Contents

Functional Testing

Functional testing involves testing the application against the business requirements. The goal of functional testing is to verify that the application is behaving the way it was designed to.

Functional testing ensures that your software is ready to for release to the public. It also verifies that all the specified requirements have been incorporated.

Another way of stating this is that functional testing is "the customer test". But even this is misleading. Developers need a benchmark during all development stages — a developer-independent benchmark — to tell them what they have and have not achieved. Functional testing begins as soon as there is a function to test and continues through the application's completion and first customer contact.

The expression, "the customer test", can be misleading in another way. Some people think of functional testing as mimicking a user and checking for an expected output. But the real customer is not just someone feeding commands into the application. They are running the application on a system, simultaneously with other applications, with constant fluctuations in user load. The application, of course, should be crash-resistant in the face of these conditions.

More and more, the user interface of our applications is supplied by pre-tested components, which produce few surprises once they are integrated correctly. On the other hand, we are constantly asked to write custom applications for systems of ever-increasing complexity. Therefore, the central functions tested by a functional test are increasingly system-related rather than user-related.

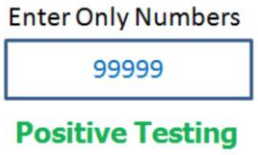
There are two major categories of functional testing: positive and negative functional testing. Positive functional testing involves inputting valid inputs to see how the application responds to these and also testing to determine if outputs are correct. Negative functional testing involves using different invalid inputs, unanticipated operating conditions and other invalid operations.

Positive Testing

Positive testing, or "Happy path testing", is the type of testing that can be performed on the system by providing the valid data as input. It checks whether an application behaves as expected with the positive input.

Positive testing determines that your application works as expected. If an error is encountered during positive testing, the test fails.

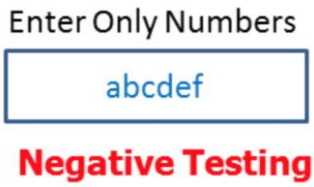
Example:



Negative Testing

Negative testing ensures that your application can gracefully handle an invalid input or unexpected user behavior. For example, if a user tries to type a letter in a numeric field, the correct behavior in this case would be to display the "Incorrect data type, please enter a number" message.

Example:



The purpose of negative testing is to detect such situations and prevent applications from crashing. Also, negative testing helps you improve the quality of your application and find its weak points.

Negative testing is aimed at detecting possible application crashes in different situations. Below are several possible examples of such situations:

Populating required fields – Some applications and web pages contain fields that are marked as required. To check an application's behavior, create a test that leaves the required fields empty and analyzes the tested application's response.

For example, the application can show a message box requesting a user to populate an appropriate field. After that, your test must interpret the application's behavior as correct when working with invalid data.

Correspondence between data and field types – Typically, dialog boxes and forms contain controls that can accept data of a specific type (for example, numeric, date, text, etc.).

To verify whether the application functions properly, you can create a test that enters incorrect data into a control, for example, a test that enters a letter into an UpDown edit box or the value of 13/30/2010 to a date field.

Allowed number of characters – Some applications and web pages contain fields allowing a limited number of characters to be entered, for example, the value of the User Name field for applications and web pages must contain less than 50 characters.

To check the behavior of the application, create a test that enters more characters into the field than is allowed.

Allowed data bounds and limits – Applications can use input fields that accept data in a certain range. For example, there can be an edit box into which you enter an integer number from 10 to

50, or an edit box that accepts text of a specific length. To check the application's behavior, create a negative test that enters a value smaller than the lower bound or greater than the upper bound of the specified field.

Another example of this negative test case is entering data that exceeds the data type limits. For instance, an integer value can normally contain values in the range of -2,147,483,648..2,147,483,647 (the size is limited by the number of bytes in memory).

To check the application's behavior, you can create a negative test that enters a value exceeding the bounds. For instance, the test can enter a large number (100,000,000,000) into an integer field.

Reasonable data – Some applications and web pages include fields that have a reasonable limit, for example, entering 200 or a negative number as the value for the "Your age:" field is not allowed. To check the application's behavior, create a negative test that enters invalid data into the specified field.

Web session testing – Some web browsers require that you log in before the first web page is opened. To check that these browsers' function correctly, create a test that tries to open web pages in the tested application without logging in.

