**Black Box Testing**

Black box testing is a testing technique in which the internal workings of the software are not known to the tester. The tester only focuses on the input and output of the software.

Prerequisite - [Software Testing | Basics](https://www.geeksforgeeks.org/software-testing-basics/)

**Table of Content**

* [What is Black Box Testing?](https://www.geeksforgeeks.org/software-engineering-black-box-testing/?ref=lbp#what-is-black-box-testing)
* [Types Of Black Box Testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/?ref=lbp#types-of-black-box-testing)
* [Advantages of Black Box Testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/?ref=lbp#advantages-of-black-box-testing)
* [Disadvantages of Black Box Testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/?ref=lbp#disadvantages-of-black-box-testing)
* [Black Box and White Box Testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/?ref=lbp#black-box-and-white-box-testing)
* [Grey Box Testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/?ref=lbp#grey-box-testing)
* [Ways of Black Box Testing Done](https://www.geeksforgeeks.org/software-engineering-black-box-testing/?ref=lbp#ways-of-black-box-testing-done)
* [Tools Used for Black Box Testing:](https://www.geeksforgeeks.org/software-engineering-black-box-testing/?ref=lbp#tools-used-for-black-box-testing)
* [What can be identified by Black Box Testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/?ref=lbp#what-can-be-identified-by-black-box-testing)
* [Features of black box testing](https://www.geeksforgeeks.org/software-engineering-black-box-testing/?ref=lbp#features-of-black-box-testing)

**What is Black Box Testing?**

Black-box testing is a type of software testing in which the tester is not concerned with the internal knowledge or implementation details of the software but rather focuses on validating the functionality based on the provided specifications or requirements.

**Types Of Black Box Testing**

The following are the several categories of black box testing:

1. [**Functional Testing**](https://www.geeksforgeeks.org/software-testing-functional-testing/)
2. [**Regression Testing**](https://www.geeksforgeeks.org/software-engineering-regression-testing/)
3. [**Nonfunctional Testing (NFT)**](https://www.geeksforgeeks.org/software-testing-non-functional-testing/)

**Functional Testing:**

Functional testing is defined as a type of testing that verifies that each function of the software application works in conformance with the requirement and specification. This testing is not concerned with the source code of the application. Each functionality of the software application is tested by providing appropriate test input, expecting the output, and comparing the actual output with the expected output. This testing focuses on checking the user interface, APIs, database, security, client or server application, and functionality of the Application Under Test. Functional testing can be manual or automated. It determines the system’s software functional requirements.

**Regression Testing:**

Regression Testing is the process of testing the modified parts of the code and the parts that might get affected due to the modifications to ensure that no new errors have been introduced in the software after the modifications have been made. Regression means the return of something and in the software field, it refers to the return of a bug. It ensures that the newly added code is compatible with the existing code. In other words, a new software update has no impact on the functionality of the software. This is carried out after a system maintenance operation and upgrades.

**Nonfunctional Testing:**

Non-functional testing is a software testing technique that checks the non-functional attributes of the system. Non-functional testing is defined as a type of software testing to check non-functional aspects 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. Non-functional testing is as important as functional testing. Non-functional testing is also known as NFT. This testing is not functional testing of software. It focuses on the software’s performance, usability, and scalability.

**Advantages of Black Box Testing**

* The tester does not need to have more functional knowledge or programming skills to implement the Black Box Testing.
* It is efficient for implementing the tests in the larger system.
* Tests are executed from the user’s or client’s point of view.
* Test cases are easily reproducible.
* It is used to find the ambiguity and contradictions in the functional specifications.

**Disadvantages of Black Box Testing**

* There is a possibility of repeating the same tests while implementing the testing process.
* Without clear functional specifications, test cases are difficult to implement.
* It is difficult to execute the test cases because of complex inputs at different stages of testing.
* Sometimes, the reason for the test failure cannot be detected.
* Some programs in the application are not tested.
* It does not reveal the errors in the control structure.
* Working with a large sample space of inputs can be exhaustive and consumes a lot of time.

**Black Box and White Box Testing**

Black box testing is a testing technique in which the internal workings of the software are not known to the tester. The tester only focuses on the input and output of the software. Whereas, White box testing is a testing technique in which the tester has knowledge of the internal workings of the software, and can test individual code snippets, algorithms and methods.

* **Testing objectives:**Black box testing is mainly focused on testing the functionality of the software, ensuring that it meets the requirements and specifications. White box testing is mainly focused on ensuring that the internal code of the software is correct and efficient.
* **Knowledge** **level**: Black box testing does not require any knowledge of the internal workings of the software, and can be performed by testers who are not familiar with programming languages. White box testing requires knowledge of programming languages, software architecture and design patterns.
* **Testing methods:**Black box testing uses methods like equivalence partitioning, boundary value analysis, and error guessing to create test cases. Whereas, white box testing uses methods like control flow testing, data flow testing and statement coverage.
* **Scope**: Black box testing is generally used for testing the software at the functional level. White box testing is used for testing the software at the unit level, integration level and system level.

