Introduction To Software Testing

Testing is a process of verifying and validating if the developed computer software is correct, complete and has the quality which is acceptable. That means, it is checking if a software system meets specifications and that it fulfills its intended purpose. Hence, it's a universally accepted (and most debated topic) that testing computer software can never be completely established.

Software testing is nothing but an art of investigating software to ensure that its quality under test is in line with the requirement of the client. Software testing is carried out in a systematic manner with the intent of finding defects in a system. It is required for evaluating the system.

What is Software Testing?

What does this defect mean in Software Testing?

The main purpose of Software Testing is to identify the **defects**.

Defect:-A flow in a component or system that can cause the component or system to fail to perform its required function.

Fault:- Fault is similar to a **defect.**

Failure:-Deviation of the component or system from its expected delivery, service or result.

Error:-A human action that produces an incorrect result.

Bug:- Bug is similar to that of an defect.

II.Isolate the defects.

Isolating means seperation or dividing the defects.

These isolated defects are collected in the **Defect Profile**

What is **Defect Profile Document?**

a.Defect Profile is a document with many columns in Software Testing.
b.This is a template provided by the company.

III.Subjected for rectification

The Defect Profile is **subjected for rectification** that means it is send to developer

IV.Defects are rectified

After getting from the developer make sure all the defects are rectified, before defining it as a **Quality** product.

What is Quality in Testing?

Quality is defined as justification of user requirements or satisfaction of user requirements.

When all the 4 steps are completed we can say that Software Testing is completed.

SOFTWARE TESTING MAIN DEFINITION----

This is the process in which the defects are identified, isolated , and subjected for rectification and finally make sure that all the defects are rectified , in order to ensure that the product is a Quality product.

Why testing code is important ?

(A) Software testing is really required to point out the **defects** and errors that were made during the **development phases**. Example: Programmers may make a mistake during the implementation of the software. There could be many reasons for this like lack of experience of the programmer, lack of knowledge of the programming language, insufficient experience in the domain, incorrect implementation of the algorithm due to complex logic or simply human error.

(B) It’s essential since it makes sure that the customer finds the organization reliable and their satisfaction in the application is maintained. If the customer does not find the testing organization reliable or is not satisfied with the quality of the deliverable, then they may switch to a competitor organization.

Sometimes contracts may also include monetary penalties with respect to the timeline and quality of the product. In such cases, if proper software testing may also prevent monetary losses.

(C) It is very important to ensure the Quality of the product. Quality product delivered to the customers helps in gaining their confidence. (Know more about **Software Quality**) As explained in the previous point, delivering good quality product on time builds the customers confidence in the team and the organization.

(D) Testing is necessary in order to provide the facilities to the customers like the delivery of high quality product or software application which requires lower maintenance cost and hence results into more accurate, consistent and reliable results.High quality product typically has fewer defects and requires lesser maintenance effort, which in turn means reduced costs.

(E) Testing is required for an effective performance of software application or product.

It’s important to ensure that the application should not result into any **failures** because it can be very expensive in the future or in the later stages of the development.

- Proper testing ensures that bugs and issues are detected early in the life cycle of the product or application.
- If defects related to requirements or design are detected late in the life cyle, it can be very expensive to fix them since this might require redesign, re-implementation and retesting of the application.

(F) It’s required to stay in the business. Users are not inclined to use software that has bugs. They may not adopt a software if they are not happy with the stability of the application.

In case of a product organization or startup which has only one product, poor quality of software may result in lack of adoption of the product and this may result in losses which the business may not recover from.

Verification and Validation

Verification is the process of checking that a software achieves its goal without any bugs. It is the process to ensure whether the product that is developed is right or not. It verifies whether the developed product fulfills the requirements that we have. Verification is static testing. Verification means **Are we building the product right?**

Validation is the process of checking whether the software product is up to the mark or in other words product has high level requirements. It is the process of checking the validation of product i.e. it checks what we are developing is the right product. it is validation of actual and expected product. Validation is the dynamic testing.

