## Overview White Box Testing

The box testing approach in software testing includes both black box testing and white box testing. This discussion focuses on white box testing, which is also known by several other names, including glass box testing, structural testing, clear box testing, open box testing, and transparent box testing.

- **Definition**:
  White box testing examines the internal coding and structure of software, concentrating on verifying predefined inputs against expected outputs. This testing method is based on the inner workings of an application and focuses on testing its internal structure.
- **Requirements**:
  To design effective test cases in white box testing, programming skills are essential. The primary objective of this approach is to monitor the flow of inputs and outputs through the software while enhancing its security.
- **Terminology**:
  The term "white box" refers to the tester's internal perspective of the system. Names like clear box, white box, or transparent box highlight the ability to see beyond the software's exterior and into its internal mechanisms.
- **Testing Process**:
  White box testing is typically performed by developers, who examine every line of the program's code. After conducting this testing, developers hand over the application or software to the testing team, which conducts black box testing to verify compliance with requirements, identify bugs, and report them back to the developers.
- **Bug Fixing and Retesting**:
  Once the developers address the reported bugs, they perform another round of white box testing to ensure that the issues have been resolved. This process confirms that the identified bugs have been fixed and that the related features are functioning correctly in the application.

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

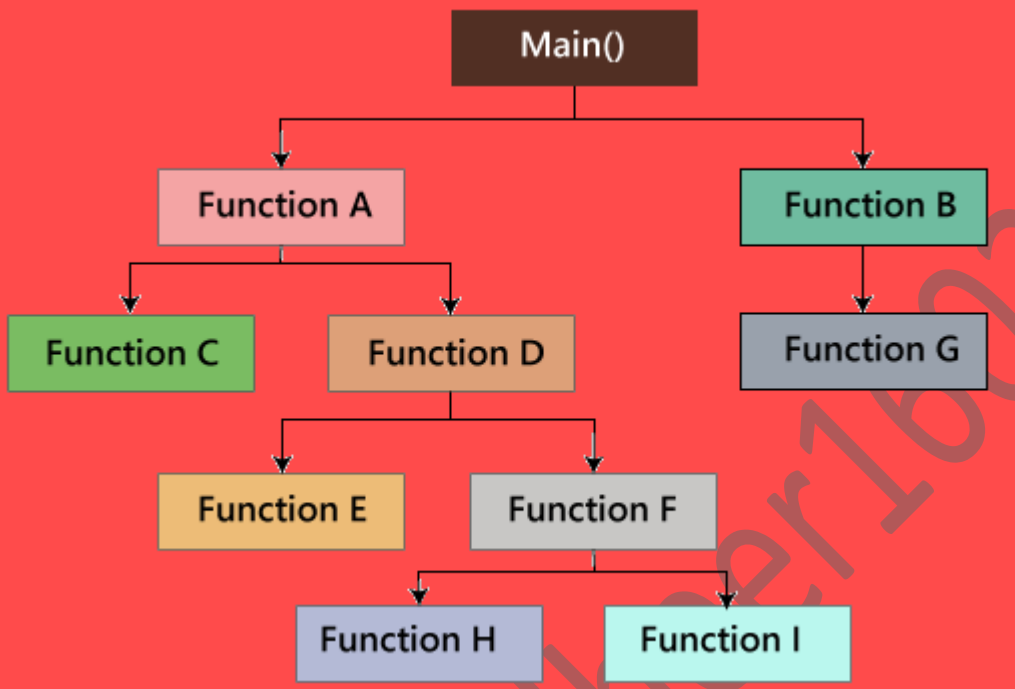
- Fixing the bug might interrupt the other features. Therefore, the test engineer should always find the bugs, and developers should still be doing the bug fixes.
- If the test engineers spend most of the time fixing the defects, then they may be unable to find the other bugs in the application.

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

**Path Testing**

In path testing, we create flow graphs to test all independent paths within a program. These flow graphs visually represent the flow of the program and illustrate how different components are interconnected. This method helps ensure comprehensive coverage of all execution paths, allowing us to verify that each path behaves as expected.




Testing all independent paths means that, for example, if there is a path from main() to function G, we first set the parameters and verify whether the program operates correctly along that specific path. Similarly, we will test each of the other paths, ensuring that all routes through the program are evaluated and any bugs are identified and resolved.

