

# [Black box and white box comparison computer science](https://assignbuster.com/black-box-and-white-box-comparison-computer-science/)

Testing is an important activity for checking the correctness of system implementations. We need to find out bugs in the system on unit level as well as on the integration or system level of testing. At unit level, we use white box testing approach to find bugs in the developed code whereas at integration or system level we use black box testing approach to validate the system functionality. In this paper, we will focus on these two widely used black box and white box approaches of software testing. We will compare various features of both approaches and find out which approach is useful on different stages of testing to find bugs in the softwares.

Software testing is an important component of software engineering field and very important technique for improving software quality. There are two main techniques for software testing, white box testing and black box testing. Black box testing deals with the functionality of the system and is based on the software requirements. Black box testing is performed in user perspective while giving some input data and getting the valid output. White box testing is concerned with the inner design and structure of the system. White box testing is very helpful to find bugs in the softwares at initial level of testing.

A couple of papers describe the black box and white box testing techniques while highlighting different features of both techniques [1] [2]. These papers also tell the advantages and disadvantages of both approaches and explained on which stages these approaches can be used. The papers also mentioned the testing cost and the tester skills for both the testing types.

In this paper, we will look into detail in black box and white box testing. We will highlight their main technique and different features for both approaches. Based on their features, we will take a comparison of both techniques and we will analyze which technique is feasible for testing on certain stages of the software testing. As white box testing is used on the unit level and black box testing is used in integration or system level testing. So, using only white box or black box testing is not sufficient to find all bugs in the software. Both approaches have their own importance in testing softwares and finding bugs on different levels of the software testing.

This paper is organized as follows. Section 2 of this paper describes black box testing method in detail. Section 3 includes the detail of white box testing approach used to test software projects. Section 4 describes a comparison of both the testing approaches white box and black box testing. Section 5 concludes the paper.

BLACK BOX TESTING

“ Black box testing treats the system as aA “ black-box”, so it doesn’t explicitly use Knowledge of the internal structure or code.” [1] Therefore black box testing is performed while giving inputs and getting the expected output. Black box testing verifies the correctness of the software’s behaviour and is referred as behaviour coverage. The focus of the black box testing is on functionality of the system. Black box testing requirement includes software requirements, uses cases only executable program and its data. These requirements are usually met during the middle phase of the development lifecycle when large pieces of code begin to operate together.

“ Black box testing is performed when we do not have the source code. As the programming completes, it’s packaged as larger components of the code. This package is built by compiling, linking or binding the executable codes to demonstrate the collective code execution behaviour with inputs and outputs.” [3] Here is the Black box testing diagram.

Input Test dataSystemOutput Test results

Diagram . 1, Black box Testing

Black Box Testing Techniques

Developers usually focus on the positive testing by validating the expected behaviour of the code. Tester add values to black box testing by executing both positive and negative testing because the tester knows that most of the defects are found from the unexpected behaviour. Here are the techniques of Black box testing.

Equivalence class technique

In this technique the software data is divided into partitions of data and test cases are derived from it. This technique ensures that test cases are derived to cover partitioned data. According to this method one test case is enough for each partition to check the behaviour of the program. By using equivalence class technique test cases can be reduced because all test cases of a partition will not find any new faults in the program. The recommended black box approach for selecting equivalence class values includes values at the beginning, in the middle and end of the data range.

We can understand this technique more with the help of the example. [3] Let us assume that a security system have to validate 5000 log on IDs when in production. Now tester can validate each logon IDs or on the other hand tester knows that there are only three categories of IDs clerks, supervisors, managers. Now sample from each category will be sufficient for the validation. Let us consider 50 logon IDs from each category. During validation if any of the 50 logon IDs for a given category is not accepted, increase the sample size of logon ID’s for that category and continue testing. If the initial samples of 150 IDs for all three categories are correctly accepted then the tester has achieved the desired test results using only 3% of the possible test data.

Boundary Value Analysis technique

In this technique the analysis is extended of beginning and ending inputs value possibilities of an equivalence class. The boundary values are around the beginning and ending of a portion and functional errors from input and output data occurs around these boundaries. The boundary analysis begins by identifying smallest value increment in a specific equivalence class. This increment of smallest value is called boundary value epsilon. [3] It is used to calculate +/- value around the beginning and ending values in an equivalence class. If we take the above example then the boundary value for each category would be 49, 50, 51 for 50 value and 149, 150, 151 for 150 equivalence value. Equivalence classes remain smaller even by the addition of boundary test values.

Expected result coverage Technique

The expected result coverage technique focus on output test values for related input values. To get the expected results we need to find the business rules in the application requirement. The difference between expected results and actual results for any combination of input should cause further analysis to determine if the difference is faulty test design, unclear ambiguous rules or program error. [3]

Error Guessing

In this technique, Tester judge where bugs or errors can be hidden. Tester tests the application with the past experience and creates test cases to validate the different tasks. There is no specific tool used for this technique.