Validation means **Are we building the right product?**

The difference between Verification and Validation is as follow:

VERIFICATION	VALIDATION
It includes checking documents, design, codes and programs.	It includes testing and validating the actual product.
Verification is the static testing.	Validation is the dynamic testing.
It does not include the execution of the code.	It includes the execution of the code.
Methods used in verification are reviews, walkthroughs, inspections and desk-checking.	Methods used in validation are Black Box Testing, White Box Testing and non-functional testing.

It checks whether the software conforms to specifications or not.	It checks whether the software meets the requirements and expectations of a customer or not.
It can find the bugs in the early stage of the development.	It can only find the bugs that could not be found by the verification process.
The goal of verification is application and software architecture and specification.	The goal of validation is an actual product.
Quality assurance team does verification.	Validation is executed on software code with the help of testing team.
It comes before validation.	It comes after verification.

What is Quality Assurance?

Quality Assurance is popularly known as QA Testing, is defined as an activity to ensure that an organization is providing the best possible product or service to customers.

What is Quality Control?

Quality control popularly abbreviated as QC. It is a Software Engineering process used to ensure quality in a product or a service. It does not deal with the processes used to create a product; rather it examines the quality of the "end products" and the final outcome.

Quality Assurance (QA)	Quality Control (QC)
•It is a procedure that focuses on providing assurance that quality requested will be achieved	•It is a procedure that focuses on fulfilling the quality requested.
•QA aims to prevent the defect	•QC aims to identify and fix defects
•It is a method to manage the quality- Verification	•It is a method to verify the quality-Validation
•It does not involve executing the program	•It always involves executing a program
•It's a Preventive technique	•It's a Corrective technique
•It's a Proactive measure	•It's a Reactive measure

•It is the procedure to create the deliverables	•It is the procedure to verify that deliverables
•QA involves in full software development life cycle	•QC involves in full software testing life cycle
•In order to meet the customer requirements, QA defines standards and methodologies	•QC confirms that the standards are followed while working on the product
•It is performed before Quality Control	•It is performed only after QA activity is done
•It is a Low-Level Activity, it can identify an error and mistakes which QC cannot	•It is a High-Level Activity, it can identify an error that QA cannot

•Its main motive is to prevent defects in the system. It is a less time-consuming activity	•Its main motive is to identify defects or bugs in the system. It is a more time-consuming activity
•QA ensures that everything is executed in the right way, and that is why it falls under verification activity	•QC ensures that whatever we have done is as per the requirement, and that is why it falls under validation activity
•It requires the involvement of the whole team	•It requires the involvement of the Testing team
•The statistical technique applied on QA is known as SPC or Statistical Process Control (SPC)	•The statistical technique applied to QC is known as SQC or Statistical Quality Control

Seven Principles of Software Testing

1. Testing shows the presence of bugs

Testing an application can only reveal that one or more defects exist in the application, however, testing alone cannot prove that the application is error free. Therefore, it is important to design test cases which find as many defects as possible.

2. Exhaustive testing is impossible

Unless the application under test (AUT) has a very simple logical structure and limited input, it is not possible to test all possible combinations of data and scenarios. For this reason, risk and priorities are used to concentrate on the most important aspects to test.

3. Early testing

The sooner we start the testing activities the better we can utilize the available time. As soon as the initial products, such the requirement or design documents are available, we can start testing. It is common for the testing phase to get squeezed at the end of the development lifecycle, i.e. when development has finished, so by starting testing early, we can prepare testing for each level of the development lifecycle.

Another important point about early testing is that when defects are found earlier in the lifecycle, they are much easier and cheaper to fix. It is much cheaper to change an incorrect requirement than having to change a functionality in a large system that is not working as requested or as designed!

4. Defect clustering

During testing, it can be observed that most of the reported defects are related to small number of modules within a system. i.e. small number of modules contain most of the defects in the system. This is the application of the Pareto Principle to software testing: approximately 80% of the problems are found in 20% of the modules.

5. The pesticide paradox

If you keep running the same set of tests over and over again, chances are no more new defects will be discovered by those test cases. Because as the system evolves, many of the previously reported defects will have been fixed and the old test cases do not apply anymore. Anytime a fault is fixed or a new functionality added, we need to do regression testing to make sure the new changed software has not broken any other part of the software. However, those regression test cases also need to change to reflect the changes made in the software to be applicable and hopefully fine new defects.

