Code No: 154CQ R18

JAWAHARLAL NEHRU TECHNOLOGICAL UNIVERSITY HYDERABAD
B. Tech II Year II Semester Examinations, April/May - 2023
SOFTWARE ENGINEERING
(Common to CSBS, CSIT, CSE(AIML), CSE(DS))

Time: 3 Hours Max. Marks: 75

Note: i) Question paper consists of Part A, Part B.
ii) Part A is compulsory, which carries 25 marks. In Part A, Answer all questions.
iii) In Part B, Answer any one question from each unit. Each question carries 10 marks and may have a, b as sub questions.

PART – A

(25 Marks)

1.a) Define software engineering and explain. [2]
b) What is process assessment? Why do we need to asses software process? [3]
c) What is interface specification? [2]
d) What is feasibility study? What are its types? Explain. [3]
e) What is Software Architecture? Define. [2]
f) What are called interaction diagrams? Explain their types. [3]
g) Why do we need to test software? Explain. [2]
h) What is validation testing? Explain. [3]
i) What is the purpose of software measurement? Explain. [2]
j) What is software review? Explain different types of software reviews. [3]

PART – B

(50 Marks)

2.a) Compare and contrast different software process models.
b) With a neat sketch, explain Capability Maturity Model Integration (CMMI). [5+5]

OR

3. Explain in detail about personal and team process models. [10]

4.a) What is the goal of requirements analysis phase? Why the requirements analysis phase is a difficult one? Justify your answer.

b) Explain the differences between functional requirements and non-functional requirements by giving suitable examples. [5+5]

OR

5.a) What are the desirable characteristics of a good software requirement specification document? Discuss.

b) Draw a process model showing how a requirements review might be organized. [5+5]

6.a) Draw the class diagram and explain various relationships that exists between classes.

b) Distinguish between sequence and collaboration diagrams. [5+5]

OR

7.a) Draw and explain Use case diagrams and its development process with suitable illustrations.

b) What are the design principles of a good software design?
Explain. [5+5]

8.a) What are the various testing strategies used for testing conventional software? Discuss.

b) What is software quality? What are the metrics used for source code analysis? Explain. [5+5]

OR

9.a) What is white box testing? How white box testing is carried out? Demonstrate with example.

b) Discuss about the metrics used for software design model. [5+5]

10.a) What is risk identification? Discuss various methods used for risk identification.

b) Discuss the role of formal technical reviews in achieving good quality software.
[5+5]

OR

11.a) Explain ISO 9000 Quality standards.

b) What is Statistical Software Quality Assurance? Discuss its objectives.
[5+5]

---ooOoo---

1.a) Define software engineering and explain.

Definition: Software engineering is the systematic application of engineering principles to the design, development, maintenance, testing, and evaluation of software. Explanation: It aims to produce high-quality software that meets user requirements, is reliable, and maintainable by following established methodologies and practices.

1.b) What is process assessment? Why do we need to assess software processes?

Definition: Process assessment is the evaluation of software processes to determine their capability and maturity. Need:

1. Identify strengths and weaknesses in current processes.
2. Ensure compliance with industry standards.
3. Improve process efficiency and effectiveness.

1.c) What is interface specification?

Definition: Interface specification describes the details of how software components interact with each other. Explanation: It includes inputs, outputs, and the behavior of the interface to ensure consistent and correct communication between system components.

1.d) What is feasibility study? What are its types? Explain.

Definition: A feasibility study is an analysis to determine whether a project is viable and worth pursuing. Types:

1. Technical Feasibility: Assesses technological capabilities.
2. Economic Feasibility: Evaluates financial viability.
3. Operational Feasibility: Examines operational suitability.
4. Schedule Feasibility: Determines time frame viability.

1.e) What is Software Architecture? Define.

Definition: Software architecture is the high-level structure of a software system, consisting of software elements, their relationships, and properties. Explanation: It provides a blueprint for constructing and evolving a system, ensuring it meets both functional and non-functional requirements.

1.f) What are called interaction diagrams? Explain their types.

Definition: Interaction diagrams are UML diagrams that visualize the flow of control and data among system components. Types:

1. Sequence Diagrams: Show object interactions in a sequence of time.
2. Collaboration Diagrams: Emphasize structural organization and message passing among objects.

