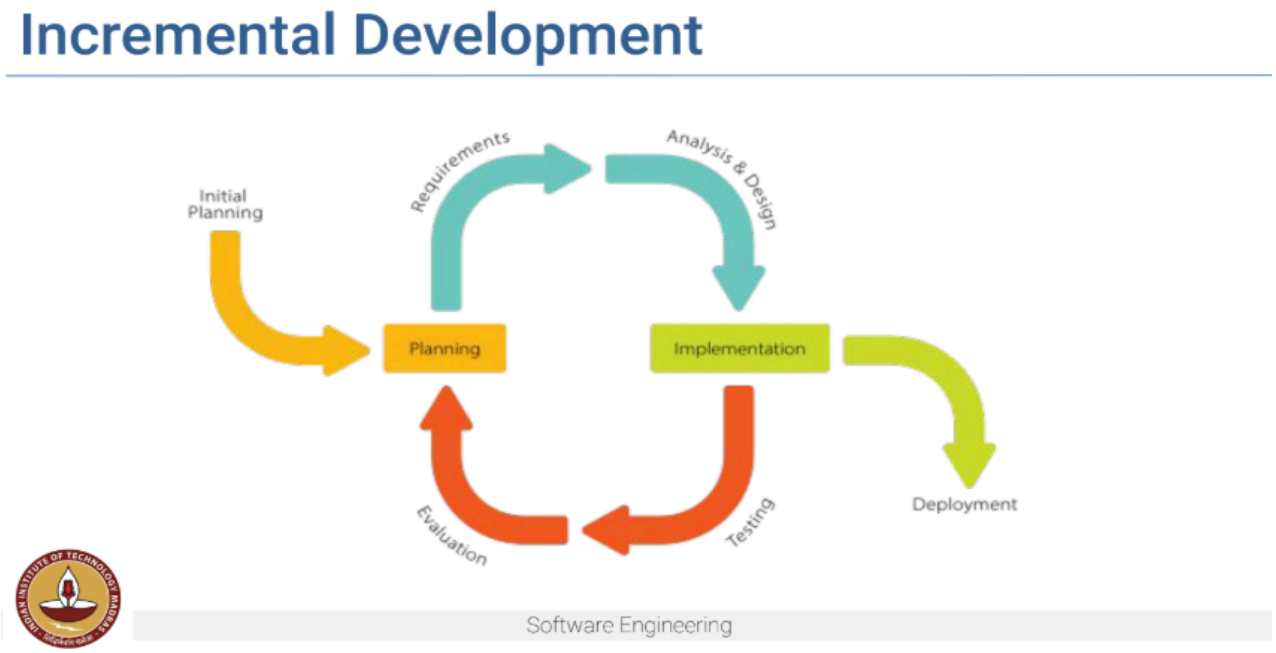
- Sequential phases: Requirements → Design → Development → Testing → Release.
- **Drawbacks**:
    - Long development cycles (6–18 months).
    - Expensive and time-consuming to make changes later.
    - Clients may not know their exact needs initially.
    - **Example**: If the client dislikes the final product, revising it is costly.

Prototype Model

- Build a **prototype** to get feedback before actual development.
- **Advantages**:
    - Helps understand unclear requirements.
    - Collects early client feedback.
- **Disadvantages**:
    - High initial costs.
    - Bugs may persist into final development.
    - **Example**: Creating a prototype of a chatbot interface for feedback.

Spiral Model

- **Iterative and incremental** development with **regular feedback.**
- Refines the software with each iteration.
- **Example**: Developing a photo editing app where features like filters and cropping are added in stages.



Agile Model

- Focuses on **collaboration, flexibility, and delivering small, functional software increments.**
- **Principles**:
  - Individuals and interactions over processes and tools.
  - Working software over comprehensive documentation.
  - Customer collaboration over contract negotiation.
  - Responding to change over following a plan.
- **Practices**:
  - User stories
  - , sprints,
  - daily stand-up meetings,
  - test-driven development.
- **Examples**:
  - **Scrum**: Regular sprints to deliver small increments.
  - **Kanban**: Managing tasks on a visual board.