6. Testing is context dependent

Different methodologies, techniques and types of testing is related to the type and nature of the application. For example, a software application in a medical device needs more testing than a games software. More importantly a medical device software requires risk based testing, be compliant with medical industry regulators and possibly specific test design techniques. By the same token, a very popular website, needs to go through rigorous performance testing as well as functionality testing to make sure the performance is not affected by the load on the servers.

7. Absence of errors fallacy

Just because testing didn't find any defects in the software, it doesn't mean that the software is ready to be shipped. Were the executed tests really designed to catch the most defects? or where they designed to see if the software matched the user's requirements? There are many other factors to be considered before making a decision to ship the software.

Other principles to note are:

Testing must be done by an independent party.

Testing should not be performed by the person or team that developed the software since they tend to defend the correctness of the program.

Assign best personnel to the task.

Because testing requires high creativity and responsibility only the best personnel must be assigned to design, implement, and analyze test cases, test data and test results.

Test for invalid and unexpected input conditions as well as valid conditions.

The program should generate correct messages when an invalid test is encountered and should generate correct results when the test is valid.

Keep software static during test.

The program must not be modified during the implementation of the set of designed test cases.

Provide expected test results if possible.

A necessary part of test documentation is the specification of expected results, even if providing such results is impractical.

Objective of Software Testing

- Understand the difference between verification and validation testing activities
- Understand what benefits the V model offers over other models.
- Be aware of other models in order to compare and contrast.
- Understand the cost of fixing faults increases as you move the product towards live use.
- Understand what constitutes a master test plan in Software Testing.
- Understand the meaning of each testing stage in Software Testing.

What is STLC V-Model?

One of the major handicaps of waterfall STLC model was that defects were found at a very later stage of the development process since testing was done at the end of the development cycle. It became very challenging and costly to fix the defects since it was found at a very later stage. To overcome this problem, a new development model was introduced called the **“V Model”**

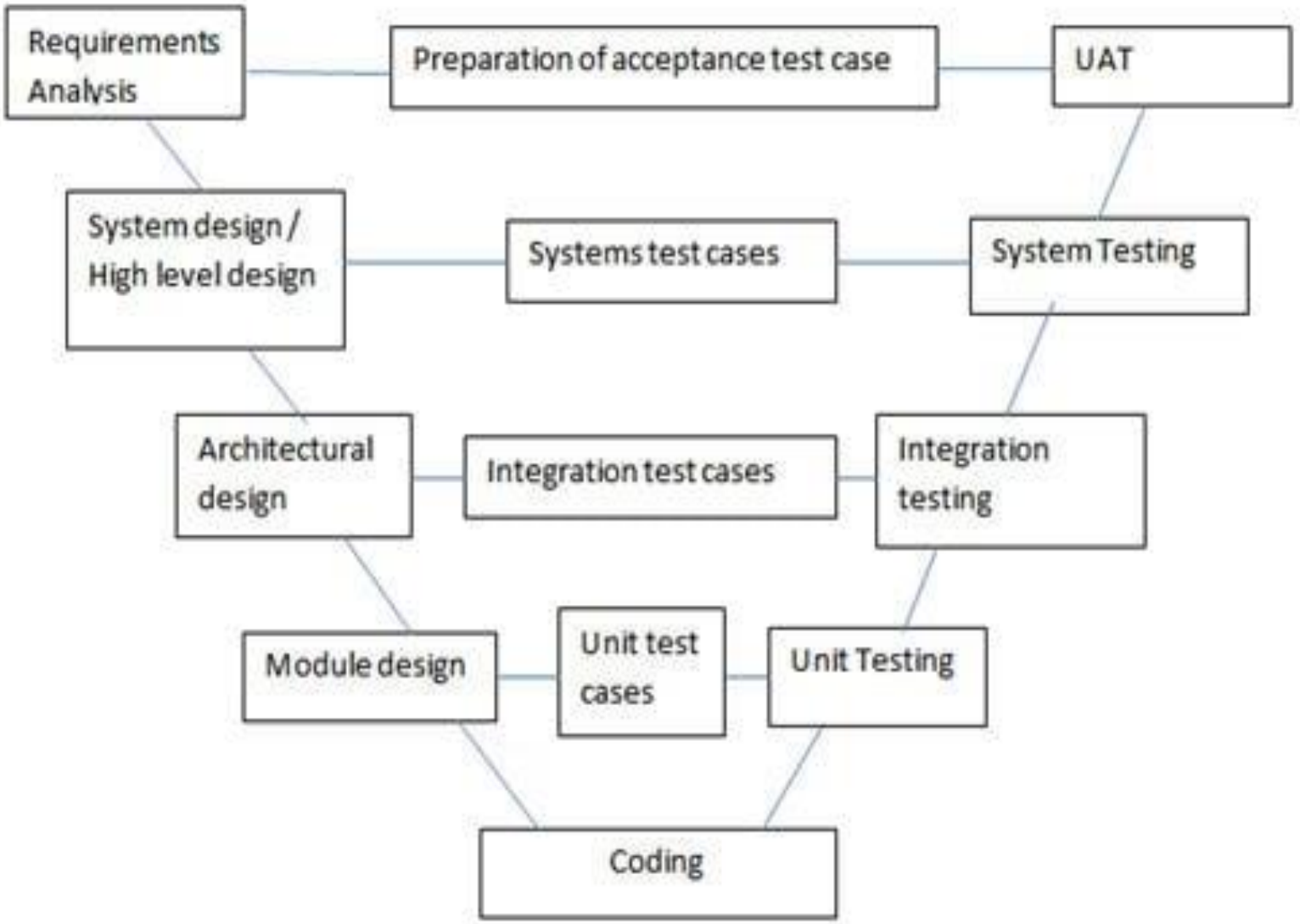
V model is also called a **verification and validation** model.