1.g) Why do we need to test software? Explain.

Definition: Software testing is essential to identify defects and ensure the software meets specified requirements. Explanation: It helps to verify functionality, performance, and reliability, and prevents potential failures, ensuring high-quality software delivery.

1.h) What is validation testing? Explain.

Definition: Validation testing ensures that the software meets the needs and requirements of the end-users. Explanation: It involves evaluating the software during or at the end of the development process to ensure it fulfills the intended use and user expectations.

1.i) What is the purpose of software measurement? Explain.

Definition: The purpose of software measurement is to quantify attributes of software products and processes. Explanation: It helps in monitoring progress, evaluating quality, estimating resources, and improving processes through data-driven decision-making.

1.j) What is software review? Explain different types of software reviews.

Definition: A software review is a process to evaluate software products and artifacts to identify defects and improvements. Types:

1. Peer Reviews: Informal reviews by colleagues.
2. Walkthroughs: Semi-formal reviews to examine the software.

3. Inspections: Formal, structured reviews to detect defects.
4. Audits: Reviews to ensure compliance with standards and procedures.

2.a) Compare and contrast different software process models.

Waterfall Model:

1. **Sequential Phases:** Requirements, Design, Implementation, Testing, Maintenance.
2. **Pros:** Simple, easy to understand and manage, clear milestones.
3. **Cons:** Inflexible to changes, difficult to go back to previous phases, late testing phase.

Iterative Model:

1. **Repetitive Cycles:** Develop a small part, test, review, and repeat.
2. **Pros:** Allows for early testing and feedback, adaptable to changes, reduces risk.
3. **Cons:** Requires good planning, can be resource-intensive, scope creep risk.

Agile Model:

1. **Incremental Development:** Short iterations (sprints) with frequent reassessment and adaptation.
2. **Pros:** Highly adaptable to change, promotes customer collaboration, fast delivery.
3. **Cons:** Requires disciplined teams, less predictable, can be chaotic if not well managed.

Spiral Model:

1. **Risk-Driven:** Combines iterative development with risk assessment.
2. **Pros:** Focuses on risk management, flexible, handles changes well.
3. **Cons:** Complex to manage, costly, requires expertise in risk analysis.
4. **V-Model:**
5. **Verification and Validation:** Emphasizes testing at each development stage.
6. **Pros:** High-quality testing, clear stages and milestones, easy to manage.
7. **Cons:** Inflexible, similar to waterfall, changes are difficult to implement.

2.b) With a neat sketch, explain Capability Maturity Model Integration (CMMI).

CMMI Model:

Levels: Initial, Managed, Defined, Quantitatively Managed, Optimizing.

Description:

1. **Initial (Level 1):** Processes are unpredictable, poorly controlled, reactive.
2. **Managed (Level 2):** Projects are managed; processes are planned, documented, and followed.
3. **Defined (Level 3):** Organization-wide standards and processes are established and improved.
4. **Quantitatively Managed (Level 4):** Processes are measured and controlled.
5. **Optimizing (Level 5):** Focus on continuous process improvement through innovation and feedback.

3. Explain in detail about personal and team process models.

Personal Software Process (PSP):

1. **Focus:** Individual's ability to manage and improve their work.
2. **Phases:** Planning, High-Level Design, High-Level Design Review, Development, Postmortem.
3. **Activities:** Estimating time, tracking defects, analyzing productivity.
4. **Benefits:** Improved personal productivity, quality, and predictability.

Team Software Process (TSP):

1. **Focus:** Team-based development, based on PSP principles.
2. **Phases:** Launch, High-Level Design, Implementation, Integration, Testing, Postmortem.
3. **Activities:** Defining roles, setting goals, planning tasks, monitoring progress, conducting reviews.
4. **Benefits:** Improved team performance, reduced defects, higher project visibility.

4.a) What is the goal of requirements analysis phase? Why the requirements analysis phase is a difficult one? Justify your answer.

Goal:

Understanding Requirements: Elicit, analyze, and document the software requirements to ensure clarity, completeness, and feasibility.

Difficulty:

1. **Complexity:** Diverse and conflicting requirements from various stakeholders.
2. **Ambiguity:** Unclear, incomplete, or changing requirements.
3. **Communication:** Misunderstandings between stakeholders and developers.
4. **Scope Creep:** Constantly changing requirements can lead to project delays and increased costs.

