Testing Terms

**Error**: Any Human Mistake in Coding part and developers finds Errors.

**Bug**: Mismatch between actual and expected result before shippin g the product to client and found by Testers.

**Defect**: Mismatch between actual and expected result after shipping the product to client and found by Client or end users.

**Fault:** Fault is a condition that causes a system to fail in performing its required functionality. Its is static characteristic of a software. In a sy-stem

**Failure: Failure is a** characteristic of program behavior and it occurs only when a fault is executed.

**When developer commits an Error it leads to a fault and execution of a fault results a failure.**

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

**Ad hoc testing**: is performed without any plan and FRS or documents.

**Beta testing:** testing that is done at Client site.

**Aplha testing:** testing that is done at development site.

**Black box testing:** check the functionality or external structure of app.

**White Box testing:** testing on internal structure like loop,statement code part.

Concurrency testing: multiuser testing – performed to indentified the defects in an application when multiple users login to the application.

**Dynamic testing:** testing is done with code execution.

**End-to-end testing:** technique used to test the functionality of an application from start to finish

**Functional testing**: check the functionality of an application.

**Glass Box testing:** white box

**Gorilla testing:** testing a particular module with various range of valid and invalid inputs.

**Gray Box testing:** combination of black and white, in which tester has limited knowledge of coding part or internal structure.

**Integration testing**: Individuals modules are combined together and tested as a group.

**Installation testing: is to check that s/w application is successfully installed or not.**

**Monkey testing:**test with random inputs either valid or invalid across each and every module.

**Negative testing:** testing aimed at showing s/w doen’t work.

Performance testing:

**Positive testing:** aimed at showing s/w doen’t work.

**Recovery testing:** how quickly the system can recover after gone through the system crash.

**Retesting:** using the same test cases , to verify whether the defects have been fixed or not.

**Regression testing:** performed the testing on modify build to check the changes doesn’t make adverse effect of other working module.

**Sanity testing:** brief test of major functional of any app to determine if its basically operational.

**Security testing:** to check whether the application is secure or not.if any one hack the application without any authorization.

**Smoke testing:** surface level testing to verify the build provided by development to QA is ready to accept for further testing.

**Exploratory testing:** explore the whole app, what a s/w does and what doesn’t what works and what works not. Testers are involved in minimum planning and max. execution.

**Usability testing:** testers test the easy use of application for end users.

**Static testing:** without code execution, Verify the documents, verification , Inspection .

**Early testing:** conducting testing as soon as possible in development life cycle to find defects at early stages of STLC.

**Load testing:** measures s/w ability to handle varied load, large no. of simultaneous users

**2. Introduction to Software Testing**

Testing is a process of executing a program with the intent of finding an error. Software testing is about testing a feature with varying test data to get a result and then comparing the actual result with expected result.

Software testing is often used in association with terms verification and validation:-

|  |
| --- |
| Validation: Are we doing the right job?  Verification: Are we doing the job right? |

**2.1 Basics Of Testing**

**a) Error**

An error is a human action that produces incorrect results**.** An error is usually a programmer action or omission that results in a fault. In software development one error may cause one or more defects in requirements, designs, programs, or tests.

**b) Bug**

Presence of an error at the time of execution of the software is known as Bug or defect.

Software bugs have following symptoms:

* The software is not according to the product specification
* The software is doing something that is not mentioned in the product specification
* The software is doing something that is not to be done according to the product specification
* The software is hard to use and understand

**c) Failure**

It is deviation of the software from its expected delivery or service. It is an event. Failure is the unacceptable departure of a program operation from program requirements.

**d) Fault**

A fault is a software defect that causes a failure. Fault is a state of software caused by an error. It can be defined as accidental condition that causes the failure of a component in the implementation model to perform its required behavior.

**e) Verification and Validation (V&V)**

**Verification** is the process of confirming that software meets its specification. It involves reviews and meetings to evaluate documents, plans, code, requirements and specification.

It is examination of process and checks if we are building the product right?

