Name: Resmi Mariyil

UID: 179270

Software Testing

Software testing is classified into two types of testing, which are **Manual testing and Automation testing**. Both manual and automation testing have their characteristics and approaches that make both the testing technique different from each other.The box testing approach of software testing consists of black box testing and white box testing.

White Box Testing

White box testing is also known as **glass box testing, structural testing, clear box testing, open box testing and transparent box testing**.

It tests internal coding and infrastructure of a software focus on checking of predefined inputs against expected and desired outputs. It is based on inner workings of an application and revolves around internal structure testing. The primary goal of white box testing is to focus on the flow of inputs and outputs through the software and strengthening the security of the software.

The term 'white box' is used because of the internal perspective of the system. The clear box or white box or transparent box name denote the ability to see through the software's outer shell into its inner workings.

Developers do white box testing, the developer will test every line of the code of the program. The developers perform the White-box testing and then send the application or the software to the testing team, where they will perform the black box testing and verify the application along with the requirements and identify the bugs and sends it to the developer.

The developer fixes the bugs and does one round of white box testing and sends it to the testing team. Here, fixing the bugs implies that the bug is deleted, and the particular feature is working fine on the application.

Here, the test engineers will not include in fixing the defects for the following reasons:

- Fixing the bug might interrupt the other features. Therefore, the test engineer should always find the bugs, and developers should still be doing the bug fixes.
- If the test engineers spend most of the time fixing the defects, then they may be unable to find the other bugs in the application.

The white box testing contains various tests, which are as follows:

- Path testing
- Loop testing
- Condition testing
- Testing based on the memory perspective
- Test performance of the program
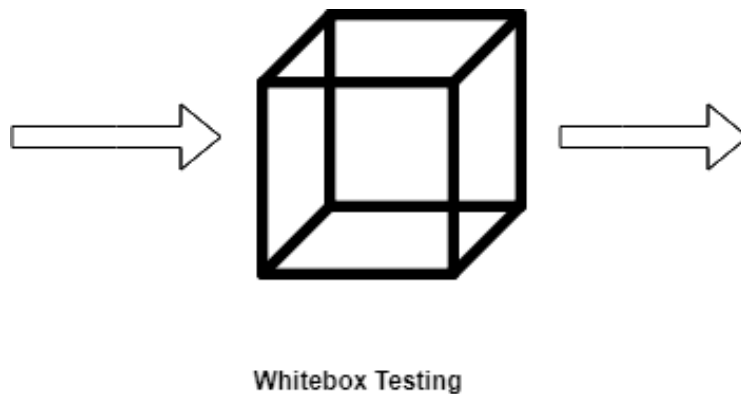
Whitebox Testing

Generic steps of white box testing

- Design all test scenarios, test cases and prioritize them according to high priority number.
- This step involves the study of code at runtime to examine the resource utilization, not accessed areas of the code, time taken by various methods and operations and so on.
- In this step testing of internal subroutines takes place. Internal subroutines such as nonpublic methods, interfaces are able to handle all types of data appropriately or not.

- o This step focuses on testing of control statements like loops and conditional statements to check the efficiency and accuracy for different data inputs.
- o In the last step white box testing includes security testing to check all possible security loopholes by looking at how the code handles security.

Reasons for white box testing

- o It identifies internal security holes.
- o To check the way of input inside the code.
- o Check the functionality of conditional loops.
- o To test function, object, and statement at an individual level.

Advantages of White box testing

- o White box testing optimizes code so hidden errors can be identified.
- o Test cases of white box testing can be easily automated.
- o This testing is more thorough than other testing approaches as it covers all code paths.
- o It can be started in the SDLC phase even without GUI.

Disadvantages of White box testing

- o White box testing is too much time consuming when it comes to large-scale programming applications.
- o White box testing is much expensive and complex.
- o It can lead to production error because it is not detailed by the developers.
- o White box testing needs professional programmers who have a detailed knowledge and understanding of programming language and implementation.

Techniques Used in White Box Testing

Data Flow Testing	Data flow testing is a group of testing strategies that examines the control flow of programs in order to explore the sequence of variables according to the sequence of events.