4.b) Explain the differences between functional requirements and non-functional requirements by giving suitable examples.

Functional Requirements:

1. **Definition:** Specific behaviors or functions of a system.
2. **Examples:** User authentication, data processing, report generation.
3. **Focus:** What the system should do.
4. **Non-Functional Requirements:**
5. **Definition:** Quality attributes, constraints, and standards.
6. **Examples:** Performance (system should handle 1000 transactions per second), security (data encryption), usability (user-friendly interface).
7. **Focus:** How the system should perform its functions.

5.a) What are the desirable characteristics of a good software requirement specification document? Discuss.

Characteristics:

Clear and Unambiguous: Each requirement should be stated clearly and precisely to avoid misinterpretation.

Complete: All necessary requirements should be included, covering all aspects of the system.

Consistent: Requirements should not conflict with each other.

Verifiable: Each requirement should be testable, with a clear way to verify its implementation.

Modifiable: The document should be easy to update as requirements change.

Traceable: Each requirement should be traceable back to its origin and throughout the development process.

5.b) Draw a process model showing how a requirements review might be organized.

Process Model:

1. Preparation:
2. Collect requirements.
3. Prepare review materials.
4. Review Meeting:
5. Present requirements.
6. Discuss each requirement.
7. Identify issues and concerns.
8. Analysis:
9. Analyze feedback.
10. Prioritize and categorize issues.
11. Revision:
12. Update requirements document based on feedback.
13. Follow-up:
14. Verify changes.
15. Ensure all issues are addressed.

6.a) Draw the class diagram and explain various relationships that exist between classes

Relationships:

Association: A general connection between classes (e.g., Person and Address).

Aggregation: A special form of association with a whole-part relationship, but parts can exist independently.

Composition: A stronger form of aggregation where parts cannot exist without the whole.

Inheritance: A relationship where one class (child) inherits attributes and methods from another class (parent).

6.b) Distinguish between sequence and collaboration diagrams.

Sequence Diagrams:

1. **Focus:** Time sequence of messages between objects.

2. **Elements:** Objects, lifelines, messages, activation bars.
3. **Use:** Visualizing the order of interactions.

Collaboration Diagrams:

1. **Focus:** Structural organization and interaction.
2. **Elements:** Objects, links, messages.
3. **Use:** Showing how objects interact and their relationships.

7.a) Draw and explain Use Case Diagrams and its development process with suitable illustrations.

Use Case Diagram:

plaintext

Copy code

```
+---------------------+        +-----------------+
|   Customer        |        |  System      |
+---------------------+        +-----------------+
      ^                   |
      |                  |
      | performs            | performs
      |                  |
      +------- (Use Case)---------+
              Purchase Item
```

Development Process:

1. **Identify Actors:** Determine users and external systems interacting with the system.
2. **Identify Use Cases:** Define functionalities provided by the system.
3. **Draw Diagram:** Connect actors with their respective use cases using associations.
4. **Review and Refine:** Ensure completeness and accuracy of the diagram.

7.b) What are the design principles of a good software design? Explain.

Principles:

1. **Modularity:** Dividing the system into smaller, manageable modules.
2. **Encapsulation:** Hiding the internal details of modules and exposing only necessary interfaces.

3. **Separation of Concerns:** Distinguishing different aspects of the software to manage complexity.
4. **Coupling and Cohesion:** Minimizing dependencies (low coupling) and maximizing the relatedness of elements within a module (high cohesion).
5. **Abstraction:** Simplifying complex systems by modeling essential features while ignoring non-essential details.
6. **Reusability:** Designing components that can be reused in different contexts to reduce redundancy.

8.a) What are the various testing strategies used for testing conventional software? Discuss.

Testing Strategies:

1. **Unit Testing:** Testing individual components or modules for correctness.
2. **Integration Testing:** Testing combined parts of the system to ensure they work together.
3. **System Testing:** Testing the complete system for compliance with requirements.
4. **Acceptance Testing:** Testing the system in real-world scenarios to ensure it meets user needs.
5. **Regression Testing:** Re-testing after changes to ensure no new defects are introduced.

8.b) What is software quality? What are the metrics used for source code analysis? Explain.

