Testing Terminologies

1. Software Testing

Software testing is a process to ensure that the system or application under test is working as per requirements or specifications. This process intends to find bugs within the system/application before it is launched to Production. It is to make sure that:

- All the mentioned specifications are working as expected
- All the bugs are uncovered
- None of the features are behaving what it is not expected to do

2. Software Testing Life Cycle

Software Testing Life Cycle (STLC) is imprtant of Software Testing terms defines process to carry out all the testing activities in systematic and planned manner. All the activities go in the sequence where they are termed as Phases. Below are the different phases of STLC (in sequence):

- Requirement Analysis
- Test Planning / Test Strategy
- Test Case Development
- Environment Setup
- Test Execution
- Test Cycle Closure

3. Test Strategy

Test strategy is a set of guidelines that explain test design and determines how testing needs to be done. Components of Test Strategy includes: objectives and scope, documentation formats, test processes/approach, team reporting structure, client communication strategy

4. Test Plan

Test Plan is a document that defines everything about the project with respect to testing activities. As per IEEE 829 standards, below are the components of a Test Plan document:

- Test Plan identifier
- References
- Introduction
- Test Items

- Risks
- Items to be tested
- Features excluded for testing
- Testing Approach
- Test Pass and Fail Criteria
- Resumption/Suspension Criteria
- Test deliverables
- Test Environment Setup
- Training and Staffing
- Team Member Responsibilities
- Testing Schedule
- Planning for Risks and Contingency Plans
- Approvals

5. Static Testing

Static testing is the testing of the software manually by reviewing or with a set of tools, but they are not executed. It is the verification process which does not include actual execution of component/system as a whole. Different types of reviews are conducted to ensure the correctness.

- Informal Reviews
- Technical Reviews
- Walkthrough
- Inspection
- Static Code Review

6. Informal Reviews

Informal Review is the type of review which doesn't follow any process to find errors in the document. Here documents are just reviewed and comments are given informally. It does not follow any process as such. So there will be no documentation available for components reviewed/review comments / review comments incorporation. This type of review cannot be tracked to measure efforts and metrics.

7. Technical Reviews

Technical specification of the software is reviewed to check whether it is suitable for the implementation. Peers do this review and reports discrepancies in a formal way. It follows standards and mainly concentrates on technical documents related to the software such as Test Strategy, Test Plan, and requirement specification documents.

8. Walkthrough

The product is explained to the team by the author, who leads the meeting for this review type. Other participants can discuss over their questions, and the review comments are noted by Scribe (one who notes down all the discussion).

9. Inspection

Trained Moderator leads the meeting to find defects. It is a formal type of review which follows a strict process to find the defects. Reviewers have a checklist to review the work products. They record the defect and inform the participants to rectify those errors.

10. Static Code Review

The source code is systematically reviewed without executing it. Syntax, coding standards, code optimizations, etc. are reviewed at any point during the development phase. This is also termed as White box testing

11. Dynamic Testing

Dynamic testing (or dynamic analysis) is the testing of the dynamic behavior of the code. The actual functionality of the system is tested for correctness, completeness, reliability, and resistance. Physical response to a wide set of input data is analyzed. Below are the techniques used:

- Unit Testing
- Integration Testing
- System Testing

12. Unit Testing

Testing of individual software components or modules to ensure that it is working correctly as a component alone. Typically done by the programmer and not by testers, as it requires detailed knowledge of the internal program.

13. Integration Testing

Individual modules are integrated to verify that the functionality between the modules works correctly after integration is done. Usually, two or more components are integrated and data flow between them is tested.

14. System Testing

The entire system is tested to check whether it meets the requirement specifications as intended. All the components of the system are grouped together and critical business scenarios are tested firstly. Then the entire system is tested for all possible executions. This is the extension to integration testing.

15. Test Case (Important Software Testing Terms)

A test case is a set of steps and preconditions which a tester uses to execute software tests. Components of a Test Case Document include: Test Case Identifier, Requirement ID to map, Precondition, Input data, Steps to execute, Expected Result, Execution Status (PASS / FAIL / BLOCKED)

16. Test Suite

The test suite is a container that has a set of tests. A Test case can be added to multiple tests. The test suite is created after Test Plan creation.

17. Test case Design Techniques

- Deriving test cases directly from a requirement specification or black box test design technique: Boundary Value Analysis (BVA), Equivalence Partitioning (EP), Decision Table Testing, State Transition Diagrams, Use Case Testing
- Deriving test cases directly from the structure of a component or system: Statement Coverage, Branch Coverage, Path Coverage
- Deriving test cases based on tester's experience on similar systems or testers intuition: Error Guessing, Exploratory Testing

18. Black box testing

Black-box testing is a technique that verifies the functionality of the application. It does not require knowledge of internal structures used, memory and source code.

19. Equivalence Partitioning

Equivalence Partitioning is the technique to divide input data/test conditions into the groups of equivalence classes. This can be applied where there is a range in the input field.

