White Box Testing Techniques:

* **Statement Coverage -** This technique is aimed at exercising all programming statements with minimal tests.
* **Branch Coverage -**This technique is running a series of tests to ensure that all branches are tested at least once.
* **Path Coverage -**This technique corresponds to testing all possible paths which means that each statement and branch is covered.

Calculating Structural Testing Effectiveness:

Statement Testing = (Number of Statements Exercised / Total Number of Statements) x 100 %

Branch Testing = (Number of decisions outcomes tested / Total Number of decision Outcomes) x 100 %

Path Coverage = (Number paths exercised / Total Number of paths in the program) x 100 %

Advantages of White Box Testing:

* Forces test developer to reason carefully about implementation.
* Reveals errors in "hidden" code.
* Spots the Dead Code or other issues with respect to best programming practices.

Disadvantages of White Box Testing:

* Expensive as one has to spend both time and money to perform white box testing.
* Every possibility that few lines of code are missed accidentally.
* In-depth knowledge about the programming language is necessary to perform white box testing.

**How to Perform White Box Testing**

**White box and**[**black box testing**](http://www.softwaretestinghelp.com/black-box-testing/)**:**

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

It’s the counterpart of Black box testing.

In simple words – In Black box testing, we test the software from a user’s point of view, but in White box, we see and test the actual code. In Black box, we do testing without seeing the internal system code, but in White box we do see and test the internal code.

White box testing technique is used by both developers as well as testers. It helps them understand which line of code is actually executed and which is not. This may indicate that there is either missing logic or a typo, which eventually can lead into some negative consequences.

**Steps to perform White box testing:**

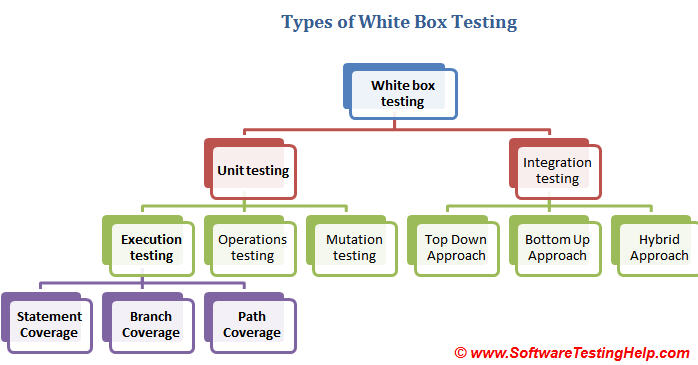
**Step #1** – Understand the functionality of the application through its source code. Having said that, it simply means that the tester must be well versed with the programming language and other tools and techniques used to develop the software.

**Step #2**– Create the tests and execute them.

When we discuss about testing, “[coverage](http://www.softwaretestinghelp.com/practical-approaches-to-improving-your-testing-by-maximizing-code-coverage/)” is the most important factor. Here I will explain how to have maximum coverage in the context of White box testing.

**Types of white box testing:**

There are different types and different methods for each white box testing type. See below image. *(click on image to enlarge)*

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/02/White-box-testing-types.jpg)

***Today, we are going to focus mainly on the execution testing types of ‘Unit testing white box technique’.***

**The three main White box testing Techniques are:**

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

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

**#1 Statement coverage**

In programming language, 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 running mode.

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

**#2 Branch Coverage**

“Branch” in programming language is like the “IF statements”. If statement has two branches: true and false.

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

In case of a “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 when executed ensures that each 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 that Branch coverage. This technique is useful for testing the complex programs.

Let’s take a simple example to understand all these white box testing techniques.

**White box testing example**

Consider below simple pseudo code:

|  |  |  |
| --- | --- | --- |
| 1 | INPUT A & B | |
| 2 | C = A + B |

|  |  |
| --- | --- |
| 3 | IF C>100 |
| 4 | PRINT “IT’S DONE” | |
|  |  | |

For **Statement Coverage** – we would need only one test case to check all the lines of code.

That means:

If I consider *TestCase\_01 to be (A=40 and B=70),* then all the lines of code will be executed

Now the question arises:

Is that sufficient?

What if I consider my Test case as A=33 and B=45?

Because Statement coverage will only cover the true side, for the pseudo code, only one test case would NOT be sufficient to test it. As a tester, we have to consider the negative cases as well.

Hence for maximum coverage, we need to consider **“Branch Coverage”**, which will evaluate the “FALSE” conditions.

In real world, you may add appropriate statements when condition fails.

