**BLOCK BOX TESTING:**

**What is Black Box Testing?**

Black Box Testing is also known as behavioral, opaque-box, closed-box, specification-based or eye-to-eye testing.

It is a Software Testing method that analyzes the functionality of a software/application without knowing much about the internal structure/design of the item that is being tested and compares the input value with the output value.

**The main focus of Black Box Testing is on the functionality of the system as a whole.** The term **‘Behavioral Testing’** is also used for Black Box Testing.

EX:***Most of us perform Black Box Testing every day!***

Whether we have learned or not, we all have performed Black box Testing many times in our day to day life!!

From the name itself we can probably understand that it implicates interacting with the system that you are testing as a mystery box. **It means that you are not knowledgeable enough about the internal working of the system but you know how it should behave.**

If we take an **example** to test our car or bike, we always drive it to make sure that it doesn’t behave in an unusual way.

### Types of Black Box Testing

Practically, there are several types of Black Box Testing that are possible, but if we consider a major variant of it then only the below mentioned are the two fundamental ones.

#### **#1) Functional Testing**

This testing type deals with the functional requirements or specifications of an application. Here, different actions or functions of the system are being tested by providing the input and comparing the actual output with the expected output.

**For example**, when we test a Dropdown list, we click on it and verify if it expands and all the expected values are showing in the list.

**Few major types of Functional Testing are:**

* Smoke Testing
* Sanity Testing
* Integration Testing
* System Testing
* Regression Testing
* User Acceptance Testing

#### **#2) Non-Functional Testing**

Apart from the functionalities of the requirements, there are even several non-functional aspects that are required to be tested to improve the quality and performance of the application.

**Few major types of Non-Functional Testing include:**

* Usability Testing
* Load Testing
* Performance Testing
* Compatibility Testing
* Stress Testing
* Scalability Testing

**Black Box Testing Techniques**

In order to systematically test a set of functions, it is necessary to design test cases. Testers can create test cases from the requirement specification document using the following Black Box Testing techniques:

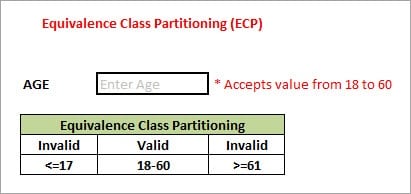
* Equivalence Partitioning
* Boundary Value Analysis
* Decision Table Testing
* State Transition Testing
* Error Guessing
* Graph-Based Testing Methods
* Comparison Testing

#### **1) Equivalence Partitioning**

This technique is also known as Equivalence Class Partitioning (ECP). **In this technique, input values to the system or application are divided into different classes or groups based on its similarity in the outcome.**

Hence, instead of using each and every input value, we can now use any one value from the group/class to test the outcome. This way, we can maintain test coverage while we can reduce the amount of rework and most importantly the time spent.

**For Example:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/03/Equivalence-Partitioning.jpg)

As present in the above image, the “AGE” text field accepts only numbers from 18 to 60. There will be three sets of classes or groups.

**Two invalid classes will be:**

a) Less than or equal to 17.

b) Greater than or equal to 61.

A valid class will be anything between 18 and 60.

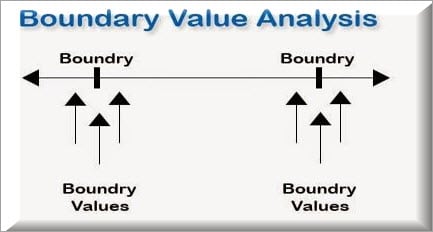
We have thus reduced the test cases to only 3 test cases based on the formed classes thereby covering all the possibilities. So, testing with any one value from each set of the class is sufficient to test the above scenario.

#### **#2) Boundary Value Analysis**

The name itself defines that in this technique, we focus on the values at boundaries as it is found that many applications have a high amount of issues on the boundaries.

**Boundary refers to values near the limit where the behavior of the system changes**. In boundary value analysis, both valid and invalid inputs are being tested to verify the issues.

**For Example:**

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/03/Boundary-Value-Analysis.jpg)

If we want to test a field where values from 1 to 100 should be accepted, then we choose the boundary values: 1-1, 1, 1+1, 100-1, 100, and 100+1. Instead of using all the values from 1 to 100, we just use 0, 1, 2, 99, 100, and 101.

