UNIT – V

SOFTWARE TESTING

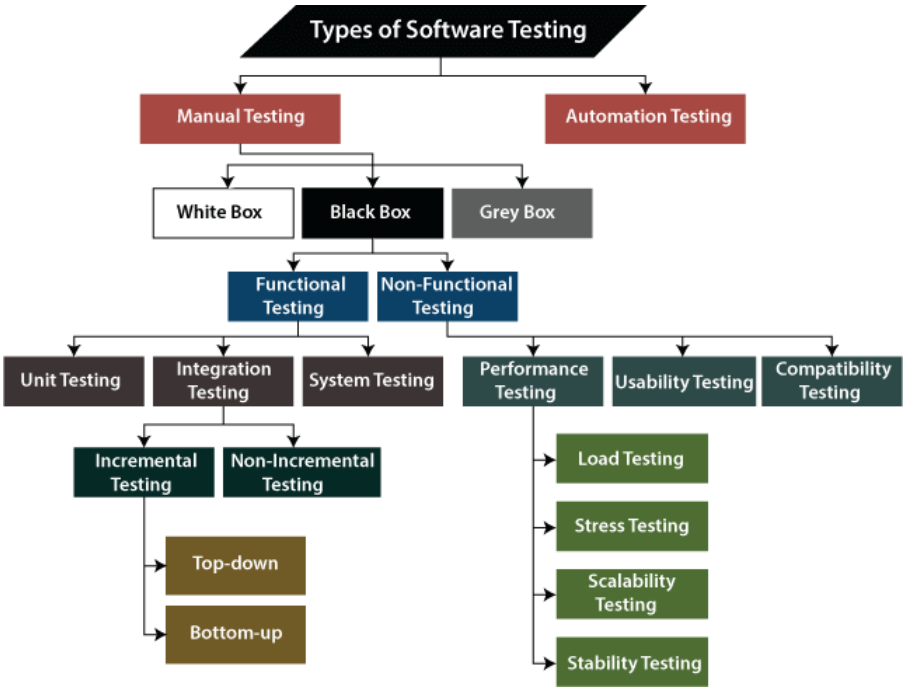
What is Software Testing?

Software Testing is a method to assess the functionality of the software program. The process checks whether the actual software matches the expected requirements and ensures the software is bug-free. The purpose of software testing is to identify the errors, faults, or missing requirements in contrast to actual requirements. It mainly aims at measuring the specification, functionality, and performance of a software program or application.

Software testing can be divided into two steps:

1. **Verification:** It refers to the set of tasks that ensure that the software correctly implements a specific function. It means "Are we building the product right?".
2. **Validation:** It refers to a different set of tasks that ensure that the software that has been built is traceable to customer requirements. It means "Are we building the right product

Types Of Software Testing



Software Testing can be broadly classified into 3 types:

1. **Functional Testing:** Functional testing is a type of software testing that validates the software systems against the functional requirements. It is performed to check whether the application is working as per the software's functional requirements or not. Various types of functional testing are Unit testing, Integration testing, System testing, Smoke testing, and so on.
2. **Non-functional Testing:** Non-functional testing is a type of software testing that checks the application for non-functional requirements like performance, scalability, portability, stress, etc. Various types of non-functional testing are Performance testing, Stress testing, Usability Testing, and so on.
3. **Maintenance Testing:** Maintenance testing is the process of changing, modifying, and updating the software to keep up with the customer's needs. It involves regression testing that verifies that recent changes to the code have not adversely affected other previously working parts of the software.

Manual Testing:

Manual testing includes testing software manually, i.e., without using any automation tool or script. In this type, the tester takes over the role of an end-user and tests the software to identify any unexpected behavior or bug. There are different stages for manual testing such as unit testing, integration testing, system testing, and user acceptance testing. Testers use test plans, test cases, or test scenarios to test software to ensure the completeness of testing. Manual testing also includes
exploratory testing, as testers explore the software to identify errors in it.