20. Boundary Value Analysis / Boundary Testing

Boundary testing is the process of testing between the extreme ends or boundaries between partitions of the input values. Selection of input variable values in the range will be as below (example: 0 – 99)

- Minimum (0)
- Just above the minimum (1)
- A nominal value (2 – 97)
- Just below the maximum (98)
- Maximum (99)

21. Decision Table testing

Decision table testing is a technique to determine the test scenarios for complex business logic. Combination of inputs is tried to produce different results. This technique is sometimes also referred to as a 'cause-effect' table.

22. State Transition Diagrams

State Transition Diagram is a test design technique in which changes in input conditions cause state changes in the Application under Test (AUT). – Example: All possible states in ATMs, Coffee Vending machines, etc.

23. Use Case Testing

Use Case Testing is a technique to identify test cases that cover the entire system, on a transaction by transaction basis from start to the finishing point. It is used widely in developing tests at system or acceptance level. All possible business critical flows are considered as a priority to identify.

24. White box testing

White-box testing (also known as clear box testing, glass box testing, and transparent box testing, and structural testing) is a method of testing software that tests internal structures or workings of an application. This does not check how the application actually behaves functionality-wise.

25. Statement Coverage / Line coverage / Segment coverage

Statement coverage is a white box testing technique that executes all the statements at least once in the source code. The number of statements in true condition executed is calculated for coverage.

26. Branch Coverage / Decision coverage / All-edges coverage

Branch coverage is a testing method to ensure that each possible branch in the source code is executed at least once. It is to ensure all possible branches (i.e., True and False) are reachable and executable.

27. Path Coverage

Path coverage is a white box testing technique that executes all the possible paths of the software under test. Path here ensures code reachability and executability for complete flow in each possible case

28. Error Guessing

Error guessing is a test method in which test cases are designed based on the experience of a tester. It requires domain knowledge and makes use of a tester’s skill to guess where the errors may occur.

29. Exploratory Testing

Exploratory testing is a random way of testing the application where testers test the application without domain knowledge, no clear understandings on specifications. The application is explored as it is being used for the first time. Minimum planning maximum execution is involved in this testing. Time allocated to this will usually be less.

30. Verification

Verification is the process to make sure that the product satisfies all the conditions that are specified. This is to ensure product behaves the way it has to. Here actual testing does not happen. It is more of review kind of activity.

31. Validation

Validation is the process to make sure the product satisfies the specified requirements by actually testing it. This is to ensure the product is built as per customer requirements

32. Incremental integration testing

This is generally called Bottom up approach for testing. Involves continuous testing of an application as and when new functionality gets added. Usually done by programmers or by testers.

33. Functional testing (Important Software Testing Terms)

Testing the functionality of the system by providing input to it and to ensure that it works according to the requirement specifications. This is also known as black box testing.

34. End-to-end testing

The complete application is tested the way as real-world uses it. Complete flow is tested here. Apart from just functionality, there are other components which are given importance: Database, network communications, request-response, interactions with other applications and hardware.

35. Smoke Testing (Important Software Testing Terms)

When the build is deployed to the testing environment, a basic check is performed to ensure that the critical and major business functionalities are working fine. Here environment issues and blockers are resolved immediately if encountered.

36. Sanity testing (Important Software Testing Terms)

When the build is deployed to the testing environment, a basic check is performed to ensure that the application is loading correctly i.e., all the pages, page components like buttons, widgets, touts, rails, segments are loaded in time and correctly in position. It is to mainly ensure the clean visibility of the application.

37. Regression testing

Testing the application for each and every functionality. This is performed by executing test cases that are identified. It ensures that the functionalities of the application are working fine under all input conditions and there is no impact from the new or modified features. It is quite difficult to cover the entire application in regression testing so typically automation tools are used for this purpose.

38. Acceptance testing

Testing to ensure that the system meets all specified requirements. All critical cases, corner cases, major flows are tested. Specialized testing team or Customers do this testing to determine whether to accept or reject the system. This testing results in Go / No-Go decision.

39. Adhoc Testing

Testing the system randomly with good knowledge on the domain and requirements. This is more like exploratory testing but the tester should have a good hold on the system's requirement specifications.

40. Load testing

The system is tested under heavy load to ensure that it is resistant and working correctly as expected. This is to determine at what point the system's response time degrades or fails.

41. Stress testing (Important Software Testing Terms)

The system is tested beyond its load specifications to check at what point of load it fails. This includes usually putting a load to exceed storage capacity, complex database queries, continuous input to the system, continuous add/updates to the database.

42. Performance testing

Testing the system under heavy load to determine at what point of load the performance of system degrades/breaks down

43. Usability testing

Testing the application to ensure that it is understandable easily by the end-users. This is supported with user manuals, technical documentation on the feature, etc., with detailed steps and screenshots where-ever required.

44. Install / Uninstall testing

The testing software installs/uninstall / updates on different operating systems, hardware, and other software environments. This is to ensure that install / uninstall / updates do not fail in any environment. This requires at least minimum knowledge of system specifications – Registry, Event Logs, Memory usage, etc. – as they are required to analyze any failure that occurs.

