TESTING ESSENTIALS

1.0 INTRODUCTION TO SOFTWARE TESTING

1.1 Testing History and Description

Software testing evolved from simple "debugging" activities to a structured, well-defined engineering discipline. Initially, testing was done only after development, but modern practices integrate testing throughout the Software Development Life Cycle (SDLC). Software testing is the process of evaluating a system or application to verify that it meets specified requirements and to identify defects.

1.2 Need for Testing

Testing is required to:

- Ensure the product works as per requirements
- Identify defects before release
- Improve quality, reliability, and performance
- Reduce business risk
- Increase customer satisfaction

1.3 Defect Description

A defect (bug) is any deviation of the actual behavior from the expected behavior. Defects can arise due to requirement gaps, design issues, coding mistakes, or environment problems.

1.4 Principles of Testing

- Testing shows presence of defects, not absence
- Exhaustive testing is impossible
- Early testing saves time and cost
- Defect clustering
- Pesticide paradox
- Testing is context dependent

- Absence-of-errors fallacy

1.5 Quality Assurance and Quality Control

- **Quality Assurance (QA):** Process-oriented, focuses on preventing defects
- **Quality Control (QC):** Product-oriented, focuses on identifying defects in the product

1.6 Scope of Testing

Defines what will be tested and what will not be tested, including features, non-functional aspects, and constraints.

1.7 Occurrence of Testing

Testing occurs at all stages: requirement phase, design phase, development phase, and maintenance phase.

1.8 Constraints of Testing

- Time limitation
- Budget limitation
- Resource limitation
- Incomplete requirements

1.9 Software Tester Roles

- Understand requirements
- Design test cases
- Execute tests
- Log and track defects
- Report test status
- Support release decisions

1.10 SDLC Overview

SDLC defines the phases involved in software development: Requirement, Design, Development, Testing, Deployment, and Maintenance.

1.11 Life Cycle Models

- Waterfall
- V-Model
- Iterative Model
- Spiral Model
- Agile Model

1.12 STLC Overview

Software Testing Life Cycle includes:

- Requirement analysis
- Test planning
- Test design
- Test execution
- Defect tracking
- Test closure

1.13 Agile Testing

Testing is continuous, integrated with development, and focuses on fast feedback, automation, and collaboration.

2.0 TEST PLANNING

2.1 Test Strategy and Planning

Test strategy defines overall testing approach. Test plan describes scope, objectives, resources, schedule, and deliverables.

2.2 Customizing Test Process

Tailoring testing activities based on project size, risk, domain, and timeline.

2.3 Budgeting Overview

Includes cost for resources, tools, environment, and effort.

2.4 Scheduling

Defines timelines for test design, execution, defect fixing, and re-testing.

2.5 Risk and Configuration Management

- Risk management identifies and mitigates testing risks
- Configuration management controls versions of builds, test cases, and documents

---

3.0 DESIGN OF TESTING

3.1 Test Scenarios

High-level test conditions that describe what to test.

3.2 Test Cases and Test Data

- Test case: Step-by-step procedure to validate a feature
- Test data: Input values used to execute test cases

3.3 Difference Between Test Case and Test Scenario

- Scenario: What to test
- Test case: How to test

3.4 Test Case Creation for Application

Includes:

- Test case ID
- Description
- Preconditions
- Steps
- Expected result
- Actual result

- Status

3.5 Traceability Matrix (RTM)

RTM maps requirements to test cases to ensure complete coverage and to track impact of changes.

4.0 TECHNIQUES OF TESTING (DYNAMIC TECHNIQUES)

4.1 Black-Box / Specification-Based Techniques

Testing based on requirements without knowing internal code.

4.2 Boundary Value Analysis (BVA)

Testing at the edges of input ranges.

4.3 Decision Table Testing

Used when different combinations of inputs produce different outputs.

4.4 Equivalence Partitioning

Dividing inputs into valid and invalid groups and testing one value from each group.

4.5 Experience-Based Techniques

4.5.1 Error Guessing

Based on the tester's experience to predict defect-prone areas.

4.5.2 Exploratory Testing

Simultaneous learning, test design, and execution without predefined scripts.

5.0 STATIC TECHNIQUES

5.1 Importance of Reviews in STLC

Early detection of defects in requirements, design, and code without executing the software.

5.2 Review Activities

- Planning
- Overview meeting
- Preparation
- Review meeting
- Rework
- Follow-up

5.3 Roles and Responsibilities During Review

- Author
- Reviewer
- Moderator
- Scribe
- Manager

6.0 LEVELS OF TESTING

6.1 Unit Testing

Testing individual components or modules.

6.2 Integration Testing

Testing interaction between integrated modules.

6.3 System Testing

Testing the complete system against requirements.

6.4 User Acceptance Testing (UAT)

Testing by end users to validate business requirements.

