

Software Testing Glossary

Acceptance Testing: Testing conducted to enable a user/customer to determine whether to accept a software product. Normally performed to validate the software meets a set of agreed acceptance criteria.

Accessibility Testing: Verifying a product is accessible to the people having disabilities (deaf, blind, mentally disabled etc.).

Ad Hoc Testing: A testing phase where the tester tries to 'break' the system by randomly trying the system's functionality. Can include negative testing as well. See also Monkey Testing.

Agile Testing: Testing practice for projects using agile methodologies, treating development as the customer of testing and emphasizing a test-first design paradigm. See also Test Driven Development.

Application Binary Interface (ABI): A specification defining requirements for portability of applications in binary forms across different system platforms and environments.

Application Programming Interface (API): A formalized set of software calls and routines that can be referenced by an application program in order to access supporting system or network services.

Automated Software Quality (ASQ): The use of software tools, such as automated testing tools, to improve software quality.

Automated Testing:

- Testing employing software tools which execute tests without manual intervention. Can be applied in GUI, performance, API, etc. testing.
- The use of software to control the execution of tests, the comparison of actual outcomes to predicted outcomes, the setting up of test preconditions, and other test control and test reporting functions.

B

Backus-Naur Form: A Meta language used to formally describe the syntax of a language.

Basic Block: A sequence of one or more consecutive, executable statements containing no branches.

Basis Path Testing: A white box test case design technique that uses the algorithmic flow of the program to design tests.

Basis Set: The set of tests derived using basis path testing.

Baseline: The point at which some deliverable produced during the software engineering process is put under formal change control.

Benchmark Testing: Tests that use representative sets of programs and data designed to evaluate the performance of computer hardware and software in a given configuration.

Beta Testing: Testing of a re-release of a software product conducted by customers.

Binary Portability Testing: Testing an executable application for portability across system platforms and environments, usually for conformation to an ABI specification.

Black Box Testing: Testing based on an analysis of the specification of a piece of software without reference to its internal workings. The goal is to test how well the component conforms to the published requirements for the component.

Bottom Up Testing: An approach to integration testing where the lowest level components are tested first, then used to facilitate the testing of higher level components. The process is repeated until the component at the top of the hierarchy is tested.

Boundary Testing: Test which focus on the boundary or limit conditions of the software being tested. (Some of these tests are stress tests).

Boundary Value Analysis: In boundary value analysis, test cases are generated using the extremes of the input domain, e.g. maximum, minimum, just inside/outside boundaries, typical values, and error values. BVA is similar to Equivalence Partitioning but focuses on "corner cases".

Branch Testing: Testing in which all branches in the program source code are tested at least once.

Breadth Testing: A test suite that exercises the full functionality of a product but does not test features in detail.

Bug: A fault in a program which causes the program to perform in an unintended or unanticipated manner.

C

CAST: Computer Aided Software Testing.

Capture/Replay Tool: A test tool that records test input as it is sent to the software under test. The input cases stored can then be used to reproduce the test at a later time. Most commonly applied to GUI test tools.

CMM: The Capability Maturity Model for Software (CMM or SW-CMM) is a model for judging the maturity of the software processes of an organization and for identifying the key practices that are required to increase the maturity of these processes.

Cause Effect Graph: A graphical representation of inputs and the associated outputs effects which can be used to design test cases.

Code Complete: Phase of development where functionality is implemented in entirety; bug fixes are all that are left. All functions found in the Functional Specifications have been implemented.

Code Coverage: An analysis method that determines which parts of the software have been executed (covered) by the test case suite and which parts have not been executed and therefore may require additional attention.

Code Inspection: A formal testing technique where the programmer reviews source code with a group who ask questions analyzing the program logic, analyzing the code with respect to a checklist of historically common programming errors, and analyzing its compliance with coding standards.

Code Walkthrough: A formal testing technique where source code is traced by a group with a small set of test cases, while the state of program variables is manually monitored, to analyze the programmer's logic and assumptions.

Coding: The generation of source code.

Compatibility Testing: Testing whether software is compatible with other elements of a system with which it should operate, e.g. browsers, Operating Systems, or hardware.