Control Flow Testing	Control flow testing determines the execution order of statements or instructions of the program through a control structure. The control structure of a program is used to develop a test case for the program. In this technique, a particular part of a large program is selected by the tester to set the testing path. Test cases represented by the control graph of the program.
Branch Testing	Branch coverage technique is used to cover all branches of the control flow graph. It covers all the possible outcomes (true and false) of each condition of decision point at least once.
Statement Testing	Statement coverage technique is used to design white box test cases. This technique involves execution of all statements of the source code at least once. It is used to calculate the total number of executed statements in the source code, out of total statements present in the source code.
Decision Testing	This technique reports true and false outcomes of Boolean expressions. Whenever there is a possibility of two or more outcomes from the statements like do while statement, if statement and case statement (Control flow statements), it is considered as decision point because there are two outcomes either true or false.

Examples:-

1. Data Flow Testing

```
1. read x, y;
2. if(x>y)
3.      a = x+1
4. else
5.      a = y-1
6. print a;
```
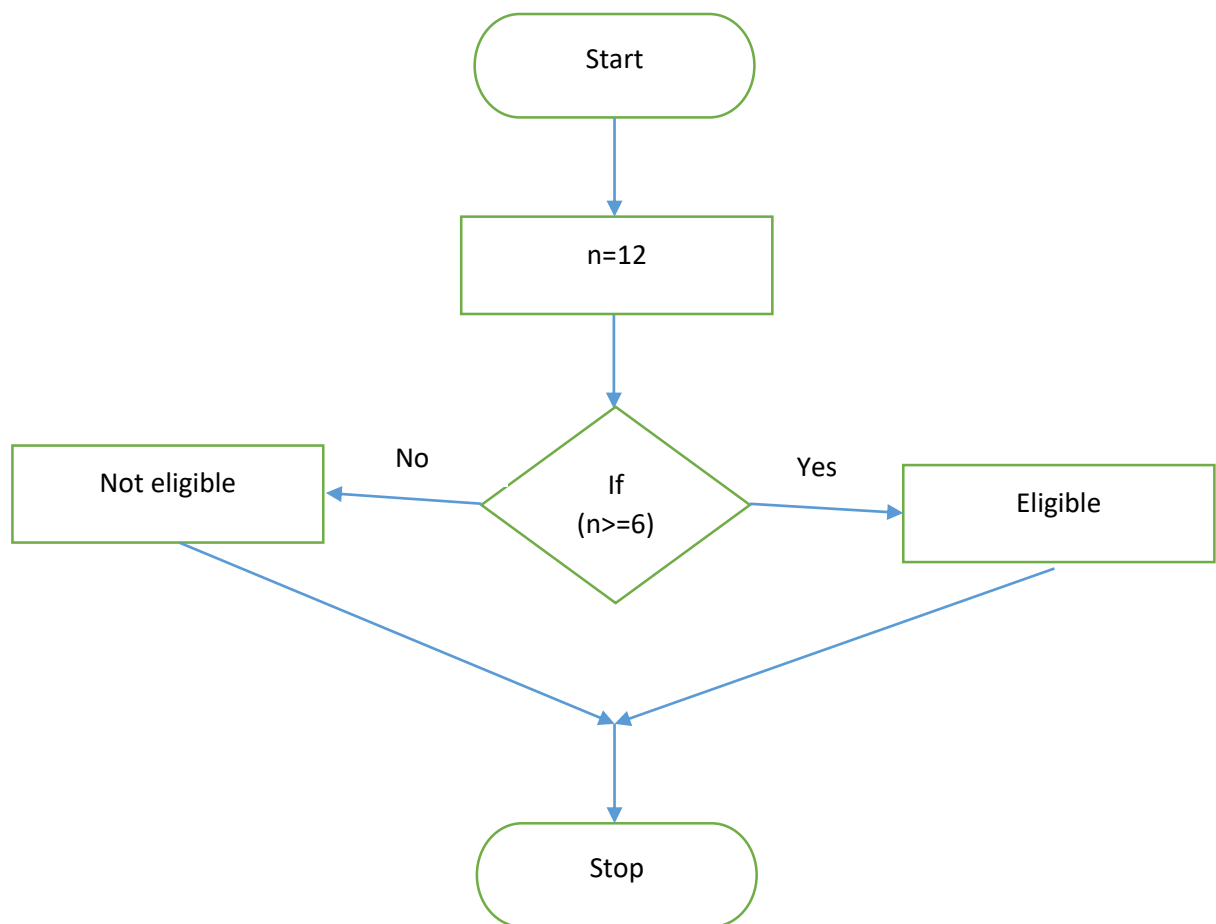
```
Association
    1, 2, t, x, y
    1, 2, f, x, y
    1, 3, x, y
    1, 4, x, y
    1, 6, a, x, y
```
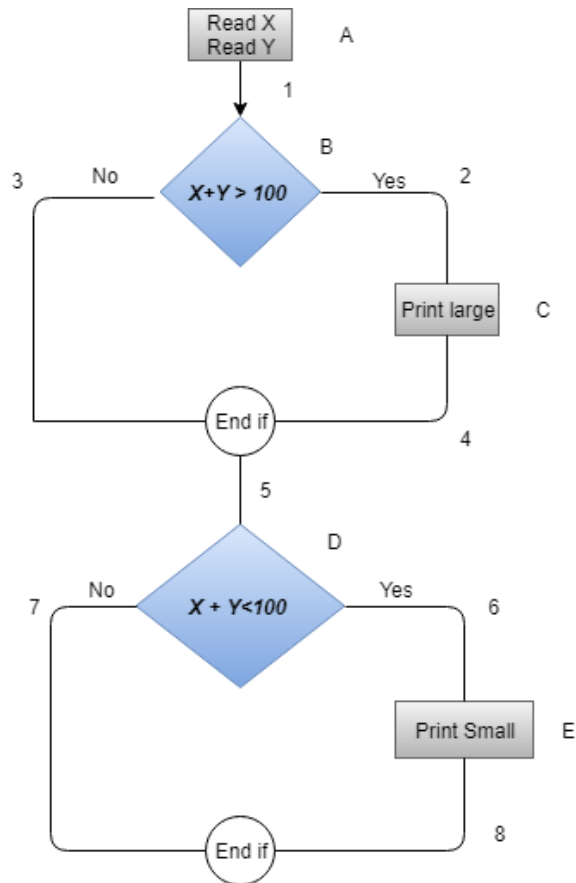
2. Control flow Testing

```
1. public class Admission{
2. public static void main(String []args){
3. int n=12;
4. if(n>=6)
5. System.out.println("You are eligible for admission");
6.  else
7. System.out.println("You are not eligible for admission");
8. }
```

```mermaid
Start
  |
n=12
  |
If (n>=6)
  No --> Not eligible
  Yes --> Eligible
  Stop
```

