Software Testing

Software Engineering

06-Nov-2025

Software Quality

Software quality refers to the degree to which a software product meets specified requirements and user expectations.

- Product Quality: Reliability, usability, maintainability, efficiency.
- Process Quality: Adherence to standards and methodologies used during development.

Dimensions of Quality (McCall's Model & ISO 9126)

Dimension	Description	Example
Functionality	Performs intended functions	Calculator app performing correct operations
Reliability	Consistent performance under conditions	Banking app uptime
Usability	Ease of use and learning	Intuitive UI
Efficiency	Optimal use of resources	Fast response under heavy load
Maintainability	Ease of modification	Modular code
Portability	Works across environments	Runs on Windows, Linux, macOS

Software Quality Metrics

Metrics quantify quality in measurable terms.

a. Product Metrics
b. Process Metrics
c. Project Metrics

Product Metrics

- LOC (Lines of Code) – size metric.
- Cyclomatic Complexity (McCabe)
  - measures decision points in code.
  - Formula:[V(G) = E - N + 2P]
  - where E = edges, N = nodes, P = connected components.

Process Metrics

- Defect density = (Number of defects) / (Size of software in KLOC).
- Defect removal efficiency = (Defects removed before release / Total defects) × 100.

Project Metrics

Schedule variance (SV) = EV - PV

Cost variance (CV) = EV - AC

Where EV = Earned Value, PV = Planned Value, AC = Actual Cost

Example:

If a project was planned for 100 hours (PV = 100), completed 80 hours of work (EV = 80), and consumed 110 hours (AC = 110):

SV = -20 (behind schedule)

CV = -30 (over budget)

Coding Style

A consistent style improves readability, maintainability, and debugging.

Key Guidelines:

- Follow naming conventions (e.g., camelCase in Java, snake_case in Python).
- Use indentation (4 spaces).
- Add comments and docstrings for clarity.
- Avoid long functions (≤ 50 lines).

Static Analysis

Static analysis inspects code without executing it to detect potential issues early.

Common Checks:

- Unused variables
- Memory leaks
- Security vulnerabilities
- Code complexity
- Violation of coding standards

Example Tools

Tool	Language	Purpose
SonarQube	Multi-language	Code quality dashboard
Pylint	Python	Enforces coding conventions
Checkstyle	Java	Style compliance
PMD	Java	Detects code smells
Clang Static Analyzer	C/C++	Security and memory checks

Verification & Validation

Aspect	Verification	Validation
Question answered	Are we building the product right?	Are we building the right product?
Activity type	Static	Dynamic
Performed by	QA, reviewers	Testers, users
Methods	Reviews, inspections	Testing, prototyping

Continued…

Example:

Verification → Checking requirements documents for completeness.

Validation → Running the final product to ensure it meets user needs.

Requirements	↔	Acceptance Testing
Design	↔	System Testing
Coding	↔	Unit Testing

Verification Methods

- Walkthroughs – Developer-led discussion
- Inspections – Formal peer review
- Checklists – To ensure standards compliance

Validation Methods

- Black-box testing
- System testing
- User acceptance testing (UAT)

Testing Techniques

- Black-box Testing – Tests functionality without seeing the code.
  - Equivalence Partitioning: Divide input into valid/invalid partitions.
  - Boundary Value Analysis (BVA): Test values at limits.Example: For valid input range 1–10, test 0,1,10,11.
- White-box Testing – Tests internal logic.
  - Statement coverage: Execute every statement.
  - Branch coverage: Cover all possible decision branches.
- Gray-box Testing – Mix of both, often used in integration testing.

Test Case Generation

Steps:

1. Identify test objectives from requirements.
2. Define input data and expected results.
3. Document each case with unique ID.
4. Include preconditions and postconditions.

Example

Test ID	Input	Expected Output
TC01	Valid username/password	Dashboard loads
TC02	Invalid password	Error message shown
TC03	Empty fields	Prompt to fill required fields

Version Control

Tracks and manages changes in software artifacts.

Types:

- Centralized: SVN
- Distributed: Git

Git Commands Example:

```
git init

git add .

git commit -m "Initial commit"

git branch feature-login

git merge feature-login
```

Version Numbering

- Major.Minor.Patch (e.g., 3.1.4)
- Semantic versioning:
  - Major → incompatible changes
  - Minor → backward-compatible features
  - Patch → bug fixes

Continuous Integration (CI)

- Developers integrate code frequently into a shared repository.
- Each integration triggers an automated build and test.

CI Pipeline Stages:

1. Commit
2. Build
3. Test
4. Deploy

Tools: Jenkins, Travis CI, GitHub Actions, GitLab CI/CD

Project Management

Phases of Software Project Management

1. Initiation – Define goals, feasibility.
2. Planning – Schedule, resources, budget.
3. Execution – Development and testing.
4. Monitoring – Track progress, mitigate issues.
5. Closure – Deliver, review, and archive.

Risk Analysis

Risk = Probability × Impact

Risk Types:

Technical: New technology failures

Schedule: Slippages

Cost: Budget overruns

Personnel: Staff turnover

Risk Management Steps:

1. Identify
2. Analyze
3. Prioritize
4. Mitigate
5. Monitor

Configuration Management

Configuration Management (CM) ensures consistency of a product's performance and functional attributes throughout its life.

- Configuration Identification: Label and define configuration items (CIs).
- Configuration Control: Approve and document changes.
- Status Accounting: Record changes and versions.
- Configuration Audit: Verify completeness and consistency.

Tools: Git, Subversion, IBM ClearCase

Example:

Tracking multiple releases of an app — v1.0 (initial), v1.1 (bug fixes), v2.0 (new UI).

Cost Analysis and Estimation

Need for Estimation

- Predict effort, cost, and schedule.
- Avoid overruns and resource bottlenecks.

COCOMO Model

Basic formula:

Effort = a **x** (KLOC) **^** b]where a and b depend on project type.

Project Type	a	b
Organic	2.4	1.05
Semi-detached	3.0	1.12
Embedded	3.6	1.20

Summary

Module	Key Focus	Tools/Examples
Software Quality & Metrics	Reliability, maintainability	Cyclomatic complexity
Coding & Static Analysis	Coding standards	Pylint, SonarQube
Verification & Validation	Development-Testing	Reviews, UAT
Testing Techniques	Black-box, White-box	Test case design
Versioning & CI	Git, Jenkins	CI pipeline demo
Risk & Configuration Management	Risk register, CMP	Jira, Git
Cost Estimation	COCOMO, Delphi	Effort calculation

Thank You!