INTRODUCTION TO TESTING

Mansi Thakur

Table of Contents

1. What is Software Testing?

Software testing is the process of evaluating a software application to identify any gaps, errors, or missing requirements compared to the actual requirements. It involves executing the application with the intent of finding defects and ensuring the final product is fit for use.

1.1 Key Concepts in Testing

1. **Validation:** Are we building the right product?
2. **Verification:** Are we building the product, right?
3. **Defect:** Any deviation from expected behavior or requirements.

1.2 Additional Terminologies

- **Test Case:** A set of conditions under which a tester determines whether a feature works as intended.
- **Bug/Defect:** An issue where the application does not meet requirements.
- **Test Environment:** The setup required to execute test cases, including hardware, software, network, and configurations.

2. Why is Software Testing Important?

Software testing plays a crucial role in delivering high-quality products to end users. Here's why it matters:

1. **Ensures Functionality:** Confirms that the software works as expected and meets business goals.
2. **Improves Security:** Identifies vulnerabilities to protect data and users from threats.
3. **Cost Efficiency:** Fixing bugs during development is far cheaper than post-release fixes.
4. **User Satisfaction:** A bug-free product provides a seamless user experience, increasing trust and loyalty.
5. **Compliance:** Ensures the software adheres to legal, regulatory, and industry standards.

Real-Life Example:
A major bug in an e-commerce application caused payment gateway failures during peak sales hours. This issue, discovered during testing, saved millions in potential revenue loss.

3. Who Performs Software Testing?

Testing involves collaboration between different roles:

1. **Testers (QA Engineers):**
   - Write and execute test cases.
   - Identify, log, and verify defects.
   - Collaborate with developers to resolve issues.
2. **Developers:**
   - Conduct **unit testing** to ensure individual code modules work correctly.
   - Fix defects identified during testing phases.

3. **Business Analysts (BAs):**
   - Validate that the system meets business requirements.
   - Review test cases to ensure alignment with user stories or requirements.
4. **Test Managers:**
   - Plan, monitor, and control the testing process.
   - Allocate resources and manage timelines.
5. **Beta Testers (End Users):**
   - Provide feedback on the application's usability and performance in real-world conditions.

4. When to Start Testing?

Testing should begin as early as possible in the **Software Development Life Cycle (SDLC)** to minimize defects and costs.

4.1 Phases for Starting to Test

1. Requirement Analysis:
   - Testers collaborate with stakeholders to identify testable requirements.
   - Example: Ensuring that a "Forgot Password" feature is included in specifications.
2. Design Phase:
   - Testers review wireframes or system designs to identify ambiguities.
   - Example: Verifying that a user interface supports accessibility standards.
3. Development Phase:
   - Developers conduct **unit tests** to verify their code.
   - Testers create **test cases** based on features being developed.

4.2 Benefits of Early Testing

- Reduces the cost of fixing defects.
- Improves understanding of potential risks.
- Enhances the overall quality of the product.

5. When to Stop Testing?

Testing cannot go on indefinitely. The decision to stop is based on several criteria:

5.1 Exit Criteria for Stopping Testing

1. **All Test Cases Executed:** Ensure that all planned test cases are executed, and critical ones pass.
2. **No High-Severity Defects:** Ensure all critical and major defects are resolved.
3. **Test Coverage Achieved:** Confirm that all requirements have been tested (e.g., 90% coverage in functional areas).
4. **Time Constraints:** If deadlines or release schedules are met, testing stops with sign-off from stakeholders.
5. **Diminishing Returns/MTBF(Meantime between failure):** If additional testing finds few or no defects, it may be time to stop.

6. Software Testing Life Cycle or How is Testing Done?

Testing follows a systematic process to ensure thorough evaluation of the software. The STLC consists of six major stages. Each stage contributes to the overall quality of the software and includes specific tasks and deliverables:

6.1 Phases in the Testing Process

A) Requirement Analysis

This is the initial phase where the testing team analyzes and understands the requirements to identify testable features.

Activities:

- Review requirements documents (e.g., Business Requirement Document (BRD), System Requirement Specification (SRS)).
- Identify testable requirements.
- Collaborate with stakeholders to clarify any ambiguities.
- Identify the types of testing to be performed (e.g., functional, performance, security).

