Software Engineering

MOD I - 2 marks

1. What is the Personal Software Process (PSP)?

- PSP is a structured process for individual software engineers to improve productivity and quality.
- It includes planning, development, and postmortem phases to track defects and time spent.
- Uses statistical methods for performance measurement and continuous improvement.
- Example: A developer records time spent on coding and defects found to optimize future projects.

---

2. What is the Software Development Life Cycle (SDLC)?

- SDLC is a systematic approach to developing software through defined phases.
- Phases include planning, analysis, design, implementation, testing, and maintenance.
- Helps in improving software quality and project management.
- Example: In SDLC, a banking app undergoes requirement gathering before coding starts.

---

3. What are umbrella activities in software engineering?

- Umbrella activities support core SDLC phases to ensure software quality.
- Includes risk management, configuration management, quality assurance, and project tracking.
- These activities run parallel to development activities for better process control.
- Example: Formal technical reviews are conducted to identify defects early.

---

4. What are process patterns in software engineering?

- Process patterns provide reusable solutions for common software development problems.
- They help in improving efficiency, reducing errors, and maintaining consistency.
- Classified into task patterns, stage patterns, and phase patterns.
- Example: A team follows the "Code Review" pattern to ensure bug-free code before deployment.

MOD II - 2 marks

---

1. What is the Agile Manifesto?

- The Agile Manifesto emphasizes flexibility, collaboration, and customer satisfaction.
- It values individuals & interactions, working software, customer collaboration, and responding to change.
- Focuses on iterative development with continuous feedback.
- Example: Agile teams adapt requirements frequently based on stakeholder feedback.

---

2. What are Prescriptive & Specialized Process Models?

- Prescriptive models define structured phases for software development (e.g., Waterfall, Incremental).
- Specialized models address specific needs, like Component-Based Development and Formal Methods.
- Prescriptive models provide discipline, while specialized models offer flexibility for unique cases.
- Example: The Spiral model is a prescriptive model, while Agile is a specialized model.

---

3. What are the phases of the Waterfall model?

- **Requirement Gathering:** Understanding project needs and constraints.

- **Design:** Creating system architecture and detailed specifications.
- **Implementation & Testing:** Writing code and validating it against requirements.
- **Deployment & Maintenance:** Delivering the software and handling updates.
- Example: A payroll system follows Waterfall, where requirements are fixed before development.

---

4. What is concurrent development in software engineering?

- Concurrent development allows multiple software processes to execute simultaneously.
- Activities like design, coding, and testing happen in parallel to speed up delivery.
- Helps in handling dependencies and reducing bottlenecks in large projects.
- Example: A team develops the UI and backend of an e-commerce website concurrently.

---

5. What is iterative development?

- Iterative development builds software in repeated cycles, refining features progressively.
- Each iteration results in a functional version with incremental improvements.
- Helps in adapting to changing requirements without reworking the entire system.
- Example: A mobile app releases a basic version first, then adds features like payments in later iterations.

---

6. What are the roles in a Scrum team?

- **Product Owner:** Defines the product vision and prioritizes the backlog.
- **Scrum Master:** Facilitates the process, removes obstacles, and ensures Agile principles.
- **Development Team:** Cross-functional members who design, code, and test the product.
- **Stakeholders:** Provide feedback and influence product decisions.

- Example: In a startup, the CEO may act as the Product Owner while developers form the team.

MOD III - 2 marks

---

1. What are requirements in software engineering?

- Requirements define what a system should do and how it should perform.
- They are categorized as functional (what it does) and non-functional (how it performs).
- Serve as a foundation for design, development, and testing.
- Example: A banking app requires login authentication (functional) and must respond within 2 seconds (non-functional).

---

2. What is feasibility study implementation?

- Evaluates if a project is viable before development begins.
- Includes technical, economic, legal, operational, and schedule feasibility.
- Helps in decision-making and risk assessment.
- Example: A company checks if it has enough resources to build an AI chatbot before starting development.

---

3. What are the different origins of changes in software?

- **Business Changes:** Market demands or company strategies evolve.
- **Technology Changes:** New tools or platforms replace old ones.
- **Customer Feedback:** Users request modifications or improvements.
- **Regulatory Changes:** Laws and standards require compliance updates.
- Example: A payment app updates its security features due to new banking regulations.

