BG/K/TY

Reg. No: 24BBS0135

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

Final Assessment Test – November 2025

Course: **CBS1005** - **Software Engineering Methodologies**

Class NBR(s): **3268** Slot: **E1**

Time: **Three Hours** Max. Marks: **100**

- **KEEPING MOBILE PHONE/ANY ELECTRONIC GADGETS, EVEN IN 'OFF' POSITION IS TREATED AS EXAM MALPRACTICE**
- **DON'T WRITE ANYTHING ON THE QUESTION PAPER**

COs	CO Statements
CO1	Apply the system development life cycle for any Business system.
CO2	Establish software project management activities such as planning, scheduling and Estimation for the business system.
CO3	Specify the business requirements through appropriate system analysis and design. Request e-signatures.
CO4	Adapt good programming and documentation standards.
CO5	Implement and demonstrate any business system software from specification to validation and verification.

BL – Blooms Taxonomy Level (1 – Remember, 2 – Understand, 3 – Apply, 4 – Analyse, 5 – Evaluate, 6 – Create)

Answer ALL Questions

(10 X 10 = 100 Marks)

1. Consider a project where requirements are frequently changing, often mid-sprint, leading to incomplete stories and frustrated team members. Construct a response plan for this. How the team leader act to improve the handling of volatile requirements, referencing relevant agile principles and practices. **CO1 BL2**

2. A software team is reviewing a C++ module named InventoryManager used in a retail system. The class handles product data, sales processing, report generation, and file logging. The team observes that changes in one functionality often break unrelated features, and unit testing is difficult due to intertwined logic. **CO3 BL4**

Analyze the following C++ code snippet and answer the questions:

```
class InventoryManager {
public:
  void addProduct(Product p); // Product management
  void processSale(int productID, int quantity); // Sales logic
  void generateReport(); // Reporting
  void logToFile(std::string msg); // File logging
};
```

i) Given the code snippet, identify and justify the main type of cohesion present. **[2]**

ii) Given the code snippet, identify and justify the four types of coupling present. **[4]**

iii) Suggest the refactoring to improve either cohesion or coupling, and briefly justify how your change improves maintainability and testability. **[4]**

3. A university is developing an online examination system where students can register, take timed tests, and view results. During development, the team discovers late-stage flaws: missed requirements for time extension for students with accommodations, inconsistent system behaviours in concurrent user sessions, and vague definitions for system performance under load. As a requirement engineer, explain which combination of elicitation techniques and modelling methods (e.g., decision tables, event tables, state transition tables, Petri nets), and standards like SRS templates would help uncover and clarify such requirements before development. Support your answer using specific, system-relevant examples and outline how you would document these in the SRS and use cases. **CO3 BL3**

4. i) A critical client demands both early delivery and very high software quality. How would the project manager use software quality frameworks and maturity models to negotiate a realistic quality/release plan? **[4] CO4 BL3**

ii) What is software reliability? Compare at least two software reliability models, explaining the strengths, and limitations. **[6]**

5. A government agency launched a large-scale digital health record system to integrate patient data across hospitals. Despite a skilled team and significant funding, the project was delayed by over two years, exceeded its budget by 80%, and delivered a system with frequent crashes, poor usability, and incomplete data integration. Stakeholders reported that requirements kept changing, testing was insufficient, and there was no clear process for managing quality or releases. **CO1 BL4**

Analyse this failure using principles of software engineering. In your answer:

i. Explain how the absence of software engineering practices—particularly in software quality management and timely delivery—contributed to the project's failure, citing relevant statistics. **[5]**

ii. Discuss the role of software engineering as a discipline in preventing such failures in large projects. **[5]**

Support your response with real-world insights and data from known project failures.

6. A travel agency is developing a modular booking platform for flights, hotels, and taxis. During object- oriented analysis, the team identifies overlapping features in their FlightBooking & HotelBooking classes but struggles with maintaining high cohesion and low coupling. As senior designer, explain how the principles of *abstraction*, *modularity*, *specification*, *encapsulation*, and *information hiding* can be applied to restructure these classes. Illustrate your solution using a CRC card for Booking, and a UML class diagram showing improved relationships. Discuss how your approach improves code maintainability and design quality. **CO3** **BL3**

7. A banking system project team is assigned to develop an online account opening workflow, including KYC verification and fraud detection. Design a sequence diagram for the above-described activities. Also, Describe how state transition tables could model the workflow logic, documenting the KYC verification process. **[7+3]** **CO3** **BL5**

8. Explain Garvin's five dimensions of software quality with real-world software examples. How do these dimensions help organizations frame software quality goals? **[7+3]** **CO4** **BL2**

9.a) You are managing the development of a safety-critical aerospace control system with strict regulatory requirements and high uncertainty in initial design specifications. Answer the following: **CO2** **BL4**

i) Which software development life cycle model would you recommend? Justify your choice based on project characteristics **[5]**

ii) Explain how it supports iterative refinement and risk mitigation. **[5]**

OR

9.b) A software project is estimated to be 120 KLOC (Kilo Lines of Code) and is classified as a Semi-Detached mode project (a = 3.0, $b = 1.12$, $c = 2.5$, $d = 0.35$). The following cost drivers have been assessed with their respective ratings: **CO2** **BL4**

- Required Software Reliability: High (1.15)
- Database Size: Nominal (1.00)
- Product Complexity: High (1.15)
- Execution Time Constraints: Nominal (1.00)
- Main Storage Constraints: Low (1.02)
- Virtual Machine Experience: High (0.87)
- Programming Language Experience: Low (1.07)
- Application of Software Engineering Methods: High (0.82)
- Use of Software Tools: High (0.83)
- Required Development Schedule: Nominal (1.00)

All other cost drivers are assumed to be at Nominal level (1.00).

Apply COCOMO model and find the following:

i. The Effort Adjustment Factor (EAF). [3]

ii. The total Effort (E) in person-months. [3]

iii. The Development Time (Tdev) in months. [2]

iv. The number of Average Staff Required. [2]

10.a) Consider the following code CO5 BL3

```
int process(int a, int b, int c) {
  1. if (a > b) {
  2.    if (b > c) {
  3.       return a;
  4.    } else {
  5.       return c;
  6.    }
  7. } else {
  8.    if (a > c) {
  9.       return b;
  10.   } else {
  11.      return c;
  12.   }
  13. }
}
```

i. Draw the Control Flow Graph (CFG) for the above code. [3]

ii. Calculate the Cyclomatic Complexity. [2]

iii. How many test cases are needed for basis path testing? [1]

iv. List all the independent paths. [4]

OR

10.b) **Part 1:** A function calculate Grade(int marks) returns: CO5 BL3

- 'F' if marks < 40
- 'C' if 40 ≤ marks ≤ 59
- 'B' if 60 ≤ marks ≤ 79
- 'A' if 80 ≤ marks ≤ 100

Any value outside 0–100 is invalid. By applying black box testing technique,

i. List all boundary values for valid and invalid partitions. [3]

ii. Design test cases for these boundaries. [3]

Part 2:

During a code review, a tester finds that the login module uses plain-text password storage, violating the security requirement document which mandates hashing.

i. Is this a *verification* or *validation* issue? Justify. [2]

ii. Suggest two static testing techniques to detect such violations. [2]

⇔⇔⇔ BG/K/TY ⇔⇔⇔