Component: A minimal software item for which a separate specification is available.

Component Testing: See Unit Testing.

Concurrency Testing: Multi-user testing geared towards determining the effects of accessing the same application code, module or database records. Identifies and measures the level of locking, deadlocking and use of single-threaded code and locking semaphores.

Conformance Testing: The process of testing that an implementation conforms to the specification on which it is based. Usually applied to testing conformance to a formal standard.

Context Driven Testing: The context-driven school of software testing is a flavor of Agile Testing that advocates continuous and creative evaluation of testing opportunities in light of the potential information revealed and the value of that information to the organization right now.

Conversion Testing: Testing of programs or procedures used to convert data from existing systems for use in replacement systems.

Cyclomatic Complexity: A measure of the logical complexity of an algorithm, used in white-box testing.

D

Data Dictionary: A database that contains definitions of all data items defined during analysis.

Data Flow Diagram: A modeling notation that represents a functional decomposition of a system.

Data Driven Testing: Testing in which the action of a test case is parameterized by externally defined data values, maintained as a file or spreadsheet. A common technique in Automated Testing.

Debugging: The process of finding and removing the causes of software failures.

Defect: Nonconformance to requirements or functional / program specification

Dependency Testing: Examines an application's requirements for pre-existing software, initial states and configuration in order to maintain proper functionality.

Depth Testing: A test that exercises a feature of a product in full detail.

Dynamic Testing: Testing software through executing it. See also Static Testing.

E

Emulator: A device, computer program, or system that accepts the same inputs and produces the same outputs as a given system.

Endurance Testing: Checks for memory leaks or other problems that may occur with prolonged execution.

End-to-End testing: Testing a complete application environment in a situation that mimics real-world use, such as interacting with a database, using network communications, or interacting with other hardware, applications, or systems if appropriate.

Equivalence Class: A portion of a component's input or output domains for which the component's behaviour is assumed to be the same from the component's specification.

Equivalence Partitioning: A test case design technique for a component in which test cases are designed to execute representatives from equivalence classes.

Exhaustive Testing: Testing which covers all combinations of input values and preconditions for an element of the software under test.

F

Functional Decomposition: A technique used during planning, analysis and design; creates a functional hierarchy for the software.

Functional Specification: A document that describes in detail the characteristics of the product with regard to its intended features.

Functional Testing: See also Black Box Testing.

- Testing the features and operational behavior of a product to ensure they correspond to its specifications.
- Testing that ignores the internal mechanism of a system or component and focuses solely on the outputs generated in response to selected inputs and execution conditions.

G

Glass Box Testing: A synonym for White Box Testing.

Gorilla Testing: Testing one particular module, functionality heavily.

Gray Box Testing: A combination of Black Box and White Box testing methodologies: testing a piece of software against its specification but using some knowledge of its internal workings.

H

High Order Tests: Black-box tests conducted once the software has been integrated.

I

Independent Test Group (ITG): A group of people whose primary responsibility is software testing,

Inspection: A group review quality improvement process for written material. It consists of two aspects; product (document itself) improvement and process improvement (of both document production and inspection).

Integration Testing: Testing of combined parts of an application to determine if they function together correctly. Usually performed after unit and functional testing. This type of testing is especially relevant to client/server and distributed systems.

Installation Testing: Confirms that the application under test recovers from expected or unexpected events without loss of data or functionality. Events can include shortage of disk space, unexpected loss of communication, or power out conditions.

J, K, L

Load Testing: See Performance Testing.

Localization Testing: This term refers to making software specifically designed for a specific locality.

Loop Testing: A white box testing technique that exercises program loops.

M

Metric: A standard of measurement. Software metrics are the statistics describing the structure or content of a program. A metric should be a real objective measurement of something such as number of bugs per lines of code.

Monkey Testing: Testing a system or an Application on the fly, i.e just few tests here and there to ensure the system or an application does not crash out.

Mutation Testing: Testing done on the application where bugs are purposely added to it.

N

Negative Testing: Testing aimed at showing software does not work. Also known as "test to fail". See also Positive Testing.

