REG.NO.:

SLOT: B1

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING
CONTINUOUS ASSESSMENT TEST – I KEY
WINTER SEMESTER 2025-2026

Programme Name & Branch	**: B. Tech Computer Science and Engineering**	
Course Code and Course Name	**: CSI1007 - Software Engineering Principles**	
Date of Examination	**: January 28, 2026**	
Exam Duration	**: 90 minutes**	**Maximum Marks: 50**

1. a. Match the following Software Process Models with Software Systems.

a-iv, b-i, c-v, d-ii, e-iii

1. b. Bring out the strengths and weaknesses of the following process models: waterfall, Incremental, prototype, RAD and Spiral.

1. Waterfall Model

Strengths

1. Simple and easy to understand and use
2. Well-defined stages with clear documentation
3. Easy to manage due to rigid structure
4. Suitable for **small projects** with **stable and well-understood requirements**
5. Progress is measurable at each phase

Weaknesses

1. Inflexible; difficult to accommodate changes once a phase is completed
2. Late discovery of errors (testing comes very late)
3. Not suitable for complex or long-term projects
4. Customer involvement is minimal after requirements phase
5. High risk if requirements are misunderstood

2. Incremental Model

Strengths

1. Delivers working software early and continuously
2. Easier to manage risk as functionality is added in increments
3. Customer feedback is available after each increment
4. Changes can be accommodated between increments
5. Suitable for large systems divided into smaller modules

Weaknesses

1. Requires good planning and architecture upfront
2. Integration of increments may be complex
3. Total cost may be higher than Waterfall
4. Not suitable if the system cannot be modularized easily

3. Prototyping Model

Strengths

1. Helps in **clear understanding of user requirements**
2. Early user involvement improves user satisfaction
3. Reduces risk of requirement mismatch
4. Useful when requirements are unclear or incomplete
5. Early visibility of the system

Weaknesses

1. Users may mistake the prototype for the final system
2. Poorly designed prototypes may lead to inefficient final systems

3. Increased development time and cost
4. Documentation may be neglected
5. Not ideal for very large systems

4. RAD (Rapid Application Development) Model

Strengths

1. Very fast development using reusable components
2. High user involvement through workshops and feedback
3. Reduces development time significantly
4. Encourages component reuse
5. Suitable for time-critical applications

Weaknesses

1. Requires highly skilled developers and designers
2. Not suitable for large, complex, or performance-critical systems
3. High dependency on user availability
4. Expensive due to tools and skilled manpower
5. Not ideal where technical risks are high

5. Spiral Model

Strengths

1. Strong emphasis on **risk analysis and management**
2. Flexible and supports changes
3. Combines benefits of waterfall and prototyping
4. Suitable for large, complex, and high-risk projects
5. Continuous customer involvement

Weaknesses

1. Complex and difficult to manage
2. Requires expertise in risk assessment
3. Expensive and time-consuming
4. Not suitable for small or low-risk projects
5. Heavy documentation and planning overhead

2. <u>Explain the Unified Process (UP) Model in Software Engineering. Describe its phases, key characteristics, and advantages.</u>

Unified Process (UP) Model

The Unified Process (UP) is an iterative and incremental software development process framework that provides a disciplined approach to assigning tasks and responsibilities within a development organization. It is use-case driven, architecture-centric, and risk-focused. The Unified Process was developed by Rational Software and is the foundation for the Rational Unified Process (RUP).

Key Characteristics of the Unified Process

1. Use-Case Driven
   i. System requirements are captured and documented using use cases.
   ii. Use cases guide analysis, design, implementation, and testing activities.
2. Architecture-Centric
   i. A robust software architecture is established early.

ii. Architecture is continuously refined throughout the project.

3. Iterative and Incremental
   i. Development is divided into multiple iterations.
   ii. Each iteration delivers a partially complete but executable system.
4. Risk-Focused
   i. High-risk elements are identified and addressed early in the project.

Phases of the Unified Process

The Unified Process consists of four phases, each having specific objectives.

1. Inception Phase

1. Defines the project scope and objectives
2. Identifies key use cases and stakeholders
3. Performs feasibility analysis
4. Estimates, cost and schedule

Deliverables: Vision document, initial use cases, project plan

2. Elaboration Phase

1. Establishes the system architecture
2. Refine requirements and use cases
3. Identifies and mitigates major risks
4. Creates detailed project plan

Deliverables: Software architecture document, refined use cases, prototype

3**. Construction Phase**

1. Actual coding and implementation
2. System features are developed in iterations
3. Testing and integration are carried out

Deliverables: Executable software increments, test cases

4. **Transition Phase**

1. System is deployed to users
2. User training and documentation
3. Bug fixing and performance tuning

Deliverables: Final product, user manuals, release notes

Advantages of the Unified Process

i. Encourages early risk reduction
ii. Supports changing requirements
iii. Ensures early delivery of working software
iv. Improves software quality through continuous testing
v. Suitable for large and complex projects

The Unified Process is a flexible and structured software development model that combines the benefits of iterative development, risk management, and strong architectural focus, making it highly suitable for modern software engineering projects.

3. <u>Explain the role of prototyping in requirements engineering with a suitable example.</u>

Prototyping is a technique used in Requirements Engineering (RE) to build an early, simplified version of a software system. Its main purpose is to understand, validate, and refine user requirements before actual development begins.

1**. Clarifies User Requirements**

1. Users often find it difficult to express requirements in words.
2. A prototype gives a visual and functional representation of the system.
3. Helps users identify missing, unclear, or incorrect requirements.

Example:

In a hospital management system, doctors may not initially specify all report formats. A prototype screen showing patient records helps them suggest improvements.