Verification and Validation

- **Verification**: Verification is a static analysis technique. In this technique, testing is done without executing the code. Examples include – Reviews, Inspection, and walkthrough.
- **Validation**: Validation is a dynamic analysis technique where testing is done by executing the code. Examples include functional and non-functional testing techniques.

V-Model

In the V model, the **development and QA activities are done simultaneously**. There is **no discrete phase called Testing**, rather testing starts right from the requirement phase. The verification and validation activities go hand in hand.



STLC - V Model - © www.SoftwareTestingHelp.com

In a typical development process, the **left-hand side shows the development activities** and the **right hand side shows the testing activities**

Pros and Cons of using V model

PROS	CONS
- Development and progress is very organized and systematic	-Not suitable for bigger and complex projects
- Works well for smaller to medium sized projects.	- Not suitable if the requirements are not consistent.
- Testing starts from beginning so ambiguities are identified from the beginning.	- No working software is produced in the intermediate stage.
- Easy to manage as each phase has well defined objectives and goals.	- No provision for doing risk analysis so uncertainty and risks are there.

What is Manual Testing?

- Manual testing is testing of the software where tests are executed manually by a QA Analysts. It is performed to discover bugs in software under development.
- In Manual testing, the tester checks all the essential features of the given application or software. In this process, the software testers execute the test cases and generate the test reports without the help of any automation software testing tools.
- It is a classical method of all testing types and helps find bugs in software systems. It is generally conducted by an experienced tester to accomplish the software testing process.

What is Automation Testing?

- In Automated Software Testing, testers write code/test scripts to automate test execution. Testers use appropriate automation tools to develop the test scripts and validate the software. The goal is to complete test execution in a less amount of time.
- Automated testing entirely relies on the pre-scripted test which runs automatically to compare actual result with the expected results. This helps the tester to determine whether or not an application performs as expected.
- Automated testing allows you to execute repetitive task and regression test without the intervention of manual tester. Even though all processes are performed automatically, automation requires some manual effort to create initial testing scripts.