So now the pseudo code becomes:

|  |  |  |
| --- | --- | --- |
| 1 | INPUT A & B | |
| 2 | C = A + B |

|  |  |
| --- | --- |
| 3 | IF C>100 |
| 4 | PRINT “IT’S DONE” | |

|  |  |
| --- | --- |
| 5 | ELSE |
| 6 | PRINT “IT’S PENDING” | |

Since Statement coverage is not sufficient to test the entire pseudo code, we would require Branch coverage to ensure maximum coverage**.**

------------

So for Branch coverage, we would require two test cases to complete testing of this pseudo code.

**TestCase\_01**: A=33, B=45

**TestCase\_02**: A=25, B=30

With this, we can see that each and every line of code is executed at least once.

**Here are the conclusions so far:**

* Branch Coverage ensures more coverage than Statement coverage
* Branch coverage is more powerful than Statement coverage,
* 100% Branch coverage itself means 100% statement coverage,
* 100 % statement coverage does not guarantee 100% branch coverage

Now let’s move on to the **Path Coverage**:

As said earlier, Path coverage is used to test the complex code snippets, which basically involves loop statements or combination of loops and decision statements.

**Consider this pseudo code:**

|  |  |  |
| --- | --- | --- |
| 1 | INPUT A & B | |
| 2 | C = A + B |

|  |  |
| --- | --- |
| 3 | IF C>100 |
| 4 | PRINT “IT’S DONE” | |

|  |  |
| --- | --- |
| 5 | END IF |
| 6 | IF A>50 | |

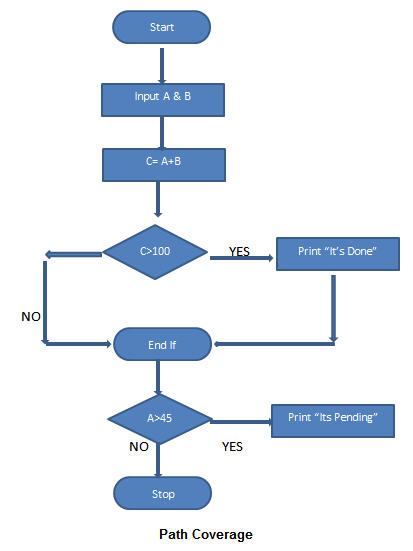
|  |  |  |
| --- | --- | --- |
| 7 | PRINT “IT’S PENDING” | |
| 8 | END IF |

Now to ensure maximum coverage, we would require 4 test cases.

How?

Simply – there are 2 decision statements, so for each decision statement we would need to branches to test. One for true and other for false condition. So for 2 decision statements, we would require 2 test cases to test the true side and 2 test cases to test the false side, which makes total of 4 test cases.

**To simplify this lets consider below flowchart of the pseudo code we have:**

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/02/Path-coverage-1.jpg)

So, In order to have the full coverage, we would need following test cases:

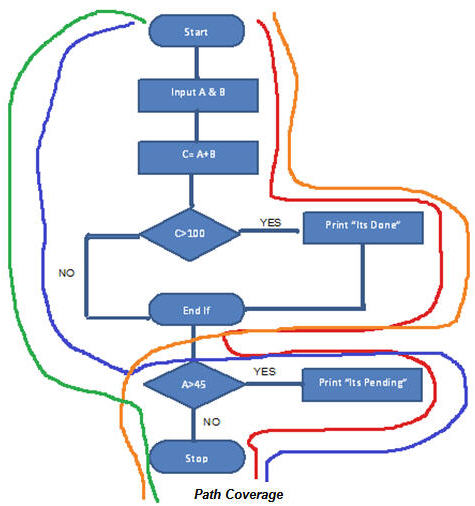
**TestCase\_01:** A=50, B=60

**TestCase\_02**: A=55, B=40

**TestCase\_03:**A=40, B=65

**TestCase\_04:**A=30, B=30

So the path covered will be:

[](http://cdn2.softwaretestinghelp.com/wp-content/qa/uploads/2015/02/Path-coverage-2.jpg)

Red Line – TestCase\_01 = (A=50, B=60)

Blue Line = TestCase\_02 = (A=55, B=40)

Orange Line = TestCase\_03 = (A=40, B=65)

Green Line = TestCase\_04 = (A=30, B=30)

**See also =>**[**Different Types of testing**](http://www.softwaretestinghelp.com/types-of-software-testing/)

**Conclusion**

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.

Relying only on black box testing is not sufficient for maximum test coverage. We need to have combination of both black box and white box testing techniques to [cover maximum defects](http://www.softwaretestinghelp.com/tips-to-find-valid-defects-in-any-application/).

If done properly, White box testing will certainly contribute to the software quality. It’s also good for testers to participate in this testing as it can provide the most “unbiased” opinion about the code.