

Software Engineering

1st Internal Assessment Test

2 marks

1. Define Software.

- Software is more than just a program code. A program is an executable code, which serves some computational purpose. Software is considered to be collection of executable programming code, associated libraries and documentations. Software, when made for a specific requirement is called software product.

2. List the different types of softwares

- Based on application, softwares are classified into two:
 1. System Software
 2. Application Software
- Based on usage, softwares are classified into two:
 1. Generic software
 2. Bespoke software

3. Mention the steps in generic process model.

- The generic process model encompasses the following five steps:
 - Communication
 - Planning
 - Modelling
 - Construction
 - Deployment

4. Define requirement engineering

- Requirements engineering is the systematic process of defining, documenting, and maintaining system needs to ensure a software product meets stakeholder expectations.

5. Mention the key stages of requirement engineering process.

- Inception
- Elicitation
- Analysis & Elaboration
- Negotiation
- Specification
- Validation
- Management

6. List the key requirement elicitation techniques in requirement engineering.

- Interviews
- Workshops and Brainstorming
- Observation (Contextual Inquiry)
- Prototyping
- Document Analysis
- Use Cases and Scenarios

7. Define a. Scenario Based Model

- Scenario-based models: Show how users interact with the system using use cases, activity diagrams, and user stories.

b. Behavioral Model

- Behavioral models: Depict state changes and how the system reacts to external events using state diagrams.

1. Briefly explain the various software myths

- Software myths are beliefs about software and the process used to build it. They appear reasonable but create false expectations.

Management Myths

- Standards and procedures provide everything developers need. In reality, standards may be unused, outdated, incomplete, inflexible, or inefficient.
- Adding programmers to a late project will help it catch up. In reality, educating newcomers reduces productive development effort and can make the project later.
- Outsourcing allows management to relax. In reality, an organization that cannot manage software internally will also struggle to control outsourced work.

Customer Myths

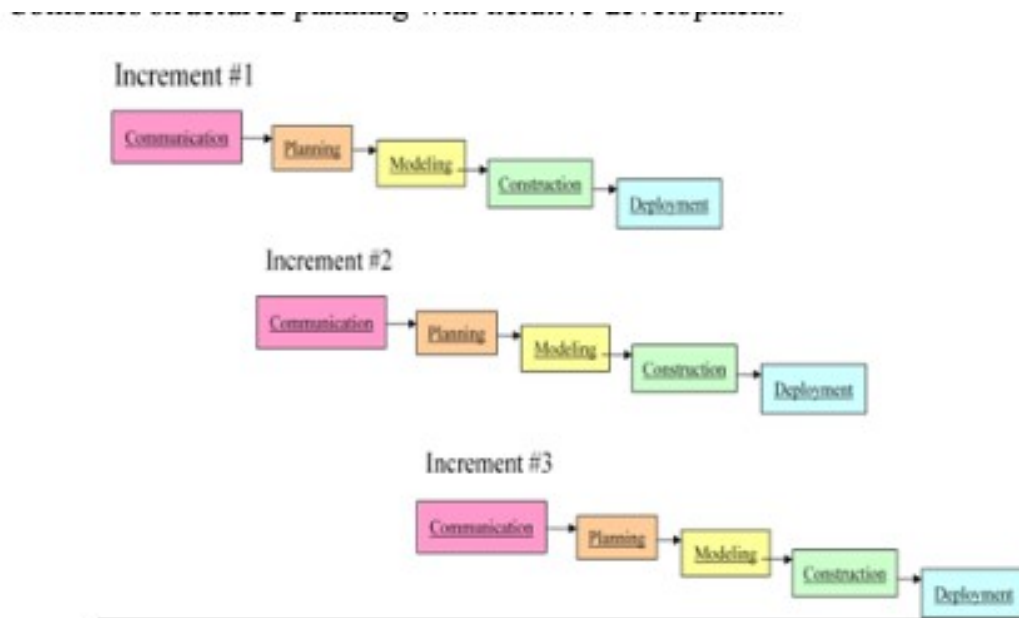
- A general statement of objectives is enough to begin coding. In reality, clear requirements require effective and continuous customer-developer communication.
- Requirement changes are easy because software is flexible. In reality, the cost and disruption of change increase when it is introduced later.

