SEN-2023-May-PYQ

Q1. [20 Marks]

a. Describe the advantages and limitations for large sized software projects?

Q2. [10 Marks]

a. Explain the Principles of Agile methodology? Discuss the difference between Agile and Evolutionary Process Model?

b. Explain V Model in detail?

Q3. [10 Marks]

a. Build an SRS Document for online student feedback system?

b. What is Feasibility Study? Discuss the different types of feasibility study.

Q4. [10 Marks]

a. Explain LOC

b. Explain different types of coupling and cohesion?

Q6. [10 Marks]

a. Explain software testing strategy and its techniques?

Q1. [20 Marks] - Answers

a. Describe the advantages and limitations for large sized software projects?

Advantages:

1. **Comprehensive Functionality:**
    - Large projects can handle **complex business processes** and serve multiple user needs.
2. **High Scalability:**
    - Designed to support **many users, transactions, and modules**, making them suitable for enterprise use.
3. **Better Resource Utilization:**
    - Allows **specialized teams** (developers, testers, designers) to work on different modules efficiently.
4. **Long-Term Value:**
    - Once developed, large systems can serve an organization for **many years** with updates.
5. **Integration Capabilities:**
    - Can **integrate with other systems** like databases, ERP, or third-party applications.

Limitations:

1. **High Development Cost:**
    - Requires **significant investment** in manpower, tools, and infrastructure.
2. **Longer Development Time:**
    - Involves **complex planning, coordination, and testing**, leading to long delivery cycles.
3. **Maintenance Complexity:**
    - Difficult to **identify and fix errors** due to system size and interdependencies.
4. **Risk of Failure:**

- Mismanagement or unclear requirements can lead to **cost overruns and project failure**.

5. **Communication Challenges:**
   - Coordination between large teams can lead to **delays and misunderstandings**.

Q2. [10 Marks] - Answers

a. Explain the Principles of Agile methodology? Discuss the difference between Agile and Evolutionary Process Model?

Principles of Agile Methodology

Agile is built on the **Agile Manifesto**, which values individuals, working software, customer collaboration, and responsiveness to change. Its **12 guiding principles** include:

1. Customer Satisfaction

- Deliver valuable software **early and continuously**.

2. Welcome Changing Requirements

- Embrace changes even late in development for **competitive advantage**.

3. Frequent Delivery

- Deliver working software in **short cycles** (e.g., 2–4 weeks).

4. Collaboration

- Developers and business stakeholders must **work together daily**.

5. Working Software Is the Measure of Progress

- Focus on **functionality**, not documentation.

6. Sustainable Development

- Maintain a **consistent pace** indefinitely.

7. Technical Excellence

- Continuous attention to **design and quality** enhances agility.

8. Self-Organizing Teams

- Best architectures and designs emerge from **autonomous teams**.

Agile vs. Evolutionary Process Model – Comparison Table

Feature	**Agile Methodology**	**Evolutionary Process Model**
Approach	Iterative and incremental	Iterative with gradual refinement
Customer Involvement	High – continuous feedback	Moderate – feedback after each prototype
Flexibility	Highly adaptive to change	Adaptive but slower to respond
Delivery	Frequent working software	Prototype evolves into final system
Planning	Lightweight and ongoing	Initial planning followed by refinement
Team Collaboration	Core principle	Less emphasized
Examples	Scrum, Kanban, XP	Spiral Model, Prototyping Model

Explain V Model in detail?

V-Model (Verification and Validation Model)

The **V-Model** is a software development model that emphasizes a parallel relationship between development activities and testing activities. It is an extension of the **Waterfall model**, where each development phase has a corresponding testing phase.

Structure of the V-Model

The model is shaped like the letter "V", representing the flow of development on the left side and testing on the right side.

Left Side – **Verification Phases**

These phases ensure that the product is being built correctly.

Phase	Description
Requirement Analysis	Gather and analyze user needs.
System Design	Define system architecture and modules.
High-Level Design	Design module interactions and interfaces.
Low-Level Design	Design internal logic of each module.
Coding	Actual implementation of the design.

Right Side – **Validation Phases**

These phases ensure that the product meets user expectations.

Phase	Corresponding to...	Description
Unit Testing	Low-Level Design	Test individual modules for correctness.
Integration Testing	High-Level Design	Test interactions between modules.
System Testing	System Design	Test the complete system functionality.
Acceptance Testing	Requirement Analysis	Validate system against user needs.