**White Box Testing**

**White box testing** techniques analyze the internal structures the used data structures, internal design, code structure, and the working of the software rather than just the functionality as in black box testing. It is also called glass box testing or clear box testing or structural testing. White Box Testing is also known as transparent testing or open box testing.

White box testing is a software testing technique that involves testing the internal structure and workings of a software application. The tester has access to the source code and uses this knowledge to design test cases that can verify the correctness of the software at the code level.

White box testing is also known as structural testing or code-based testing, and it is used to test the software’s internal logic, flow, and structure. The tester creates test cases to examine the code paths and logic flows to ensure they meet the specified requirements.

*Prerequisite –*[*Software Testing | Basics*](https://www.geeksforgeeks.org/software-testing-basics/)

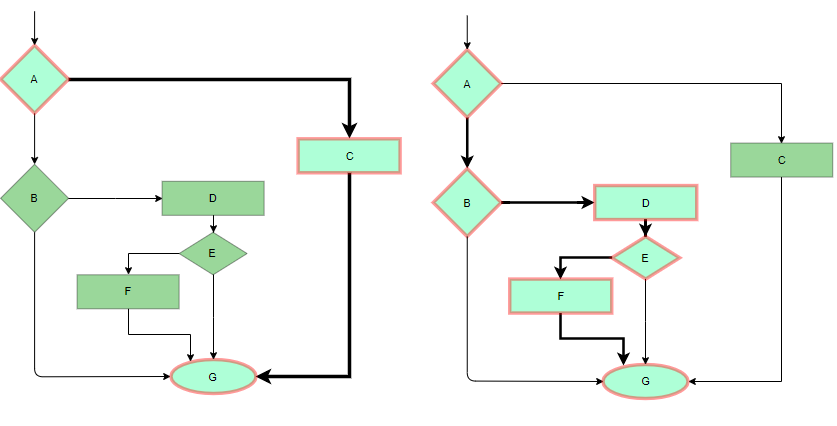
**Process of White Box Testing**

1. **Input:** Requirements, Functional specifications, design documents, source code.
2. **Processing:** Performing risk analysis to guide through the entire process.
3. **Proper test planning:** Designing test cases to cover the entire code. Execute rinse-repeat until error-free software is reached. Also, the results are communicated.
4. **Output:** Preparing final report of the entire testing process.

**Testing Techniques**

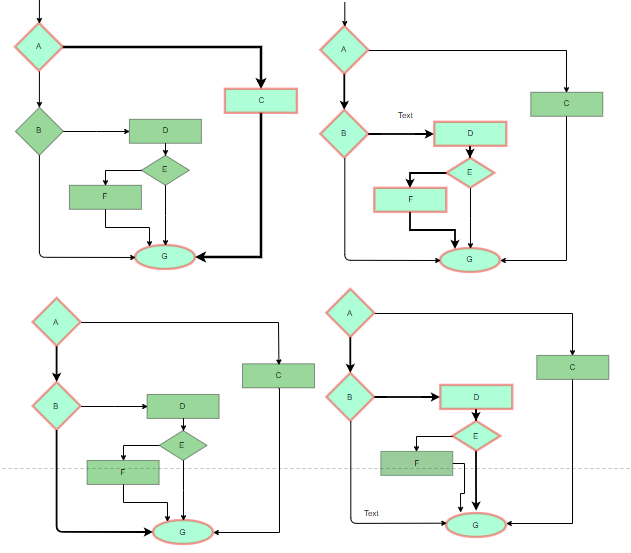
**1. Statement Coverage**

In this technique, the aim is to traverse all statements at least once. Hence, each line of code is tested. In the case of a flowchart, every node must be traversed at least once. Since all lines of code are covered, it helps in pointing out faulty code.



*Statement Coverage Example*

**2. Branch Coverage:**

In this technique, test cases are designed so that each branch from all decision points is traversed at least once. In a flowchart, all edges must be traversed at least once. 

*4 test cases are required such that all branches of all decisions are covered, i.e, all edges of the flowchart are covered*

**3. Condition Coverage**

In this technique, all individual conditions must be covered as shown in the following example:

* READ X, Y
* IF(X == 0 || Y == 0)
* PRINT ‘0’
* #TC1 – X = 0, Y = 55
* #TC2 – X = 5, Y = 0

**4. Multiple Condition Coverage**

In this technique, all the possible combinations of the possible outcomes of conditions are tested at least once. Let’s consider the following example:

* READ X, Y
* IF(X == 0 || Y == 0)
* PRINT ‘0’
* #TC1: X = 0, Y = 0
* #TC2: X = 0, Y = 5
* #TC3: X = 55, Y = 0
* #TC4: X = 55, Y = 5

