SOFTWARE ENGINEERING

DEC-2020

PART-A

1. **What is Software Engineering?**
   - **Software Engineering** is the systematic, disciplined, and structured approach to the **development, operation, and maintenance** of software.
   - It applies engineering principles to software creation to ensure that the final product is **reliable, efficient, cost-effective, and meets user requirements**.
   - Software engineering involves processes such as **requirements analysis, design, coding, testing, and deployment**.
   - Its goal is to produce high-quality software that is easy to maintain and works effectively in real-world environments.

2. **Define the term-Managerial issue.**
   - **Managerial issue** refers to the problems, challenges, or concerns that managers face while planning, organizing, directing, and controlling a project or an organization.
   - These issues can involve **decision-making, resource allocation, scheduling, team coordination, risk management, communication, and maintaining productivity**.
   - Managerial issues affect how smoothly a project runs and how successfully goals are achieved. They require proper planning, monitoring, and leadership to ensure effective project execution.

3. **What is Planning Activity?**
   - **Planning Activity** refers to the process of defining the **goals, tasks, resources, schedules, and strategies** required to complete a project successfully.
   - It involves identifying what needs to be done, how it will be done, who will do it, and when it must be completed.
   - Planning activities help in **estimating costs, assigning responsibilities, managing risks, and setting timelines**.
   - The main purpose of planning is to ensure that the project proceeds in a structured and organized manner, reducing uncertainties and improving the chances of achieving project objectives efficiently.

4. **What is meant by Maintenance cost?**
   - **Maintenance cost** refers to the total expense involved in **modifying, updating, and supporting software after it has been delivered to the user**.
   - These costs arise due to fixing errors, improving performance, adapting the software to new hardware or operating systems, and adding new features as user needs change.
   - Maintenance cost is often a significant part of the software lifecycle because software must remain **reliable, secure, and up-to-date** throughout its usage.
   - It includes expenses for **bug fixes, upgrades, technical support, documentation updates, and system enhancements**.

5. **What is Hierarchical Team Structure?**
   - **Hierarchical Team Structure** refers to a team organization method where members are arranged in **levels of authority**, forming a pyramid-like structure.
   - At the top is the project manager or leader, followed by sub-leaders, and then team members at the lower levels. Each level has specific duties and reports to the level above it.
   - This structure ensures **clear communication, well-defined roles, and controlled decision-making**.

- ❖ It is useful in large software projects because it allows efficient **task delegation, supervision, coordination**, and helps maintain discipline within the team.
- ❖ The hierarchical structure also supports better **monitoring of progress**, as every member knows their responsibilities and who they must report to.

6. What is Decision Table?

Decision Table is a systematic and visual method used to represent complex decision-making logic in the form of a table. It lists all possible **conditions**, their different **combinations**, and the corresponding **actions or outcomes** to be taken. A decision table typically has four parts:

1. **Condition Stub** – lists all the conditions.
2. **Condition Entries** – show the possible values for each condition.
3. **Action Stub** – lists all possible actions.
4. **Action Entries** – indicate which action applies for each condition combination.

Decision tables help ensure that no condition is missed, reduce ambiguity, and make decision logic easy to understand. They are very useful for representing **business rules, validation checks, and complex logical scenarios**, helping analysts, designers, and programmers to implement clear and error-free logic.

7. Define the term–Data Flow Diagram.

- ❖ **Data Flow Diagram (DFD)** is a graphical tool used to represent how data moves through a system. It shows the **flow of information**, the **processes** that transform the data, the **data stores** where information is kept, and the **external entities** that interact with the system.
- ❖ A DFD helps to understand the system's functionality by focusing on *what* happens to data rather than *how* it happens.
  DFDs are drawn at different levels—**Context Level (Level 0)** provides a broad overview, while **Level 1 and Level 2** diagrams show detailed processes.
- ❖ They are widely used in system analysis to identify requirements, improve clarity, detect redundancies, and communicate system behavior effectively among developers and stakeholders.

8.What is Type less language?

- o **Typeless language** (also called *untyped language*) is a programming language in which variables do not have a fixed or predefined data type.
- o In such languages, the type of data is determined only at **runtime**, not during compilation.
- o This means a variable can store **different types of values at different times**, such as numbers, strings, or lists, without needing type declarations.
- o Typeless languages provide high flexibility and faster development because programmers do not need to specify data types.
- o However, they may lead to runtime errors and make debugging more difficult since type-related mistakes are not caught early.
- o Examples of typeless or weakly typed languages include **JavaScript, PHP, and early versions of BASIC**.