Difference Between Manual Testing and Automation Testing

Parameter	Automation Testing	Manual Testing
Definition	Automation Testing uses automation tools to execute test cases.	In manual testing, test cases are executed by a human tester and software.
Processing time	Automated testing is significantly faster than a manual approach.	Manual testing is time-consuming and takes up human resources.
Exploratory Testing	Automation does not allow random testing	Exploratory testing is possible in Manual Testing

Initial investment	The initial investment in the automated testing is higher. Though the ROI is better in the long run.	The initial investment in the Manual testing is comparatively lower. ROI is lower compared to Automation testing in the long run.
Reliability	Automated testing is a reliable method, as it is performed by tools and scripts. There is no testing Fatigue.	Manual testing is not as accurate because of the possibility of the human errors.

UI Change	For even a trivial change in the UI of the AUT, Automated Test Scripts need to be modified to work as expected	Small changes like change in id, class, etc. of a button wouldn't thwart execution of a manual tester.
Investment	Investment is required for testing tools as well as automation engineers	Investment is needed for human resources.

Cost-effective	Not cost effective for low volume regression	Not cost effective for high volume regression.
Test Report Visibility	With automation testing, all stakeholders can login into the automation system and check test execution results	Manual Tests are usually recorded in an Excel or Word, and test results are not readily/ readily available.

Parallel Execution	This testing can be executed on different operating platforms in parallel and reduce test execution time.	Manual tests can be executed in parallel but would need to increase your human resource which is expensive
Batch testing	You can Batch multiple Test Scripts for nightly execution.	Manual tests cannot be batched.
Programming knowledge	Programming knowledge is a must in automation testing.	No need for programming in Manual Testing.

Human observation	Automated testing does not involve human consideration. So it can never give assurance of user-friendliness and positive customer experience.	The manual testing method allows human observation, which may be useful to offer user-friendly system.
Performance Testing	Performance Tests like Load Testing, Stress Testing, Spike Testing, etc. have to be tested by an automation tool compulsorily.	Performance Testing is not feasible manually

Set up	Automation test requires less complex test execution set up.	Manual testing needs have a more straightforward test execution setup
Engagement	Done by tools. Its accurate and never gets bored!	Repetitive Manual Test Execution can get boring and error-prone.
Ideal approach	Automation testing is useful when frequently executing the same set of test cases	Manual testing proves useful when the test case only needs to run once or twice.
Build Verification Testing	Automation testing is useful for Build Verification Testing (BVT).	Executing the Build Verification Testing (BVT) is very difficult and time-consuming in manual testing.

Deadlines	Automated Tests have zero risks of missing out a pre-decided test.	Manual Testing has a higher risk of missing out the pre-decided test deadline.
Framework	Automation testing uses frameworks like Data Drive, Keyword, Hybrid to accelerate the automation process.	Manual Testing does not use frameworks but may use guidelines, checklists, stringent processes to draft certain test cases.

Documentation	Automated Tests acts as a document provides training value especially for automated unit test cases. A new developer can look into a unit test cases and understand the code base quickly.	Manual Test cases provide no training value
Test Design	Automated Unit Tests enforce/drive Test Driven Development Design.	Manual Unit Tests do not drive design into the coding process

Devops	Automated Tests help in Build Verification Testing and are an integral part of DevOps Cycle	Manual Testing defeats the automated build principle of DevOps
When to Use?	Automated Testing is suited for Regression Testing, Performance Testing, Load Testing or highly repeatable functional test cases.	Manual Testing is suitable for Exploratory, Usability and Adhoc Testing. It should also be used where the AUT changes frequently.

Tools used for automation testing

- HP Quick Test Professional
- Selenium
- IBM Rational Functional Tester
- SilkTest
- TestComplete
- Testing Anywhere
- WinRunner
- LoadRunner
- Visual Studio Test Professional
- WATIR

What is White Box Testing?

White Box Testing is defined as the testing of a software solution's internal structure, design, and coding. In this type of testing, the code is visible to the tester. It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security. White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. It is usually performed by developers.

It is one of two parts of the "Box Testing" approach to software testing. Its counterpart, Blackbox testing, involves testing from an external or end-user type perspective. On the other hand, Whitebox testing is based on the inner workings of an application and revolves around internal testing.