45. Recovery testing

Testing to ensure system recovers from crashes, hardware failures, or other catastrophic problems.

46. Security testing

Testing the system's security level to ensure that it cannot be hacked by different penetration techniques. This determines system protects against unauthorized internal or external access, attacks.

47. Compatibility testing

Testing system performance under a wide variety of hardware/software/operating system/network environment and different combinations of them.

48. Comparison testing

Comparing the system's strength, weakness with its previous versions or similar systems in the market.

49. Alpha testing

Testing the system in the production type of environment, usually called Staging, Failover environment. This environment requires special access for anyone accessing it. All business critical, major, important flows are tested.

50. Beta testing

Testing the system by exposing it to real-world. Here end-users test the system and provide feedback/reports issues back. The system here is usually referred as the beta version.

51. Bug (Important Software Testing Terms)

Anything that produces incorrect results is called Bug. This is the terminology used by Testers.

52. Defect

Flaws in the system which results in deviation of actual results from expected results is called a defect.

53. Error

Mistake is coding is called error

54. Missing

Any requirement specification that is missed out during implementation is called Missing

55. Fault

Any condition that results in the system to fail to produce expected results is called fault

56. Failure

System, when fails to perform as required within specified performance requirements, is called failure

57. Bug Life Cycle

Bugs found during testing undergoes a cycle from logging to the closure of it which is termed as Bug Life cycle. Different phases of the bug during its lifetime goes as below: New, Assigned, Open, Fixed, Verified, Closed, Re-open, Duplicate, Need More Info, Deferred, Invalid

58. Severity (Important Software Testing Terms)

Impact of bug on the system determines severity. It is a measure to represent how severe the bug is. Critical, Major, Normal, Minor are different severity levels of a bug

59. Priority (Important Software Testing Terms)

Order in which the bug has to be fixed determines priority. It is the representation to indicate how much early the bug fixing has to be done. Higher the priority sooner the bug has to be fixed

60. Start Testing

When the build is ready and deployed to the testing environment, testing starts with basic checks like Smoke / Sanity testing. Once the Smoke / Sanity passes, regression testing starts on the build by executing test cases

61. Suspend Testing

Testing suspends/gets hold when there encounters blockers, frequent environment issues, unresolved dependencies.

62. Resume Testing

Suspended testing resumes back once the issues that resulted in suspending testing gets resolved.

63. Stop Testing (Important Software Testing Terms)

Testing activity stops under different conditions:

- Bug logging rate falls below (Testing not leading to more than 5% Critical or Major bugs)
- Budget Constraints
- Time Limits
- Management decision based on Customer’s inputs
- Test Cases completed with certain Pass %

64. Scope of Testing

Determines what has to be tested within and around the system. All requirement specifications will fall into scope of testing

65. Out-of-Scope

Any testing that has not to be performed is determined as Out-of-Scope. This will usually be third-party components, plug-ins, etc.

66. Execution

Manually performing tests following steps mentioned in the test case is called execution. Before starting execution, one should make sure there are no blockers, environment issues, major dependencies.

67. Retesting (Important Software Testing Terms)

When any bug is fixed, it will be assigned back to the tester to verify and either close or re-open it. This involves testing only for what the bug was.

68. Status report (Important Software Testing Terms)

The report that contains detailed updates on the current status of all testing activities is called Status report. It will majorly send to all stakeholders by Leads / Managers, either daily or weekly.

69. Risk

Possible problems that may occur resulting in endangering the objectives of the project and stakeholders. This is a possibility.

70. Risk Management

Process of identifying, analyzing, responding to risk factors in a systematic manner throughout the life of project is called Risk management

71. Risk mitigation

Taking appropriate steps in right time to reduce the adverse effects of risk factors is called Risk Mitigation

72. Contingency Plan

Software Testing terms Contingency Plan is where there is no change, made to risk factors but control the adverse effects of risk factors is called Contingency plan

73. Test Estimation (Important Software Testing Terms)

Estimating approximate time and effort required for testing tasks is termed Test Estimation.

74. Delphi Technique

Estimating time and effort for testing based on surveys and data collection from Experts. The task is assigned to each team member and over multiple rounds of surveys on the task, final estimation is reached.

75. Work Breakdown Structure

The complex system is divided into smaller modules. Each module is further divided into sub-modules, followed by each sub-modules divided into functionalities, and each functionality divided into sub-functionalities. Estimation is provided for each sub-functionalities and total final estimation is reached.

76. Root Cause Analysis (Important Software Testing Terms)

Analyzing the reason for the occurrence of post-production issues. This is to identify flaws in the process followed, lack of testing, and issues that caused post-production issues. Corrective measures are taken to avoid these in future.

77. Bug Triage Meeting

Meeting in which open bugs are analyzed and discussed to finalize on their severity and priority. Also, sometimes impacts are provided to be tested along with bug verification when fixed. Leads, Managers, Customers will be part of this meeting.