Software Quality:

1. **Definition:** The degree to which software meets specified requirements, customer needs, and expectations.
2. **Attributes:** Functionality, reliability, usability, efficiency, maintainability, portability.
3. **Metrics for Source Code Analysis:**
4. **Lines of Code (LOC):** Measures the size of the codebase.
5. **Cyclomatic Complexity:** Measures the complexity of the code by counting the number of linearly independent paths.
6. **Code Coverage:** Measures the extent to which the source code is tested.
7. **Code Churn:** Measures the amount of code changed over time.
8. **Technical Debt:** Measures the implied cost of future rework due to shortcuts taken during development.

9.a) What is white box testing? How is white box testing carried out? Demonstrate with an example.

Definition: White box testing is a testing technique that examines the internal structure and workings of a software application.

Process:

1. **Understanding the Code:** Review the source code to understand its logic and structure.
2. **Design Test Cases:** Create test cases based on code paths, conditions, loops, and branches.
3. **Execute Tests:** Run the tests and observe the outcomes.
4. **Analyze Results:** Identify and fix any defects revealed by the tests.

Example: Consider a function that calculates the factorial of a number:

```
def factorial(n):
   if n == 0:
      return 1
   else:
      return n * factorial(n-1)
```

Test Cases:

1. Input: n = 0, Expected Output: 1
2. Input: n = 5, Expected Output: 120
3. Check conditions for n > 0

9.b) Discuss the metrics used for software design model.

Metrics:

1. **Complexity Metrics:** Measure the complexity of design (e.g., Cyclomatic Complexity).
2. **Cohesion Metrics:** Measure how related the responsibilities of a single module are.
3. **Coupling Metrics:** Measure the degree of interdependence between modules.
4. **Size Metrics:** Measure the size of design elements (e.g., number of classes, methods).
5. **Encapsulation Metrics:** Measure the degree to which data and functions are hidden within a module.

6. **Inheritance Metrics:** Measure the use of inheritance in the design (e.g., depth of inheritance tree).

10.a) What is risk identification? Discuss various methods used for risk identification.

Definition: Risk identification is the process of determining risks that could potentially prevent the project from achieving its objectives.

Methods:

1. **Brainstorming:** Group discussions to identify potential risks.
2. **Delphi Technique:** Experts provide risk insights anonymously, and the results are compiled and shared.
3. **Checklists:** Using predefined lists of potential risks based on past projects.
4. **SWOT Analysis:** Identifying risks by analyzing strengths, weaknesses, opportunities, and threats.
5. **Interviews:** Gathering risk-related information from stakeholders and team members.
6. **Historical Data:** Analyzing data from previous projects to identify recurring risks.

10.b) Discuss the role of formal technical reviews in achieving good quality software.

Role:

1. **Defect Detection:** Identifying and fixing defects early in the development process.
2. **Improving Quality:** Ensuring that the software meets quality standards and requirements.
3. **Knowledge Sharing:** Facilitating knowledge transfer among team members.
4. **Compliance:** Ensuring adherence to standards, guidelines, and best practices.
5. **Documentation:** Providing a record of review processes and outcomes for future reference.
6. **Risk Mitigation:** Identifying potential risks and issues early to mitigate them effectively.

11.a) Explain ISO 9000 Quality Standards.

ISO 9000:

Definition: A set of international standards for quality management systems.

Purpose: Ensure organizations meet customer and regulatory requirements and improve quality.

Principles:

1. Customer focus
2. Leadership
3. Engagement of people
4. Process approach
5. Improvement
6. Evidence-based decision making
7. Relationship management

Key Standards:

1. **ISO 9001:** Specifies requirements for a quality management system.
2. **ISO 9004:** Provides guidelines for performance improvement.

11.b) What is Statistical Software Quality Assurance? Discuss its objectives.

Definition: Statistical Software Quality Assurance (SSQA) uses statistical methods to monitor and control software quality.

Objectives:

1. **Quantitative Analysis:** Use of metrics to measure software quality attributes.
2. **Process Improvement:** Identify trends and patterns to improve software development processes.
3. **Defect Prevention:** Use statistical data to predict and prevent defects.
4. **Performance Monitoring:** Continuously monitor software performance and quality.
5. **Decision Making:** Provide data-driven insights for making informed decisions about quality assurance activities.