Testing Techniques used for both Positive and Negative Testing

Unit Testing

Unit testing is usually performed by the developer who writes different code units that could be related or unrelated to achieve a particular functionality.

Therefore, this usually entails writing unit tests which would call the methods in each unit and validate that when the needed parameters are passed, its return value is as expected. Code coverage is an important part of unit testing where test cases need to exist to cover the below three:

- Line coverage
- Code path coverage
- Method coverage

Sanity Testing

Testing that is done to ensure that all major and vital functionalities of the application/system are working correctly. This is generally done after a smoke test.

Smoke testing

Testing that is done after each build is released to test to ensure build stability. It's also called build verification testing.

Regression tests

Testing performed to ensure that the adding new code, enhancements, fixing of bugs is not breaking existing functionality or cause instability and still works according to the specifications. Regression tests need not be as extensive as the actual functional tests but should ensure just the amount of coverage to certify that the functionality is stable.

Integration tests

When the system relies on multiple functional modules that might individually work perfectly, but have to work coherently when clubbed together to achieve an end to end scenario, validation of such scenarios is called integration testing or system integration test.

Beta/Usability testing

Product is exposed to the actual customer in a production like an environment and they test the product. The user's comfort is derived from this and feedback is taken. This is similar to User Acceptance testing.

Positive Testing vs. Negative Testing

The core difference between positive testing and negative testing is that throwing an exception is not an unexpected event in the latter. When you perform negative testing, exceptions are expected – they indicate that the application handles improper user behavior correctly.

It is generally considered a good practice to combine both the positive and the negative testing approaches. This strategy provides higher tested application coverage as compared to using only one of the specified automated testing methodologies.

#	Positive Testing	Negative Testing
1	Positive Testing means testing the application or system by giving valid data.	Negative Testing means testing the application or system by giving invalid data.
2	It is always done to verify the known set of Test Conditions.	It is always done to check if the application handles improper user behavior correctly.
3	Positive testing ensures that the business use case is validated.	Negative testing ensures that the delivered software has no flaws that can be a deterrent in its usage by the customer.

Non-Functional Testing

In non-functional testing the quality characteristics of the component or system is tested. Non-functional refers to aspects of the software that may not be related to a specific function or user action such as scalability or security (e.g. how many people can log in at once?).

Non-functional testing is also performed at all levels like functional testing.

While functional testing is concerned about business requirements, non-functional testing is designed to figure out if your product will provide a good user experience. For example, non-functional tests are used to determine how fast the product responds to a request or how long it takes to do an action.

Examples of non-functional tests include:

Reliability testing

Reliability Testing is about exercising an application so that failures are discovered and removed before the system is deployed. The purpose of reliability testing is to determine product reliability, and to determine whether the software meets the customer's reliability requirements.

Usability testing

In usability testing basically the testers tests the ease with which the user interfaces can be used. It tests that whether the application or the product built is user-friendly or not.

Usability testing includes the following five components:

- Learnability
  - How easy is it for users to accomplish basic tasks the first time they encounter the design?
- Efficiency
  - How fast can experienced users accomplish tasks?
- Memorability
  - When users return to the design after a period of not using it, does the user remember enough to use it effectively the next time, or does the user have to start over again learning everything?
- Errors
  - How many errors do users make, how severe are these errors and how easily can they recover from the errors?
- Satisfaction
  - How much does the user like using the system?

Maintainability testing

It basically defines that how easy it is to maintain the system. This means that how easy it is to analyze, change and test the application or product.

Portability testing

It refers to the process of testing the ease with which a computer software component or application can be moved from one environment to another, e.g. moving of any application from Windows 2000 to Windows XP. This is usually measured in terms of the maximum amount of effort permitted. Results are measured in terms of the time required to move the software and complete the and documentation updates.

Baseline testing

It refers to the validation of documents and specifications on which test cases would be designed. The requirement specification validation is baseline testing.

Compliance testing

It is related with the IT standards followed by the company and it is the testing done to find the deviations from the company prescribed standards.

Documentation testing

As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.