Start

n=12

No — Not eligible

If
(n>=6)

Yes — Eligible

Stop

3. Brach Testing

```
1. Read X
2. Read Y
3. IF X+Y > 100 THEN
4. Print "Large"
```

```
5. ENDIF
6. If X + Y<100 THEN
7. Print "Small"
8. ENDIF
```

Read X
Read Y   A

         1

              B
3   No   Yes   2
    X+Y > 100

                Print large   C

                        4
    End if

         5

              D
7   No   Yes   6
    X + Y<100

                Print Small   E

                        8
    End if

4. Statement Testing

1. print (int a, int b) {
2. int sum = a+b;
3. if (sum>0)
4. print ("This is a positive result")
5. else
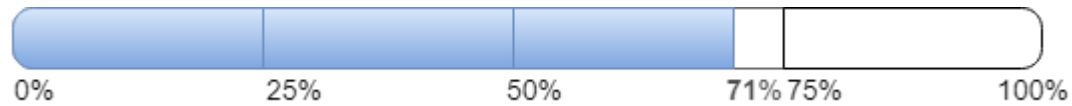6. print ("This is negative result")
7. }

Scenario
If a = 5, b = 4

1. Total number of statements = 7
2. Number of executed statements = 5

$$Statement\ coverage = \frac{Number\ of\ executed\ statements}{Total\ number\ of\ statements} * \mathbf{100}$$

