**White Box Testing** is a testing technique in which software’s internal structure, design, and coding are tested to verify input-output flow and improve design, usability, and security. In white box testing, code is visible to testers, so it is also called Clear box testing, Open box testing, Transparent box testing, Code-based testing, logic driven, path driven , structural and Glass 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 to perform white box testing:**

* Understand the source code
* Create test cases and execute

**WhiteBox Testing Example**

Consider the following piece of code

Printme (int a, int b) { ------------ Printme is a function

int result = a+ b;

If (result> 0)

Print ("Positive", result)

Else

Print ("Negative", result)

} ----------- End of the source code

The goal of White Box testing in software engineering is to verify all the decision branches, loops, and statements in the code.

To exercise the statements in the above white box testing example, WhiteBox test cases would be

* A = 1, B = 1
* A = -1, B = -3

**White Box Testing Techniques**

A major White box testing technique is **Code Coverage analysis**. Code Coverage analysis eliminates gaps in a [Test Case](https://www.guru99.com/test-case.html) 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](https://www.guru99.com/code-coverage.html). Below are a few coverage analysis techniques a box tester can use:

**Statement Coverage**:- This technique requires every possible statement in the code to be tested at least once during the testing process of [software engineering](https://www.guru99.com/what-is-software-engineering.html).

**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.**

Following are important WhiteBox Testing Techniques:

* Statement Coverage
* Decision Coverage
* Branch Coverage
* Condition Coverage
* Multiple Condition Coverage
* Finite State Machine Coverage
* Path Coverage
* Control flow testing
* Data flow testing

**1. Code Coverage**

**2. Segment coverage:** Ensure that each code statement is executed once.

**3. Branch Coverage or Node Testing:** Coverage of each code branch in from all possible was.

**4. Compound Condition Coverage:** For multiple conditions test each condition with multiple paths and combination of the different path to reach that condition.

**5. Basis Path Testing:** Each independent path in the code is taken for testing.

**6. Data Flow Testing (DFT):** In this approach you track the specific variables through each possible calculation, thus defining the set of intermediate paths through the code. DFT tends to reflect dependencies but it is mainly through sequences of data manipulation. Inshort, each data variable is tracked and its use is verified. This approach tends to uncover bugs like variables used but not initialize, or declared but not used, and so on.

**7. Path Testing:** Path testing is where all possible paths through the code are defined and covered. It’s a time-consuming task.

**8. Loop Testing:** These strategies relate to testing single loops, concatenated loops, and

nested loops. Independent and dependent code loops and values are tested by this approach.

**Types of White Box Testing**

* Unit Testing
* Testing for memory leaks
* WB Penetration testing
* WB Mutation testing

Below is a list of top **white box testing tools.**

* [EclEmma](https://www.eclemma.org/download.html)
* [NUnit](http://nunit.org/)
* [PyUnit](https://www.guru99.com/python-unit-testing-guide.html)
* [HTMLUnit](http://htmlunit.sourceforge.net/)
* [CppUnit](https://sourceforge.net/projects/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](https://www.guru99.com/software-development-life-cycle-tutorial.html) even if GUI is not available.
* 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**

* 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 and 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.
* 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.

**Why perform White Box Testing**

To ensure:

* That all independent paths within a module have been exercised at least once.
* All logical decisions verified on their true and false values.
* All loops executed at their boundaries and within their operational bounds internal data structures validity.

To discover the following types of bugs:

* Logical error tend to creep into our work when we design and implement functions, conditions or controls that are out of the program
* The design errors due to difference between logical flow of theprogram and the actual implementation
* Typographical errors and syntax checking

**Conclusion:**

White box testing can be quite complex. The complexity involved has a lot to do with the application being tested. A small application that performs a single simple operation could be white box tested in few minutes, while larger programming applications take days, weeks, and even longer to fully test.

White box testing in software testing should be done on a software application as it is being developed after it is written and again after each modification.

**Black Box Testing** is a software testing method in which the functionalities of software applications are tested without having knowledge of internal code structure, implementation details and internal paths. Black Box Testing mainly focuses on input and output of software applications and it is entirely based on software requirements and specifications. It is also known as Behavioral Testing, opaque box, closed box, specification based or eye to eye testing.

The **term ‘Behavioral Testing’ is also used for Black Box Testing**. Behavioral test design is slightly different from the black-box test design because the use of internal knowledge isn’t strictly forbidden, but it’s still discouraged.

For Example, an operating system like Windows, a website like Google, a database like Oracle or even your own custom application. Under Black Box Testing, you can test these applications by just focusing on the inputs and outputs without knowing their internal code implementation.

**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.

There are **many types of Black Box Testing** but the following are the prominent ones –

* **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](https://www.guru99.com/regression-testing.html) is done after code fixes, upgrades or any other system maintenance to check the new code has not affected the existing code.

Here are the generic steps followed to **carry out any type of Black Box 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.

**Tools used for Black box testing** largely depends on the type of black box testing you are doing.

* For Functional/ Regression Tests you can use – [**QTP**](https://www.guru99.com/quick-test-professional-qtp-tutorial.html)**,** [**Selenium**](https://www.guru99.com/selenium-tutorial.html)
* For Non-Functional Tests, you can use – [**LoadRunner**](https://www.guru99.com/loadrunner-v12-tutorials.html)**,** [**Jmeter**](https://www.guru99.com/jmeter-tutorials.html)

**Grey Box Testing or** Gray box testing is a software testing technique to test a software product or application with partial knowledge of internal structure of the application. The purpose of grey box testing is to search and identify the defects due to improper code structure or improper use of applications.

In this process, context-specific errors that are related to web systems are commonly identified. It increases the testing coverage by concentrating on all of the layers of any complex system.

Gray Box Testing is performed for the following reason,

* 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.

To perform Gray box Testing-

* 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](https://www.guru99.com/orthogonal-array-testing.html) 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](https://www.guru99.com/testing-tools.html) 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

The test cases for grey box testing may include, GUI related, Security related, Database related, Browser related, Operational system related, etc.

**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.

**Summary:**

* The overall cost of system defects can be reduced and prevented from passing further with Grey box testing
* Grey box testing is suited more for GUI, [Functional Testing](https://www.guru99.com/functional-testing.html), security assessment, web applications, web-services, etc.
* Techniques used for Grey box Testing
  + Matrix Testing
  + [Regression Testing](https://www.guru99.com/regression-testing.html)
  + OAT or Orthogonal Array Testing
  + Pattern Testing