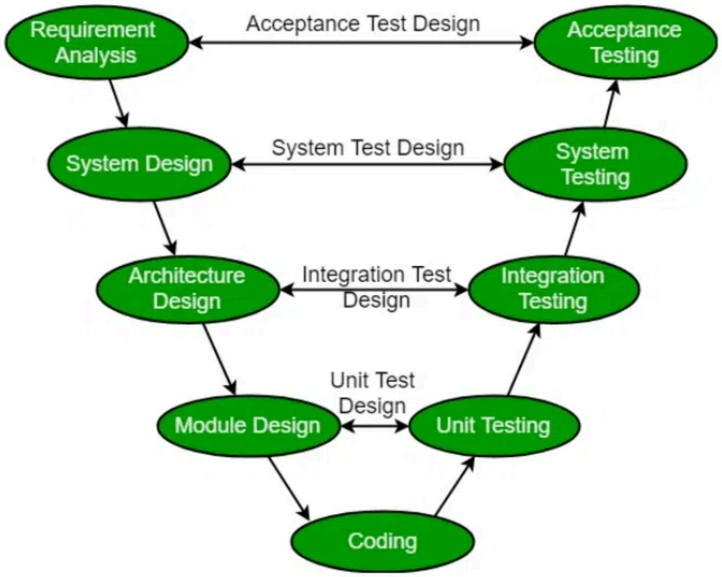
Key Features of V-Model

- Testing is planned **parallel to development**.
- Each development phase has a **matching test phase**.

- Promotes **early defect detection**.
- Suitable for **well-defined and stable requirements**.

When to Use V-Model

- Suitable for **small to medium projects** with well-defined requirements.
- Best for projects where **quality and documentation** are critical.



Q3. [10 Marks] - Answers

a. Build an SRS Document for online student feedback system?

Software Requirement Specification (SRS)

Project Title: Online Student Feedback System

1. Introduction

1.1 Purpose

The purpose of this system is to provide a platform where students can submit feedback about courses, faculty, and facilities online. It ensures transparency, quick analysis, and better decision-making for academic improvement.

1.2 Scope

- Students can log in and submit feedback securely.
- Faculty and administrators can view summarized reports.
- The system supports anonymity to encourage honest feedback.
- Feedback is stored in a database for analysis and reporting.

1.3 Objectives

- Automate the feedback collection process.
- Provide real-time analysis of student opinions.
- Improve academic quality and teaching standards.

2. Overall Description

2.1 Product Perspective

- Web-based application accessible via browsers.
- Integrated with a secure database for storing feedback.
- Role-based access: Student, Faculty, Admin.

2.2 Product Features

- Student login and feedback submission.
- Feedback categories: Course, Faculty, Infrastructure.

- Admin dashboard for viewing reports.
- Graphical representation of feedback trends.

2.3 User Characteristics

- **Students:** Basic computer literacy, submit feedback.
- **Faculty/Admin:** Moderate technical knowledge, view reports.

2.4 Constraints

- Internet connectivity required.
- Secure authentication to prevent misuse.
- Feedback must remain confidential.

3. Specific Requirements

3.1 Functional Requirements

- **FR1:** Student login using unique credentials.
- **FR2:** Submit feedback form with rating and comments.
- **FR3:** Store feedback in database with timestamp.
- **FR4:** Admin can generate reports and view statistics.
- **FR5:** Faculty can view feedback related to their courses.

3.2 Non-Functional Requirements

- **Performance:** System should handle multiple submissions simultaneously.
- **Security:** Data encryption for login and feedback storage.
- **Usability:** Simple and intuitive interface.
- **Reliability:** Ensure accurate storage and retrieval of feedback.

4. System Models

4.1 Use Case Diagram (Textual Description)

- **Actors:** Student, Admin, Faculty.
- **Use Cases:** Login, Submit Feedback, View Reports, Generate Statistics.

4.2 Deployment Diagram (Textual Description)

- **Nodes:**
  - Student Device (Browser)
  - Web Server (Application Logic)
  - Database Server (Feedback Storage)
  - Admin Device (Dashboard Access)

b. What is Feasibility Study? Discuss the different types of feasibility study

Feasibility Study

A **feasibility study** is an analysis conducted at the early stage of software development to determine whether a proposed project is **practical, achievable, and worth pursuing**.
It evaluates the project from multiple perspectives to minimize risks and ensure successful implementation.

Types of Feasibility Study

1. **Technical Feasibility**

- Examines whether the required technology, tools, and expertise are available.

- Example: Can the system be developed using existing programming languages, hardware, and networks?

