SEMESTER PROJECT

Subject: Software Quality Engineering

Submitted to: Dr. Sumaira Nazir

Section: BSSE-6th (Evening)

Dated: 14th December, 2023

Submitted by:

- **Soma Nasir (12322)**
- **Noor ud Din (12293)**
- **Talib ul Moula (12313)**

Department of Software Engineering, National University of Modern Languages, Islamabad

Fall, 2023

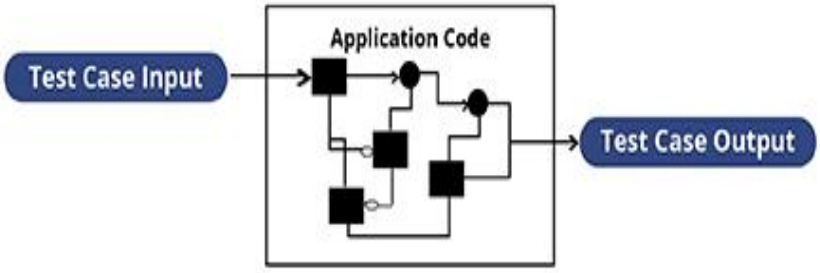
WHITE BOX TESTING

What is White Box Testing?

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, and
- Glass box testing.



White Box vs. Black Box Testing

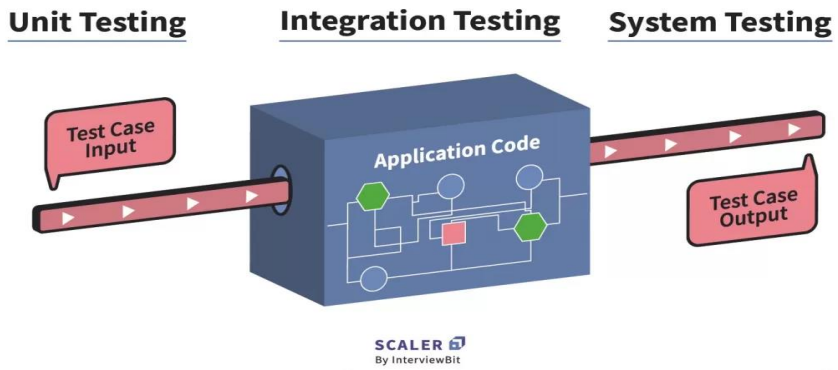
Black Box Testing	White Box Testing
• It involves testing from an external or end-user perspective.	• It is based on the inner workings of an application and revolves around internal testing.
• The Black Box name symbolizes not being able to see the inner working of the software so that only the end-user experience can be tested.	• White Box name symbolizes the ability to see through the software's outer shell into its inner working.

What do you verify in White 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 white-box 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.



Working Process of White Box Testing:

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

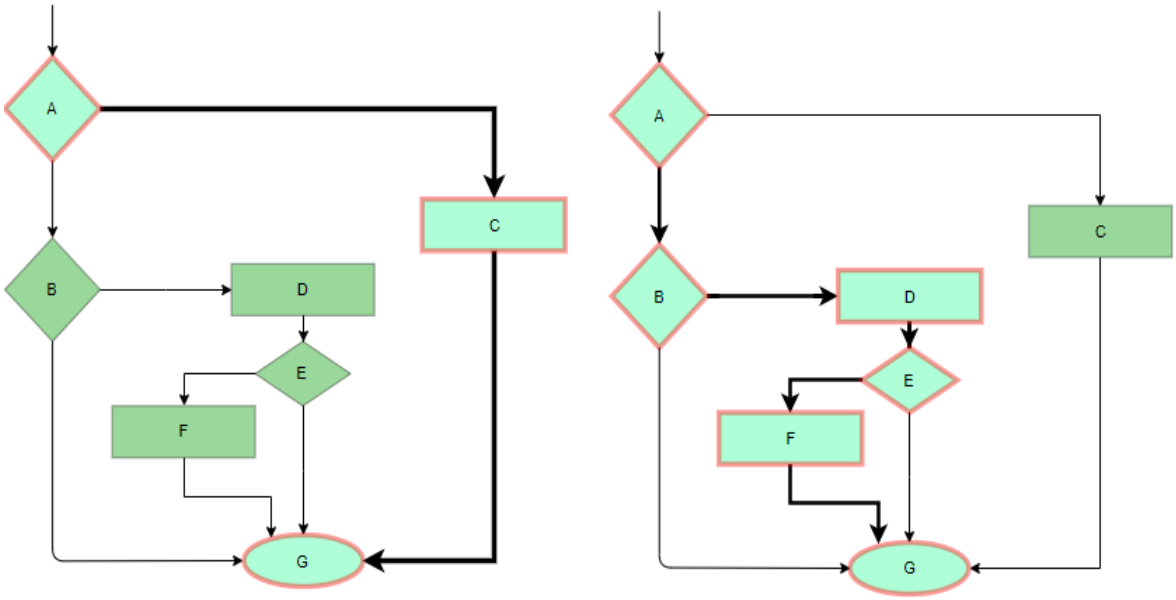
White Box Testing Techniques

1) Statement Coverage
2) Branch Coverage
3) Condition Coverage
4) Multiple Condition Coverage