Deliverables:

- Requirement traceability matrix (RTM): Maps requirements to test cases.
- Test strategy (high-level testing approach).

Entry Criteria:

- Approved requirements documentation.

Exit Criteria:

- Requirements are clear and testable.

B) Test Planning

In this phase, the test strategy and plan are developed to define the scope, approach, resources, and schedule for testing.

Activities:

- Develop a detailed **Test Plan** document.
- Identify test tools (e.g., Selenium, JMeter).
- Estimate testing effort (using techniques like Work Breakdown Structure (WBS) or past data).
- Allocate roles and responsibilities.

Deliverables:

- Test Plan document.
- Effort estimation report.

Entry Criteria:

- Clear requirements and test strategy.

Exit Criteria:

- Approved Test Plan.

C) Test Case Development

In this stage, detailed test cases and test scripts are designed based on the identified requirements.

Activities:

- Write detailed test cases (including steps, expected results, and preconditions).

- Prepare test data for each test case.
- Review and finalize test cases with stakeholders.
- Create automation test scripts (if applicable).

Deliverables:

- Test cases and test scripts.
- Test data.

Entry Criteria:

- Approved test plan.
- Requirement traceability matrix (RTM).

Exit Criteria:

- Reviewed and finalized test cases.

D) Test Environment Setup

This phase involves preparing the test environment where testing will be executed. It ensures that all required resources are available and configured correctly.

Activities:

- Define hardware and software requirements.
- Configure test servers, databases, and network settings.
- Install required tools and frameworks.
- Verify environment readiness through smoke testing.

Deliverables:

- Test environment setup document.
- Test data loaded into the environment.

Entry Criteria:

- Test plan and test cases.
- Availability of the test environment.

Exit Criteria:

- Test environment is ready and validated.

E) Test Execution

This phase involves executing the test cases in the prepared environment and logging any defects that are discovered.

Activities:

- Execute manual and automated test cases.
- Log defects into a defect tracking tool (e.g., Jira, Bugzilla).
- Retest defects after they are fixed.
- Perform regression testing to ensure no new defects were introduced.

Deliverables:

- Test execution report.
- Defect logs.

Entry Criteria:

- Test cases and test scripts are ready.
- Test environment is set up.

Exit Criteria:

- All planned test cases are executed.

F) Test Closure

The final phase involves evaluating testing activities and documenting the results. It focuses on ensuring that all objectives have been met.

Activities:

- Collect test metrics (e.g., defect density, test coverage, defect leakage).
- Prepare a **Test Summary Report** outlining key findings.
- Conduct a retrospective to identify lessons learned.
- Archive test artifacts for future reference.

Deliverables:

- Test Summary Report.
- Final RTM (to ensure all requirements are tested).
- Lessons learned document.

Entry Criteria:

- All planned test cases are executed.
- Defect reports are closed or deferred.

Exit Criteria:

- Stakeholders sign off on the testing phase.

6.2 Key Artifacts in STLC

1. **Test Plan:** A document detailing the testing scope, objectives, schedule, and approach.
2. **Requirement Traceability Matrix (RTM):** Ensures test coverage by mapping requirements to test cases.
3. **Test Cases and Scripts:** Step-by-step procedures for verifying each requirement.
4. **Defect Reports:** Detailed logs of issues, including severity and status.
5. **Test Summary Report:** High-level overview of testing activities and results.

6.3 Benefits of Following STLC

1. **Structured Process:** Ensures testing activities are systematic and organized.
2. **Early Defect Detection:** Issues are identified early, reducing costs.
3. **Improved Quality:** Each phase focuses on a specific aspect of quality.
4. **Metrics and Insights:** Provides valuable insights for process improvement.
5. **Traceability:** Ensures all requirements are tested and accounted for.

6.4 Deliverables in Testing

1. **Test Plan:** Defines the testing scope, objectives, and schedule.
2. **Test Cases:** Detailed steps for validating each feature or functionality.
3. **Defect Reports:** Logs of identified issues with severity and reproduction steps.
4. **Test Summary Report:** A high-level overview of testing progress, results, and status.