**5. Basis Path Testing**

In this technique, control flow graphs are made from code or flowchart and then Cyclomatic complexity is calculated which defines the number of independent paths so that the minimal number of test cases can be designed for each independent path. **Steps:**

* Make the corresponding control flow graph
* Calculate the cyclomatic complexity
* Find the independent paths
* Design test cases corresponding to each independent path
* V(G) = P + 1, where P is the number of predicate nodes in the flow graph
* V(G) = E – N + 2, where E is the number of edges and N is the total number of nodes
* V(G) = Number of non-overlapping regions in the graph
* #P1: 1 – 2 – 4 – 7 – 8
* #P2: 1 – 2 – 3 – 5 – 7 – 8
* #P3: 1 – 2 – 3 – 6 – 7 – 8
* #P4: 1 – 2 – 4 – 7 – 1 – . . . – 7 – 8

**6. Loop Testing**

Loops are widely used and these are fundamental to many algorithms hence, their testing is very important. Errors often occur at the beginnings and ends of loops.

* **Simple loops:** For simple loops of size n, test cases are designed that:

1. Skip the loop entirely
2. Only one pass through the loop
3. 2 passes
4. m passes, where m < n
5. n-1 ans n+1 passes

* **Nested loops:** For nested loops, all the loops are set to their minimum count, and we start from the innermost loop. Simple loop tests are conducted for the innermost loop and this is worked outwards till all the loops have been tested.
* **Concatenated loops:** Independent loops, one after another. Simple loop tests are applied for each. If they’re not independent, treat them like nesting.

**White Testing is performed in 2 Steps**

1. Tester should understand the code well
2. Tester should write some code for test cases and execute them

**Tools required for White box testing:**

* PyUnit
* Sqlmap
* Nmap
* Parasoft Jtest
* Nunit
* VeraUnit
* CppUnit
* Bugzilla
* Fiddler
* JSUnit.net
* OpenGrok
* Wireshark
* HP Fortify
* CSUnit

**Features of White box Testing**

1. **Code coverage analysis:** White box testing helps to analyze the code coverage of an application, which helps to identify the areas of the code that are not being tested.
2. **Access to the source code:** White box testing requires access to the application’s source code, which makes it possible to test individual functions, methods, and modules.
3. **Knowledge of programming languages:** Testers performing white box testing must have knowledge of programming languages like Java, C++, Python, and PHP to understand the code structure and write tests.
4. **Identifying logical errors:** White box testing helps to identify logical errors in the code, such as infinite loops or incorrect conditional statements.
5. **Integration testing:**White box testing is useful for integration testing, as it allows testers to verify that the different components of an application are working together as expected.
6. **Unit testing:** White box testing is also used for unit testing, which involves testing individual units of code to ensure that they are working correctly.
7. **Optimization of code:**White box testing can help to optimize the code by identifying any performance issues, redundant code, or other areas that can be improved.
8. **Security testing:** White box testing can also be used for security testing, as it allows testers to identify any vulnerabilities in the application’s code.
9. **Verification of Design:** It verifies that the software’s internal design is implemented in accordance with the designated design documents.
10. **Check for Accurate Code:** It verifies that the code operates in accordance with the guidelines and specifications.
11. **Identifying Coding Mistakes:** It finds and fix programming flaws in your code, including syntactic and logical errors.
12. **Path Examination:** It ensures that each possible path of code execution is explored and test various iterations of the code.
13. **Determining the Dead Code:** It finds and remove any code that isn’t used when the programme is running normally (dead code).

**Advantages of Whitebox Testing**

1. **Thorough Testing**: White box testing is thorough as the entire code and structures are tested.
2. **Code Optimization:** It results in the optimization of code removing errors and helps in removing extra lines of code.
3. **Early Detection of Defects:** It can start at an earlier stage as it doesn’t require any interface as in the case of black box testing.
4. **Integration with SDLC:** White box testing can be easily started in Software Development Life Cycle.
5. **Detection of Complex Defects:** Testers can identify defects that cannot be detected through other testing techniques.
6. **Comprehensive Test Cases:** Testers can create more comprehensive and effective test cases that cover all code paths.
7. Testers can ensure that the code meets coding standards and is optimized for performance.

**Disadvantages of White box Testing**

1. **Programming Knowledge and Source Code Access:** Testers need to have programming knowledge and access to the source code to perform tests.
2. **Overemphasis on Internal Workings:** Testers may focus too much on the internal workings of the software and may miss external issues.
3. **Bias in Testing:** Testers may have a biased view of the software since they are familiar with its internal workings.
4. **Test Case Overhead:** Redesigning code and rewriting code needs test cases to be written again.
5. **Dependency on Tester Expertise:** Testers are required to have in-depth knowledge of the code and programming language as opposed to black-box testing.
6. **Inability to Detect Missing Functionalities:** Missing functionalities cannot be detected as the code that exists is tested.
7. **Increased Production Errors:** High chances of errors in production.