5) Basis Path Testing

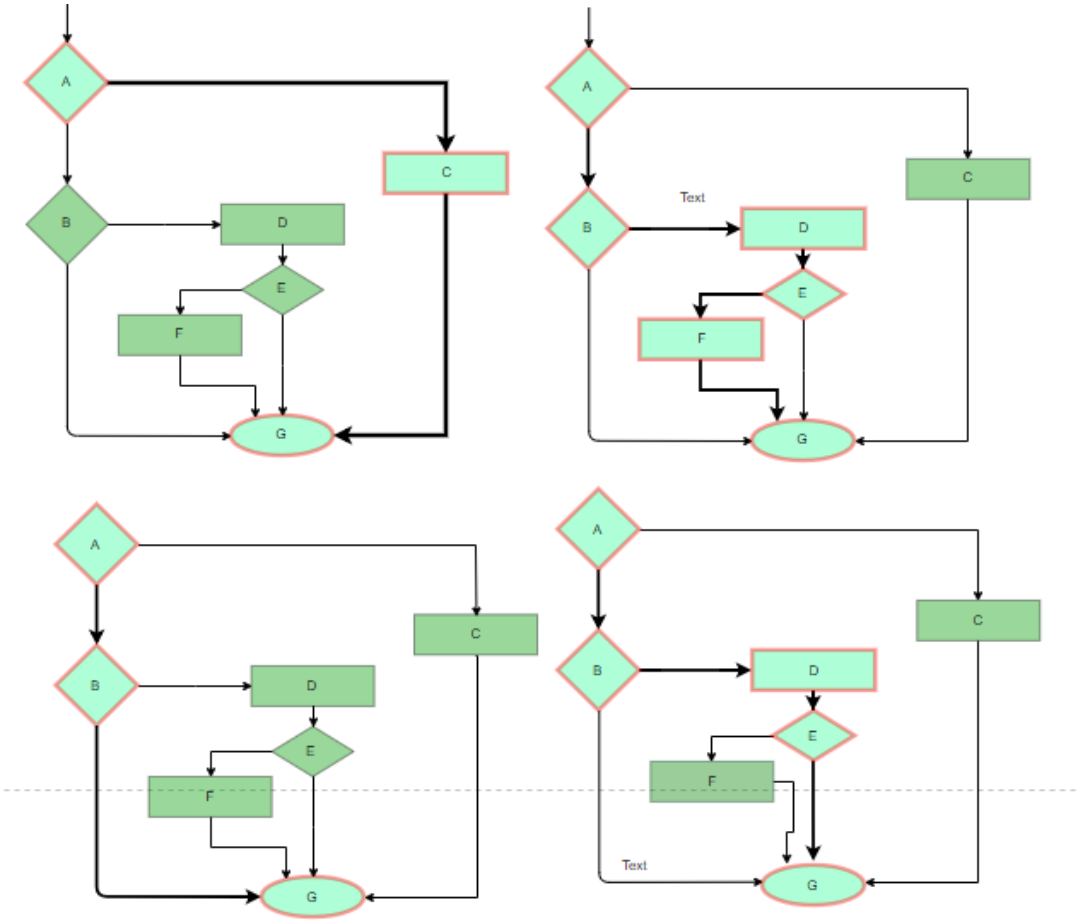
1) Statement Coverage

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, helps in pointing out faulty code.



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.



3) Condition Coverage

This technique is used to cover all conditions. It is also known as predicate coverage in which each one of the Boolean expressions has been evaluated to be both True and False.

Example

```
read a, b, c;
if (a == 0 || b == 0)
{
        print 1;
}
else
{
```

```
        If (c == 0 && d == 0)
        {
        print 2;
        }
}
```

4) Multiple Condition Coverage

In this technique, all the possible combinations of outcomes of conditions in a decision (therefore the complete decision table) are tested at least once. Since there are only two possible outcomes of a condition i.e. true or false, so 2 is the basis for the number of test situations that can be created. The maximum number of test situations depends on the number of conditions: 2n, where N is the number of conditions.

Example

Let's consider the following example:

1. If (a > 7 || b > 40 && c == 0)
2. { }
3. else
4. { }

2^3 = 8 Test Cases

Decision Table

Test Case Id	a > 7	b > 40	c == 0
1	T	T	T
2	T	T	F
3	T	F	T
4	T	F	F
5	F	T	T
6	F	T	F

7	F	F	T
8	F	F	F

5) Basis Path Testing

Basis Path Testing is a technique in which test cases are defined based on flows or logical paths that can be taken through the program. The objective of basis path testing is to define the number of independent paths so that the number of test cases needed can be defined explicitly to maximize test coverage. It involves the execution of all possible blocks in a program and achieves maximum path coverage with the least number of test cases

Steps for Basis Path Testing

The basic steps involved in basis path testing include:

1. Draw the control flow graph of the program.
2. Calculate the cyclomatic complexity of the control flow graph. This will be the maximum number of independent paths in the graph.
3. Identify independent paths in the control flow graph.
4. Design test cases based on the independent paths identified so that the test cases execute all independent paths.

Advantages of Basis Path Testing:

The advantages of conducting basis path testing are:

- Basis path testing reduces the number of redundant tests.
- All program statements are executed and tested at least once.
- It guarantees complete branch coverage.

Example

Let’s consider an example:

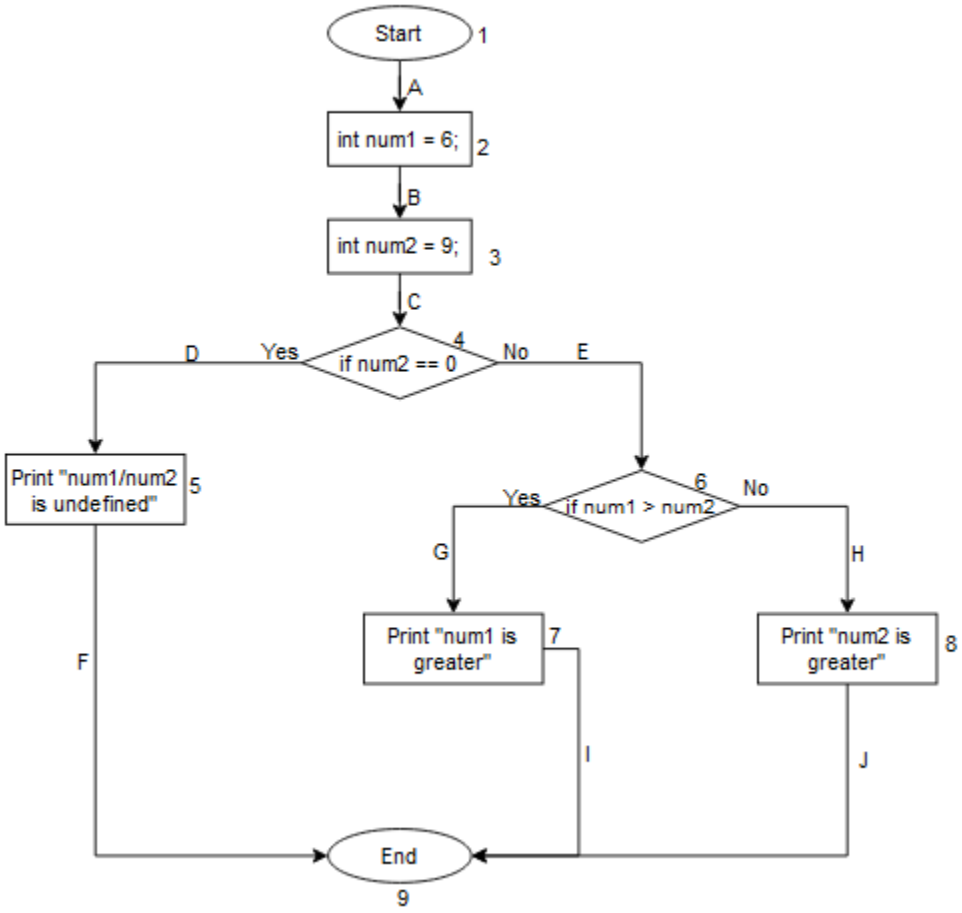
```
int num1 = 6;
int num2 = 9;

if(num2 == 0)
{
  cout<<"num1/num2 is undefined"<<endl;
```

```
}
Else
{
  if(num1 > num2)
  {
    cout<<"num1 is greater"<<endl;
  }
  Else
  {
    cout<<"num2 is greater"<<endl;
  }
}
```

Step - 1: Draw Control Flow Graph

The control flow graph of the code above will be as follows:



Step - 2: Calculate Cyclomatic Complexity

There are three ways to calculate the cyclomatic complexity of a program:

1. Create blocks or regions
2. Predicate statement (conditional statements), i.e. PS +1
3. E-N+2P (number of edges, nodes, and ways to exit the program)

Here,

E = Number of edges in the control flow graph.

N = Number of nodes in the control flow graph.

P = Number of connected components in the control flow graph.

The cyclomatic complexity of the control flow graph above will be:

*Cyclomatic complexity = E - N + 2*P*

*Cyclomatic complexity = 10 - 9 + 2*1*

Cyclomatic complexity = 3

Step - 3: Identify Independent Paths

The independent paths in the above control flow graph are as follows:

- **Path 1:** 1A-2B-3C-4D-5F-9
- **Path 2:** 1A-2B-3C-4E-6G-7I-9
- **Path 3:** 1A-2B-3C-4E-6H-8J-9

Step - 4: Design Test Cases

So, we have to create three different test use-case for each independent path.

During testing, if any test case fails, it means we are making a mistake in our code, if not then our code is correct.

The test cases to execute all paths above will be as follows:

Path	Input values
Path 1: 1A-2B-3C-4D-5F-9	• num1 = 9 • num2 = 0
Path 2: 1A-2B-3C-4E-6G-7I-9	• num1 = 4 • num2 = 2
Path 3: 1A-2B-3C-4E-6H-8J-9	• num1 = 6 • num2 = 8

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

Advantages of White Box Testing

- Code optimization by finding hidden errors.
- White box test cases can be easily automated.
- Testing is more thorough as all code paths are usually covered.
- Testing can start early in SDLC even if GUI is not available.

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 time to test fully.

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 a 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.