The term "WhiteBox" was used because of the see-through box concept. The clear box or WhiteBox name symbolizes the ability to see through the software's outer shell (or "box") into its inner workings. Likewise, the "black box" in "Black Box Testing" symbolizes not being able to see the inner workings of the software so that only the end-user experience can be tested.

What do you verify in White Box Testing?

White box testing involves the testing of the software code for the following:

- Internal security holes
- Broken or poorly structured paths in the coding processes
- The flow of specific inputs through the code
- Expected output
- The functionality of conditional loops
- Testing of each statement, object, and function on an individual basis

The testing can be done at system, integration and unit levels of software development. One of the basic goals of whitebox testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

How do you perform White Box Testing?

STEP 1) UNDERSTAND THE SOURCE CODE

The first thing a tester will often do is learn and understand the source code of the application. Since white box testing involves the testing of the inner workings of an application, the tester must be very knowledgeable in the programming languages used in the applications they are testing. Also, the testing person must be highly aware of secure coding practices. Security is often one of the primary objectives of testing software. The tester should be able to find security issues and prevent attacks from hackers and naive users who might inject malicious code into the application either knowingly or unknowingly.

Step 2) CREATE TEST CASES AND EXECUTE

The second basic step to white box testing involves testing the application's source code for proper flow and structure. One way is by writing more code to test the application's source code. The tester will develop little tests for each process or series of processes in the application. This method requires that the tester must have intimate knowledge of the code and is often done by the developer. Other methods include Manual Testing, trial, and error testing and the use of testing tools as we will explain further on in this article.

White Box Testing Techniques

- A major White box testing technique is Code Coverage analysis. Code Coverage analysis eliminates gaps in a Test Case suite. It identifies areas of a program that are not exercised by a set of test cases. Once gaps are identified, you create test cases to verify untested parts of the code, thereby increasing the quality of the software product
- There are automated tools available to perform Code coverage analysis. Below are a few coverage analysis techniques
- **Statement Coverage**:- This technique requires every possible statement in the code to be tested at least once during the testing process of software engineering.

- **Branch Coverage** - This technique checks every possible path (if-else and other conditional loops) of a software application.
- Apart from above, there are numerous coverage types such as Condition Coverage, Multiple Condition Coverage, Path Coverage, Function Coverage etc. Each technique has its own merits and attempts to test (cover) all parts of software code. Using Statement and Branch coverage you generally attain 80-90% code coverage which is sufficient.

Types of White Box Testing

- Unit Testing: It is often the first type of testing done on an application. Unit Testing is performed on each unit or block of code as it is developed. Unit Testing is essentially done by the programmer. As a software developer, you develop a few lines of code, a single function or an object and test it to make sure it works before continuing Unit Testing helps identify a majority of bugs, early in the software development lifecycle. Bugs identified in this stage are cheaper and easy to fix.
- Testing for Memory Leaks: Memory leaks are leading causes of slower running applications. A QA specialist who is experienced at detecting memory leaks is essential in cases where you have a slow running software application.

White Box Testing Tools

Below is a list of top white box testing tools.

- Veracode
- EclEmma
- RCUNIT
- Nunit
- JSUnit
- Junit
- CppUnit

Advantages of White Box Testing

- Code optimization by finding hidden errors.
- White box tests cases can be easily automated.
- Testing is more thorough as all code paths are usually covered.
- Testing can start early in SDLC even if GUI is not available.

Disadvantages of WhiteBox Testing

- White box testing can be quite complex and expensive.
- Developers who usually execute white box test cases detest it. The white box testing by developers is not detailed can lead to production errors.
- White box testing requires professional resources, with a detailed understanding of programming and implementation.
- White-box testing is time-consuming, bigger programming applications take the time to test fully.

What is Black Box Testing?

Black box testing is defined as a testing technique in which functionality of the Application Under Test (AUT) is tested without looking at the internal code structure, implementation details and knowledge of internal paths of the software. This type of testing is based entirely on software requirements and specifications. In BlackBox Testing we just focus on inputs and output of the software system without bothering about internal knowledge of the software program.