Stress testing

It involves testing beyond normal operational capacity, often to a breaking point, in order to observe the results. It is a form of testing that is used to determine the stability of a given system. It put greater emphasis on robustness, availability, and error handling under a heavy load, rather than on what would be considered correct behavior under normal circumstances. The goals of such tests may be to ensure the software does not crash in conditions of insufficient computational resources (such as memory or disk space).

Load testing

A load test is usually conducted to understand the behavior of the application under a specific expected load. Load testing is performed to determine a system's behavior under both normal and at peak conditions. It helps to identify the maximum operating capacity of an application as well as any bottlenecks and determine which element is causing degradation. E.g. If the number of users are increased then how much CPU, memory will be consumed, what is the network and bandwidth response time

Performance testing

Performance testing is testing that is performed, to determine how fast some aspect of a system performs under a particular workload. It can serve different purposes like it can demonstrate that the system meets performance criteria. It can compare two systems to find which performs better. Or it can measure what part of the system or workload causes the system to perform badly.

Compatibility testing

Compatibility testing is basically the testing of the application or the product built with the computing environment. It tests whether the application or the software product built is compatible with the hardware, operating system, database or other system software or not.

Security testing

Security testing is basically to check that whether the application or the product is secured or not. Can anyone come tomorrow and hack the system or login the application without any authorization. It is a process to determine that an information system protects data and maintains functionality as intended.

Scalability testing

It is the testing of a software application for measuring its capability to scale up in terms of any of its non-functional capability like load supported, the number of transactions, the data volume etc.

Recovery testing

Recovery testing is done in order to check how fast and better the application can recover after it has gone through any type of crash or hardware failure etc. Recovery testing is the forced failure of the software in a variety of ways to verify that recovery is properly performed. For example, when an application is receiving data from a network, unplug the connecting cable. After some time, plug the cable back in and analyze the application's ability to continue receiving data from the point at which the network connection got disappeared. Restart the system while a browser has a definite number of sessions and check whether the browser is able to recover all of them or not.

Internationalization testing and Localization testing

Internationalization is a process of designing a software application so that it can be adapted to various languages and regions without any changes. Whereas Localization is a process of adapting internationalized software for a specific region or language by adding local specific components and translating text.

Functional vs Non-Functional Testing

The major difference between functional and non-functional testing is this: Functional testing ensures that your product meets customer and business requirements, and doesn't have any major bugs. On the other hand, non-functional testing wants to see if the product stands up to customer expectations.

Basically, functional testing is designed to determine that the application's features and operations perform the way they should. Non-functional testing wants to know that the product "behaves" correctly.

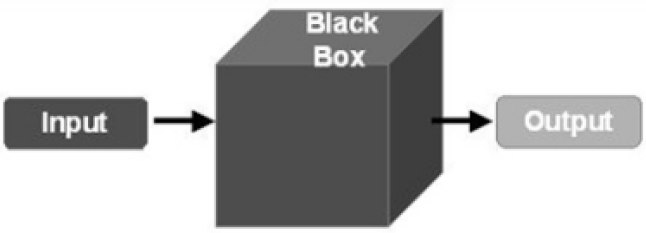
Your application needs to pass both categories of testing to ensure that your consumers have a good experience with your product. Failure to release a working product that meets the needs of consumer demands can damage your company's reputation and reduce overall product sales.

#	Functional Testing	Non-Functional Testing

1	Functional testing is performed using the functional specification provided by the client and verifies the system against the functional requirements.	Non-Functional testing checks the Performance, reliability, scalability and other non-functional aspects of the software system.
2	Functional testing is executed first	Non-Functional testing should be performed after functional testing
3	Manual testing or automation tools can be used for functional testing	Using automation tools will be effective for non-functional testing
4	Business requirements are the inputs to functional testing	Performance parameters like speed, scalability are inputs to non-functional testing.
5	Functional testing describes what the product does	Non-Functional testing describes how good the product works
6	Easy to do manual testing	Tough to do manual testing

Black Box Testing

In Black Box Testing, the tester tests an application without knowledge of the internal workings of the application being tested. Data are entered into the application and the outcome is compared with the expected results; what the program does with the input data or how the program arrives at the output data is not a concern for the tester performing black box testing. All that is tested is the behavior of the functions being tested.



