Unit- III
White box testing

- Dr. Shivani Budhkar

What is White Box Testing?

- White box testing (also known as clear, glass box or structural testing) is a testing technique which evaluates the code and the internal structure of a program.
- White box testing involves looking at the structure of the code. When you know the internal structure of a product, tests can be conducted to ensure that the internal operations performed according to the specification. And all internal components have been adequately exercised.
- 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.
- It is one of two parts of the Box Testing approach to software testing. Its counterpart, Black box testing, involves testing from an external or end-user type perspective. On the other hand, White box testing is based on the inner workings of an application and revolves around internal testing.

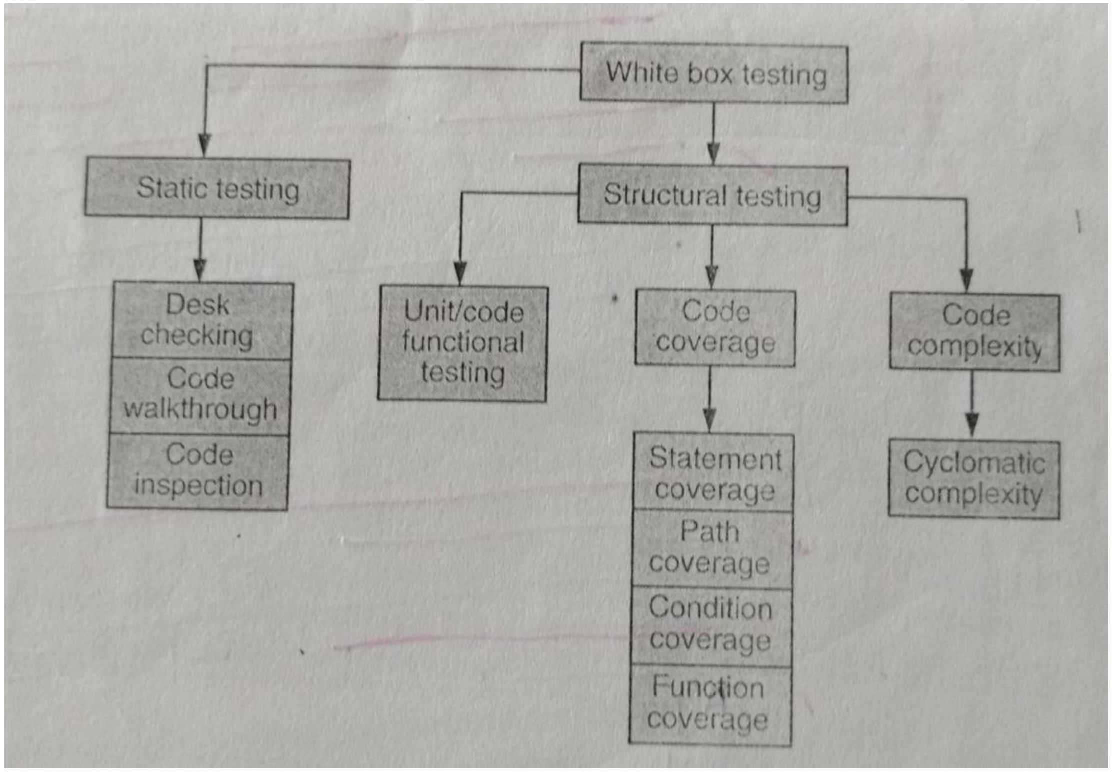
What do you verify in White Box Testing?

- 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 whitebox 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.

Why we perform WBT?

- **To ensure:**
  - That all independent paths within a module have been exercised at least once.
  - All logical decisions verified on their true and false values.
  - All loops executed at their boundaries and within their operational bounds internal data structures validity.
- **To discover the following types of bugs:**
  - Logical error tend to creep into our work when we design and implement functions, conditions or controls that are out of the program
  - The design errors due to difference between logical flow of the program and the actual implementation
  - Typographical errors and syntax checking

Classification of white box testing



Static Testing

- Requires only source code of product
- Does not involve executing the program on computers

1. Static testing by Humans

- Methods to achieve static testing
  - Desk checking
  - Code walkthrough
  - Code inspection

2. Static Analysis tools

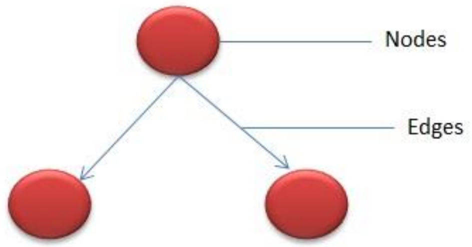
   To reduce manual and performance analysis of code various static analysis tools are used

