Unit 7 Software Testing

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. It is the process of executing software with the intent of finding software bugs and to ensure that it satisfies the specified requirements. Test is conducted for executing a system in order to identify any gaps, error or missing requirements in contrary to the actual desire or requirements.

Software is tested for **validating and verifying** software to ensure the business and technical requirements that guided its design and development and works as expected. It also provides an objective, independent view of software to allow the business to appreciate and understand the risk of software implementation.

Software Testing Objectives

The major objectives of software testing are as follows:

- Finding defects which may get created by the programmer while developing the software.
- Gaining confidence in and providing information about the level of quality.
- To make sure that the end result meets the business and user requirement.
- To gain the confidence of the customers by providing them a quality product
- To ensure effective performance of software product.

Software Verification and Validation

Software verification and validation activities check the software against its specification. Every project must verify and validate the software it produces.

Verification

Verification refers to the set of activities that ensure that software correctly implements specific functions. It ensures that the product has been built according to the requirement and design specification. It makes sure that the product is designed to deliver all functionality to the

customer. Verification is done at the development process and includes reviews and meeting, inspection etc. to evaluate documents, plan code, requirements and specifications. Suppose you are constructing a table then verification is about checking all the parts of table, whether all legs are of correct size or not. It one leg is not of the correct size it will imbalance the end product. Verification is the demonstration of consistency, completeness and correctness of the software at each stage and between each stage of the development lifecycle. It is expressed as: Are we building the product right?

Advantages

- Verification helps in lowering down the count of the defect in the later stages of development.
- Verifying the product at the starting phase of development will help in understanding the product in better way.
- It reduces the chance of failure in software product.
- It helps in building the product as per the customer specification and needs.

Validation

Validation refers to the set of activities that ensure that software that has been built is traceable to the customer requirements. It is intended to show that the software does what the user requires. Validation concerns with determining if the system complies with the requirements and performs functions for which it is intended and meets the organizations goals and user needs. It is done at the end of the development process and takes place after verifications are completed. It answers the question like: Are we building the right product?

Advantages

- It some defects are missed in verification, then during validation defect can be caught as failures.
- If specification is misunderstood and development had happened then during validation process while executing that functionality the difference between the actual result and expected result can be understood.

- Validation helps in building the right product as per the customer's requirement and helps in satisfying their needs.
- Validation is basically done by the testers during the testing.

Software testing process

Software testing process starts with preparation of test cases. Test cases are specification of the inputs to the test and the expected output from the system together with what is being tested. From the design of test cases we get test case list. After that we prepare the test data. Test data are the inputs which have been devised to test the system. Test data can be generated automatically or manually. From this step we get test data set. Then we run the program with test data for all test cases and generate the result. Finally we compare the test result to test cases and generate the test report.

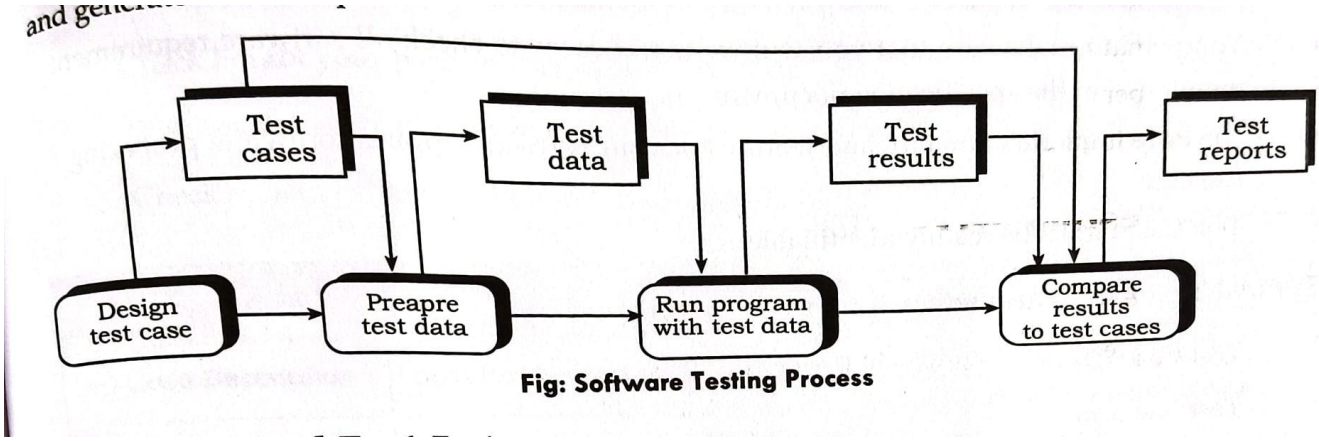


Fig: Software Testing Process

Test Case and Test Data

A **Test Case** is a set of actions executed to verify a particular feature or functionality of your software application. It is a document, which has a set of test data, preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement. Test Case acts as the starting point for the test execution, and after applying a set of input values; the application has a definitive outcome and leaves the system at some end point or also known as execution post condition.

