Part-A

(R25CS58122PE)

1. a) Define software Quality

A: Software Quality is the degree to which a software product satisfies specified requirements and meets customer expectations without defects.

According to IEEE:

Software quality is the degree to which a system, component or process meets specified requirements and customer needs

Key Points:

- Conformance to requirements
- fitness for use
- Reliability & Performance
- Defect-free operation.

b) Defect Prevention Techniques

A: Defect prevention focuses on avoiding defects before they occur.

Techniques:

1. Root cause Analysis (RCA): Identify the main cause of defects
2. Process Improvement: Improve development process
3. code reviews & Inspections - Detect defect early.
4. Prototyping - clarify requirements early
5. Automation Tools: static analysis tools

c) How to Improve Software Quality ?

A: Software quality can be improved by :

1. Clear requirements Specification
2. Quality Planning
3. Continuous testing
4. Use of Quality models
5. Metrics & Measurement

Common improvement approaches :

- International organization for standardization (iso standards)
- Capability Maturity Model integration (CMMI)
- Six sigma
- TQM

d) Why is software testing required

A: Software testing is required to :

1. Detect defects & errors
2. Reduce risk of failure
3. Ensure Security & Safety
4. verify and validate the system
5. Ensure software meets requirements

e.) Limitations of checklist - Based Testing

A: Check-list based testing uses a predefined list of test cases

Limitations:

1. Does not cover all possible scenarios
2. Depends on experience of tester
3. May miss complex defects
4. Check list may become outdated
5. Focuses only on known issues.

Part-B

2: Examine different perspectives & expectations of Software quality Elaborate how the customer, developer & user perspective differ.

A: Software quality is the degree to which a software product meets specified requirements and satisfies customer and user needs without defects.

Software quality is viewed different stake holders

the three major perspective are

1. Customer Perspective
2. Developer perspective
3. User perspective

Each group has different expectations based on their role & interest

1. Customer perspective:

The customer is the person or organization that pays for the software

Expectations:

- High reliability & security
- Good return on investment
- Easy maintainance & scalability

Focus:

- cost effectiveness
- Bussiness value
- Complance with contract

2) Developer Perspective:

The developer focuses on building the software correctly.

Expectations:

- Clear and complete requirements
- well-structured design
- Fewer defects

Focus:

- code quality
- process quality
- Documentation.

conclusion: Software quality is multi-dimensional & product is is truly high-quality only when.

- It satisfies customer requirements
- It is technically sound.

3) User perspective:

The user is the person who actually uses the software

Expectations:

- Easy to use
- Reliability
- Helpful support

Focus:

- User interface
- Performance
- functionality.

4) Analyze the defect reduction techniques. Inspect software inspection and testing as methods of direct fault detection and failure observation.

A: Defect reduction is the process of identifying, detecting and removing defects from software after they are introduced during development.

Defect reduction mainly uses:

1. Software Inspection
2. Software Testing.

1. Software Inspection:

Software inspection is a static verification technique used to detect defects directly in documents, design, or code without executing the Program.

It was formalized by micheal Fagan at IBM

Purpose:
- Detect faults early
- Improve quality before testing phase
- Reduce cost of fixing defects.

Inspection Process:

1: Planning: Select document
2: Rework: Fix defects
3: Follow-up: Verify corrections

Characteristics:

- Early defect removal
- Reduces rework cost
- Static.

Example: If a design document misses a required feature, inspection detects it before coding starts.

Software Testing:

Software testing is a dynamic technique where the software is executed to identify failures and detect defects.

Purpose:
- Observe failured during execution
- Identify faults causing those failures
- Validate software behaviour.

Testing Process:
- Test case design
- Debugging
- Retesting

characteristics:
- Dynamic
- Detects runtime defects
- Validates functionality.

Example:

If an e-commerce website crashes during Payment, testing reveals the failure and debugging identifies the defect.

6. Demonstrate the activity and process of quality engineering in detail.

A: Introduction to Quality Engineering:

Quality Engineering is a systematic approach to ensure that software products meet required quality standards throughout the software development life cycle.

It focuses on:

- Defect prevention
- Process control
- customer satisfaction

Activities of quality Engineering:

Quality Engineering activities are carried out through out the SDLC

1. Quality Planning

Purpose:

To define quality goals and strategies.