---

4. What are the information domain characteristics of function points?

- **External Inputs (EI):** User-provided inputs processed by the system.
- **External Outputs (EO):** Processed data delivered to the user.
- **Internal Logical Files (ILF):** Data stored within the system.
- **External Interface Files (EIF):** Data exchanged with other systems.
- Example: An inventory system has ILFs for stock data and EO for generating reports.

---

5. What are the activities for Requirement Elicitation and Analysis?

- **Stakeholder Interviews:** Gathering insights from users and clients.
- **Surveys & Questionnaires:** Collecting structured feedback.
- **Prototyping:** Developing mock-ups for better understanding.
- **Document Analysis:** Reviewing existing system records.
- Example: A software team uses prototyping to refine a hospital management system's interface.

---

6. What is requirement validation in requirements engineering activity?

- Ensures that gathered requirements are correct, complete, and feasible.
- Involves checks for consistency, correctness, and testability.
- Uses techniques like reviews, prototyping, and test case generation.
- Example: A banking system requirement is validated by simulating a money transfer process.

Software Engineering

MOD I - 5 marks

1. What is Layered Technology in software engineering?

1. Layered technology in software engineering provides a structured approach to software development.
2. It consists of multiple layers, each handling specific tasks to ensure systematic development.
3. The four key layers are **Quality Focus, Process, Methods, and Tools**.
4. **Quality Focus** ensures that every phase follows high standards and minimizes defects.
5. **Process Layer** defines the software development framework, guiding planning, execution, and management.
6. **Methods Layer** includes techniques for analysis, design, coding, and testing.
7. **Tools Layer** provides automation for tasks such as coding, debugging, and version control.
8. This layered approach ensures efficient development, better maintainability, and reduced errors.
9. It helps teams follow industry standards and best practices throughout development.
10. Example: A software company uses Agile (process layer), UML diagrams (methods layer), and Git (tools layer) to develop applications.

---

2. What is the Process Framework in software engineering?

1. A process framework defines a structured sequence of activities for software development.
2. It consists of five core phases: **Communication, Planning, Modeling, Construction, and Deployment**.
3. **Communication** involves gathering requirements and discussing project scope with stakeholders.
4. **Planning** establishes project timelines, resources, cost estimation, and risk analysis.

5. **Modeling** covers system design, architecture, and database structure.
6. **Construction** includes coding, integration, and testing of software components.
7. **Deployment** involves delivering the final product, collecting feedback, and performing maintenance.
8. Process frameworks can follow different models, such as Waterfall, Agile, or Spiral.
9. It ensures a systematic approach to software development, reducing project failure risks.
10. Example: A banking software follows a structured process framework for requirement gathering, planning, designing, coding, and testing.

---

3. What is the concept of Process Patterns?

1. Process patterns are reusable solutions that help in structuring software development processes.
2. They act as best practices for common problems in software engineering.
3. Process patterns are categorized into **Task Patterns, Stage Patterns, and Phase Patterns**.
4. **Task Patterns** define reusable solutions for specific tasks like code reviews or bug tracking.
5. **Stage Patterns** help in structuring phases, such as requirement gathering or testing.
6. **Phase Patterns** cover broader stages of software development, like iterative or waterfall approaches.
7. They improve efficiency by reducing redundancy and providing tested solutions.
8. Process patterns ensure consistency in development across different projects.
9. They help in project planning, resource allocation, and risk management.
10. Example: A team follows the "Daily Stand-up Meeting" pattern in Agile to improve team communication.

---

4. How do ISO standards improve software process quality?

1. ISO (International Organization for Standardization) provides global standards for software quality improvement.
2. **ISO 9001** focuses on overall quality management in software engineering.
3. **ISO/IEC 12207** defines standard processes for software lifecycle activities.
4. **ISO/IEC 15504 (SPICE)** evaluates and improves software process maturity.
5. These standards help in setting benchmarks for process efficiency and consistency.
6. Compliance with ISO standards improves reliability, security, and maintainability of software.
7. It enhances customer satisfaction by ensuring high-quality software delivery.
8. ISO certification boosts an organization's credibility and competitiveness in the market.
9. Standards provide guidelines for documentation, testing, and risk management.
10. Example: A company follows ISO 9001 to improve software quality by establishing a structured development process.