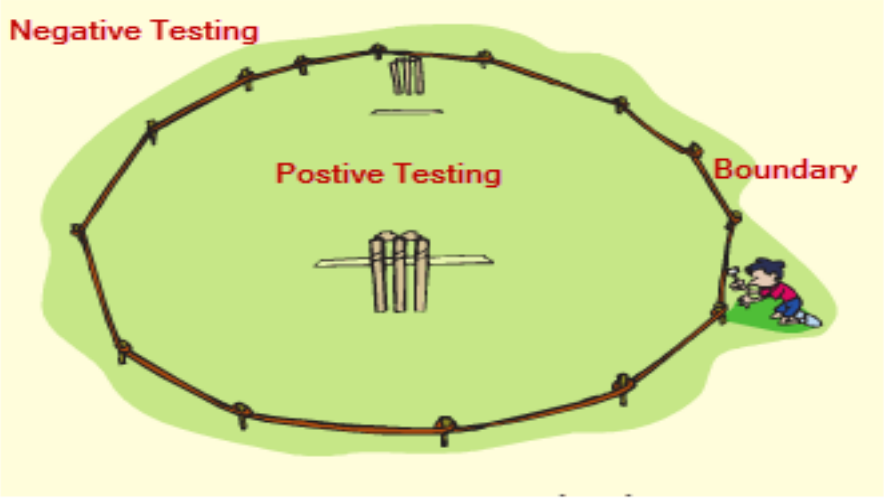
This is why black box testing is also known as functional testing which tests the functionality of a program. Note we can also have non-functional black box testing, such as performance testing which is a type of black box testing but instead of verifying the behavior of the system, it tests how long it takes for a function to respond to user's inputs and how long it takes to process the data and generate outputs.

Because black box testing is not concerned with the underlying code, then the techniques can be derived from the requirement documents or design specifications and hence testing can start as soon as the requirements are written.

Black Box Testing Techniques

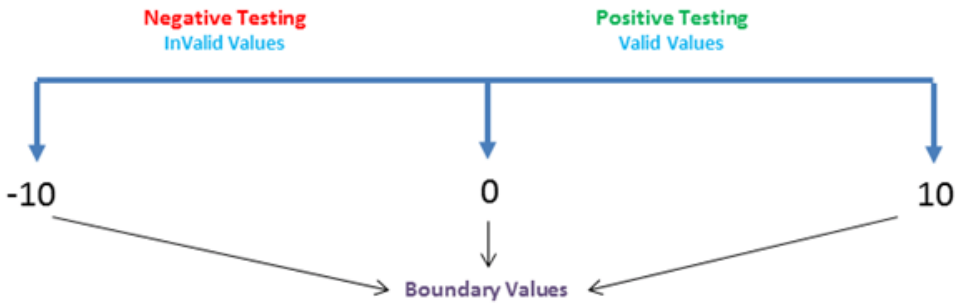
Boundary Value Analysis:

This is one of the software testing technique in which the test cases are designed to include values at the boundary. If the input data is used within the boundary value limits, then it is said to be Positive Testing. If the input data is picked outside the boundary value limits, then it is said to be Negative Testing.



Example:

A system can accept the numbers from 0 to 10 numeric values. All other numbers are invalid values. Under this technique, boundary values 0, 10 and -10 will be tested.