## Loop Testing

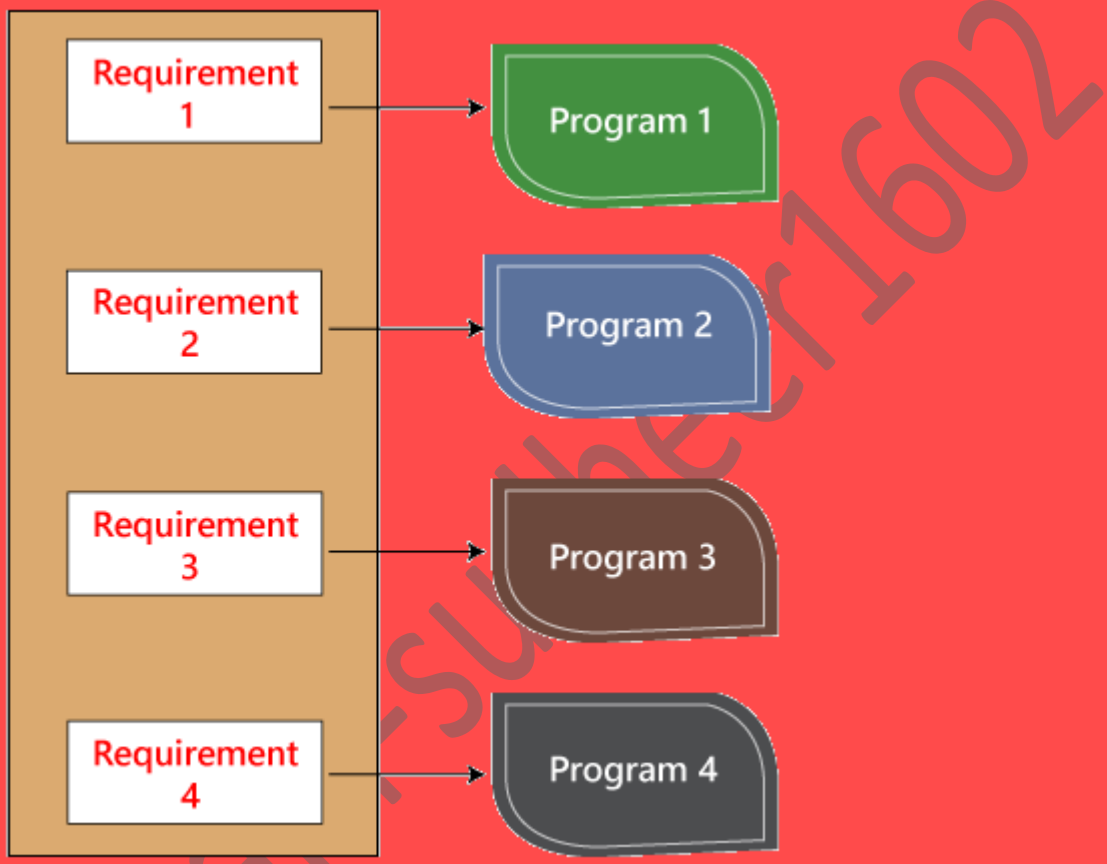
In loop testing, we focus on evaluating various types of loops, including while, for, and do-while loops. This involves checking whether the termination conditions are functioning correctly and verifying that the size of the conditions is appropriate for the intended use. The goal is to ensure that the loops behave as expected under different scenarios.

1. {
2. while(50,000)
3. ……
4. ……
5. }

Testing a program manually for all 50,000 loop cycles is impractical. Therefore, we create a smaller program to automate the testing of all 50,000 cycles. In the example below, the test program P is written in a language similar to the source code of the original program. This automated testing approach is known as a unit test and is typically developed by the programmers themselves.

1. Test P
2. {
3. ……
4. …… }

In the image below, we see several requirements labeled as 1, 2, 3, and 4. Correspondingly, the developer writes programs—Program 1, Program 2, Program 3, and Program 4—to address these parallel conditions. The application consists of hundreds of lines of code to meet these requirements.