---

5. What is project planning and risk management in software engineering?

1. **Project Planning** involves defining the project's scope, schedule, budget, and resources.
2. It includes identifying deliverables, setting milestones, and allocating tasks to team members.
3. Planning ensures that development follows a structured and time-efficient approach.
4. **Risk Management** identifies potential risks and formulates strategies to mitigate them.
5. Risks in software development can include scope creep, technical failures, and resource shortages.
6. It involves **Risk Identification, Risk Analysis, Risk Mitigation, and Risk Monitoring**.
7. Risk analysis determines the probability and impact of each identified risk.
8. Mitigation plans are developed to reduce or eliminate risks before they affect the project.
9. Regular monitoring helps in detecting new risks and adjusting strategies accordingly.

10. Example: A software company anticipates a possible delay in API integration and prepares a backup solution in advance.

MOD II - 5 marks

---

1. What is the difference between the Waterfall model and the Spiral model?

Feature	Waterfall Model	Spiral Model
Approach	Follows a linear, sequential process	Uses an iterative, risk-driven approach
Flexibility	Changes are difficult to incorporate	Allows modifications at every phase
Risk Management	Minimal risk analysis	Strong focus on risk identification
User Involvement	User involvement only at the beginning and end	Continuous user feedback in every iteration
Cost Estimation	Cost is fixed early but may change later	Cost estimation is refined at every cycle
Prototyping	No prototyping involved	Includes prototypes for better understanding
Best for	Well-defined and stable projects	Large and high-risk projects
Iteration	No iterations; each phase follows the next	Multiple iterations allow gradual refinement
Error Detection	Errors are found late in the development cycle	Errors are detected early through iterations
Example	Banking system with fixed requirements	AI-based application with evolving needs

---

2. What is the Component Assembly Model?

1. The Component Assembly Model focuses on building software using pre-built, reusable components.
2. It follows the principle of "**software reuse**" to save development time and costs.
3. Components are modular and can be assembled into different applications as needed.
4. The model improves maintainability, scalability, and reduces redundant coding efforts.

5. It includes processes like **component selection, customization, and integration**.
6. **Middleware technologies** like CORBA, COM, and JavaBeans support component-based development.
7. This approach reduces development complexity by using tested and validated components.
8. It is commonly used in **enterprise software, web applications, and distributed systems**.
9. Challenges include **component compatibility issues and dependency management**.
10. Example: An e-commerce platform uses pre-built payment gateway and inventory management components.

---

3. What is the Concurrent Process Model?

1. The Concurrent Process Model allows multiple software activities to execute in parallel.
2. Unlike sequential models, different phases like design, coding, and testing can overlap.
3. It improves development speed by reducing wait times between phases.
4. It is particularly useful in **real-time systems and complex software projects**.
5. Each process state (e.g., Under Development, Awaiting Changes) is tracked separately.
6. The model supports iterative refinements, allowing dynamic adjustments.
7. It is suitable for projects with **multiple teams working on different features**.
8. Communication and coordination among teams are crucial for success.
9. Challenges include **managing dependencies and avoiding conflicts between concurrent processes**.
10. Example: A team develops the UI and backend of a hospital management system simultaneously.

---

4. What is the Agile Development Model?

1. Agile follows an **iterative and incremental** approach to software development.

2. It focuses on **customer collaboration, adaptability, and rapid delivery**.
3. The Agile Manifesto values **working software over documentation**.
4. Development is broken into short cycles called **sprints** (1-4 weeks long).
5. Teams prioritize tasks based on a **product backlog** managed by a Product Owner.
6. Agile frameworks include **Scrum, Kanban, XP (Extreme Programming), and SAFe**.
7. Agile enables quick adaptation to **changing customer requirements**.
8. It emphasizes **continuous feedback, frequent releases, and iterative improvements**.
9. Agile requires **self-organizing teams and active customer involvement**.
10. Example: A mobile app development team releases an MVP first and then updates it based on user feedback.