**Validation** is the process of confirming that software meets the user’s requirements. Validation typically involves actual testing and takes place after verifications are completed.

It is execution of the process and checks if we are building the right product?

**f) Quality Assurance (QA)**

Software quality assurance (SQA) is defined as a planned and systematic approach to the evaluation of the quality and adherence to software product standards, process and procedures.

**g) Quality Control (QC)**

Quality control describes the directed use of testing to measure the achievement of a specified standard. QC makes sure the results of what you’ve done are what you expected. QC focuses on testing for quality and hence detecting defects. It deals with product but QA deals with process.

**Software Testing has two methods:**

* 1. Manual Testing
  2. Automation Testing

**2.1 Manual Testing** Manual testing is the process of manually testing software for defects. It requires a tester to play the role of an end user, and use most of all features of the application to ensure correct behavior.

Tools used for manual testing:-

1. Test Case Management System (TCMS Version 1.5.1).
2. Bugzilla (Version 2.18rc1)

**2.2 Automation Testing** The use of software to control the execution of tests, the comparison of actual outcomes to predicated outcomes, the setting up of test preconditions and other test control and test reporting functions.

Tool used for automation testing:-

* Selenium

**2.3 Testing Principles**

* All tests should be traceable to customer requirements.
* Tests should be planned long before testing begins.
* Testing should begin “in small” and progress toward testing “in large”.
* To be most effective, testing should be conducted by an independent third party.

**2.4 Objectives of testing:**

* To ensure that application works as specified in requirements document.
* To provide a bug-free application (though we know no application can be 100% bug free, but still user acceptance is fine. .
* To establish confidence in software.
* To evaluate properties of software

**2.5 Test Levels**

There are four test levels in each development cycle as mentioned below.

1. Component Testing
2. Integration testing
3. System testing
4. Acceptance Testing

**a) Component testing**

The testing of those test objects which are separately testable as a single unit without integrating with other components (e.g. modules, programs, objects, classes, etc.).

**b) Integration Testing**

During Integration testing, tests are executed to verify the interaction between different components or with different parts of the system and functionality of different components.

**c) System Testing**

System testing is defined as the testing of functionality of whole system/product as specified in functional requirements of the product. The whole components of the system are tested and their interaction with each other, behavior with other systems is verified.

**d) Acceptance Testing**

As the name suggests, this testing is used to accept the product depending upon acceptance criteria defined. During testing phase, it is performed to accept the build for further testing by QA Team. Alpha and beta (or field) testing are also forms of Acceptance testing.

**2.6 Types of testing**

**2.6.1 White Box Testing Types**

1. Unit Testing
2. Structural Testing

**2.6.1 White Box testing**

1. **Unit Testing:** Unit testing is a procedure used to validate that individual units of source code are working properly. For Example: - A function, method, Loop or statement in program is working fine. Tests the smallest testable part of an application (i.e. a unit).Usually done by programmers.
2. **Structural Testing:** During Development of a project, Code is written based on the design and follows some structure/tiers/layers in it. Structural testing is based on the architecture of the system, such as a calling hierarchy. Structural testing is used to verify the functionality of a particular structure of the code and to ensure that every line of code has been covered during testing. During Structural testing, tests are executed to exercise the code in different layers and measure the Code coverage of the structure.

**2.6.2 Black Box Testing Types**

1. Functional Testing
2. Smoke Testing
3. Sanity Testing
4. Integration Testing
5. Usability Testing
6. Compatibility Testing
7. Configuration Testing
8. Concurrency Testing
9. Security Testing

**2.6.2 Black Box Testing Types**

**1. Functional Testing:** Functional testing covers how well the system executes the functions it is supposed to execute—including user commands, data manipulation, searches and business processes, user screens, and integrations. Functional testing is more important because it verifies that your system is ready for release.

**2. Smoke Testing:** Smoke testing is the surface-level testing to verify that build provided by development to QA is ready to accept for further testing.

**3. Sanity Testing:** Sanity testing is the surface level testing where QA engineer verifies that all the menus, functions, commands available in the product are working fine. Sanity testing is performed after the build has passed the Smoke test and has been accepted by QA team for further testing.