9. What is Test Plan?

- **Test Plan** is a formal document that describes the **overall strategy, objectives, resources, schedule, and scope** of testing a software product.
- It outlines **what will be tested, how it will be tested, who will perform the tests, and when the tests will take place**.
- A test plan includes details such as test items, testing methods, required tools, test environment, entry and exit criteria, and responsibilities of team members.
- The main purpose of a test plan is to ensure that testing is carried out in a **systematic, organized, and controlled** manner.
- It helps identify risks, allocate resources efficiently, and provide a clear roadmap for achieving software quality before release.

---

10. What is Verification?

- **Verification** is the systematic process of evaluating software work-products (such as requirement documents, design diagrams, and source code) to ensure they meet the specified requirements and follow the correct development standards.
- It answers the question: **"Are we building the product right?"**. Verification focuses on checking the correctness and quality of intermediate outputs before the software is executed.
- It mainly involves **static techniques** such as reviews, inspections, walkthroughs, desk checking, and static code analysis.
- These methods help identify defects early, reduce development cost, and improve product quality. Verification ensures that each phase of the software development lifecycle is completed properly, the documentation is accurate, and the design logic is sound.
- It plays a crucial role in preventing errors from spreading to later stages and ensuring reliability in the final software product.

---

11. Define Quality Assurance.

- **Quality Assurance (QA)** is a planned and systematic set of activities designed to ensure that the software development process is carried out according to defined standards and procedures.
- Its main focus is on **preventing defects** by improving the processes used to develop and maintain software.
- QA ensures that every stage of the Software Development Life Cycle (SDLC) follows best practices and meets quality guidelines.
- QA includes activities such as **process definition, audits, reviews, training, documentation checking, and compliance monitoring**.
- It works on the principle of "**Do it right the first time**," ensuring that methods, tools, and techniques are applied correctly.
- The goal of QA is to ensure that the final software product is **reliable, consistent, efficient, secure, and satisfies customer expectations**.
- It also helps reduce development costs, improves productivity, and increases customer confidence in the product.

12.Define-Acceptance Testing.

- **Acceptance Testing** is the final phase of software testing conducted to determine whether the software meets the **business requirements and expectations** of the end users or clients.

- It is performed after system testing and before the software is released. The main purpose of acceptance testing is to confirm that the system is **ready for deployment** and works correctly in real-world conditions.
- It ensures that the software is **complete, correct, usable, and satisfies all functional and non-functional requirements**.
- Acceptance testing is usually carried out by the customer, user, or client, and includes methods such as **User Acceptance Testing (UAT)**, **Alpha Testing**, and **Beta Testing**.
- If the software passes acceptance testing, it is approved for delivery or implementation.

DEC 2021
PART-A

1. What is Software Engineering?
Software Engineering is the systematic application of engineering principles to the development, operation, and maintenance of software.
It aims to produce reliable, efficient, and cost-effective software that meets user requirements.
It includes various stages such as requirement analysis, design, coding, testing, and maintenance.
Software Engineering also emphasizes project management, quality assurance, and documentation.
In short, it brings discipline, methods, and tools to software development to ensure product quality and timely delivery.

---

2. What is meant by Module?
A module is a **self-contained unit** or part of a larger software system that performs a specific function.
It helps divide a large program into smaller, manageable sections.
Each module can be developed, tested, and maintained independently.
Modules communicate with each other through well-defined interfaces.
This approach improves **reusability, maintainability, and readability** of the software.

---

3. What is Planning Activity?
Planning activity refers to the process of defining the scope, objectives, schedule, and resources required for a software project.
It helps in identifying tasks, estimating time and cost, and assigning responsibilities.
Proper planning reduces project risks and ensures smooth progress.
It includes creating a **project plan**, **risk management plan**, and **quality plan**.
Thus, planning is the foundation for the successful execution of a software project.

---

4. What is Documentation?
Documentation is the process of recording all important information related to software development.
It includes requirements documents, design details, source code comments, test cases, and user manuals.
Good documentation helps developers understand the system and maintain it easily in the future.
It ensures knowledge sharing and supports new team members.
Hence, documentation improves the clarity, consistency, and maintainability of the software.

---

5. What is Hierarchical Team Structure?
A hierarchical team structure is an organization model where team members are arranged in levels based on authority and responsibility.
At the top is the project manager, followed by team leaders and then developers or testers.

This structure ensures clear communication, proper supervision, and accountability.
Each level reports to the one above it, allowing efficient control and coordination.
It is suitable for large projects that require clear roles and responsibilities.

---

6. What is Software Maintenance?
Software maintenance is the process of modifying a software product after its delivery to correct faults, improve performance, or adapt it to a new environment.
It includes **corrective**, **adaptive**, **perfective**, and **preventive** maintenance.
Maintenance ensures that the software continues to function effectively over time.
It consumes a major part of the software's total cost.
Thus, maintenance is essential for extending the life and usefulness of software.