7.0 TYPES OF TESTING

- Regression Testing: Ensure new changes do not break existing features
- Smoke Testing: Basic build verification
- Database Testing: Validate data integrity and queries
- Load Testing: Check behavior under expected load
- Performance Testing: Check speed, response time, and stability
- Compatibility Testing: Check across devices, OS, browsers
- Security Testing: Identify vulnerabilities
- Volume Testing: Test with large data volumes
- Stress Testing: Test beyond limits
- Usability Testing: Check user-friendliness
- Internationalization Testing: Check support for multiple languages
- Localization Testing: Check region-specific behavior

8.0 EXECUTING TEST

8.1 Overview on Build and Release

Build is a deployable version of the application provided by development for testing.

8.2 Release Notes

Document describing new features, changes, fixes, and known issues.

8.3 Pre-QA Checklist

Ensures test environment, test data, and build readiness.

8.4 Entry and Exit Criteria

- Entry: Conditions to start testing
- Exit: Conditions to stop testing

8.5 Test Execution

Running test cases, recording results, and logging defects.

---

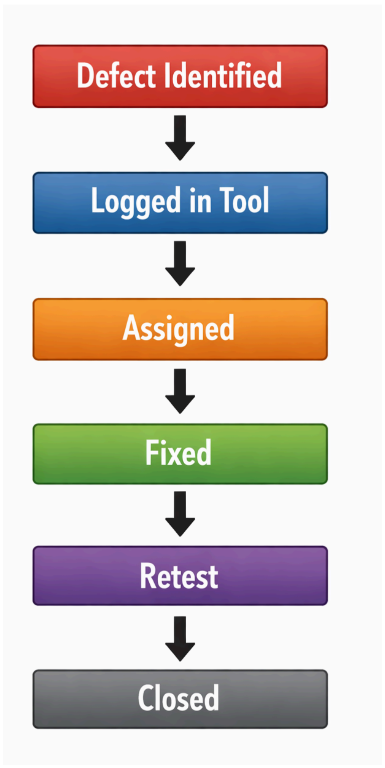
9. Managing Defects

Defect management is a structured process used to identify, document, track, prioritize, and resolve defects throughout the testing life cycle. It ensures defects are properly communicated between testers and developers and are fixed within agreed timelines to maintain product quality.

Example

Search button not working → defect logged → developer fixes → tester retests → defect closed

FLOW



9.1. Defect Prevention

Defect prevention is a proactive quality assurance approach that focuses on preventing defects from being introduced into the software rather than detecting them later. It involves process improvements, reviews, and best practices to reduce errors at early stages.

Example

Reviewing requirements prevents incorrect feature implementation

9.2. Defect Discovery

Defect discovery is the activity of identifying flaws, errors, or deviations in software during various testing and review stages. It ensures that problems are detected early so they can be corrected before release.

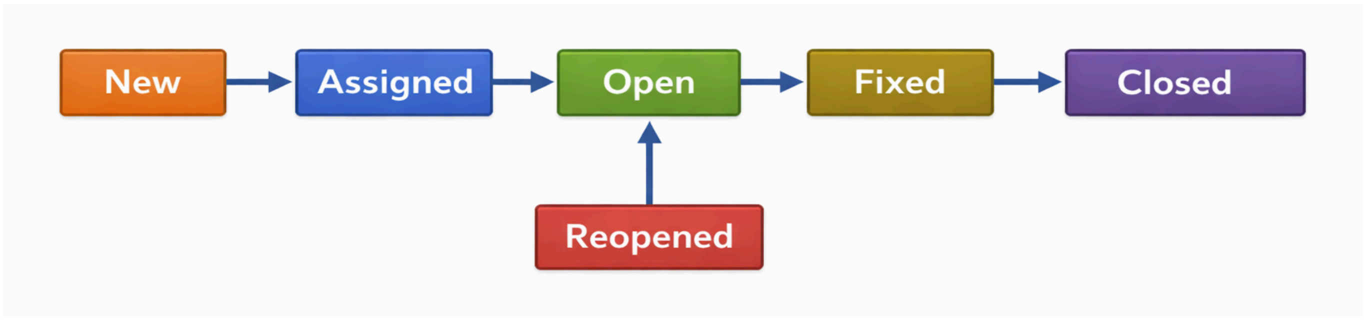
Example

Tester finds validation error during form testing

9.3. Defect Life Cycle

The defect life cycle describes the sequence of states a defect passes through from discovery to closure. It ensures every defect is tracked, reviewed, fixed, retested, and verified properly.

Flow



9.4. Severity and Priority

Severity defines the technical impact of a defect on the system’s functionality, whereas priority defines how urgently the defect should be fixed from a business perspective.

Example

App crash → High severity, High priority
Spelling mistake → Low severity, Low priority

9.5. Overview on RCA (Root Cause Analysis)

Root Cause Analysis is a systematic approach used to identify the underlying cause of defects rather than just fixing symptoms. RCA helps teams prevent the same issues from occurring again by improving processes and practices.

Example

Repeated calculation error → RCA finds incorrect business rule interpretation

9.6. Hands-on: Identify and Log Defects

This is the practical activity where testers identify defects during test execution, collect necessary evidence, and log defects with clear information so developers can easily reproduce and fix them.