How to do BlackBox Testing

- Initially, the requirements and specifications of the system are examined.
- Tester chooses valid inputs (positive test scenario) to check whether SUT processes them correctly. Also, some invalid inputs (negative test scenario) are chosen to verify that the SUT is able to detect them.
- Tester determines expected outputs for all those inputs.
- Software tester constructs test cases with the selected inputs.
- The test cases are executed.
- Software tester compares the actual outputs with the expected outputs.
- Defects if any are fixed and re-tested.

Types of Black Box Testing

- **Functional testing** - This black box testing type is related to the functional requirements of a system; it is done by software testers.
- **Non-functional testing** - This type of black box testing is not related to testing of specific functionality, but non-functional requirements such as performance, scalability, usability.
- **Regression testing** - Regression Testing is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Black Box Testing Techniques

- **Equivalence Class Testing:** It is used to minimize the number of possible test cases to an optimum level while maintains reasonable test coverage.
- **Boundary Value Testing:** Boundary value testing is focused on the values at boundaries. This technique determines whether a certain range of values are acceptable by the system or not. It is very useful in reducing the number of test cases. It is most suitable for the systems where an input is within certain ranges.
- **Decision Table Testing**: A decision table puts causes and their effects in a matrix. There is a unique combination in each column.

Black Box Testing	White Box Testing
the main focus of black box testing is on the validation of your functional requirements.	White Box Testing (Unit Testing) validates internal structure and working of your software code
Black box testing gives abstraction from code and focuses on testing effort on the software system behavior.	To conduct White Box Testing, knowledge of underlying programming language is essential. Current day software systems use a variety of programming languages and technologies and its not possible to know all of them.
Black box testing facilitates testing communication amongst modules	White box testing does not facilitate testing communication amongst modules

What is Gray Box Testing?

- Gray Box Testing is a technique to test the software product or application with partial knowledge of the internal workings of an application.
- In this process, context-specific errors that are related to web systems are commonly identified. It will increase the testing coverage by concentrating on all of the layers of any complex system.
- Gray Box Testing is a software testing method, which is a combination of both White Box Testing and Black Box Testing method.

- In White Box testing internal structure (code) is known
- In Black Box testing internal structure (code) is unknown
- In Grey Box Testing internal structure (code) is partially known.


Black Box Testing
+
White Box Testing
=
Gray Box Testing

In Software Engineering, Gray Box Testing gives the ability to test both sides of an application, presentation layer as well as the code part. It is primarily useful in **Integration Testing and Penetration Testing.**

Example of Gray Box Testing: While testing websites feature like links or orphan links, if tester encounters any problem with these links, then he can make the changes straightaway in HTML code and can check in real time.

Why Gray Box Testing

- It provides combined benefits of both black box testing and white box testing both
- It combines the input of developers as well as testers and improves overall product quality
- It reduces the overhead of long process of testing functional and non-functional types
- It gives enough free time for a developer to fix defects
- Testing is done from the user point of view rather than a designer point of view

Gray Box Testing Strategy

To perform Gray box testing, it is not necessary that the tester has the access to the source code. A test is designed based on the knowledge of algorithm, architectures, internal states, or other high -level descriptions of the program behavior.

It applies a straightforward technique of black box testing

It is based on requirement test case generation, as such, it presets all the conditions before the program is tested by assertion method.

Techniques used for Grey box Testing are-

Matrix Testing: This testing technique involves defining all the variables that exist in their programs.

Regression Testing: To check whether the change in the previous version has regressed other aspects of the program in the new version. It will be done by testing strategies like retest all, retest risky use cases, retest within a firewall.

Orthogonal Array Testing or OAT: It provides maximum code coverage with minimum test cases.

Pattern Testing: This testing is performed on the historical data of the previous system defects. Unlike black box testing, gray box testing digs within the code and determines why the failure happened

Usually, Grey box methodology uses automated software testing tools to conduct the testing. Stubs and module drivers are created to relieve tester to manually generate the code.