2. **Economic Feasibility**

- Also called **Cost-Benefit Analysis**.
- Evaluates whether the project is financially viable.
- Example: Will the benefits (automation, efficiency) outweigh the development and maintenance costs?

3. **Operational Feasibility**

- Assesses whether the proposed system will function effectively in the organization.
- Example: Will users accept and adapt to the new system?

4. **Legal Feasibility**

- Ensures the project complies with laws, regulations, and licensing requirements.
- Example: Data privacy laws, intellectual property rights.

5. **Schedule Feasibility**

- Determines if the project can be completed within the required timeframe.
- Example: Can the system be delivered before the academic year starts?

6. **Social Feasibility**

- Evaluates the impact of the system on stakeholders and society.
- Example: Will the system improve student satisfaction and faculty transparency?

Q4. [10 Marks] - Answers

a. Explain LOC.

Definition:

Lines of Code (LOC) is a **software size estimation metric** that measures the **number of lines written in the source code** of a program.

Key Points:

1. **Purpose:**
    - Used to **estimate effort, cost, and productivity** in software development.
    - Example: Higher LOC generally means more work and higher cost.
2. **Types of LOC:**
    - **Physical LOC:** Counts every line (including comments and blanks).
    - **Logical LOC:** Counts only **executable statements**.
3. **Effort Estimation:**
    - Total effort = Productivity rate × LOC
    - Example: If 5000 LOC are written and the productivity rate is 10 LOC/hour,

      → Effort = 500 hours.
4. **Advantages:**
    - Simple to calculate and understand.
    - Useful for comparing productivity across projects.
5. **Limitations:**
    - Varies with programming language.
    - Does not measure **code quality or complexity**.
    - Not suitable for early design stages when code isn't written yet.

b. Explain different types of coupling and cohesion?

Coupling and Cohesion in Software Engineering

Coupling and **cohesion** are key design principles that determine the quality and maintainability of software modules.

Types of Coupling

Coupling refers to the degree of interdependence between software modules. Lower coupling is preferred for flexible and maintainable systems.

Type	Description
Content Coupling	One module directly modifies or relies on the internal workings of another. Most tightly coupled and least desirable.
Common Coupling	Modules share global data. Changes in shared data affect all modules.
External Coupling	Modules share externally imposed data formats or communication protocols.
Control Coupling	One module controls the flow of another by passing control information (e.g., flags).
Stamp Coupling	Modules share composite data structures but only use part of the data.
Data Coupling	Modules share only necessary data through parameters. Most desirable form of coupling.

Types of Cohesion

Cohesion refers to how closely related the functions within a single module are. Higher cohesion is preferred for clarity and reusability.

Type	Description
Coincidental Cohesion	Random functions grouped together. No meaningful relationship.
Logical Cohesion	Functions perform similar activities (e.g., input/output) but are unrelated.

Type	Description
Temporal Cohesion	Functions executed at the same time (e.g., initialization).
Procedural Cohesion	Functions follow a specific sequence of execution.
Communicational Cohesion	Functions operate on the same data set.
Sequential Cohesion	Output from one function is input to the next.
Functional Cohesion	All functions contribute to a single, well-defined task. Most desirable form of cohesion.

Q.6 [10 Marks] - Answers

a. Explain software testing strategy and its techniques?

What Is a Software Testing Strategy?

A **software testing strategy** defines the overall approach to testing a software product. It ensures that testing is systematic, efficient, and aligned with project goals.

Objectives:

- Detect defects early.
- Validate software against requirements.
- Ensure reliability, performance, and usability.
- Minimize risks before deployment.

Common Software Testing Techniques

Here are the major techniques used within a testing strategy:

1. **Unit Testing**

- Tests individual components or functions.
- Performed by developers.
- Ensures each unit works as expected.

2. **Integration Testing**

- Verifies interactions between modules.
- Detects interface defects and data flow issues.
- Can be top-down, bottom-up, or sandwich approach.

3. **System Testing**

- Tests the complete software system.
- Validates overall functionality, performance, and compliance.
- Includes functional and non-functional testing.

4. **Acceptance Testing**

- Conducted by end users or clients.
- Confirms the system meets business requirements.
- Includes alpha and beta testing.

5. **Regression Testing**

- Ensures new changes don't break existing functionality.
- Re-runs previous test cases after updates.

6. **Smoke Testing**

- Quick check to verify basic functionality.

- Performed before deeper testing begins.

7. **Sanity Testing**

- Focused testing on specific areas after minor changes.
- Ensures targeted fixes work correctly.