Steps

- Identify defect
- Reproduce
- Document steps
- Assign severity/priority
- Submit defect

9.7. Bugzilla Tool

Bugzilla is a web-based defect tracking system used to record, manage, and monitor software defects throughout their life cycle. It helps teams maintain transparency and accountability in defect resolution.

10. Team Collaboration & Reporting

Team collaboration and reporting involve continuous communication among testers, developers, managers, and stakeholders to share testing progress, risks, and quality status. Effective collaboration ensures faster issue resolution and smooth project delivery.

10.1. Test Status Reports

A test status report is a periodic document that provides stakeholders with real-time information on testing progress, test execution results, defect trends, and overall quality status.

10.2. Test Closure Reports

Test closure report is the final document prepared at the end of the testing phase, summarizing all testing activities, results, defects, metrics, lessons learned, and sign-off details.

10.3. Tester and Developer Collaboration

Tester–developer collaboration is a continuous partnership where testers identify quality issues and developers resolve them efficiently. This collaboration reduces rework, speeds up delivery, and improves software quality.

10.4. Client Interaction

Client interaction refers to communication between the QA team and the customer to understand requirements, clarify doubts, provide testing updates, and ensure client expectations are met.

10.5. Onshore/Offshore Model

The onshore/offshore delivery model is a global development approach where different project activities are distributed across locations to reduce cost, improve productivity, and ensure 24/7 work cycles.

10.6. Mitigate Current Challenges

Mitigating challenges means identifying risks in testing (time constraints, requirement changes, environment issues) and applying strategies such as automation, agile practices, and better communication to minimize impact.

---

11. Overview on Metrics & Measurements

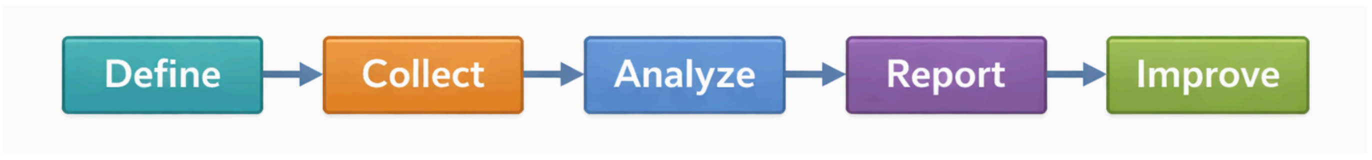
Metrics and measurements are quantitative indicators used to evaluate the effectiveness, efficiency, and quality of testing activities. They help teams make informed decisions and improve processes.

11.1. Test Metrics Benefits

Test metrics provide visibility into testing progress, quality risks, defect trends, and productivity, enabling management to take corrective actions early.

11.2. Life Cycle of Metrics

The metrics life cycle defines how metrics are planned, collected, analyzed, reported, and used for continuous improvement in testing.



11.3. Test Metrics Types

Process Metrics

Measure testing process efficiency (execution %, defect removal efficiency)

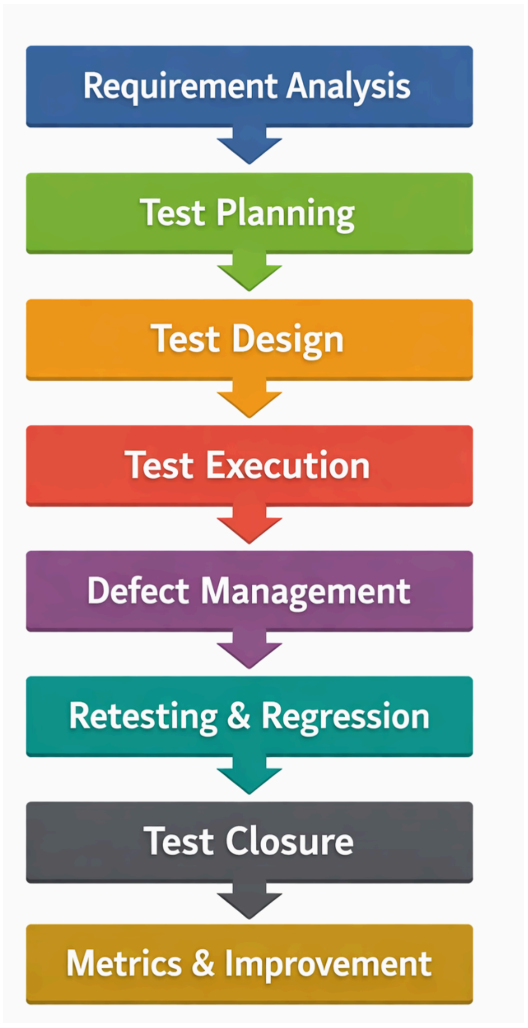
Product Metrics

Measure product quality (defect density, test coverage)

Project Metrics

Measure project performance (effort, schedule variance)

TESTING FLOW



CONCLUSION

This document provides a complete overview of software testing concepts, processes, techniques, execution, defect management, collaboration, and metrics. It serves as a strong foundation for understanding manual testing and test process management in real-time projects.