The test case includes specific variables or conditions, using which a testing engineer can compare expected and actual results to determine whether a software product is functioning as per the requirements of the customer. A Test Scenario is defined as any functionality that can be tested. It is a collective set of test cases which helps the testing team to determine the positive and negative characteristics of the project. Test Scenario gives a high-level idea of what we need to test.

Test Data in Software Testing is the input given to a software program during test execution. It represents data that affects or affected by software execution while testing. Test data is used for both positive testing to verify that functions produce expected results for given inputs and for negative testing to test software ability to handle unusual, exceptional or unexpected inputs. Poorly designed testing data may not test all possible test scenarios which will hamper the quality of the software.

Level of testing:

There are different levels of testing which can be described below:-

Unit testing

Unit testing is a procedure used to validate that a particular module of source code is working properly. The procedure is to write test cases for all function and methods so that whenever a change cases a regression, it can be quickly identified and fixed.

Integrated testing

It is a systematic technique for validating the construction of the overall software structure while at the same conducting test to uncover errors associated with interfacing.

System testing

System testing is a testing conducted or a complete integrated system to evaluate the systems complains with its specified requirements. System testing falls within the scope black box testing and as such should require no knowledge of inner design of the code.

Acceptance Testing

When the software is ready to hand over to the customer it has to go through last phase of testing where it is tested for user-interaction and response. This is important because even if the software matches all user requirements and if user does not like the way it appears or works, it may be rejected.

- **Alpha testing** - The team of developer themselves perform alpha testing by using the system as if it is being used in work environment. They try to find out how user would react to some action in software and how the system should respond to inputs.
- **Beta testing** - After the software is tested internally, it is handed over to the users to use it under their production environment only for testing purpose. This is not as yet the delivered product. Developers expect that users at this stage will bring minute problems, which were skipped to attend.

Testing Methods

Testing can be performed without having any knowledge of the interior workings of the application as well as having knowledge of implementation detail of software system. There are two testing methods which are black box testing and white box testing,

Black Box Testing

Black box testing is the technique of software testing in which without having knowledge of the interior working, application is tested to check the relationship between its input and output. In this testing, knowing the specified function that a product has been design to perform, test can be conducted that demonstrate each function is fully operational. The tester is oblivious to the system architecture and does not have access to the source code. Typically, when performing a black box test, a testing will interact with the system's user interface by providing inputs and examining output without knowing how and where the inputs are worked upon. In black box testing, system is like black box whose behavior can only be determined by studying its input and related outputs.

Black box testing is also called functional or behavioral testing focuses on functional requirement of software i.e. tester is only concerned with functionality not the implementation of software. This testing is especially applicable to perform system testing. Black box testing attempts to find errors in the following categories:

- Incorrect or missing functions.
- Interface error.
- Error in data structure.
- Behavior or performance
- Initialization or termination error.

White box testing

White box testing is an approach to testing where the tests are derived from knowledge of software's structure and implementation. It is the detailed investigation of internal logic and structure of the software system. In this testing, knowing the internal working of the product, test case can be conducted to ensure that, internal operations are performed according to specifications and all internal components have been adequately exercised. In order to perform white box testing in an application, the tester needs to process knowledge of internal workings of the code, the tester needs to have look inside the sources code and find out which unit chunk of the code is behaving in appropriately. Using white box testing, the software can derive test cases that:

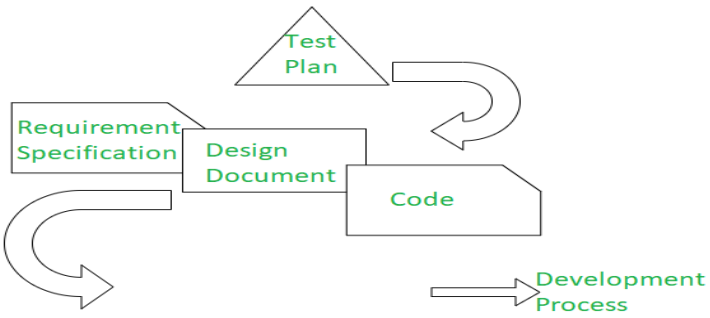
- Guarantee that all independent paths with a module have been exercised at least once.
- Exercise all logical decisions on their true and false value.
- Execute all loops at their boundaries.

This method is usually applied to relatively small program unit, subroutine and operations such as unit testing, component testing and integration testing. It is also known as clear box testing, glass box testing or structural testing.