Statement coverage = 5/7*100

= 500/7

= 71%

| 0% | 25% | 50% | 71% 75% | 100% |

5. Decision Testing

1. Test (**int** a)
2. {
3. If(a>4)
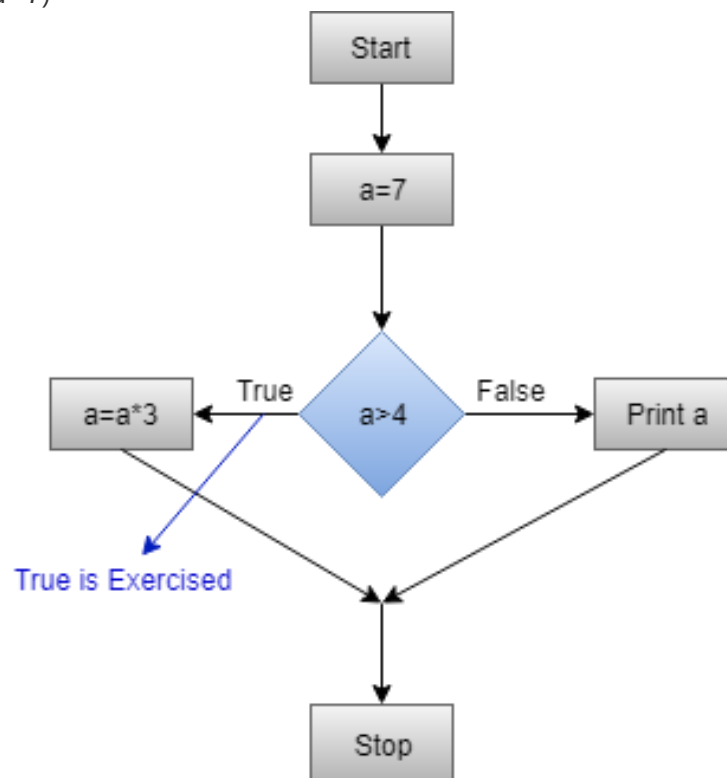4. a=a*3
5. Print (a)
6. }

Scenario

Value of a is 7 (a=7)

Calculation of Decision Coverage percent:

$$\text{Decision Coverage} = \frac{\text{Number of Decision Outcomes Exercised}}{\text{Total Number of Decision Outcomes}} *100$$

Decision Coverage = ½*100 (Only "True" is exercised)

         =100/2

         = 50

Decision Coverage is 50%

Black box testing

Black box testing is a technique of software testing which examines the functionality of software without peering into its internal structure or coding. The primary source of black box testing is a specification of requirements that is stated by the customer.

In this method, tester selects a function and gives input value to examine its functionality, and checks whether the function is giving expected output or not.

If the function produces correct output, then it is passed in testing, otherwise failed. The test team reports the result to the development team and then tests the next function. After completing testing of all functions if there are severe problems, then it is given back to the development team for correction.

Generic steps of black box testing

- o The black box test is based on the specification of requirements, so it is examined in the beginning.
- o In the second step, the tester creates a positive test scenario and an adverse test scenario by selecting valid and invalid input values to check that the software is processing them correctly or incorrectly.
- o In the third step, the tester develops various test cases such as decision table, all pairs test, equivalent division, error estimation, cause-effect graph, etc.
- o The fourth phase includes the execution of all test cases.
- o In the fifth step, the tester compares the expected output against the actual output.
- o In the sixth and final step, if there is any flaw in the software, then it is cured and tested again.

Test procedure

The test procedure of black box testing is a kind of process in which the tester has specific knowledge about the software's work, and it develops test cases to check the accuracy of the software's functionality. It does not require programming knowledge of the software. All test cases are designed by considering the input and output of a particular function. A tester knows about the definite output of a particular input, but not about how the result is arising

Test cases

Test cases are created considering the specification of the requirements. These test cases are generally created from working descriptions of the software including requirements, design parameters, and other specifications. For the testing, the test designer selects both positive test scenario by taking valid input values and adverse test scenario by taking invalid input values to determine the correct output. Test cases are mainly designed for functional testing but can also be used for non-functional testing. Test cases are designed by the testing team, there is not any involvement of the development team of software.