**4. Regression Testing:** Regression testing is the re-testing of features to ensure that features working earlier are still working fine as desired. It is performed when any new build comes to QA.

**5. Integration Testing:** Integration Testing: Tests interactions between two or more units or components. It is the phase of software testing in which individual software modules are combined and tested as a group.

6. **Usability Testing:**"Usability Testing measures the usability of the system". Usability Testing means a way to measure how end users find the system easy, moderate or hard to interact with and use the system keeping its purpose in mind. We all know that Software is written to be used.

**7. Compatibility Testing:** Software compatibility testing means checking that your software interacts with and shares information correctly with other software. For example: different browser versions, different OS, different data bases etc.

**8. Configuration Testing:** Configuration Testing is the process of checking the operation of the software you're testing with various types of hardware. It is the system testing of different variations of an integrated, black box application against its configurability requirements.

9. **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.

**2.6.2.1 BLACK BOX TESTING TECHNIQUES**

1. Equivalence Partitioning
2. Boundary Value Analysis
3. Cause-Effect Graphing Techniques
4. Decision Table Testing
5. Error Guessing
6. **Equivalence Partitioning**

Equivalence partitioning is a software testing technique to minimize number of permutation and combination of input data. In equivalence partitioning, data is selected in such a way that it gives as many different outputs as possible with the minimal set of data. Now, data from these classes can be representative of all the input values that your software expects. For equivalence classes, it can be assumed that software will behave in exactly same way for any data value from the same partition.

1. **Boundary Value Analysis**

Boundary value analysis is the technique of making sure that behaviour of system is predictable for the input and output boundary conditions. Reason why boundary conditions are very important for testing is because defects could be introduced at the boundaries very easily.

1. **Cause-Effect Graphing Techniques**

Cause-effect graphing is a test case design approach that offers a concise depiction of logical conditions and associated actions. The approach has four stages:

1. **Decision Table Testing**

Decision Table Testing involves testing the behavior of a system when any component involves logical conditions in it and system needs to be verified for different combinations of the conditions.

1. **Error Guessing**

Error Guessing involves making an itemized list of the errors expected to occur in a particular area of the system and then designing a set of test cases to check for these expected errors. Error Guessing is more testing art than testing science but can be very effective given a tester familiar with the history of the system. Error guessing involves asking "what if?” Example: The statement might be made "the user must input a valid zip code". What if the user enters no zip code? What if the zip doesn't match the state? What if the zip code doesn't exist? Etc...

**2.6.3 Non-Functional Testing Types**

1. Security Testing
2. Performance Testing
3. Load Testing
4. Stress Testing
5. Volume Testing
6. Recovery Testing

**2.6.3 NON-FUNCTIONAL TESTING TYPES**

Non-Functional Testing Types describes the tests used to measure the software characteristics such as Response times, Page Load times, Peak Load limit, Threshold limit for optimum performance of the Software product on varying scale etc.

**1.) Security Testing:** Security Testing: Process to determine that an IS (Information System) protects data and maintains functionality as intended.

The basic security concepts that need to be covered by security testing are:

* 1. Confidentiality
  2. Integrity
  3. Authentication and Authorisation

**2.) Performance Testing:** Performance Testing is done to determine the benchmark data of various critical business processes and transactions while the system is under low load.

**3.) Load Testing:** Load testing is used to test that if the application works fine with the loads that result from large number of simultaneous users, transactions and to determine whether it can handle peak usage periods. Load testing measures the system's ability to handle varied workloads.

**4.) Stress Testing:** When the load placed on the system is raised beyond normal usage patterns, in order to test the system's response at unusually high or peak loads, it is known as stress testing.

**5.) Volume Testing:** Volume testing, subjects the software to larger and larger amounts of data to determine its point of failure. Volume testing refers to testing a software application for a certain data volume.

**6.) Recovery Testing In s/w testing**, recovery testing is the activity of testing how well the software is able to recover from crashes, hardware failures and other similar problems.