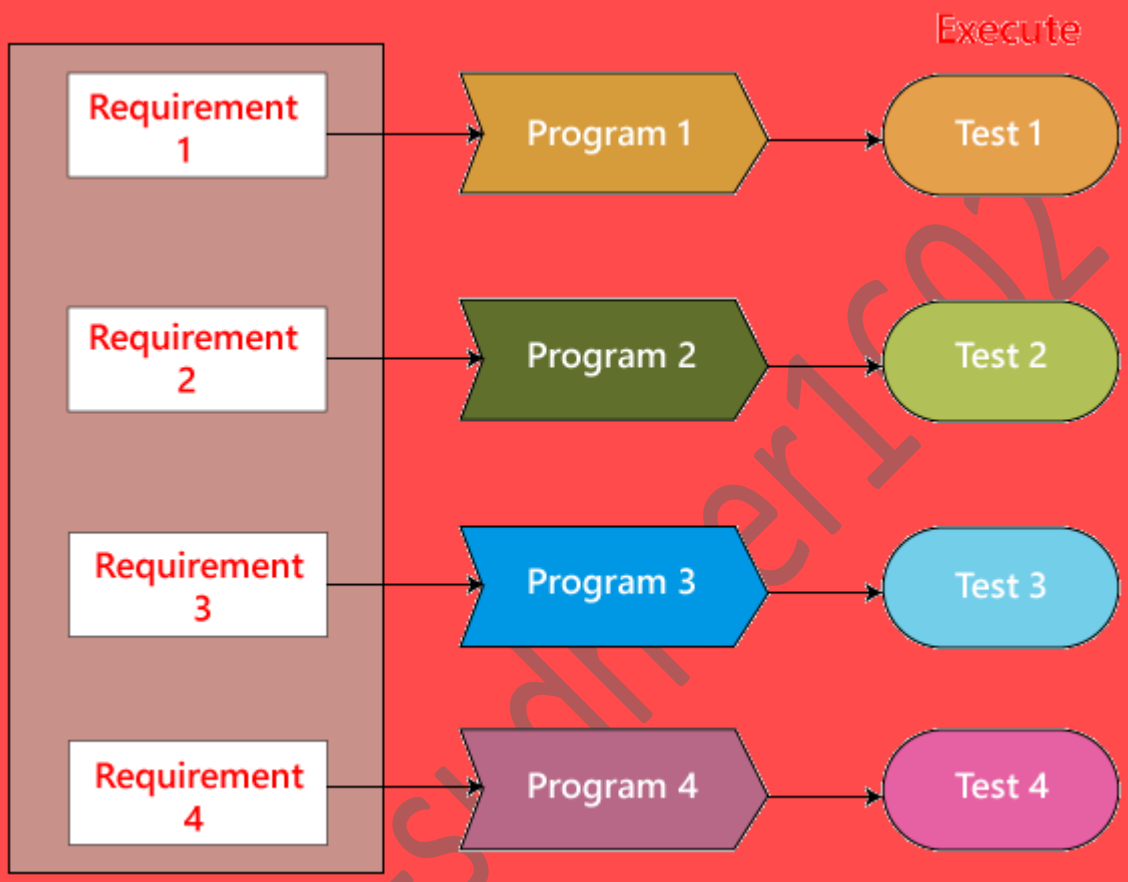



The developer conducts white box testing by examining each line of code in all five programs to identify any bugs. If a bug is discovered in any of the programs, it is corrected, and the system undergoes retesting. This process can be time-consuming and requires significant effort, which can delay the product release timeline.

In another scenario, if clients request modifications to the requirements, the developer must implement the necessary changes and retest all four programs. This, too, demands considerable time and effort, further impacting the overall development schedule.

These issues can be addressed in the following ways:

We can create tests for the relevant programs by writing test code in the same language as the source code. Developers then execute these test codes, commonly referred to as unit test programs. These test programs are linked to the main program and function alongside it, allowing for efficient testing and verification of the application.




## Condition testing

In this, we will test all logical conditions for both **true** and **false** values; that is, we will verify for both **if** and **else** condition.

**For example:**

1. if(condition) - true
2. {
3. .....
4. ......
5. ......
6. }
7. else - false
8. {

9. …..
10. ……
11. ……
12. }

The above program will work fine for both the conditions, which means that if the condition is accurate, and then else should be false and conversely.

## Generic Steps of White Box Testing

1. **Test Scenario and Case Design**:
    - Design all test scenarios and test cases, prioritizing them according to their importance.
2. **Code Analysis at Runtime**:
    - Examine the code during execution to assess resource utilization, identify unaccessed areas of the code, and measure the time taken by various methods and operations.
3. **Testing Internal Subroutines**:
    - Test internal subroutines, such as non-public methods and interfaces, to ensure they can handle all types of data appropriately.
4. **Control Statement Testing**:
    - Focus on testing control statements, including loops and conditional statements, to evaluate their efficiency and accuracy with different data inputs.
5. **Security Testing**:
    - Conduct security testing to identify potential security loopholes by analyzing how the code handles security measures.

**Reasons for White Box Testing**

- Identifies internal security vulnerabilities.
- Examines the input handling within the code.
- Assesses the functionality of conditional loops.
- Tests functions, objects, and statements at an individual level.

## Advantages of White Box Testing

- Optimizes code, enabling the identification of hidden errors.
- Test cases can be easily automated.
- More comprehensive than other testing approaches as it covers all code paths.

- Can be initiated during the Software Development Life Cycle (SDLC) even without a graphical user interface (GUI).

## Disadvantages of White Box Testing

- Time-consuming, especially for large-scale programming applications.
- Can be expensive and complex to implement.
- May lead to production errors due to insufficient detail by developers.
- Requires professional programmers with extensive knowledge of programming languages and implementations.