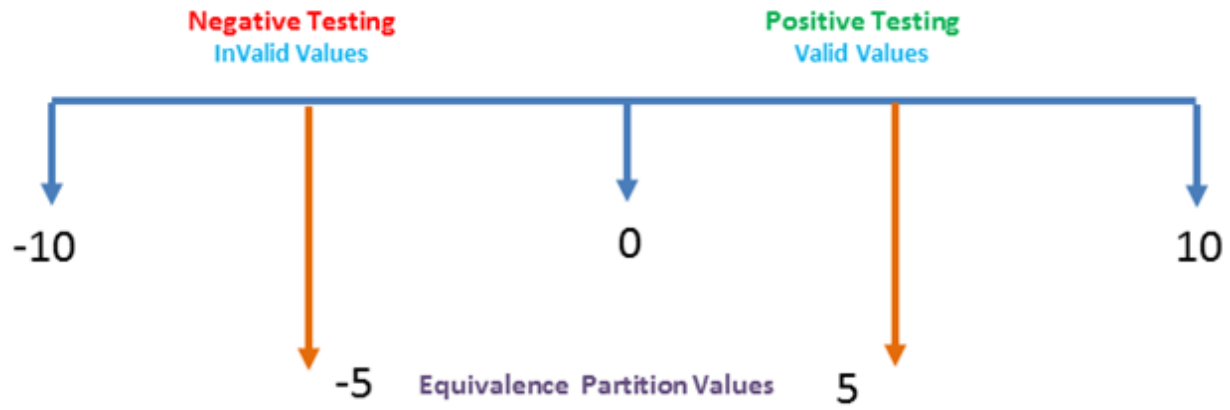


Equivalence Partitioning:

This is a software testing technique which divides the input date into many partitions. Values from each partition must be tested at least once. Partitions with valid values are used for Positive Testing. While, partitions with invalid values are used for negative testing.

Example:

Numeric values 0 to 10 can be divided to two (or three) partitions. In our case we have two partitions -10 to -1 and 0 to 10. Sample values (5 and -5) can be taken from each part to test the scenarios.



Advantages and disadvantages of Black Box Testing

Advantages of Black Box Testing

- The test is unbiased because the designer and the tester are independent of each other
- The tester does not need knowledge of any specific programming languages
- The test is done from the point of view of the user, not the designer
- Test cases can be designed as soon as the specifications are complete

Disadvantages of Black Box Testing

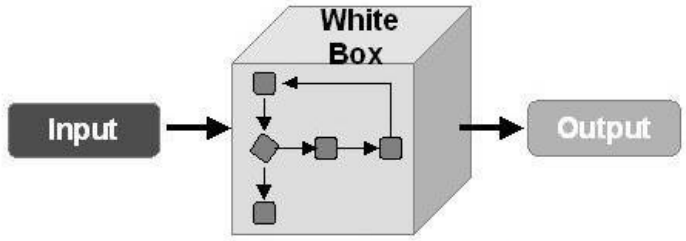
- The test can be redundant if the software designer has already run a test case
- The test cases are difficult to design
- Testing every possible input stream is unrealistic because it would take a inordinate amount of time; therefore, many program paths will go untested

White Box Testing

White Box Testing (WBT) is also known as Code-Based Testing or Structural Testing. White box testing is the software testing method in which internal structure is being known to tester who is going to test the software.

A short definition for white box testing would be “Testing based on an analysis of the internal structure of the component or system”.

In this method of testing the test cases are calculated based on analysis internal structure of the system based on Code coverage, branches coverage, paths coverage, condition Coverage etc.



White box testing involves the testing by looking at the internal structure of the code and when you are completely aware of the internal structure of the code then you can run your test cases and check whether the system meets the requirements mentioned in the specification document. Based on derived test cases the user exercised the test cases by giving the input to the system and checking for expected outputs with actual output. In this is testing method user has to go beyond the user interface to find the correctness of the system.

Typically, such a method is used at Unit Testing of the code but this different as Unit testing done by the developer and White Box Testing done by the testers, this is learning the part of the code and finding out the weakness in the software program under test.

For tester to test the software application under test is like a white/transparent box where the inside of the box is clearly seen to the tester (as tester is aware/access of the internal structure of the code), so this method is called as White Box Testing.

The White-box testing is one of the best method to find out the errors in the software application in early stage of software development life cycle. In this process the deriving the test cases is most important part. The test case design strategy includes such that all lines of the source code will be executed at least once or all available functions are executed to complete 100% code coverage of testing.

White Box Testing Techniques

Here are some examples of White Box Testing Techniques.

Statement Coverage

This technique tries to cover 100% statement coverage of the code, which means to test that every possible statement in the code is executed at least once.

Decision Coverage

This technique tries to cover 100% decision coverage of the code. This means to test that every possible decision conditions (example: if-else, for loop and other conditional loops) in the code is executed at least once.

Condition Coverage

This technique tries to cover 100% Condition coverage of the code. This means to test that every possible condition in the code is executed at least once.

Code Coverage analysis helps to identifying the gaps in a test case suite. It allows you to find the area in the code to which is not executed by given set of test cases. Upon identifying the gaps in the test case suite you can add the respective test case. So it helps to improve the quality of the software application.