Activities:

- Identify quality objectives
- Define quality metrics
- Prepare Quality Assurance plan.

Example:

Defining acceptable defect density before project starts.

2) Qaality Assurance

Purpose:

Ensure processes are followed correctly.

Activities:

- Process audits
- Standards compliance
- Documentation review.

3) Quality control

Purpose:

Detect detects in the product

Activities:

-) Code reviews
-) Inspections
-) Testing.

4) Defect Prevention

Activities:

- Root cause analysis (RCA)
- Process improvement
- Standardization.

Goal: Reduce defect introduction

Step 1: Requirement Analysis

- Validate requirements
- check clarity and completeness
- Identify risks.

Step 2: Design Quality Assurance.

- Design reviews
- Architecture validation
- Risk analysis

Step 3: Development Quality control

- Code standards
- Peer reviews
- Unit testing.

Step 4: Testing & Verification.

- Integration testing
- System testing
- Security testing.

Step 5: Deployment & maintenance

- Monitoring
- Defect tracking

4. Diagram representation:

8: Illustrate the methodology for test planning & Preparation.

1. Introduction

Test planning and Preparation is the initial phase of the software testing process where testing objectives, scopes, strategy and resources are defined.

It ensures that testing is systematic, organized & effective.

2. Objectives of test planning

- Define testing scope & goals
- Estimate time and cost
- Identify risks

3. Methology for test planning

The methadology includes the following structed steps

i) Requirement Analysis:

- Study software requirement specification
- clarity ambiguities
- Define acceptance criteria

ii) Define test scope and objectives:

- Identify test levels
- Define quality goals

iii) Resource Planning:

- Assign roles & responsibilities
- Identify required skills
- Select testing tools

iv) Schedule and cost Estimation

- Prepare test schedule
- Estimate effort + cost
- Define milestones

v) Risk identification and management

- Identify possible risks
- Analyze impact
- Prepare mitigation plan.

vi) Prepare test plan document

- Test objectives
- Scope
- Strategy.

4) Test Preparation Activities

After planning, Preparation includes:

1) Test case design:

- Create test scenarios
- Define expected results.

2) Test data Preparation:

- Prepare valid + invalid inputs
- Ensure data coverage.

5) Design test cases based on probability

- More frequent operations → more test cases
- Rare operations → Fewer test cases.

- How usage - Based Statistical Testical works
  - Develop operational Profile
  - Execute tests
  - Measure failure rate

Advantages:

- Focuses on real-world usage
- Improves reliability prediction
- Prioritizes critical operations

Limitations:

- Difficult to estimate accurate usage data
- Rare but critical functions may get less attention

Example:

In an ATM System

- 70% users with draw cash
- 20% check balance
- 10% deposit money.

3. Steps in constructing Musa's operational Protile

i) Identity user categories.

eg: Admin

Regular user

Guest

ii) Identity system operations

eg: Login

Check balance

Paybills

iii) Determine frequency of operations

Estimate how often each operation is used

eg:

Operation	Frequency per day
Login	1000
Check balance	2000
Transfer bills	500
Pay bills	300

iv) Calculate Probability of each operation.

$$\text{Probability} = \frac{\text{operation frequency}}{\text{Total frequency}}$$

10 Enter usage - based statistical testing using Musa's operational Profile.

A: 1. Introduction:

Usage-based statistical Testing approach where test cases are selected based on how users actually use the software in real world conditions.

It focuses on:

- Realistic usage patterns
- Probability of operations

This concept was introduced by John Musa

2. Musa's operational Profile:

An operational Profile is a quantitative description of how a software system will be used in the field

It specifies

- Frequency of each operation
- Probability of occurence.
- Different operations performed by users.

3) Test Environment Setup.