Techniques Used in Black Box Testing

|	
Decision Table Technique	Decision Table Technique is a systematic approach where various input combinations and their respective system behavior are captured in a tabular form. It is appropriate for the functions that have a logical relationship between two and more than two inputs.
Boundary Value Technique	Boundary Value Technique is used to test boundary values, boundary values are those that contain the upper and lower limit of a variable. It tests, while entering boundary value whether the software is producing correct output or not.
State Transition Technique	State Transition Technique is used to capture the behavior of the software application when different input values are given to the same function. This applies to those types of applications that provide the specific number of attempts to access the application.
All-pair Testing Technique	All-pair testing Technique is used to test all the possible discrete combinations of values. This combinational method is used for testing the application that uses checkbox input, radio button input, list box, text box, etc.
Cause-Effect Technique	Cause-Effect Technique underlines the relationship between a given result and all the factors affecting the result.It is based on a collection of requirements.
Equivalence Partitioning Technique	Equivalence partitioning is a technique of software testing in which input data divided into partitions of valid and invalid values, and it is mandatory that all partitions must exhibit the same behavior.
Error Guessing Technique	Error guessing is a technique in which there is no specific method for identifying the error. It is based on the experience of the test analyst, where the tester uses the experience to guess the problematic areas of the software.
Use Case Technique	Use case Technique used to identify the test cases from the beginning to the end of the system as per the usage of the system. By using this technique, the test team creates a test

| | scenario that can exercise the entire software based on the functionality of each function from start to end. |

Difference between white-box testing and black-box testing

| White-box testing	Black box testing
The developers can perform white box testing.	The test engineers perform the black box testing.
To perform WBT, we should have an understanding of the programming languages.	To perform BBT, there is no need to have an understanding of the programming languages.
In this, we will look into the source code and test the logic of the code.	In this, we will verify the functionality of the application based on the requirement specification.
In this, the developer should know about the internal design of the code.	In this, there is no need to know about the internal design of the code.

GreyBox Testing

Greybox testing is a software testing method to test the software application with partial knowledge of the internal working structure. It is a **combination of black box and white box testing** because it involves access to internal coding to design test cases as white box testing and testing practices are done at functionality level as black box testing.

BlackBox
Testing

WhiteBox
Testing
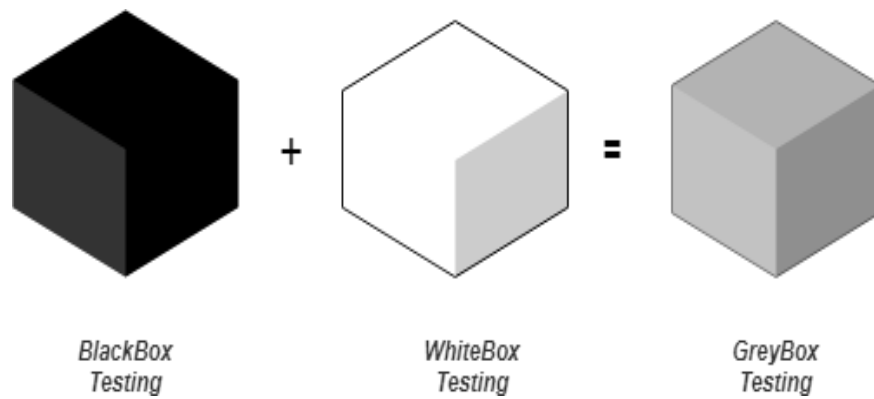
GreyBox
Testing

GreyBox testing commonly identifies context-specific errors that belong to web systems. For example; while testing, if tester encounters any defect then he makes changes in code to resolve the defect and then test it again in real time. It concentrates on all the layers of any complex software system to increase testing coverage. It gives the ability to test both presentation layer as well as internal coding structure. It is primarily used in integration testing and penetration testing.