N+1 Testing: A variation of Regression Testing. Testing conducted with multiple cycles in which errors found in test cycle N are resolved and the solution is retested in test cycle N+1. The cycles are typically repeated until the solution reaches a steady state and there are no errors. See also Regression Testing.

O

P

Path Testing: Testing in which all paths in the program source code are tested at least once.

Performance Testing: Testing conducted to evaluate the compliance of a system or component with specified performance requirements. Often this is performed using an automated test tool to simulate large number of users. Also know as "Load Testing".

Positive Testing: Testing aimed at showing software works. Also known as "test to pass". See also Negative Testing.

Q

Quality Assurance: All those planned or systematic actions necessary to provide adequate confidence that a product or service is of the type and quality needed and expected by the customer.

Quality Audit: A systematic and independent examination to determine whether quality activities and related results comply with planned arrangements and whether these arrangements are implemented effectively and are suitable to achieve objectives.

Quality Circle: A group of individuals with related interests that meet at regular intervals to consider problems or other matters related to the quality of outputs of a process and to the correction of problems or to the improvement of quality.

Quality Control: The operational techniques and the activities used to fulfill and verify requirements of quality.

Quality Management: That aspect of the overall management function that determines and implements the quality policy.

Quality Policy: The overall intentions and direction of an organization as regards quality as formally expressed by top management.

Quality System: The organizational structure, responsibilities, procedures, processes, and resources for implementing quality management.

R

Race Condition: A cause of concurrency problems. Multiple accesses to a shared resource, at least one of which is a write, with no mechanism used by either to moderate simultaneous access.

Ramp Testing: Continuously raising an input signal until the system breaks down.

Recovery Testing: Confirms that the program recovers from expected or unexpected events without loss of data or functionality. Events can include shortage of disk space, unexpected loss of communication, or power out conditions.

Regression Testing: Retesting a previously tested program following modification to ensure that faults have not been introduced or uncovered as a result of the changes made.

Release Candidate: A pre-release version, which contains the desired functionality of the final version, but which needs to be tested for bugs (which ideally should be removed before the final version is released).

S

Sanity Testing: Brief test of major functional elements of a piece of software to determine if its basically operational. See also Smoke Testing.

Scalability Testing: Performance testing focused on ensuring the application under test gracefully handles increases in work load.

Security Testing: Testing which confirms that the program can restrict access to authorized personnel and that the authorized personnel can access the functions available to their security level.

Smoke Testing: A quick-and-dirty test that the major functions of a piece of software work. Originated in the hardware testing practice of turning on a new piece of hardware for the first time and considering it a success if it does not catch on fire.

Soak Testing: Running a system at high load for a prolonged period of time. For example, running several times more transactions in an entire day (or night) than would be expected in a busy day, to identify and performance problems that appear after a large number of transactions have been executed.

Software Requirements Specification: A deliverable that describes all data, functional and behavioral requirements, all constraints, and all validation requirements for software/

Software Testing: A set of activities conducted with the intent of finding errors in software.

Static Analysis: Analysis of a program carried out without executing the program.

Static Analyzer: A tool that carries out static analysis.

Static Testing: Analysis of a program carried out without executing the program.

Storage Testing: Testing that verifies the program under test stores data files in the correct directories and that it reserves sufficient space to prevent unexpected termination resulting from lack of space. This is external storage as opposed to internal storage.

Stress Testing: Testing conducted to evaluate a system or component at or beyond the limits of its specified requirements to determine the load under which it fails and how. Often this is performance testing using a very high level of simulated load.

Structural Testing: Testing based on an analysis of internal workings and structure of a piece of software. See also White Box Testing.

System Testing: Testing that attempts to discover defects that are properties of the entire system rather than of its individual components.

T

Testability: The degree to which a system or component facilitates the establishment of test criteria and the performance of tests to determine whether those criteria have been met.

Testing:

- The process of exercising software to verify that it satisfies specified requirements and to detect errors.
- The process of analyzing a software item to detect the differences between existing and required conditions (that is, bugs), and to evaluate the features of the software item (Ref. IEEE Std 829).
- The process of operating a system or component under specified conditions, observing or recording the results, and making an evaluation of some aspect of the system or component.