---

7. What is Static Analysis?
Static analysis is a method of checking the source code **without executing** the program.
It helps find potential errors, coding standard violations, and security flaws.
Tools are often used for static analysis to scan and evaluate the code automatically.
It improves software quality early in the development process.
By detecting bugs early, static analysis reduces testing cost and effort.

---

8. What is Typeless Language?
A typeless language is one in which variables are not bound to any specific data type.
The type of data can change during program execution.
This means the same variable can hold different kinds of data at different times.
Examples include early versions of BASIC and some scripting languages.
Although it provides flexibility, it can lead to runtime errors due to type mismatches.

---

9. What is Software Design?
Software design is the process of defining the architecture, components, interfaces, and data for a system to meet specific requirements.
It acts as a bridge between requirements and implementation.
Design can be divided into **high-level design (architecture)** and **detailed design (modules and algorithms)**.
A good design ensures reliability, maintainability, and scalability.
Hence, it provides a blueprint for the coding phase of software development.

---

10. What is Verification?
Verification is the process of checking whether the software product meets its design and requirements specifications.
It ensures that "**we are building the product right**."
Verification activities include reviews, inspections, and walkthroughs.
It helps detect defects early, before the actual testing phase.
Thus, verification improves product quality and reduces overall project cost.

---

11. What is Quality Assurance?
Quality Assurance (QA) is a systematic process to ensure that software meets the required quality standards.
It focuses on preventing defects rather than detecting them.
QA involves process monitoring, audits, and continuous improvement.
It ensures that proper methods and tools are followed throughout development.
Hence, QA builds confidence that the final software will be reliable and meet user expectations.

---

12. What is Configuration Management?
Configuration Management (CM) is the process of systematically managing changes in software.
It involves identifying, controlling, and recording the configuration items like code, documents, and designs.
CM ensures consistency and integrity throughout the project life cycle.
It helps in version control, change tracking, and rollback if needed.
Thus, configuration management maintains stability and traceability in the development process.

Dec 2022

Part-A

1. Mention any two important general skills of a good Software Engineer.

- Problem-solving skill: A software engineer must analyze problems and find logical, effective software solutions.
- Communication skill: Must be able to communicate ideas and technical details clearly to clients and team members.
- Teamwork: Work well with other developers, testers, and managers to reach project goals.
- Adaptability: Should learn and adjust to new tools, languages, and technologies quickly.
- Time management: Must plan tasks and deliver work within the given time schedule.

2. Define Project Scheduling.

- Project scheduling means planning and arranging project tasks in a proper order.
- It decides who will do what task and when it should be done.
- Helps in using time, manpower, and resources efficiently.
- Used to monitor project progress and detect delays.
- Tools like Gantt charts, PERT, and CPM are used to prepare schedules.

3. Define Layers of Software Engineering.

Software Engineering is based on four layers:

- Quality Focus: Main layer ensuring software quality at every step.
- Process Layer: Provides the framework that guides development.
- Methods Layer: Includes technical methods like analysis, design, and coding.
- Tools Layer: Software tools that support the methods and process.
- All these layers together ensure efficient and high-quality software development.

4. List out the Software Cost Effects.

Answer:

- Human resources: Cost of developer and tester salaries.
- Software tools: Cost of design, coding, and testing software tools.
- Hardware: Machines, devices, and systems used for development.
- Training: Teaching new skills or tools to employees.
- Maintenance: Fixing errors and updating software after delivery.

→ These factors together determine the total cost of software.

5. What is the use of Process Specification?

- Process specification describes how a process or function works in a software system.
- Defines input, output, and operations clearly for each process.
- Helps developers understand the logic of each part of the system.
- Maintains consistency between design and implementation.
- Used for testing, validation, and documentation in later stages.

6. What is the goal of Requirements Management?

- To make sure software satisfies user needs and project goals.
- Helps in tracking, analyzing, and controlling changes to requirements.
- Keeps project documents and design consistent with updated requirements.
- Prevents scope creep and confusion among developers.
- Ensures clear understanding between customer and development team.

7. List any two Design Principles.

- Modularity: Divide system into small, independent modules for easy work.
- Abstraction: Show only essential details, hide unnecessary ones.
- Reusability: Design components that can be reused in other software.
- Encapsulation: Keep data and operations together to protect data.
- Simplicity: Create a design that is easy to understand and maintain.

8. What is a Walkthrough?

- A walkthrough is an informal meeting where the author of a document or code explains it to team members.
- Team members review and discuss to find mistakes and improvements.
- It helps detect errors early before testing starts.
- It does not need formal records or strict rules.
- Improves understanding, communication, and software quality.

9. What is Validation Testing?

- Validation testing ensures that the software meets user needs and expectations.
- It checks whether the right product is being built.
- Done after system and integration testing.
- Includes Alpha and Beta testing with real users.
- Ensures the final software satisfies functional and performance requirements.