6.5 Challenges in Software Testing

1. **Incomplete Requirements:** Leads to gaps in test coverage.
2. **Tight Deadlines:** Forces testers to prioritize tasks, potentially missing critical defects.
3. **Changing Requirements:** Requires frequent updates to test cases.
4. **Complex Environments:** Difficulties in setting up realistic test conditions.
5. **Human Errors:** Risk of mistakes in manual testing.

6.6 Best Practices in Software Testing

1. **Shift Left Testing:** Start testing as early as possible in the SDLC.
2. **Automate Repetitive Tasks:** Use automation tools for regression testing.
3. **Focus on High-Risk Areas:** Prioritize testing of critical functionalities.
4. **Encourage Collaboration:** Maintain open communication between testers, developers, and stakeholders.
5. **Document Everything:** Maintain comprehensive records of test cases, results, and defects.

7. What is a defect or bug life cycle?

Defect life cycle, also known as bug life cycle, is a systematic process that describes the various stages a defect or bug goes through during its lifetime, from its identification to its closure. It outlines the steps involved in reporting, tracking, fixing, verifying, and closing a defect within a software development or quality assurance process. The defect life cycle helps in managing and resolving issues efficiently and ensures a structured approach towards defect management.

7.1 Typical stages of a defect life cycle

New: This stage represents the initial detection or reporting of a defect. It occurs when a tester or user identifies a deviation from the expected behavior or functionality. The defect is reported with relevant details, such as a description, steps to reproduce, and any supporting attachments or screenshots.

Open: After the defect is reported, it enters the open stage. At this point, the development or testing team reviews the defect report, validates its authenticity, and assigns it to the appropriate team member responsible for further investigation and resolution.

Assigned: Once the defect is assigned, the responsible team member begins analyzing and understanding the defect's cause, impact, and severity. They may also prioritize the defect based on its criticality and assign an appropriate fix target.

In Progress: In this stage, the assigned team member actively works on fixing the defect. They modify the source code or make necessary configuration changes to address the reported issue. During this process, collaboration with other team members or stakeholders may occur to gather additional information or provide clarifications.

Fixed: When the assigned team member has completed the necessary modifications or fixes, the defect is marked as fixed. The fix undergoes internal testing or quality checks to ensure that it has indeed resolved the reported issue.

Pending Retest: After the defect is fixed, it moves to the pending retest stage. At this point, the testing team assigns the defect to a tester for verification. The tester performs retesting to confirm whether the fix has effectively resolved the defect without introducing new issues.

Retest: During the retest stage, the assigned tester executes the test cases related to the defect and verifies if the defect is no longer reproducible. If the issue is resolved successfully, the defect proceeds to the next stage. However, if the defect is still present, it is reopened and sent back to the developer for further investigation.

Reopened: The bug was previously closed but has been reopened due to the issue still being present or reoccurring. It requires further investigation and resolution.

Verified: After successful retesting, the bug is verified as resolved. It means that the fix has effectively resolved the reported issue, and the bug is ready to be closed.

Closed: Once the defect is retested and verified as resolved, it is marked as closed. The defect is considered closed when it has been adequately addressed, confirmed as fixed, and there is no further action required. The closed defects are usually documented for future reference and analysis.

Duplicate: The reported bug is a duplicate of an existing bug that has already been reported and is being tracked separately. It indicates that the same issue has been reported more than once.

Cannot Reproduce: The reported bug cannot be reproduced by the assigned team member, indicating a difficulty in identifying or understanding the issue. It may require additional information or steps to replicate the bug.

Deferred: The bug is acknowledged but postponed for a future release or a later stage of development. It means that the issue will not be addressed in the current release or iteration.

Not a Bug: The reported issue is not a bug but rather an intended or expected behavior of the software. It clarifies that the observed behavior is by design and does not require any changes.

Won't Fix: The bug has been reviewed, but the decision has been made not to fix it, either due to low priority, low impact, or other considerations. It indicates that the reported issue will not be addressed.

Pending Approval: The bug fix is awaiting approval from relevant stakeholders or a designated authority before proceeding with further actions. It may require approval before implementing the fix.

8. STLC vs. SDLC

Aspect	STLC	SDLC
Focus	Testing activities	Entire development lifecycle
Phases	Specific to testing	Covers design, development, and testing

Objective	Validate and verify quality	Deliver functional software
Stakeholders	Testers, QA Engineers	Developers, Product Owners, Testers