Automation Testing:

Automation testing, which is also known as Test Automation, is when the tester writes scripts and uses another software to test the product. This process involves the automation of a manual process. Automation Testing is used to re-run the test scenarios quickly and repeatedly, that were performed manually in manual testing.
Apart from regression testing, automation testing is also used to test the application from a load, performance, and stress point of view. It increases the test coverage, improves accuracy, and saves time and money when compared to manual testing.

Different Types of Software Testing Techniques

Software testing techniques can be majorly classified into two categories:

1. **Black Box Testing:** Black box technique of testing in which the tester doesn't have access to the source code of the software and is conducted at the software interface without any concern with the internal logical structure of the software known as black-box testing.
2. **White-Box Testing:** White box technique of testing in which the tester is aware of the internal workings of the product, has access to its source code, and is conducted by making sure that all internal operations are performed according to the specifications is known as white box testing.
3. **Grey Box Testing:** Grey Box technique is testing in which the testers should have knowledge of implementation, however, they need not be experts.

White Box Testing

The box testing approach of software testing consists of black box testing and white box testing. We are discussing here white box testing which also known as glass box is **testing, structural testing, clear box testing, open box testing and transparent box testing**. It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. In this type of testing programming skills are required to design test cases. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.

Developers do white box testing. In this, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.

The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Generic steps of white box testing

- Design all test scenarios, test cases and prioritize them according to high priority number.
- This step involves the study of code at runtime to examine the resource utilization, not accessed areas of the code, time taken by various methods and operations and so on.

- In this step testing of internal subroutines takes place. Internal subroutines such as non-public methods, interfaces are able to handle all types of data appropriately or not.
- This step focuses on testing of control statements like loops and conditional statements to check the efficiency and accuracy for different data inputs.
- In the last step white box testing includes security testing to check all possible security loopholes by looking at how the code handles security.

Reasons for white box testing

- It identifies internal security holes.
- To check the way of input inside the code.
- Check the functionality of conditional loops.
- To test function, object, and statement at an individual level.

Advantages of White box testing

- White box testing optimizes code so hidden errors can be identified.
- Test cases of white box testing can be easily automated.
- This testing is more thorough than other testing approaches as it covers all code paths.
- It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing

- White box testing is too much time consuming when it comes to large-scale programming applications.
- White box testing is much expensive and complex.
- It can lead to production error because it is not detailed by the developers.
- White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

Black box testing

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not. If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction

.

Generic steps of black box testing

- The black box test is based on the specification of requirements, so it is examined in the beginning.
- In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- The fourth phase includes the execution of all test cases.
- In the fifth step, the tester compares the expected output against the actual output.
- In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

Difference between white-box testing and black-box testing

S No.	Black Box Testing	White Box Testing
1	Internal workings of an application are not required.	Knowledge of the internal workings is a must.

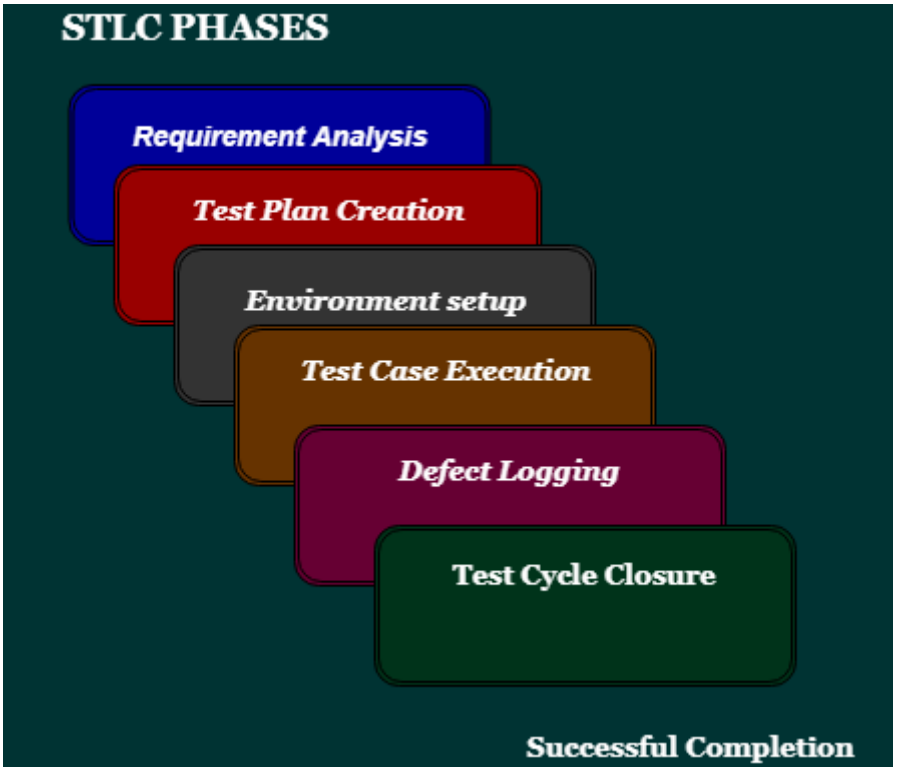
S No.	Black Box Testing	White Box Testing
2	Also known as closed box/data-driven testing.	Also known as clear box/structural testing.
3	End users, testers, and developers.	Normally done by testers and developers.
4	This can only be done by a trial and error method.	Data domains and internal boundaries can be better tested.