Steps to perform Grey box Testing are:

Step 1: Identify inputs

Step 2: Identify the outputs

Step 3: Identify the major paths

Step 4: Identify Subfunctions

Step 5: Develop inputs for Subfunctions

Step 6: Develop outputs for Subfunctions

Step 7: Execute test case for Subfunctions

Step 8: Verify the correct result for Subfunctions

Step 9: Repeat steps 4 & 8 for other Subfunctions

Step 10: Repeat steps 7 & 8 for other Subfunctions

Gray Box Testing Challenges

- When a component under test encounter a failure of some kind may lead to abortion of the ongoing operation
- When test executes in full but the content of the result is incorrect.

What is Functional Testing?

Functional Testing is defined as a type of testing which verifies that each function of the software application operates in conformance with the requirement specification. This testing mainly involves black box testing and it is not concerned about the source code of the application.

Each and every functionality of the system is tested by providing appropriate input, verifying the output and comparing the actual results with the expected results.

This testing involves checking of User Interface, APIs, Database, security, client/ server applications and functionality of the Application Under Test.

The testing can be done either manually or using automation

What do you test in Functional Testing?

- Mainline functions: Testing the main functions of an application
- Basic Usability: It involves basic usability testing of the system. It checks whether a user can freely navigate through the screens without any difficulties.
- Accessibility: Checks the accessibility of the system for the user
- Error Conditions: Usage of testing techniques to check for error conditions. It checks whether suitable error messages are displayed.

Identify test input (test data)

Compute the expected outcomes with the selected test input values

Execute test cases

Comparison of actual and computed expected result

How to perform Functional Testing: Complete Process

- Understand the Software Engineering Requirements
- Identify test input (test data)
- Compute the expected outcomes with the selected test input values
- Execute test cases
- Comparison of actual and computed expected result

Functional Testing Tools

- Ranorex Studio — all-in-one functional test automation for desktop, web, and mobile apps with built-in Selenium WebDriver.
- Selenium - Popular Open Source Functional Testing Tool
- QTP - Very user-friendly Functional Test tool by HP
- JUnit- Used mainly for Java applications and this can be used in Unit and System Testing
- soapUI - This is an open source functional testing tool, mainly used for Web service testing. It supports multiple protocols such as HTTP, SOAP, and JDBC.
- Watir - This is a functional testing tool for web applications. It supports tests executed at the web browser and uses a ruby scripting language

What is Non-Functional Testing?

- Non-functional testing is defined as a type of Software testing to check non-functional aspects (performance, usability, reliability, etc) of a software application. It is designed to test the readiness of a system as per nonfunctional parameters which are never addressed by functional testing.
- An excellent example of non-functional test would be to check how many people can simultaneously login into a software.
- Non-functional testing is equally important as functional testing and affects client satisfaction.

Objectives of Non-functional testing

- Non-functional testing should increase usability, efficiency, maintainability, and portability of the product.
- Helps to reduce production risk and cost associated with non-functional aspects of the product.
- Optimize the way product is installed, setup, executes, managed and monitored.
- Collect and produce measurements, and metrics for internal research and development.
- Improve and enhance knowledge of the product behavior and technologies in use.

Characteristics of Non-functional testing

- Non-functional testing should be measurable, so there is no place for subjective characterization like good, better, best, etc.
- Exact numbers are unlikely to be known at the start of the requirement process
- Important to prioritize the requirements
- Ensure that quality attributes are identified correctly in Software Engineering.

Non Functional Testing Parameters

Functional Testing	System Testing
1. Functional testing is a part of system testing.	1. Tests are executed to verify the compliance of the completed and integrated software with its specifications.
2. Verifies the functionality of the product and ensures it works as per its specifications.	2. Its purpose is to ensure that the completed product fulfills the stated requirements.
3. Tests the functionalities and features of the software, such as load, security, performance and more.	3. It is performed to the test the end product.
4. Manual testing and automation test tools can be used for functional testing.	4. It tests every single module, interface and internal and external aspects of the software.
5.It helps define what the product does.	5.It defines how good a software works.
6. Does not require internal knowledge of the structure.	6. System testing is performed after integration testing and before acceptance testing.