- configure hardware
- Install required

4) Review and Approval

- Review test case
- validate completeness

5) Diagram Representation

Requirement Analysis
↓
Test strategy
↓
Resource & Schedule planning
↓
Risk Analysis
↓
Test Plan Document
↓
Test case & Environment preparation

11. How to construct the operational profile? Explain.

1. Introduction

An operational profile (OP) is a quantitative description of how software will be used by users in real operating condition

The concept of operational profile was introduced by John Musa as part of software reliability

→ It helps in usage - based statistical testing and reliability estimation

2. Purpose of Constructing Operational Profile

- Identify real user behaviour
- Determine frequency of operation
- Prioritize testing effort
- Improve reliability prediction

3 Step of Construct an operational profile

Step 1: Identify Customer / User Categories

Classify different types of users

Example: Administration, Registered User, Guest User

Step 2: Identify System Operations

List all operations that users can perform

Example: (ATM System)

- Withdraw cash
- Check balance

Step 3: Determine Occurrence Rates

Estimate how often each operation is performed

Sources of data

- Historical usage records
- Customer surveys

Step 4: Calculate Operation Probability

$$\text{probability} = \frac{\text{Operation Occurrence}}{\text{Total Occurrences}}$$

Step 5: Validate and Refine the profile

- Review with stakeholders
- Compare with real data
- Update periodically

Part-B

30 Define the correctness and defects in software quality. Discuss their properties and measurement techniques with suitable examples

Ans Correctness and Defects in Software Quality

- Definition of Correctness

Correctness. Is the degree to which software performs its intended functions according to specified requirements

-> A Software System is correct if it produces the expected output for all valid inputs

Example:

If a banking application calculate intrest as per the given formula in the requirements, then considered correct

Properties of Correctness

1. Requirement Conformance: Matches specification exactly
2. Accuracy Produces correct results
3. Completeness Implements all required functions
4. Consistency No contradictory behavior
5. Logical Validity follows correct algorithm and logic

Measurement of Correctness

Correctness is measured using

1. Testing — Comparing actual output with expected output
2. Defect Density — Number of defects per KLOC
3. Reliability Matrics — failure rate, MTBF
4. formal Verfication — Mathematical proof of correctness

5Q. Interpret defect containment in software system. Elucidate the software fault tolerance safety assurance and failure containment mechanism.

Ans Defect Containment in Software System

1. Definition.

Defect Containment is a quality engineering approach that limits the impact of residual defects to prevent system-wide failures

Even after defect prevention and defect reduction some defects may remain in the system

Defect containment ensures that such defects do not cause major damage.

2. Need for Defect Containment

Defect containment is required because

- Not all defects can be completely removed
- Complex system may contain hidden faults
- Safety - critical system must avoid catastrophic failure
- It improves system reliability and availability

Example.

In online banking even if one module fails, the entire system not crash.

3. Techniques of Defect Containment

Definition

Fault tolerance is the ability of a system to continue operating properly even when faults occur

Techniques.

- Redundacy (backup system)
- Exception handling
- Recovery blocks
- Checkpointing and rollback
- N-version programming

10. Assess the procedure used for software quality assessment and improvement introduction

Ans Software Quality Assessment and improvement is a systematic procedure used to evaluate the quality of software products and process and to continuously enhance them

Quality Assessment
Measuring and evaluating current quality level

Quality Improvement
Taking corrective and preventive action to enhance quality

Procedure for software Quality Assessment
Quality assessment involves the following steps

1. Define Quality Standards and Criteria
Identify quality attributes
Select standards and guidlines
Set measurable quality goals
Standards may follow frameworks

2. Select Quality Metrics
Metrics help measure quality objectively

3. Conduct Reviews and Audits

Requirement reviews

Design inspection

Code reviews

Process audits

Purpose: Identify defects and compliance issues

9Q. When to use test automation? Describe various tools, benefits and challenges of test automation

1. Introduction

Test Automation

It is the use of software tools to execute test cases automatically, compare actual result with expected results and generate test reports

It reduces manual effort and improves testing efficiency

2. When to use Test Automation

Test automation should be used in the following

1. Repetitive Testing
   - Regression testing
   - Smoke testing
   - Re-running same test cases multiple times
2. Large and Complex Application
   - Enterprise System
   - Web application with frequent update
3. Performance and load Testing
   - Simulating multiple users
   - Measuring response time.
4. Continous Integration / DevOps Environment
   - Frequent builds
   - Continious testing requirement
5. Time - Constrained Projects
   - When manual testing is too slow

When NOT to use Automation

- One-time testing
- Frequently changing UI
- Very small project
- Exploratory testing