10. What is meant by Static Analysis?

- Static analysis means checking the program's code without executing it.
- It is used to find errors, security issues, and bad coding practices early.
- Tools automatically examine the source code.
- Helps developers correct issues before testing.
- Improves software reliability, maintainability, and coding standards.

11. What is Software Quality Assurance (SQA)?

- SQA ensures that software meets defined quality standards.
- It includes review, audit, and testing activities during development.
- Prevents defects instead of just detecting them later.
- Improves confidence in the software product.
- Ensures high-quality, reliable, and error-free software for customers.

12. What is System Testing?

- System testing is the testing of the complete, integrated system.
- It checks if all modules work correctly together.
- Verifies both functional (what it does) and non-functional (how well it works) requirements.
- Conducted after integration testing and before user acceptance testing.
- Ensures the system performs as expected by the customer.

DECEMBER 2023

1. Define Software Engineering.

Software engineering is the disciplined and systematic approach to developing, operating, and maintaining software. It applies engineering principles to produce reliable, efficient, and cost-effective software. It includes activities such as requirements analysis, design, coding, testing, and maintenance. The main aim is to deliver high-quality software that meets user needs and performs well over time.

---

2. What is Software Project?

A software project is a planned activity to develop or enhance a software system within a specific time, budget, and scope. It involves tasks such as requirement gathering, design, coding, testing, and deployment. Every software project has goals, constraints, resources, and deliverables. Proper management ensures the project is completed successfully and meets user expectations.

---

3. What is meant by module?

A module is a small, independent unit of a software system that performs a specific function. It contains related data and functions grouped together. Modules make software easier to understand, maintain, and test. Modular design improves clarity, reduces complexity, and allows multiple developers to work independently.

---

4. What is Cost Estimation?

Cost estimation is the process of predicting the total cost, effort, manpower, and time required to complete a software project. It helps in budgeting, resource allocation, and planning. Common estimation techniques include expert judgment, analogy, Delphi, and algorithm-based methods like COCOMO. Accurate cost estimation ensures the project is completed within financial limits.

---

5. What is Planning Activity?

Planning activity involves deciding the tasks, schedule, resources, and methods needed to complete a software project. It includes defining objectives, creating timelines, identifying risks, and assigning responsibilities. Good planning improves coordination, reduces project delays, and increases the chances of successful project completion.

---

6. How to maintain a software?

Software maintenance includes modifying and updating software after delivery to improve performance or fix issues. It involves corrective maintenance (fixing bugs), adaptive maintenance (updating for new environments), perfective maintenance (enhancing features), and preventive maintenance (avoiding future problems). Effective maintenance increases software lifespan and user satisfaction.

---

7. What is Software Design?

Software design is the process of defining the architecture, components, interfaces, and data of a software system. It transforms requirements into a blueprint for developers. Design includes high-level design (overall structure) and low-level design (detailed logic). A good design reduces complexity, improves maintainability, and ensures smooth development.

---

8. What is milestone?

A milestone is a significant event or checkpoint in a project used to measure progress. It marks the completion of a key phase such as requirement analysis, design, coding, or testing. Milestones help track project status, manage deadlines, and ensure timely delivery of each stage.

---

9. Define Test Plan.

A test plan is a document that outlines the strategy, scope, objectives, resources, schedule, and methods for testing a software system. It specifies what to test, how to test, who will test, and when testing will occur. A proper test plan ensures effective detection of defects and improves software quality.

---

10. What is validation?

Validation checks whether the final software meets the customer's requirements and expectations. It ensures the product is useful and correct for its intended purpose. Validation activities include user acceptance testing, system testing, and reviews. It answers the question: "Are we building the right product?"

---

11. What is Static Analysis?

Static analysis is the process of examining source code without executing it to find errors, vulnerabilities, and coding issues. Tools automatically check for syntax errors, security flaws,

unused variables, and coding standard violations. It improves code quality early in development and reduces future defects.

---

12. Differentiate between verification and validation.

- **Verification ensures the software is built correctly according to the design and specifications. It includes reviews, inspections, walkthroughs, and static analysis.**
- **Validation ensures the software meets the user's actual needs and expectations. It includes system testing and user acceptance testing.**
  In short:
  Verification checks "Are we building the product right?"
  Validation checks "Are we building the right product?"