Advantages of Black Box testing

Here are the advantages to black box testing:

Ease of Use

Black Box testing is not involved in the inner functionality of the application so it is easy for the tester to create test cases by simple working through the application. Black box testers don’t need high skills in programming rather low skilled persons can do black box testing.

Quicker Test Case Development

Test cases can be developed quickly because tester concerns only with the user interface. There is no need to identify the internal paths that may be involves in a specific process. The tester will cover the UI paths that a user may take so it makes quicker test development. [2]

Simplicity

Using black box testing even complex applications can be tested, that focuses on valid and invalid inputs and ensuring that the correct out puts are received.

Issues of Black box testing

The Followings are the few issues of black box testing approach.

Fragility

User interface interaction makes script fragile. The reason is because the UI may not be rendered constantly from time to time on different platforms. Unless the tool is capable dealing with difference in UI rendering, it is likely that test cases will fail to execute properly. [2]

Lack of introspection

One of the drawbacks of the black box testing is that it cannot look into inner logic of the application unlike the white box testing. Therefore black box testing can never test an application fully.

Failure reason

Another issue with the black box testing, it does not found the reason of the failure. It does not tell what causes the failure in the system. So pure black box testing does not guarantee 100% coverage of the tested component.

WHITE BOX TESTING

The white box testing involves the inner logic of the program and the test cases are constructed from the source code of the program. It is based on the knowledge of how the system is implemented. White box testing includes analyses of data flow, control flow, information flow, coding practice, exception and error handling within the system, to test the intended and unintended behaviour of the system. [1] White box testing requires access to the source code. When the code is developed, white box testing can be performed any time in the life cycle but it is good practice to perform white box testing during the unit level testing. The first step in white box testing is to analyze available design documentation, source code knowing what make software secure is a fundamental requirement. Second, to create tests that exploits software. Third thing is to perform testing effectively; testers need to know the techniques and tools used for the white box testing.

White box testing normally do by the developers as an extension of code debugging activity. The developers usually focus on making the code work, which gives them the tendency to debug only the code they know works. Software testers add value to developer debugging activity by helping the developer plan and debug more of the code than usual. The more the logic test coverage attain while debugging, the fewer the defect will be discovered later by other kinds of testing. The earlier these defects can be discovered, the less expensive they are to correct. “ Research has not yet produced a white box approach that guarantees 100% logic test coverage for all situations”. [3] Here is the diagram for white box testing.

Static Analysis

Manual

Automated

Inspection

Walkthrough

Symbolic Execution

Static Verification

Diagram . 2, White Box Testing

White Box Testing Techniques:

White box testing verifies the correctness of the software’s statements, code paths, conditions, loops and data flow. The prerequisites of white box testing include the software requirements, use cases, the executable program, its data and its source code. Here are the techniques used for white box testing.

Statement Coverage Technique

This technique focuses on determining what percentage of the source code lines in a program has been executed. If there are 5, 000 lines of source code in a program and you can determine manually or with a tool that you have executed 4, 537 lines of source code, then you have achieved 90. 7% statement coverage. A hypothesis in this context is that the higher the source code test coverage, the fewer will be the defects found later. The practical conclusion is the that new, unexecuted code lines are just a software time bomb waiting to explode at the most inopportune moment in production. [3] The question is not “ if” they will explode, only the question is “ when”.

Branch coverage technique

This technique focus on determining what percentage of source code branch (true/false) logic in a program has been executed. Suppose if there are 1, 500 source code branch points in a program and we can determine manually or with a tool that we have executed 1, 145 branches by counting true and false branch executions separately then we have achieved 76. 3% branch point coverage. [3] The co-occurrence of unexecuted branches with unexecuted statements is found most often in untested error recovery logic. Testing simple condition branches before the compound condition branches requires fewer initial test actions. The developer needs to choose a test value that will force a true branch and any test value that will force a false branch, just two test values per branch.

Compound Condition Coverage Technique

This technique extends the branch coverage technique to branches with compound conditions, ones that contain combinations of Boolean operators AND, OR and NOT along with pairs of parenthesis, possible nested. In this technique, the challenge is to identify all the test value combinations that will evaluate to true and false for every simple condition and every Boolean combination of simple condition.

Path Coverage Technique

This technique focus on determining what percentage of source code paths in a program have been traversed completely. “ A source code path is the sequence of program statements from the first executable statement through a series of arithmetic, replacement input/out, branch and looping statements to a return/stop/end/exit statement.” [3] If there are 943 different paths through a program and we can determine manually or with a tool that we have executed 766 of them, then we have achieved 81% path coverage. The underlying hypothesis is that the higher the path test coverage, the fewer will be the defect found later.