#### **#3) Decision Table Testing**

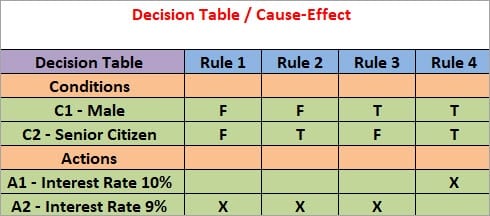
As the name itself suggests, wherever there are logical relationships like:

If  
{  
(Condition = True)  
then action1 ;  
}  
else action2; /\*(condition = False)\*/

**Then a tester will identify two outputs (action1 and action2) for two conditions (True and False).** So based on the probable scenarios a Decision table is carved to prepare a set of test cases.

**For Example:**

Take an example of XYZ bank that provides an interest rate for the Male senior citizen as 10% and 9% for the rest of the people.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2018/03/Decision-Table.jpg)

In this example condition, C1 has two values as true and false, C2 also has two values as true and false. The total number of possible combinations would then be four. This way we can derive test cases using a decision table.

#### **#4) State Transition Testing**

**State Transition Testing is a technique that is used to test the different states of the system under test.** The state of the system changes depending upon the conditions or events. The events trigger states which become scenarios and a tester needs to test them.

A systematic state transition diagram gives a clear view of the state changes but it is effective for simpler applications. More complex projects may lead to more complex transition diagrams thereby making it less effective.

#### **#5) Error Guessing**

This is a classic example of Experience-Based Testing.

In this technique, the tester can use his/her experience about the application behavior and functionalities to guess the error-prone areas. Many defects can be found using error guessing where most of the developers usually make mistakes.

**Few common mistakes that developers usually forget to handle:**

* Divide by zero.
* Handling null values in text fields.
* Accepting the Submit button without any value.
* File upload without attachment.
* File upload with less than or more than the limit size.

**#6) Graph-Based Testing Methods**

Each and every application is a build-up of some objects. All such objects are identified and the graph is prepared. From this object graph, each object relationship is identified and test cases are written accordingly to discover the errors.

**#7) Comparison Testing**

In this method, different independent versions of the same software are used to compare to each other for testing.

**WHITE BOX TESTING:**

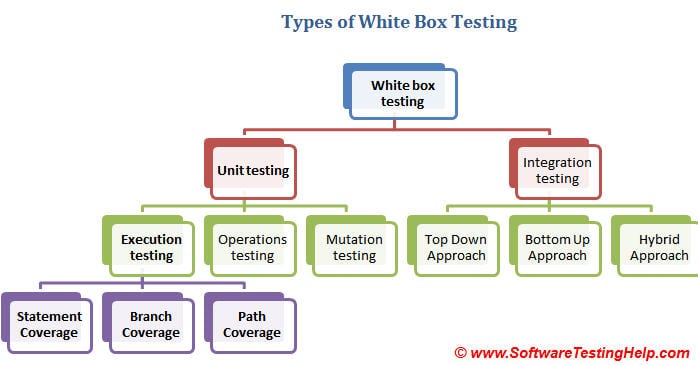
**What is White Box Testing?**

If we go by the definition, “White box testing” (also known as clear, glass box or structural testing) is a testing technique which evaluates the code and the internal structure of a program.

### Types and Techniques of White Box Testing

There are several types and different methods for each white box testing type.

See the below image for your reference.

[](https://www.softwaretestinghelp.com/wp-content/qa/uploads/2015/02/White-box-testing-types.jpg)

**3 Main White Box Testing Techniques:**

1. Statement Coverage
2. Branch Coverage
3. Path Coverage

Note that the statement, branch or path coverage does not identify any bug or defect that needs to be fixed. It only identifies those lines of code which are either never executed or remains untouched. Based on this further testing can be focused on.

Let’s understand these techniques one by one with a simple example.

**#1) Statement coverage:**

In a programming language, a statement is nothing but the line of code or instruction for the computer to understand and act accordingly. A statement becomes an executable statement when it gets compiled and converted into the object code and performs the action when the program is in a running mode.

Hence *“Statement Coverage”*, as the name itself suggests, it is the method of validating whether each and every line of the code is executed at least once.

**#2) Branch Coverage:**

“Branch” in a programming language is like the “IF statements”. An IF statement has two branches: T**rue and False**.

So in Branch coverage (also called Decision coverage), we validate whether each branch is executed at least once.

**In case of an “IF statement”, there will be two test conditions:**

* One to validate the true branch and,
* Other to validate the false branch.

Hence, in theory, Branch Coverage is a testing method which is when executed ensures that each and every branch from each decision point is executed.

**#3) Path Coverage**

Path coverage tests all the paths of the program. This is a comprehensive technique which ensures that all the paths of the program are traversed at least once. Path Coverage is even more powerful than Branch coverage. This technique is useful for testing the complex programs.