Test Automation: See Automated Testing.

Test Bed: An execution environment configured for testing. May consist of specific hardware, OS, network topology, configuration of the product under test, other application or system software, etc. The Test Plan for a project should enumerate the test beds(s) to be used.

Test Case:

- Test Case is a commonly used term for a specific test. This is usually the smallest unit of testing. A Test Case will consist of information such as requirements testing, test steps, verification steps, prerequisites, outputs, test environment, etc.
- A set of inputs, execution preconditions, and expected outcomes developed for a particular objective, such as to exercise a particular program path or to verify compliance with a specific requirement.

Test Driven Development: Testing methodology associated with Agile Programming in which every chunk of code is covered by unit tests, which must all pass all the time, in an effort to eliminate unit-level and regression bugs during development. Practitioners of TDD write a lot of tests, i.e. an equal number of lines of test code to the size of the production code.

Test Driver: A program or test tool used to execute tests. Also known as a Test Harness.

Test Environment: The hardware and software environment in which tests will be run, and any other software with which the software under test interacts when under test including stubs and test drivers.

Test First Design: Test-first design is one of the mandatory practices of Extreme Programming (XP). It requires that programmers do not write any production code until they have first written a unit test.

Test Harness: A program or test tool used to execute tests. Also known as a Test Driver.

Test Plan: A document describing the scope, approach, resources, and schedule of intended testing activities. It identifies test items, the features to be tested, the testing tasks, who will do each task, and any risks requiring contingency planning. Ref IEEE Std 829.

Test Procedure: A document providing detailed instructions for the execution of one or more test cases.

Test Scenario: Definition of a set of test cases or test scripts and the sequence in which they are to be executed.

Test Script: Commonly used to refer to the instructions for a particular test that will be carried out by an automated test tool.

Test Specification: A document specifying the test approach for a software feature or combination of features and the inputs, predicted results and execution conditions for the associated tests.

Test Suite: A collection of tests used to validate the behavior of a product. The scope of a Test Suite varies from organization to organization. There may be several Test Suites for a particular product for example. In most cases however a Test Suite is a high level concept, grouping together hundreds or thousands of tests related by what they are intended to test.

Test Tools: Computer programs used in the testing of a system, a component of the system, or its documentation.

Thread Testing: A variation of top-down testing where the progressive integration of components follows the implementation of subsets of the requirements, as opposed to the integration of components by successively lower levels.

Top Down Testing: An approach to integration testing where the component at the top of the component hierarchy is tested first, with lower level components being simulated by stubs. Tested components are then used to test lower level components. The process is repeated until the lowest level components have been tested.

Total Quality Management: A company commitment to develop a process that achieves high quality product and customer satisfaction.

Traceability Matrix: A document showing the relationship between Test Requirements and Test Cases.

U

Usability Testing: Testing the ease with which users can learn and use a product.

Use Case: The specification of tests that are conducted from the end-user perspective. Use cases tend to focus on operating software as an end-user would conduct their day-to-day activities.

User Acceptance Testing: A formal product evaluation performed by a customer as a condition of purchase.

Unit Testing: Testing of individual software components.

V

Validation: The process of evaluating software at the end of the software development process to ensure compliance with software requirements. The techniques for validation are testing, inspection and reviewing.

Verification: The process of determining whether or not the products of a given phase of the software development cycle meets the implementation steps and can be traced to the incoming objectives established during the previous phase. The techniques for verification are testing, inspection and reviewing.

Volume Testing: Testing which confirms that any values that may become large over time (such as accumulated counts, logs, and data files), can be accommodated by the program and will not cause the program to stop working or degrade its operation in any manner.

W

Walkthrough: A review of requirements, designs or code characterized by the author of the material under review guiding the progression of the review.

White Box Testing: Testing based on an analysis of internal workings and structure of a piece of software. Includes techniques such as Branch Testing and Path Testing. Also known as Structural Testing and Glass Box Testing. Contrast with Black Box Testing.

Workflow Testing: Scripted end-to-end testing which duplicates specific workflows which are expected to be utilized by the end-user.