MAY 2021

1. What is Software Project?

A software project is a planned and organized effort to develop or modify a software system. It involves a series of activities such as requirement analysis, design, coding, testing, and maintenance.
A software project has specific objectives, a defined scope, resources (like people, time, and money), and must deliver a quality product within given constraints.
Example: Developing an online banking application or a mobile ticket booking system.
Key aspects include:

- **Project goals and deliverables**
- **Schedule and milestones**
- **Budget and resources**
- **Risk management**

---

2. What is meant by Module?

A module is a separate, independent, and functional unit of a software system that performs a specific task. Software is often divided into modules to make development, testing, and maintenance easier.
Each module can be developed and tested independently before integration.
Features of a Module:

- **High cohesion (focuses on one function)**
- **Low coupling (less dependency on other modules)**
- **Reusability and maintainability**
  Example: In an e-commerce system, modules may include "User Management," "Payment Processing," and "Order Management."

---

3. What are the Size Factors in Software Engineering?

Size factors are used to estimate the effort, time, and cost required to develop a software system. They help in project estimation and planning.
Common size factors include:

1. **Lines of Code (LOC): Number of source code lines written.**
2. **Function Points (FP): Measures functionality based on user inputs, outputs, files, and interfaces.**
3. **Number of Modules or Components: More modules mean larger system size.**
4. **Complexity: Logical or computational complexity of the software.**
5. **User Interface Elements: Number of screens, forms, or reports.**
   These factors help in productivity analysis and cost estimation models like COCOMO.

---

4. What is Documentation?

Documentation refers to the collection of written materials that describe the development, design, operation, and use of software. It serves as a communication tool among developers, testers, and users.
Types of Documentation:

- **Requirement Documentation: Describes what the system should do.**
- **Design Documentation: Details system architecture and components.**
- **Technical Documentation: Includes code comments, APIs, and configuration details.**
- **User Documentation: Manuals and help files for end users.**
  Good documentation improves software maintenance, reduces errors, and helps new developers understand the system.

---

5. What is Planning Activity?

A planning activity defines the roadmap for executing a software project successfully. It includes identifying project goals, estimating resources, scheduling activities, and managing risks.
Major Steps in Planning:

1. **Define project objectives and scope.**
2. **Estimate cost, effort, and schedule.**
3. **Identify required resources (personnel, tools).**
4. **Define milestones and deliverables.**
5. **Develop risk management and communication plans.**
   Effective planning ensures that the project is completed on time, within budget, and meets quality standards.

---

6. What is Software Maintenance?

Software maintenance is the process of modifying a software product after its delivery to correct faults, improve performance, or adapt it to a changed environment.
It is the longest phase of the software lifecycle and ensures the software remains useful and efficient.
Types of Maintenance:

1. **Corrective: Fixing errors discovered after delivery.**
2. **Adaptive: Modifying software to work in new environments.**
3. **Perfective: Enhancing performance or features.**
4. **Preventive: Making changes to prevent future problems.**
   Example: Updating an app to work with a new version of Android.

---

7. What is Static Analysis?

Static analysis is a method of examining the source code of a program without executing it. The purpose is to detect coding errors, security vulnerabilities, and violations of coding standards.
Advantages:

- **Early detection of bugs and vulnerabilities**
- **Improves code quality and maintainability**
- **Reduces testing time**
  Tools: SonarQube, Coverity, or Lint.
  Example: Detecting unused variables or unreachable code using a static analysis tool.

---

8. Name any Three Maintenance Tools.

1. **Debuggers: Used to identify and fix logical or runtime errors in the code. Example: GDB, Visual Studio Debugger.**
2. **Version Control Tools: Manage different versions of the software and track changes. Example: Git, SVN.**
3. **Automated Testing Tools: Used for regression and performance testing after changes. Example: Selenium, JUnit, LoadRunner.**
   These tools make maintenance easier, faster, and more reliable.

---

9. What is Software Design?

Software design is the process of defining the architecture, components, interfaces, and data of a system to satisfy specified requirements. It acts as a blueprint for implementation.
Types of Design:

- **High-Level Design (Architectural Design): Defines system architecture, data flow, and module interactions.**
- **Low-Level Design (Detailed Design): Defines internal logic and data structures of each module.**
  Objectives:

- **Improve modularity and reusability**
- **Ensure efficiency and scalability**
  Example: Designing a layered architecture for a web application (Presentation, Business Logic, Database layers).

---

10. What is Inspection?

Inspection is a formal review process in which software documents such as code, design, or requirements are examined by a team to identify defects early in development.
It is a type of static verification technique (no execution).
Steps in Inspection:

1. **Planning**
2. **Overview meeting**
3. **Preparation**
4. **Inspection meeting**
5. **Rework and follow-up**
   Benefits:

- **Early error detection**
- **Improved software quality**
- **Reduced testing and maintenance cost**

---

11. What is Validation?

Validation is the process of evaluating software during or at the end of development to ensure it meets customer requirements and expectations.
It answers the question: "Are we building the right product?"
Validation Techniques:

- **System testing**
- **User acceptance testing (UAT)**
- **Field trials**
  Example: Checking if an online banking system correctly transfers funds between accounts as per user needs.
  Validation ensures the product fulfills its intended purpose.

---

12. What is Configuration Management?

Software Configuration Management (SCM) is the process of systematically managing changes to software to maintain integrity and traceability throughout its lifecycle.
It ensures that changes are made in a controlled manner and that every version is properly documented.
Main Activities:

1. **Version Control: Managing multiple versions of code.**
2. **Change Control: Evaluating and approving modifications.**
3. **Build Management: Ensuring consistent builds.**
4. **Release Management: Preparing and delivering final software releases.**
   Tools: Git, Subversion, Jenkins.
   SCM helps teams work collaboratively and prevents confusion caused by uncontrolled changes.

MAY-2022

1. What is Software Engineering?

Software Engineering is the systematic application of engineering principles, methods, and tools to the development and maintenance of high-quality software. It focuses on applying structured processes and methodologies to design, develop, test, and maintain software effectively. It aims to produce reliable, efficient, and cost-effective software that meets user requirements.

2. What are Small Projects?

Small projects are software projects with limited scope, budget, and duration, usually developed by a small team or even an individual. They involve fewer requirements, simple designs, minimal documentation, and short development cycles. Examples include small business applications, college management systems, or personal utility programs. They often use lightweight methodologies like Agile or Scrum.

3. What is COCOMO?

COCOMO (Constructive Cost Model) is a software cost estimation model developed by Barry Boehm. It estimates the cost, effort, and schedule of software projects based on project size (measured in KLOC – thousands of lines of code).

Types of COCOMO models:

1. **Basic COCOMO – estimates effort based on KLOC.**
2. **Intermediate COCOMO – includes cost drivers (e.g., reliability, team experience).**
3. **Detailed COCOMO – includes project phases.**

---

4. List the desirable properties of S/W requirement specification (SRS).

A good SRS should have the following properties:

- **Correctness – Accurately represents the customer's requirements.**
- **Unambiguity – Each requirement should have only one interpretation.**
- **Completeness – Includes all necessary requirements.**
- **Consistency – No conflicts among requirements.**
- **Verifiability – Each requirement should be testable.**
- **Modifiability – Easy to change or update.**

· · **Traceability – Each requirement traceable to its origin.**

---

5. What is Software Specification?

Software Specification is the process of defining what the software system should do and the constraints under which it operates. It describes the functionality, performance, and interface requirements in detail. The result of this process is the Software Requirement Specification (SRS) document, which serves as a contract between the client and the developer.

---

6. Define Inspection.

Inspection is a formal technical review process where a team of reviewers examines software documents (like code, design, or SRS) to detect defects early in development. It is a static verification technique — no code execution is required. It helps ensure quality and compliance with standards before testing begins.

---

7. What is Walkthrough?

A walkthrough is an informal review process where the author of a software document or code presents it to peers for feedback. It helps identify logical errors, inconsistencies, and improvement areas. Unlike inspections, walkthroughs are less formal and often used for educational and collaborative purposes.

---

8. List down any two design notations.

Design notations are graphical or textual representations used to describe software design. Examples:

· **Data Flow Diagram (DFD) – Shows data movement through the system.**

· · **Structure Chart – Depicts the hierarchy of modules in a program.**

· · **UML Diagrams like Class Diagram, Sequence Diagram.**

---

9. What are two types of verification?

Verification ensures the software meets its specified requirements. The two types are:

1.Static Verification – Reviewing, inspecting, or analyzing software without executing code.

2.Dynamic Verification – Involves executing software and checking output (testing).
Verification answers the question: *"Are we building the product right?"*

---

10. What is Quality Assurance (QA)?

Quality Assurance is a set of planned and systematic activities to ensure software meets required quality standards. It includes process monitoring, audits, reviews, and testing to prevent defects.

QA ensures that the development process is followed correctly to achieve consistent product quality.

---

11. What is Validation?

Validation ensures that the developed software meets the customer's needs and expectations. It answers: *"Are we building the right product?"*
It involves activities like acceptance testing, user evaluation, and prototype testing to confirm that the system performs its intended functions correctly.

---

12. What is Debugging?

Debugging is the process of identifying, analyzing, and removing errors or defects from software code. It occurs after testing detects failures. Debugging involves techniques like backtracking, cause elimination, and program tracing. The goal is to make the software execute correctly without faults.

MAY-2023

1. Define the term Project Plan.

A project plan is a formal document that defines how a project will be executed, monitored, and controlled. It outlines the project objectives, scope, schedule, resources, budget, risks, and quality standards. It serves as a roadmap for the project team and stakeholders by providing clear guidelines on tasks, roles, deadlines, and deliverables. A well-prepared project plan helps in organizing work efficiently, tracking progress, and ensuring the project is completed on time and within budget.

2. What are the layers of Software Engineering?

· Tools Layer – Software tools that support development and testing activities such as editors, debuggers, and CASE tools.

· Methods Layer – Provides technical methods for analysis, design, coding, testing, and maintenance.

· Process Layer – Defines the framework or set of activities required to build high-quality software like planning, modeling, construction, and deployment.

· Quality Focus – Lies at the core of all layers and emphasizes continuous improvement and meeting customer expectations.

---

3. What is Project Estimation?

Project estimation is the process of predicting the effort, time, cost, and resources required to complete a software project. It involves techniques like expert judgment, function point analysis, and COCOMO models. Estimation helps in setting realistic schedules, budgets, and

resource allocation. Accurate project estimation minimizes risks, avoids delays, and improves project planning and decision-making.

---

4. Define Software Maintenance.

Software maintenance refers to the process of modifying a software application after delivery to correct faults, improve performance, or adapt it to a changed environment. It includes corrective, adaptive, perfective, and preventive maintenance. Maintenance ensures the software remains functional, reliable, and up-to-date throughout its life cycle.

---

5. Define Data Design.

Data design is the process of transforming data requirements into a structured database or data model. It involves organizing data into entities, relationships, and attributes. The aim is to create a logical and physical data structure that supports efficient storage, retrieval, and processing. Good data design ensures data integrity, consistency, and scalability.

---

6. What is meant by Real-Time Design?

Real-time design refers to designing systems that must respond to inputs or events within a strict time limit. Such systems are used in areas like aviation, medical equipment, and automotive control systems. Real-time design ensures timely processing, reliability, and synchronization of hardware and software to avoid system failure.

---

7. What is Requirement Analysis?

Requirement analysis is the process of identifying, gathering, and documenting the needs and expectations of users for a software system. It involves interaction with stakeholders to understand functional and non-functional requirements. The output is a Software Requirement Specification (SRS) document. It helps in avoiding misunderstandings and forms the foundation for system design.

---

8. Define Walkthrough.

A walkthrough is a peer-review process where developers present their work (like design or code) to a group for feedback. The goal is to detect errors, improve quality, and enhance understanding. It is informal, guided by the author, and does not involve detailed testing. It helps in early error detection and improves team collaboration.

---

9. What is Testing?

Testing is the process of evaluating a software system to find defects and ensure that it meets user requirements. It involves executing the program under controlled conditions to check correctness, performance, and reliability. Types of testing include unit testing, integration testing, system testing, and acceptance testing. Testing helps in delivering a high-quality, error-free product.

10. Write the use of Static Analysis.

Static analysis is used to examine the source code without executing it. It helps in detecting errors, coding standard violations, and security vulnerabilities early in the development cycle. Tools like linters and code analyzers are used. Static analysis improves code quality, reduces debugging time, and enhances software reliability.

---

11. What is Verification?

Verification is the process of checking whether the software correctly follows the specified requirements. It answers the question, "Are we building the product right?" It includes reviews, inspections, walkthroughs, and static analysis. Verification ensures that each development phase output matches its specification before moving to the next phase.

---

12. What is Symbolic Execution?

Symbolic execution is a software analysis technique in which program inputs are treated as symbols rather than actual values. The program is executed using symbolic values, and paths are explored to detect errors, logical bugs, or security vulnerabilities. It helps in finding edge-case bugs that are hard to detect through normal testing.

MAY - 2024

PART-A

1. What is software engineering?

Definition:
Software Engineering is the **systematic application of engineering principles, methods and tools** to the development, operation and maintenance of software.

Explanation:
It involves a step-by-step approach that includes **planning, requirement analysis, design, coding, testing, and maintenance**. The main goal is to produce **high-quality, error-free, reliable, and cost-effective software** that satisfies user requirements.

Key Features

- ❖ Systematic Approach: Uses structured methods instead of random coding.
- ❖ Uses Engineering Principles: Applies concepts similar to traditional engineering (planning, designing, testing).
- ❖ Reduces Cost and Time: Helps complete software projects within budget and schedule.

2. What are the requirement of S/W project.

In Software Engineering, the requirements of a software project are the needs or expectations that the software must fulfill. These requirements help developers understand what to build and how the system should work.

Types of Software Requirements:

1.Functional Requirements

Describe what the system must do.

Example: Login feature, data entry, report generation.

2.Non-Functional Requirements

Describe how the system should work (quality aspects).

Example: Performance, security, reliability, speed.

3.User Requirements

Describes the needs of the end users in simple language.

Example: The user should be able to send messages through the app.

4.System Requirements

Define the hardware and software needed for the system to run.

Example: Operating system, memory, database server.

---

3. What is planning activity?

Definition:

Planning activity in software engineering is the process of deciding in advance what is to be done, how it will be done, when it will be done, and who will do it during a software project. It is the first step of project management.

Explanation:

During planning, the project manager prepares a project plan that includes estimation of time, cost, resources, manpower and schedule. This helps in controlling the project and ensures the work is completed within budget and deadline.

Main Activities in Planning:

Project Scope Definition – Understanding what the project should achieve.

Effort and Cost Estimation – Estimating required time, money, and resources.

Scheduling – Preparing a timeline / Gantt chart to assign tasks.

Resource Allocation – Assigning work to the team based on skills.

==========

4. What are cost estimation?

Definition:

Cost estimation in software engineering is the process of predicting the total cost required to develop, test, deliver, and maintain a software project.

Explanation:

- Cost estimation helps project managers determine:
- How much money is needed,
- How many working hours are required,
- What resources (hardware/software/manpower) are necessary.
- Objectives of Cost Estimation:
- To decide the total budget of the project.
- To ensure that the software project is completed within cost limits.
- To help in planning and resource allocation.
- To avoid financial risks and project failure.

5. What is hierarchical team structure?

Definition:

A hierarchical team structure is a team organization where members are arranged in multiple authority levels, with a clear reporting relationship.

Explanation:

In this structure, the project manager is at the top, below them are team leaders, and under them are developers, testers, and support staff. Decisions flow from top to bottom, and work progress reports move from bottom to top.

Characteristics / Key Points:

- Clear authority and responsibility – Everyone knows who they report to.
- Top-down decision-making – Project manager takes major decisions.
- Well-defined roles – Work is divided according to expertise.
- Effective supervision and control – Easy to monitor performance and progress

6. What is software maintenance?

Definition:

Software maintenance is the process of modifying, updating, and improving software after it has been delivered to the customer.

Types of Software Maintenance:

Corrective Maintenance – Fixing errors/bugs found after delivery.

Adaptive Maintenance – Updating software when the environment changes (e.g., new OS, new hardware).

Perfective Maintenance – Adding new features or improving performance.

Preventive Maintenance – Making changes to prevent future problems.

==========

7. What is DFD?

Definition:

A Data Flow Diagram (DFD) is a graphical representation of how data flows through a system. It shows where data comes from, how it is processed, and where it goes.

Symbols / Components of DFD:

Process – Shows how data is processed or transformed.

Data Flow – Indicates the movement of data between components.

Data Store – Represents storage of data (files, databases).

External Entity – Source or destination of data (users, other systems).

Advantages:

Easy to understand (visual representation).

Helps in communicating system requirements.

==========

8. What is type less language?

Definition:

A typeless language (also called untyped language) is a programming language in which variables do not have a fixed data type. The type of data is determined at runtime, and a variable can hold values of different types at different times.

Features:

- No need to declare variable types before use.
- Variables can store any kind of data (numbers, strings, etc.).
- Type checking happens during execution (runtime).
- Easy and flexible for quick coding, but may cause type errors.

Examples of Typeless / Dynamically Typed Languages:

- Python
- JavaScript
- PHP
- Ruby

---

9. What is modular design?

Definition:

Modular design is a software design technique where a large system is divided into smaller, manageable, and independent units called modules. Each module performs a specific function.

Advantages of Modular Design:

Easy to develop and maintain – Each module can be handled independently.

Improves code reusability – Modules can be reused in other projects.

Enhances teamwork – Different team members can work on different modules.

Reduces errors – If one module fails, others are not affected.

---

10. What is verification?

Definition:

Verification is the process of checking whether the software is being built correctly according to the specifications, design documents, and requirements.

Key Points:

Ensures that the software design and code follow the specified requirements.

Performed through reviews, inspections, walkthroughs, and analysis.

Done at every phase of the Software Development Life Cycle (SDLC).

---

11. What is configuration management?

Definition:
Configuration Management is the process of **systematically managing, controlling, and tracking changes** made to software during its development and maintenance.

Key Functions / Activities:

1. **Version Control:** Tracks versions of software files (e.g., using Git).
2. **Change Control:** Ensures changes are approved before implementation.
3. **Status Reporting:** Keeps records and reports on changes made.
4. **Configuration Identification:** Identifies components (files, documents, modules) that need control.

---

12. What is acceptance testing?

Definition:

Acceptance Testing is the final level of software testing performed to ensure that the software meets the user's requirements and is ready for delivery or deployment.

Objectives:

- To ensure the software satisfies business/user requirements.
- To check that the software is ready for use in the real environment.
- To confirm that there are no major defects remaining.
- To gain formal approval from the customer.
- Types of Acceptance Testing (write any):

- User Acceptance Testing (UAT)
- Customer Acceptance Testing
- Alpha and Beta Testing