Development Testing

It is a method of applying testing practices consistently throughout the software development life cycle process. This testing ensures the detection of bugs or errors at the right time which further ensures delay of any kind of risk in terms of time and cost. Development Testing aims to establish a framework to verify whether the requirements of a given project are met in accordance with the rules of the mission to be accomplished. This testing is performed by the software developers or other engineers during the construction phase of the software development lifecycle (SDLC). Development Testing is a continuous or a running process in the development of a product in the entire software development life cycle. This testing is done only once as compared to other testing which can be performed many times. To meet the deadline date, development testing is performed during the development phase of a software product,

In Development Testing, the phases are more tightly integrated so that code that is being written and checked in is automatically tested. In this way, the problems can be more quickly discovered and can be addressed.



When to perform Development Testing?

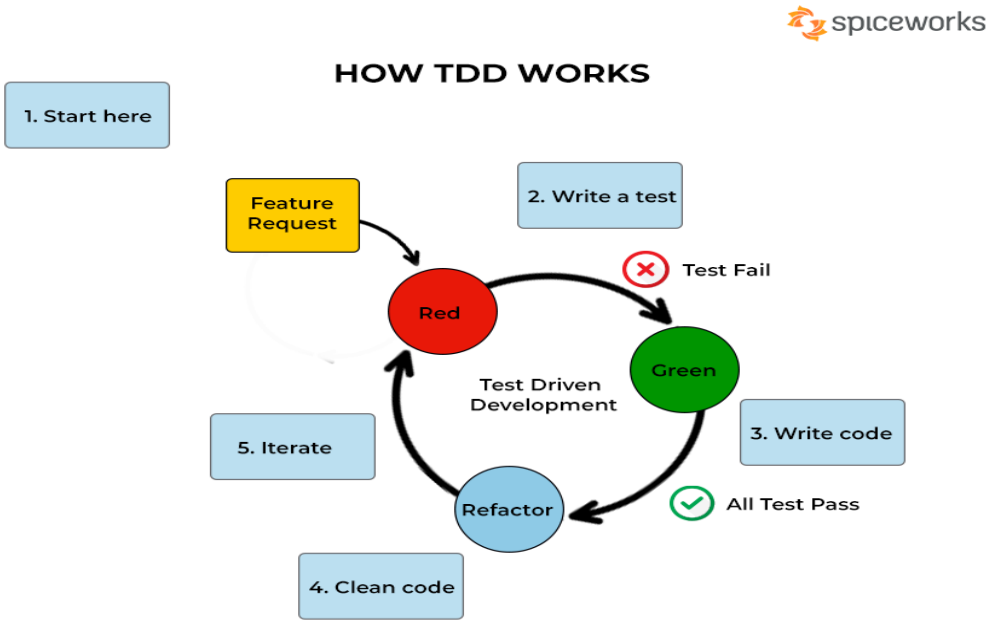
- When writing new code or building a new software product.
- When development cost is low, the client should perform development testing so that the client doesn't have to face the debugging and another testing cost.

Benefits of Development Testing

- Helps in increasing software development life cycle (SDLC) efficiency
- Helps in reducing software errors
- Speed up the delivery process
- It gives high-quality code at any time as the code is being tested continuously
- It requires less time to deploy in the market for new features

Test-Driven Development

Test-driven development (TDD) is defined as an iterative methodology that prioritizes the creation of and checking against test cases at every stage of software development, by converting each component of the application into a test case before it is built and then testing and tracking the component repeatedly.



It is a continuous process that includes refactoring, unit testing, and programming. Prior to writing any actual code, Test-driven development (TDD) emphasizes the creation of unit test cases. It requires developers to build a test first, then just enough production code to satisfy it and the ensuing reworking. The specs are used by developers, who then create tests that demonstrate how the code ought to function.

Testing, coding, and reworking are done in fast succession. The first step in test-driven development is to plan and create tests for each minor aspect of an application's functionality. The test-driven development framework tells programmers to build new code only in the event that an automated test fails. This prevents code duplication. Simply put, test cases are generated and tested for each capability first, and if the test fails, new code is produced to pass the test and provide simple, bug-free code.

Kent Beck created it as a component of extreme programming in the late 1990s. A more comprehensive approach to software design known as Extreme Programming (XP), which is a component of the Agile software development methodology, introduced test-driven development. To find problems in the tested code, testing is done on a small sample of the code.

Before functional testing, the test is written to make sure the application is appropriate for testing. To find problems in the tested code, testing is done on a small sample of the code. The feature is then put into use. The feature is then put into use. Test-driven development makes use of repeated quick development cycles and it typically uses a cycle called "Red-Green-Refactor."