Stages of Testing Development

The procedure of software testing is also known as STLC (Software Testing Life Cycle) which includes phases of the testing process.The testing process is executed in a well-planned and systematic manner. All activities are done to improve the quality of the software product

Software testing life cycle contains the following steps:

1. Requirement Analysis
2. Test Plan Creation
3. Environment setup
4. Test case Execution
5. Defect Logging
6. Test Cycle Closure



Requirement Analysis:
The first step of the manual testing procedure is requirement analysis. In this phase, tester analyses requirement document of SDLC (Software Development Life Cycle) to examine requirements stated by the client. After examining the requirements, the tester makes a test plan to check whether the software is meeting the requirements or not.

Test Plan Creation:
Test plan creation is the crucial phase of STLC where all the testing strategies are defined. Tester determines the estimated effort and cost of the entire project. This phase takes place after the successful completion of the **Requirement Analysis Phase**. Testing strategy and effort estimation documents provided by this phase. Test case execution can be started after the successful completion of Test Plan Creation

Environment setup:
Setup of the test environment is an independent activity and can be started along with **Test Case Development**. This is an essential part of the manual testing procedure as without environment testing is not possible. Environment setup requires a group of essential software and hardware to create a test environment. The testing team is not involved in setting up the testing environment, its senior developers who create it.

Test case Execution:
Test case Execution takes place after the successful completion of test planning. In this phase, the testing team starts case development and execution activity. The testing team writes down the detailed test cases, also prepares the test data if required. The prepared test cases are reviewed by peer members of the team or Quality Assurance leader. RTM (Requirement Traceability Matrix) is also prepared in this phase. Requirement Traceability Matrix is industry level format, used for tracking requirements. Each test case is mapped with the requirement specification. Backward & forward traceability can be done via RTM.

Defect Logging:
Testers and developers evaluate the completion criteria of the software based on test coverage, quality, time consumption, cost, and critical business objectives. This phase determines the characteristics and drawbacks of the software. Test cases and bug reports are analyzed in depth to detect the type of defect and its severity.
Defect logging analysis mainly works to find out defect distribution depending upon severity and types. If any defect is detected, then the software is returned to the development team to fix the defect, then the software is re-tested on all aspects of the testing.
Test Cycle Closure:
The test cycle closure report includes all the documentation related to software design, development, testing results, and defect reports. This phase evaluates the strategy of development, testing procedure, possible defects in order to use these practices in the future if there is a software with the same specification

Unit Testing

1. Unit testing involves the testing of each unit or an individual component of the software application. It is the first level of functional testing. The aim behind unit testing is to validate unit components with its performance.
2. A unit is a single testable part of a software system and tested during the development phase of the application software.
3. The purpose of unit testing is to test the correctness of isolated code. A unit component is an individual function or code of the application. White box testing approach used for unit testing and usually done by the developers.
4. Whenever the application is ready and given to the Test engineer, he/she will start checking every component of the module or module of the application independently or one by one, and this process is known as Unit testing or components testing**.**

Unit Testing Techniques:

Unit testing uses all white box testing techniques as it uses the code of software application:

- Data flow Testing
- Control Flow Testing
- Branch Coverage Testing
- Statement Coverage Testing
- Decision Coverage Testing

Advantages

- Unit testing uses module approach due to that any part can be tested without waiting for completion of another parts testing.
- The developing team focuses on the provided functionality of the unit and how functionality should look in unit test suits to understand the unit API.
- Unit testing allows the developer to refactor code after a number of days and ensure the module still working without any defect.