Loop Coverage Technique

This technique focus on determining what percentage of source code loop in a program has been cycled completely. In the programming languages, there are several loops like DO, FOR, WHILE and UNTIL. Some of the loops are a clever construct of IF statements and subsequent returns to these IF statements. The objective of the loop testing is to force the program through the loop zero time, one time, n/2 times (n is terminal loop value) n times and n+1 times. The one-time loop, n/2-time loop and n-time loop validate expected loop response at the beginning, middle and end of the longest loop. The zero-time and n+1 time loop tests for unexpected and inappropriate looping conditions.

Advantages of White Box Testing

Here is the list of the few common advantages of white box testing.

Introspection

“ Introspection is the ability to look inside the application and tester can identify objects programmatically.” [2] It is helpful when UI changes frequently or the UI yet not known, It allows testing to proceed. In some situations it decreases the fragility of the test script provided the names of an object do not change.

Stability

White box testing delivers greater stability and reusability of test cases if the objects that include an application never change.

Thoroughness

White box testing is very helpful technique, where it is essential to know that every path has been thoroughly tested, that every possible internal interaction has been examined. [2] This technique gives tester the ability to be more thorough in terms of how much an application they can test.

Issues with White box testing

The issues with the white box testing are listed below:

Complexity:

The tester must have detailed programmatic knowledge of the application in order to test with white box testing technique. White box testing involves the internal logic of the application’s code so this complexity requires highly skilled testers to develop test cases.

Fragility

Although introspection overcomes the issue of application changes breaking test scripts but the reality is that often the names of objects changes during product development. White box testing requires test scripts to be tightly tied to the underlying code of an application means that changes to the code will often cause white box test scripts to break. [2] Then it introduces a high degree of script maintenance into the testing process.

Integration

Due to the highly integrated nature of white box testing tools many do not provide support for more than one platform usually windows. In those companies where companies have applications that run on other platforms, they either need to use different tool or need to test manually. [2]

COMPARING BLACK BOX AND WHITE BOX TESTING FEATURES

Black box testing based on requirements, therefore also called behavioral or functional testing technique, whereas white box testing is based on the internal logic and structure of the project therefore also called structural testing.

Black box testing creates test data without any aspect of the software structure, so this technique does not consider how the test object has been implemented but what its requirements are. On the other hand, in white box testing, the tests are derived from the structure of the tested object and tests are based on the actual implementation. [5]

Black box testing methodology is used in integration level or system level because it does not need to look into source code but just need to execute a system by using input data and output result. On the other hand, white box testing is used for logical and analytic test in unit test level.

Black box tester can be less technical in programming and has no “ bonds” with the code so tester perception is very simple white box tester needs programming skills to identify and test all paths through the software. [3]

In black box testing, test case can be designed as soon as the functional specifications are complete whereas in white box testing, test can be designed as the program code is developed.

Black box tester used to verify contradiction in actual system and the specifications whereas white box tester chooses test case inputs to exercise paths through the code and determines the appropriate output. It reveals all errors hidden in the code. [6]

Black box testing focuses on results and the user experience whereas white box testing focuses on the internal working and making sure that the application works as efficiently as possible.

White box testing is time consuming especially when applied to the whole system but it directly identifies more bugs in the software. In black box testing, inputs need to be from large sample space to find expected results.

There are some bugs that cannot be found using only black box or only white box testing.

In black box, it is difficult to identify all possible inputs in the limited testing time so writing test cases is slow and difficult. In white box testing, it is not possible to test each and every path of the loop in a program so exhaustive testing is impossible for large systems.

In black box testing some of the code may not be reachable without extraordinary measures, whereas white box testing does not tell if all the functionality has been implemented in the program.

The comprehensive test coverage may be achieved only when black box and white box testing techniques are combined because both methodologies include various benefits of application testing.

CONCLUSION

As discussed above, Black box testing validates the functionality of the software and ensures that the system is working correctly according to the user requirements. Black box testing is less expensive as compare to white box testing and can be performed by the less technical individuals but Black box approach does not cover the comprehensive testing of the system. On the other hand, White box testing checks the internal structure, inner loops, conditions and logic of the code to verify the working of the code. White box testing is expensive and is performed by the skilled and technical individuals, but as the software size is getting huge, it is difficult to check out all parts of source code during integration or system level testing. As both testing methods have some pros and cons when applied on certain levels of testing. So using only one testing approach cannot guarantee bug free software.

In the conclusion, we can say that the comprehensive testing of the system may be achieved by using both of the techniques for the software testing. White box testing can be used with black box testing to improve overall test effectiveness.