Reasons for GreyBox testing are as follows

o It provides combined benefits of both Blackbox testing and WhiteBox testing.

o It includes the input values of both developers and testers at the same time to improve the overall quality of the product.

o It reduces time consumption of long process of functional and non-functional testing.

o It gives sufficient time to the developer to fix the product defects.

o It includes user point of view rather than designer or tester point of view.

o It involves examination of requirements and determination of specifications by user point of view deeply.
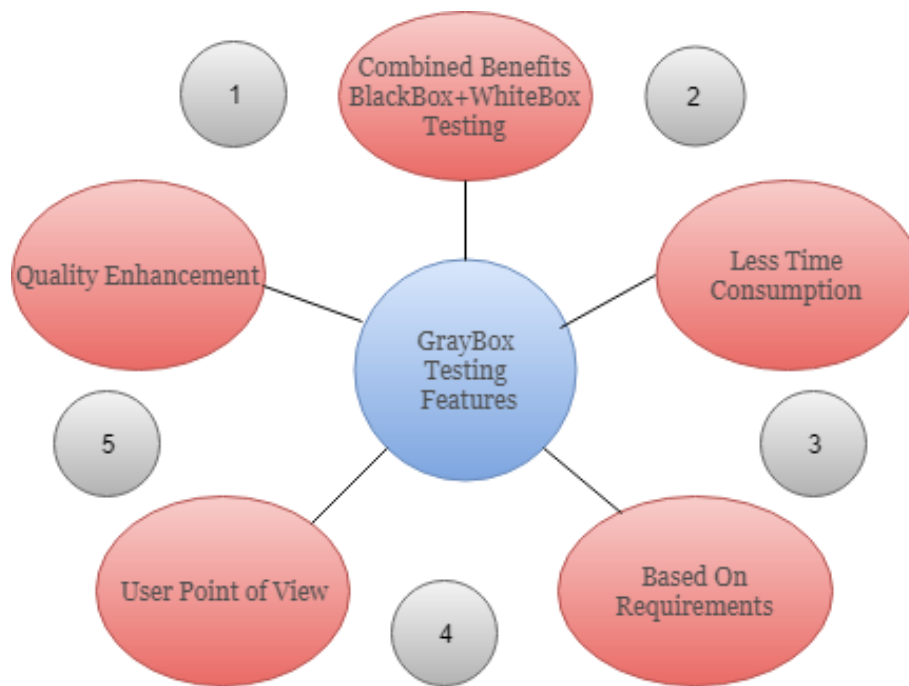
GreyBox Testing Strategy

Grey box testing does not make necessary that the tester must design test cases from source code. To perform this testing test cases can be designed on the base of, knowledge of architectures, algorithm, internal states or other high -level descriptions of the program behavior. It uses all the straightforward techniques of black box testing for function testing. The test case generation is based on requirements and preset all the conditions before testing the program by assertion method.

Generic Steps to perform Grey box Testing are:

1. First, select and identify inputs from BlackBox and WhiteBox testing inputs.
2. Second, Identify expected outputs from these selected inputs.
3. Third, identify all the major paths to traverse through during the testing period.
4. The fourth task is to identify sub-functions which are the part of main functions to perform deep level testing.
5. The fifth task is to identify inputs for subfunctions.
6. The sixth task is to identify expected outputs for subfunctions.
7. The seventh task includes executing a test case for Subfunctions.
8. The eighth task includes verification of the correctness of result.

The test cases designed for Greybox testing includes Security related, Browser related, GUI related, Operational system related and Database related testing.

Techniques of Grey box Testing

Matrix Testing

This testing technique comes under Grey Box testing. It defines all the used variables of a particular program. In any program, variable are the elements through which values can travel inside the program. It should be as per requirement otherwise, it will reduce the readability of the program and speed of the software. Matrix technique is a method to remove unused and uninitialized variables by identifying used variables from the program.

Regression Testing

Regression testing is used to verify that modification in any part of software has not caused any adverse or unintended side effect in any other part of the software. During confirmation testing, any defect got fixed, and that part of software started working as intended, but there might be a possibility that fixed defect may have introduced a different defect somewhere else in the software. So, regression testing takes care of these type of defects by testing strategies like retest risky use cases, retest within a firewall, retest all, etc.

Orthogonal Array Testing or OAT

The purpose of this testing is to cover maximum code with minimum test cases. Test cases are designed in a way that can cover maximum code as well as GUI functions with a smaller number of test cases.

Pattern Testing

Pattern testing is applicable to such type of software that is developed by following the same pattern of previous software. In these type of software possibility to occur the same type of defects. Pattern testing determines reasons of the failure so they can be fixed in the next software.

Usually, automated software testing tools are used in Greybox methodology to conduct the test process. Stubs and module drivers provided to a tester to relieve from manually code generation.