Disadvantages

- It cannot identify integration or broad level error as it works on units of the code.
- In the unit testing, evaluation of all execution paths is not possible, so unit testing is not able to catch each and every error in a program.
- It is best suitable for conjunction with other testing activities.

Component Testing

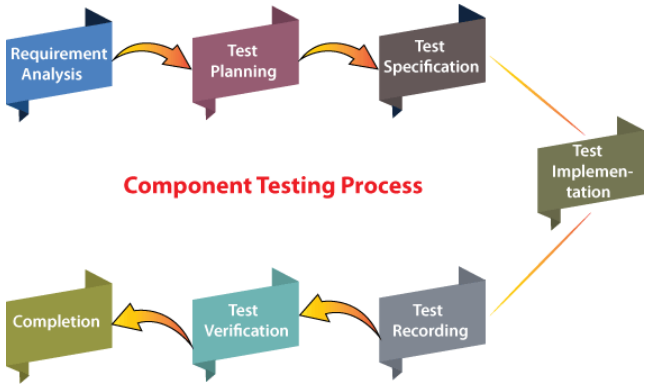
Another type of software testing is **Component Testing**. It is used to test all the components separately as well as the usability testing; interactive valuation is also done for each specific component. It is further known as **Module Testing or Program Testing and Unit Testing.**
In order to implement the component testing, all the components or modules require to be in the individual state and manageable state. And all the related components of the software should be user-understandable.

This type of testing provides a way to finding defects, which happen in all the modules. And also helps in certifying the working of each component of the software.

The Goals of Component Testing

- Reducing Risk
- Identifying defects/bugs in the component
- Validate whether the functional and non-functional performance of the component is as expected
- Developing confidence in the component's quality
- Stopping defects from escaping to higher test levels

Component Testing Process



Step1: Requirement Analysis
The first step of component testing is requirement analysis, where the user requirement associated with each component is detected.

Step2: Test Planning
Once the requirement analysis phase is done, we will mov to the next step of component testing process, which is test planning. In this phase, the test is designed to evaluate the requirement given by the users/clients.

Step3: Test Specification
Once the test planning phase is done, we will move to the next phase, known as test specification. Here, we will identify those test cases that needs to be executed and missed

Step4: Test Implementation
The forth step in the component testing process is Test implementation. When the test cases are identified as per the user requirements or the specification, then only we can implement the test cases.

Step5: Test Recording
When all the above steps have been completed successfully, we will go to the next step that is, Test Recording. In this step of the component testing process, we have the records of those defects/bugs discovered during the implementation of component testing.

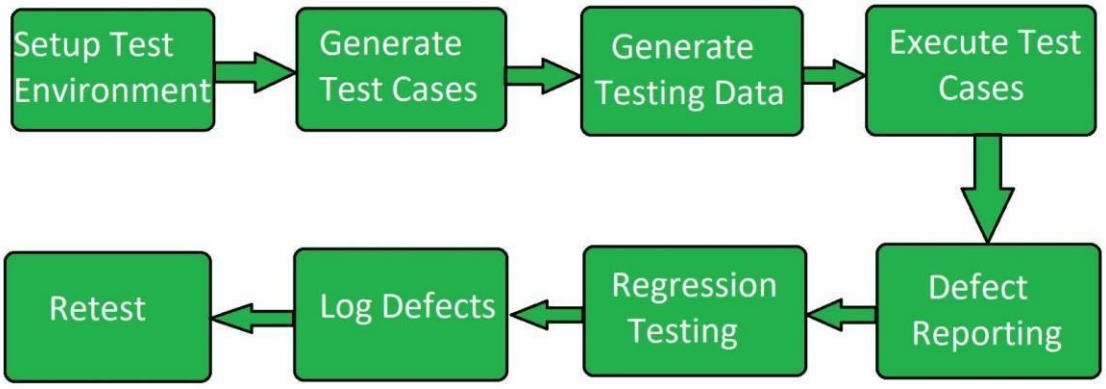
Step6: Test Verification
Once the bugs or defects have been recorded successfully, we will proceed to the test verification phase. It is the process of verifying whether the product fulfils the specification or not.