Decision/Condition Coverage

This mixed type of white box testing technique tries to cover 100% Decision/Condition coverage of the code, which means that every possible Decisions/Conditions in the code are executed at least once.

Multiple Condition Coverage

This type of testing is used to cover each entry point of the system to be execute once.

In the actual development process developers/testers make use of different combinations of white box techniques that are suitable for the software application they are building.

Advantages and Disadvantages of White Box Testing

Advantages of White Box Testing

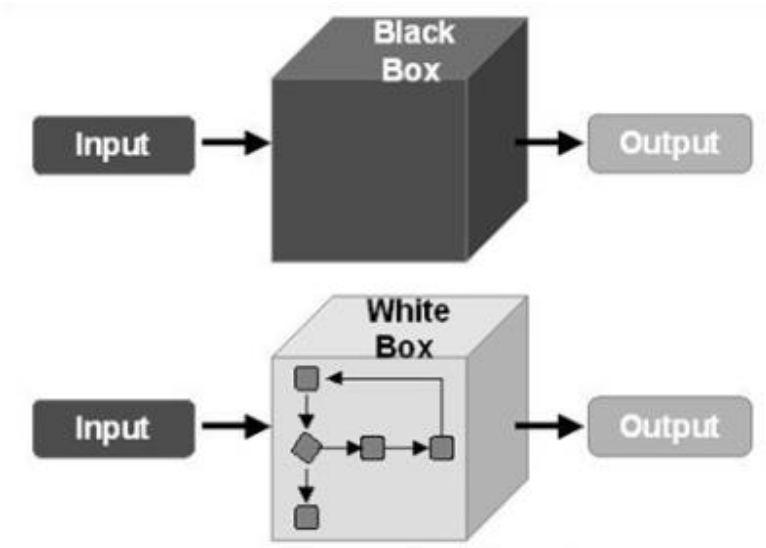
- You can start testing the piece of software without have a user interface (UI) developed.
- White Box testing techniques are often considered very thorough because all possible paths of the the code are verified.
- Tester can ask about the implementation of each section of the code, so it might be possible to remove unused lines of code which might later lead to a bug.
- As the tester is aware of the internal coding structure, then it is helpful to derive which type of input data is needed to test the software application effectively.

Disadvantages of White Box Testing

- To test the software application, a highly skilled tester is required to carry out the testing.
- If the application under test is large in size then exhaustive testing is impossible.
- To analyze line by line or path by path code is a nearly impossible work which may introduce or miss the defects in the code.

- To test each path or conditions may require different input conditions, so to fully test the application the tester needs to create a wide range of inputs which may be very time consuming.

Black Box Testing vs. White Box Testing



#	Black Box	White Box
1	Black box testing is the Software testing method which is used to test the software without knowing the internal structure of code or program.	White box testing is the software testing method in which internal structure is being known to tester who is going to test the software.
2	Implementation Knowledge is not required to carry out Black Box Testing.	Implementation Knowledge is required to carry out White Box Testing.
3	Programming Knowledge is not required to carry out Black Box Testing.	Programming Knowledge is required to carry out White Box Testing.
4	Testing is applicable on higher levels of testing like System Testing, Acceptance testing.	Testing is applicable on lower level of testing like Unit Testing, Integration testing.
5	Black box testing means functional test or external testing.	White box testing means structural test or interior testing.
6	In Black Box testing is primarily concentrate on the functionality of the system under test.	In White Box testing is primarily concentrate on the testing of program code of the system under test like code structure, branches, conditions, loops etc.
7	The main aim of this testing to check on what functionality is performing by the system under test.	The main aim of White Box testing to check on how System is performing.

8	Black Box testing can be started based on Requirement Specifications documents.	White Box testing can be started based on Detail Design documents.
9	The Functional testing, Behavior testing, Close box testing is carried out under Black Box testing, so there is no required of the programming knowledge.	The Structural testing, Logic testing, Path testing, Loop testing, Code coverage testing, Open box testing is carried out under White Box testing, so there is compulsory to know about programming knowledge.

References:

http://www.softwaretestingclass.com

http://www.guru99.com/

http://istqbexamcertification.com

https://smartbear.com