Structural Testing

- Takes into account the code, code structure, internal design and how they are coded
- Tests are actually run by computer by on the build product
- **Unit/Code Functional Testing**
- Done by developer
- This can happen by several methods
  1. obvious tests with known input variables and corresponding expected variables
  2. by putting intermediate print statements
  3. Do initial tests with debugger or IDE
- **Code Coverage Testing**
- Involves designing and executing test cases & finding out percentage of code that is covered by Testing
- Done by technique instrumentation of code using tools
- Types of coverage
  - Statement coverage
  - Path Coverage
  - Condition Coverage
  - Function coverage

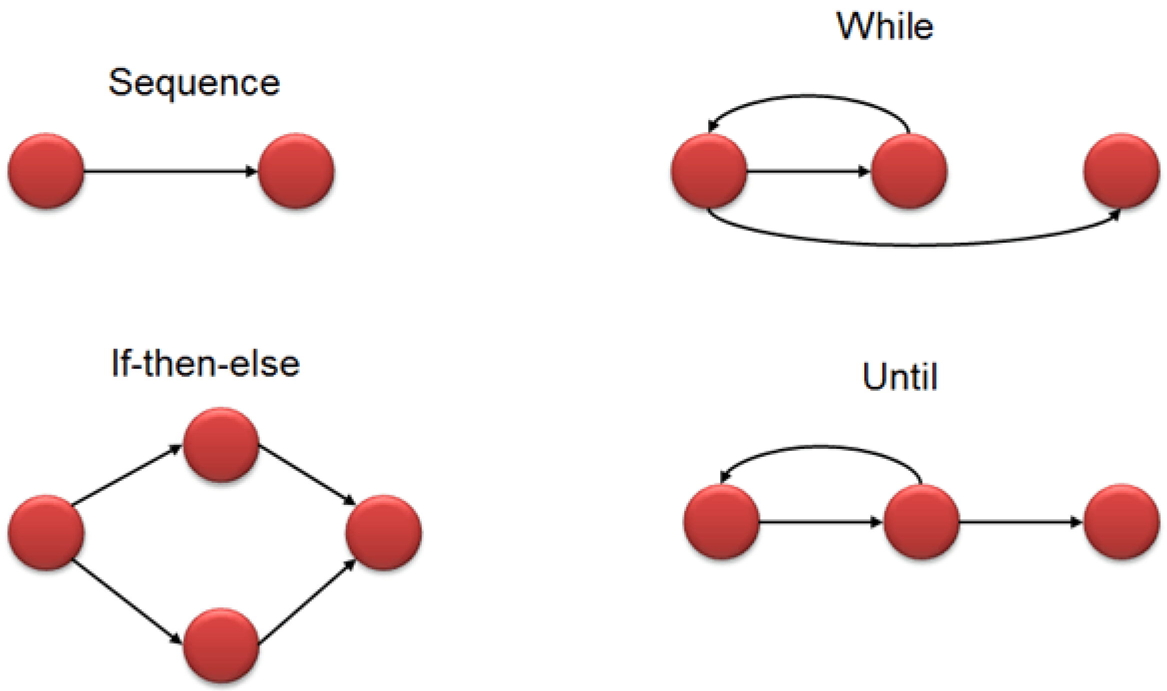
Code complexity testing

- Cyclomatic complexity is metric that quantifies complexity of program
- It is a quantitative measure of independent paths in the source code of a software program.
- Cyclomatic complexity can be calculated by using control flow graphs or with respect to functions, modules, methods or classes within a software program.
- Independent path is defined as a path that has at least one edge which has not been traversed before in any other paths.
- This metric was developed by Thomas J. McCabe in 1976
- It is based on a control flow representation of the program.
- Control flow depicts a program as a graph which consists of Nodes and Edges.
- In the graph, Nodes represent processing tasks while edges represent control flow between the nodes.



Code complexity testing

- **Flow graph notation for a program**



Code complexity testing

- **How to Calculate Cyclomatic Complexity**
- Mathematically, it is set of independent paths through the graph diagram. The Code complexity of the program can be defined using the formula –

- $V(G) = E - N + 2$
- Where,

- E – Number of edges
- N – Number of Nodes

- $V(G) = P + 1$
- Where P = Number of predicate nodes (node that contains condition)