Step7: Completion
After completed all the above steps successfully, we will come to the last step of the component testing process. In this particular step, the results will be evaluated in order to deliver a good quality product.

System Testing

- System Testing is a type of software testing that is performed on a complete integrated system to evaluate the compliance of the system with the corresponding requirements.
- In system testing, integration testing passed components are taken as input. The goal of integration testing is to detect any irregularity between the units that are integrated together.
- System testing detects defects within both the integrated units and the whole system.
- The result of system testing is the observed behavior of a component or a system when it is tested.
- System Testing is carried out on the whole system in the context of either system requirement specifications or functional requirement specifications or in the context of both.
- System testing tests the design and behavior of the system and also the expectations of the customer. It is performed to test the system beyond the bounds mentioned in the software requirements specification (SRS).
- System Testing is basically performed by a testing team that is independent of the development team that helps to test the quality of the system impartial. It has both functional and non-functional testing.
- System Testing is a black-box testing. System Testing is performed after the integration testing and before the acceptance testing.

System Testing Process:



- **Test Environment Setup:** Create testing environment for the better quality testing.
- **Create Test Case:** Generate test case for the testing process.
- **Create Test Data:** Generate the data that is to be tested.
- **Execute Test Case:** After the generation of the test case and the test data, test cases are executed.
- **Defect Reporting:** Defects in the system are detected.
- **Regression Testing:** It is carried out to test the side effects of the testing process.
- **Log Defects:** Defects are fixed in this step.
- **Retest:** If the test is not successful then again test is performed.

Advantages of System Testing:

- Verifies the overall functionality of the system.
- Detects and identifies system-level problems early in the development cycle.
- Helps to validate the requirements and ensure the system meets the user needs.
- Improves system reliability and quality.
- Facilitates collaboration and communication between development and testing teams.
- Enhances the overall performance of the system.
- Increases user confidence and reduces risks.
- Facilitates early detection and resolution of bugs and defects.
- Supports the identification of system-level dependencies and inter-module interactions.
- Improves the system’s maintainability and scalability.

Disadvantages of System Testing:

- Can be time-consuming and expensive.
- Requires adequate resources and infrastructure.
- Can be complex and challenging, especially for large and complex systems.
- Dependent on the quality of requirements and design documents.
- Limited visibility into the internal workings of the system.
- Can be impacted by external factors like hardware and network configurations.
- Requires proper planning, coordination, and execution.
- Can be impacted by changes made during development.
- Requires specialized skills and expertise.
- May require multiple test cycles to achieve desired results.

Test-driven development testing

Test-driven development, or TDD for short, is a software development process. As the name implies, involves utilizing tests to guide application development, resulting in simple, iterative implementation with good test coverage right from the start.

- Test-Driven Designing and building tests for each single function of an application is the first step in development. Only when an automated test fails, the TDD framework tell developers to write new code. It prevents code duplication.
- It is based on the simple principle of developing and correcting failed tests before writing new code (before development). We write a tiny piece of code at a time to pass tests, therefore it helps the developer to minimize duplicate code. Tests are nothing more than requirements that must be tested in order to be met.
- It is a method of creating and executing automated tests prior to the application's real development. As a result, Test First Development is also known as TDD.

The following sequence of steps is generally followed:

1. Add a test – Write a test case that describe the function completely. In order to make the test cases the developer must understand the features and requirements using user stories and use cases.
2. Run all the test cases and make sure that the new test case fails.
3. Write the code that passes the test case

•