---

5. What is the Incremental Model?

1. The Incremental Model develops software in **small, manageable parts (increments)**.
2. Each increment adds functional features to the existing system.
3. It combines elements of both **Waterfall and iterative models**.
4. Customers get **early access to a partially functional product**.
5. It is suitable for projects where **core features are needed first, followed by additional enhancements**.
6. Risk management is improved since **issues are identified incrementally**.
7. Testing is performed after each increment, ensuring better quality control.
8. Requires proper **planning and prioritization** of increments.
9. Works well for **medium to large-scale projects where early feedback is valuable**.
10. Example: A university management system starts with student registration, then adds exam scheduling, fee management, and results processing in increments.

MOD III - 5 marks

---

1. Scheduling diagram for railway ticket reservation application development

1. Scheduling diagrams help in visualizing the timeline for different tasks in software development.
2. The **Gantt chart** and **PERT chart** are commonly used for scheduling in railway ticket reservation systems.
3. The project is divided into **phases: Requirement Gathering, Design, Development, Testing, Deployment, and Maintenance**.
4. Each phase has **tasks with dependencies**, ensuring sequential execution.
5. **Critical Path Analysis (CPA)** helps identify the longest sequence of dependent tasks.
6. Parallel tasks such as **UI design and database schema creation** can run simultaneously.
7. The **Testing phase includes unit testing, integration testing, and system testing**.
8. The Deployment phase schedules **server setup, security testing, and user training**.
9. Regular progress tracking ensures **timely project completion**.
10. Example: In a **Gantt chart**, coding starts only after system architecture design is completed.

---

2. What is an Analysis Model?

1. The Analysis Model is a **graphical and textual representation of system requirements**.
2. It helps bridge the gap between **user requirements and software design**.
3. Major components include **Data Modeling, Process Modeling, and Behavioral Modeling**.
4. **Data Modeling** defines how data is stored, retrieved, and processed.
5. **Process Modeling** represents the **flow of data and operations** in the system.
6. **Behavioral Modeling** describes how the system reacts to **user actions and inputs**.
7. It uses **DFDs, ER diagrams, and Use Case diagrams** for better visualization.
8. Analysis models help in **identifying inconsistencies and missing functionalities** early.
9. They provide a **structured approach for software engineers and developers**.
10. Example: In an **e-commerce system**, an ER diagram represents relationships between users, orders, and products.

3. DFD Level 0 and Level 1 diagram for Library Management System

1. **Data Flow Diagrams (DFDs)** are used to represent the flow of information in a system.
2. **Level 0 DFD (Context Diagram)** provides a high-level view of the entire system.
3. The Library Management System has **entities like User, Librarian, and Database**.
4. **Processes** include **Issue Book, Return Book, Search Book, and Manage Users**.
5. The **Level 1 DFD** expands Level 0 into more detailed sub-processes.
6. **Example Level 1 Processes:** User Registration, Book Search, Book Issue, Fine Calculation.
7. Data flows between **User, System, and Database for managing books and records**.
8. It helps in understanding **how information is processed and stored**.
9. DFDs assist in **identifying redundancies and optimizing the system**.
10. Example: A student searches for a book, and the system checks availability before issuing it.

---

4. Differences between the Basic, Intermediate, and Detailed COCOMO Models

Feature	Basic COCOMO Model	Intermediate COCOMO Model	Detailed COCOMO Model
Purpose	Provides quick cost estimation	Considers cost drivers for better accuracy	Provides most accurate cost estimation
Complexity	Simple and easy to use	Moderate complexity	Highly complex with multiple parameters
Effort Calculation	Based on project size only	Uses project size and cost drivers	Includes all cost drivers with phase-wise effort distribution
Cost Drivers	No cost drivers	15 cost drivers like reliability, experience	Uses multiple cost drivers for each phase
Phases Covered	Overall estimation for the project	Covers different phases like design, coding	Provides estimation for each software phase

Feature	Basic COCOMO Model	Intermediate COCOMO Model	Detailed COCOMO Model
Accuracy	Low accuracy	Medium accuracy	High accuracy with detailed breakdown
Use Case	Small projects with limited requirements	Medium-scale projects	Large, complex software systems
Formula Used	Effort = a(KLOC)^b	Adds cost drivers to improve estimation	Uses effort multipliers for precise calculation
Development Type	Not suitable for iterative development	Works for both Waterfall and Incremental models	Suitable for any development approach
Example	A simple calculator application	An ERP system for a mid-sized company	A banking application with strict regulations

---