Practitioner Myths

- The job is finished when the program works. In reality, 60 to 80 percent of software effort may occur after first delivery.
- Quality cannot be assessed until the program runs. In reality, formal technical reviews can identify defects from the beginning of a project.
- The working program is the only deliverable. In reality, documentation and other software configuration elements are essential for engineering and support.
- Software engineering creates unnecessary documentation and slows delivery. In reality, better quality reduces rework and results in faster delivery.

2. With the help of an illustrative diagram explain incremental process model.

- The Incremental Process Model is a software development approach where the system is built and delivered in small, manageable parts called increments. Each increment adds new features until the complete system is developed.
- Develops the system in small, manageable increments.
- Allows changes and new features in later increments.
- Combines structured planning with iterative development.
- In this life cycle model, the software is first broken down into several modules which can be incrementally constructed and delivered.
- Used when requirements are well understood o Multiple independent deliveries are identified
- Work flow is in a linear (i.e., sequential) fashion within an increment and is staggered between increments
- Iterative in nature; focuses on an operational product with each increment
- The development team first develops the core modules of the system.
- This initial product skeleton is refined into increasing levels of capability adding new functionalities in successive versions.
- Each evolutionary version may be developed using an iterative waterfall model of development.
- Provides a needed set of functionality sooner while delivering optional components later
- Useful also when staffing is too short for a full-scale development.



3. Explain the core characteristic and key phases of CBD process model.

- Component-Based Development (CBD), also called Component-Based Software Engineering (CBSE), is an evolutionary model that constructs applications using reusable components such as COTS software or internal libraries. Its iterative structure is adapted from the Spiral Model and supports evolution, risk reduction, and continuous feedback.
- It follows a "buy, don't build" approach and uses two parallel tracks: Domain Engineering manages the component library, while Component-Based Development builds the application.

Core Characteristics of a Component

- **Encapsulated:** Internal business logic is hidden from the rest of the application.
- **Independent:** A component functions as a stand-alone service provider and can be deployed on its own.
- **Composable:** External interactions occur through publicly defined interfaces or APIs.
- **Loosely Coupled:** Changing one component does not break or alter another component's behavior.

Key Phases of the CBD Process Model

- **Requirements Specification:** Analyze user requirements and identify features that may map to existing components.
- **Component Qualification:** Search internal libraries or third-party marketplaces for suitable candidate components.
- **Component Adaptation:** Modify or wrap a near-matching component to fit the system architecture and data structures.
- **Component Creation:** Build a missing component and add it to the library for future reuse.
- **System Assembly:** Integrate individual components into a cohesive software architecture using standard protocols.
- **System Testing:** Validate the integrated application for data flow, performance, and compatibility.

3. Explain the unified process model with the help of a neat diagram

- The Unified Process (UP) in Object-Oriented Analysis and Design (OOAD) is a flexible and iterative approach to developing software. It focuses on creating working software increments, collaborating with team members, and adapting to changes.

Key Principles of Unified Process

- **Iterative and Incremental:** Unified Process divides the development process into multiple iterations, with each iteration adding new functionality incrementally.
- **Use Case Driven:** The Unified Process focuses on identifying and prioritizing use cases that represent the system's functionality from the user's perspective.
- **Architecture-Centric:** The Unified Process emphasizes defining and refining the system architecture throughout the development process.
- **Risk Management:** Unified Process identifies and manages project risks proactively to minimize their impact on the project's success.
- **Continuous Validation:** Unified Process ensures continuous validation of the system's requirements, design, and implementation through reviews, testing, and feedback.

Phases of Unified Process

1. **Inception:** This is the initial phase where the project's scope, objectives, and feasibility are determined. Key activities in this phase include identifying stakeholders, defining the initial requirements, outlining the project plan, and assessing risks.
 2. **Elaboration:** In this phase, the project requirements are analyzed in more detail, and the architecture of the system is defined. Key activities include developing use cases, creating the architectural baseline, identifying key components, and refining the project plan.
 3. **Construction:** This is the phase where the actual implementation of the system takes place. Key activities include developing, testing, and integrating the system components, as well as continuously verifying that the system meets the requirements.
 4. **Transition:** In this final phase, the software is deployed to end users. Key activities include user training, final system testing, and transitioning the system to the operations and maintenance team.
- These phases are iterative, meaning that they may be revisited multiple times throughout the project to incorporate feedback, make improvements, and address changes in requirements.

The Unified Process:

