SOFTWARE ENGINEERING – 5MARKS

2019

1. Explain COCOMO model with example.

COCOMO (Constructive Cost Model) is a model used to estimate the cost, effort, and time required to develop software. It was developed by Barry Boehm.

There are three levels:

- **Basic COCOMO** – gives rough estimate.
- **Intermediate COCOMO** – includes cost drivers (like team experience, tools).
- **Detailed COCOMO** – includes all factors and phases.

Formula (Basic):
Effort (in person-months) = $a \times (KLOC)^b$
Where KLOC = Thousands of Lines of Code, and *a, b* are constants depending on the project type.

Example:
For a 10 KLOC project and constants $a = 2.4$, $b = 1.05$ (organic type):
Effort = 2.4 × (10)^1.05 ≈ 25 person-months.

---

2. Explain spiral model in detail.

The Spiral Model combines design and prototyping in stages. It is used for large, complex, and high-risk projects.

Phases in each spiral loop:

1. **Planning** – Requirements are gathered.
2. **Risk Analysis** – Identify risks and plan how to reduce them.
3. **Engineering** – Build and test the product.
4. **Evaluation** – Customer reviews the product and feedback is taken.

Each loop represents a phase of the software, and the product gets better with each loop.

Advantages:

- Good for high-risk projects.
- Frequent customer feedback.
- Supports changes during development.

---

3. Distinguish between 'process' and 'project' metrics with example.

Metric Type	Description	Example
Process Metric	Measures the efficiency of the software process.	Defect density, code review coverage.

Metric Type	Description	Example
Project Metric	Measures characteristics of a specific project.	Project size, effort, cost, schedule.

Example:

- Process metric: Number of defects found per 1000 lines of code.
- Project metric: Actual vs estimated time to complete a project.

---

4. Explain how software quality is assured through software metrics.

Software metrics help in measuring and improving quality by providing objective data.

How they help:

- Track defects and failures.
- Measure code complexity.
- Monitor performance and reliability.
- Identify areas needing improvement.

Example: If defect density is high, the team can improve testing or review processes. Metrics help ensure the product meets user and quality requirements.

2021

1. Write in brief on different stages of waterfall model.

The **Waterfall Model** is a linear software development model with the following stages:

1. **Requirements Analysis** – Understand what the user needs.
2. **System Design** – Plan the architecture and components.
3. **Implementation (Coding)** – Write the actual code.
4. **Testing** – Find and fix bugs.
5. **Deployment** – Release the software to users.
6. **Maintenance** – Fix issues or update software after release.

Each stage must be completed before moving to the next.

---

2. What are the different types of software maintenance?

There are four types of software maintenance:

1. **Corrective Maintenance** – Fix bugs found after release.
2. **Adaptive Maintenance** – Modify software to work in new environments (like new OS).

3. **Perfective Maintenance** – Improve performance or add new features.
4. **Preventive Maintenance** – Make changes to prevent future problems.

These help keep the software useful and relevant over time.

---

3. Write short note on COCOMO model.

COCOMO (Constructive Cost Model) is used to estimate software development cost, effort, and time based on project size.

Types of COCOMO:

- **Basic** – Uses only code size (KLOC).
- **Intermediate** – Adds factors like experience and tools.
- **Detailed** – Adds phase-wise breakdown.

Formula (Basic):
Effort = $a \times (KLOC)^b$

It helps in planning, budgeting, and scheduling projects.

---

4. Define modular coupling and cohesion.

Cohesion refers to how closely the functions within a module are related. A **highly cohesive** module performs one specific task or related tasks, making it easier to understand, maintain, and reuse.

- **Example:** A module that handles only user login operations shows high cohesion.

Coupling refers to the level of interdependence between different modules. **Low coupling** means modules work independently and communicate with minimal dependency, which is ideal for flexible and maintainable software.

- **Example:** A payment module that does not directly depend on user profile details shows low coupling.

In summary:

- Aim for **high cohesion**: do one thing well.
- Aim for **low coupling**: minimize dependencies between modules.

2022

1. Discuss Spiral Model.

The **Spiral Model** is a software development model that combines the features of both **iterative development** and **risk management**. It is especially suitable for **large, complex, and high-risk projects** where requirements may change over time.

The development process in the Spiral Model is divided into **multiple loops (or spirals)**. Each loop represents a phase of the project, and with each iteration, the product is refined and improved.

Each spiral loop has four main phases:

1. **Planning:**
   Requirements are gathered and objectives are defined for that phase.
2. **Risk Analysis:**
   Risks are identified and analyzed. Solutions or alternatives are planned to minimize those risks.
3. **Engineering:**
   The software is designed, coded, and tested in this phase.
4. **Evaluation:**
   The product is shown to the customer for feedback, and decisions are made about the next phase.

Key Features:

- Allows **incremental releases** of the product.
- Emphasizes **risk analysis and user involvement**.
- Helps manage **uncertain or changing requirements**.

Conclusion:

The Spiral Model is ideal for projects where changes are expected and **risk management is critical**. It provides a flexible yet structured approach to software development.

Each loop improves the software gradually. It's useful for large, high-risk projects.

---

2. Distinguish between software verification and software validation.

Concept	Verification	Validation
Meaning	Checks if the product is built correctly.	Checks if the correct product is built.
Focus	Internal processes and design.	User needs and functionality.
Methods	Reviews, inspections.	Testing, user feedback.
Example	Code review.	System testing.

---

3. Write short note on SRS.

SRS (Software Requirements Specification) is a formal document that describes the complete behavior, features, and constraints of a software system. It acts as a **bridge between the client and the development team**, ensuring both sides have a clear understanding of what the software will do.

Key Contents of SRS:

1. **Functional Requirements:**
   Describe what the system should do — such as user login, data processing, report generation.
2. **Non-Functional Requirements:**
   Describe system performance, reliability, security, usability, etc.

3. **Interfaces:**
   Defines how the software will interact with hardware, other software, and users.
4. **Constraints:**
   Includes limitations such as programming language, hardware requirements, and legal constraints.
5. **Use Cases and Diagrams:**
   Helps to visualize different user interactions with the system.

Importance:

- Provides a **clear roadmap** for developers.
- Reduces **misunderstandings** between client and team.
- Helps in **project planning, design, and testing**.

---

4. What are the basic design principles that are followed during software design?

Some basic software design principles are:

1. **Modularity** – Break system into small parts.
2. **Abstraction** – Focus on important details.
3. **Coupling and Cohesion** – Aim for low coupling and high cohesion.
4. **Reusability** – Design components that can be reused.
5. **Simplicity** – Keep design simple and understandable.

These help in building maintainable and efficient systems.

2023

1. Write in brief on different stages of spiral model.

Each loop of the **spiral model** has these four main stages:

1. **Planning** – Define goals, requirements.
2. **Risk Analysis** – Identify potential problems and solutions.
3. **Engineering** – Develop and test the prototype or product.
4. **Evaluation** – Review results and decide whether to continue.

With each loop, the software becomes more refined.

---

2. What are the different types of risk management in software?

Risk management in software development is the process of identifying, analyzing, planning, and monitoring risks that may affect the project's success.

Main Steps in Risk Management:

1. **Risk Identification:**
   Find potential risks like delays, cost overruns, or changing requirements.
2. **Risk Analysis:**
   Evaluate the probability and impact of each risk to understand its seriousness.
3. **Risk Planning:**
   Prepare strategies to avoid, reduce, or handle the risks.
4. **Risk Monitoring:**
   Continuously track risks and update the plan as needed.

Types of Risks:

- **Technical Risks** (e.g., new technology, integration issues)
- **Project Risks** (e.g., poor planning, lack of resources)
- **Business Risks** (e.g., market changes, shifting priorities)

---

3. Write short note on COCOMO model.

COCOMO (Constructive Cost Model) is a mathematical model used to estimate the **cost, effort, and time** required to develop software. It was introduced by **Barry Boehm** in 1981 and is widely used in project planning and management.

COCOMO provides estimates based on the **size of the software project**, usually measured in **KLOC** (thousands of lines of code). There are three main levels of the model:

1. **Basic COCOMO:**
   Gives a rough estimate using only the size of the project (KLOC).
   Formula: Effort = $a \times (KLOC)^b$, where 'a' and 'b' are constants based on project type (organic, semi-detached, embedded).
2. **Intermediate COCOMO:**
   Adds 15 cost drivers such as developer experience, tools used, and complexity of the product.
3. **Detailed COCOMO:**
   Further divides the project into phases like design, coding, and testing, applying cost drivers to each phase separately.

Usefulness:

- Helps in **budgeting**, **scheduling**, and **resource planning**.
- Gives a **realistic estimate** early in the project.

---

4. Define modular coupling and cohesion.

In software design, **coupling** and **cohesion** are important principles that affect the quality, maintainability, and readability of code.

Cohesion:

Cohesion refers to how closely the tasks performed by a module are related to one another. **High cohesion** means that a module performs a single, well-defined task. High cohesion improves **readability, maintainability**, and **reusability**.

- **Example:** A module that only handles payment processing shows high cohesion.

Coupling:

Coupling refers to the **level of dependency between modules**. **Low coupling** means modules are independent and communicate only when needed. Low coupling makes the system easier to modify and test.

- **Example:** If the payment module does not depend on the user profile module directly, it has low coupling.

Summary:

- Aim for **high cohesion** (do one thing well).
- Aim for **low coupling** (minimize dependencies).
  This leads to cleaner, modular, and more reliable software.

2023-24

1. Write a short on:

a. Spiral Model:

The **Spiral Model** is a software development methodology that combines **iterative development** with **risk management**. It was introduced by **Barry Boehm** and is particularly suited for large, complex, and high-risk projects.

The process is divided into multiple **spirals (loops)**, each containing four main phases:

1. **Planning:** Gather requirements and define objectives.
2. **Risk Analysis:** Identify potential risks and plan mitigation strategies.
3. **Engineering:** Design, develop, and test the system.
4. **Evaluation:** Get user feedback and decide whether to continue to the next loop.

Each loop in the spiral model refines the product further, reducing risks and improving the system gradually.

Key Advantage:

It allows for incremental delivery and provides flexibility to make changes based on user feedback or risk identification.

---

b. Prototyping Model:

The **Prototyping Model** is an iterative software development process where a prototype (an early, simplified version of the system) is created to help users visualize the software before it's fully developed.

1. **Prototype Creation:** A prototype is built quickly based on initial requirements.
2. **User Feedback:** The prototype is shown to users for feedback.
3. **Refinement:** Based on feedback, the prototype is refined, and the process repeats until the system meets user needs.

Advantages:

- Allows users to interact with the software early in the development process.
- Reduces misunderstandings between clients and developers.
- Particularly useful when requirements are unclear or evolve during development.

---

2. Write about risk identification and assessment.

Risk Identification is the first step in risk management and involves recognizing potential risks that could affect the project. These risks could be related to technology, schedule, resources, or even external factors like market changes.

Risk Assessment is the process of evaluating the identified risks to understand their **probability** (likelihood) and **impact** (severity). This allows project managers to prioritize which risks need to be addressed first. Risk assessment can be done using qualitative (descriptive) or quantitative (numerical) methods.

Risk Management Steps:

1. **Risk Identification** - Recognize possible risks.
2. **Risk Assessment** - Evaluate how likely they are to occur and how serious they could be.
3. **Risk Planning** - Develop mitigation strategies.
4. **Risk Monitoring** - Track the risks throughout the project.

---

3. Describe SCCS (Source Code Control System).

SCCS (Source Code Control System) is a software tool used to manage changes to source code during the development process. It allows developers to track and control versions of files, ensuring that the team can manage updates, revisions, and conflicts effectively.

Key Features of SCCS:

1. **Version Control:** Keeps track of different versions of source code files, allowing rollback to previous versions.
2. **Collaboration:** Multiple developers can work on the same project without overwriting each other's changes.
3. **Tracking Changes:** Each modification is recorded, with details on who made the change and why.
4. **Conflict Resolution:** If two developers modify the same file, SCCS helps merge changes without data loss.

Examples of modern **SCCS tools** include **Git**, **Subversion (SVN)**, and **Mercurial**.

---

4. Define various testing.

Software testing is the process of verifying that a software application functions as expected and identifying any defects. There are different levels and types of testing, each serving specific purposes.

Types of Testing:

1. **Unit Testing:**
   Focuses on individual components or functions of the software. It ensures each module works as intended in isolation.

2. **Integration Testing:**
   Verifies that different components or modules of the system work together. It identifies issues in the interaction between modules.

3. **System Testing:**
   Tests the complete and integrated software system to verify it meets all requirements. It checks the overall functionality and performance.

4. **Acceptance Testing:**
   Conducted to determine whether the software meets the user's needs and whether it is ready for deployment. It's often based on user requirements.

5. **Regression Testing:**
   Ensures that new changes or enhancements do not introduce new bugs into existing functionality.

6. **Stress Testing:**
   Tests how the system behaves under extreme conditions, like high traffic or heavy load.

7. **Alpha and Beta Testing:**
   Alpha testing is done in-house, while Beta testing involves releasing the software to a limited user group for real-world feedback.