2. Improves Communication Between Stakeholders

1. Acts as a common reference point for users, analysts, and developers.
2. Reduces misunderstandings caused by technical jargon.
3. Encourages active user participation in requirements elicitation.

3. Validates Requirements Early

1. Requirements can be tested and validated before coding.
2. Help confirm whether the system meets user expectations.
3. Prevents costly changes during later development stages.

4. **Identifies Usability and User Interface Requirements**

1. Reveals usability issues early in the project.
2. Helps refine layout, navigation, and workflow.
3. Ensures the system is user-friendly.

Example:

An e-commerce prototype may show that the checkout process is confusing, leading to improved UI requirements.

5. Reduces Development Risk and Cost

1. Early detection of errors reduces rework and cost.
2. Help avoid building features that users do not need.
3. Minimize the risk of project failure.

6. Helps in Requirement Prioritization

1. Users can see which features are essential.
2. Supports better decision-making on must-have vs. optional requirements.
3. Useful when time and budget are limited.

7. Supports Handling of Changing Requirements

1. Prototypes can be easily modified to reflect new requirements.
2. Especially useful in agile and iterative development.

Prototyping plays a crucial role in requirements engineering by improving requirement clarity, enhancing stakeholder communication, validating expectations early, and reducing risks. It ensures that the final system aligns closely with real user needs, leading to higher customer satisfaction and project success.

---

4. **A university plans to develop an Online Examination System (OES) that allows students to take exams remotely. Students should be able to log in, view available exams, attempt questions within a specified time, and submit their answers. The system should handle a very large number of users simultaneously, ensure data security, and provide quick response times. Identify and explain the functional and non-functional requirements of the system.**

Functional Requirements – describe the services or functions the system must provide.
Non-Functional Requirements – describe the quality attributes and constraints of the system.

(a) Functional Requirements (FR)

Functional requirements specify the core functionalities of the Online Examination System.

1. User Authentication

The system shall allow students and administrators to log in using valid credentials.

2. Exam Management

The system shall allow administrators to create, modify, and schedule exams.

3. Exam Access

The system shall allow students to view and select available exams assigned to them.

4. Question Display and Answer Submission

The system should display multiple-choice and descriptive questions and allow students to submit answers.

5. Time Management

The system shall automatically start, and end exams based on the scheduled time.

6. Result Processing

The system should evaluate objective-type answers and generate results.

7. Data Storage

The system should store student responses, scores, and exam history.

(b) Non-Functional Requirements (NFR)

Non-functional requirements define quality attributes and performance constraints of the system.

1. Performance

The system should support at least 10,000 concurrent users with a response time of less than 2 seconds.

2. Security

User data and exam content must be encrypted.
The system should prevent unauthorized access and cheating.

3. Reliability

The system should have 99.9% uptime during examination periods.

4. Scalability

The system should handle an increase in the number of users without performance degradation.

5. Usability

The user interface should be simple and user-friendly for students with minimal training.

6. Availability

The system should be accessible 24/7 except during scheduled maintenance.

7. Maintainability

The system should allow easy updates to exam patterns and question formats.

In this scenario, functional requirements describe what the Online Examination System does, while non-functional requirements describe how well it performs. Both are essential to ensure the system meets user expectations and quality standards.

5. What are use cases? Explain the process of developing use cases in software engineering. Illustrate your answer with a suitable example.

A use case is a technique used in requirements engineering to describe how users (actors) interact with a system to achieve a specific goal. Use cases focus on functional requirements and are written from the user's perspective, independent of implementation details.

Use cases help in understanding system behavior, validating requirements, and serving as a basis for design and testing.

Steps in Developing Use Cases

1. Identify Actors

Actors are external entities that interact with the system.

i. Primary actors: Direct users of the system
ii. Secondary actors: External systems or services

Example:
In an Online Banking System:

1. Customer
2. Bank Administrator
3. Payment Gateway

2. Identify Use Cases

Use cases represent services the system provides to actors.
Example Use Cases:

i. Login
ii. View Account Balance
iii. Transfer Funds
iv. Pay Bills

3. Define Use Case Scope and Boundaries

i. Clearly define what is inside and outside the system.
ii. Helps avoid ambiguity.

Example:
The banking system handles transactions but not third-party merchant systems.

4. Write Use Case Descriptions

Each use case is described using a structured format.
Typical Use Case Format:

i. Use Case Name
ii. Actor(s)
iii. Description
iv. Pre-conditions
v. Main Flow (Basic Flow)

vi. Alternate / Exception Flows
vi. Post-conditions

5. Develop Use Case Scenarios

Example: Use Case – Transfer Funds

i. Actor: Customer
ii. Pre-condition: Customer is logged in

Main Flow:

1. Customer selects "Transfer Funds"
2. Enters beneficiary details and amount
3. Confirms transaction
4. System processes transaction
5. Confirmation message is displayed

Alternate Flow:

Insufficient balance → transaction is rejected

Post-condition:

Amount is transferred and account balance updated

6. Create Use Case Diagram

i. Graphical representation shows actors and use cases.
ii. Improves communication among stakeholders.

7. Review and Validate Use Cases

i. Review use cases with stakeholders for correctness and completeness.
ii. Modify based on feedback.

Importance of Use Cases

i. Capture functional requirements clearly
ii. Improve communication between users and developers
iii. Serve as a basis for system design and test cases
iv. Help manage scope and requirements changes

Developing cases is a systematic and user-centered approach to requirements engineering. Well-defined use cases ensure a clear understanding of system functionality and contribute significantly to the success of software projects.

****