SOFTWARE ENGINEERING

WHITE BOX TESTING

White box is a testing methodology to test the internal structures and working of software. White box testing also know as structural testing is testing based on analysis of internal logic (design, code, etc.).

LEARNING OBJECTIVES

- To focus on Program structures, Program internal logic and data structures, Program internal behaviors and states.
- To focus on internal program structure and discover all internal program errors.

DEFINITION

White box testing is testing based on analysis of internal logic (design, code, etc.). (But expected results still come from requirements.). It is also known as structural testing. White-box testing concerns techniques for designing tests; it is not a level of testing. White-box testing techniques apply primarily to lower levels of testing (e.g., unit and component). The major testing focuses on

- Program structures.
  - Program statements and branches
  - Various kinds of program paths
- Program internal logic and data structures.
- Program internal behaviors and states.
- Logic coverage.
  - **Statement:** each statement executed at least once.
  - **Branch**: each branch traversed (and every entry point taken) at least once.
  - **Condition:** each condition True at least once and False at least once.
  - **Branch/Condition**: both branch and condition coverage achieved.
  - **Compound Condition**: all combinations of condition values at every branch statement covered (and every entry point taken).
  - **Path:** all program paths traversed at least once
- Dataflow coverage.

- Path conditions and symbolic evaluation.
- Other white-box testing strategies (e.g., “fault-based testing”).

WHITE BOX TESTING

White box is a testing methodology to test the internal structures and working of software. White box testing also know as structural testing is testing based on analysis of internal logic (design, code, etc.).

Test Model: Control program chart (graph)
Test case design: Various white-box testing methods generate test cases based on a given control program graph for a program.

The goal of white box testing is to:

- Guarantee that all independent paths within a module have been exercised at least once.
- Exercise all logical decisions on their true and false sides.
- Execute all loops at their boundaries and within their operational bounds.
- Exercise internal data structures to assure their validity.
- Exercise all data define and use paths.

White-Box Software Testing Methods

- **Basis path testing**: It was first proposed by Tom McCabe [MCC76]. It can be used to derive a logical complexity measure for a procedure design and used as a guide for defining a basis set of execution path. Path testing guarantees to execute every statement in the program at least one time.

- **Branch Testing**: Branch testing exercise predicate nodes of a program flow graph to make sure that each predicate node has been exercised at least once.

- **Loop Testing:** It exercise loops of a program to make sure that the inside and outside of loop body are executed.

- **State-Based Testing**: The basic idea is to use a finite state machine as a test model to check the state behaviors of a program process.

- **Cyclomatic Complexity:** Cyclomatic complexity is software metric that provides a quantitative measure of the global complexity of a program. When this metric is used in the context of the basis path testing, the value computed for cyclomatic complexity defines the number of independent paths in the basis set of a program.

Measuring software complexity is much needed as software complexity is difficult to operationalize. Computational complexity measure is big O (or big Oh), O(n). It measures software complexity from the machine's viewpoint in terms of how the size of the input data affects an algorithm's usage of computational resources (usually running time or memory). Complexity measure in software engineering should measure complexity from the viewpoint of human developers, as computer time is cheap whereas human time is expensive.

Cyclomatic Complexity was invented by Thomas McCabe (1974) to measure the complexity of a program's conditional logic. It counts the number of decisions in the program, under the assumption that decisions are difficult for people. It makes assumptions about decision-counting rules and linear dependence of the total count to complexity.

Cyclomatic complexity of graph G equals →#edges - #nodes + 2

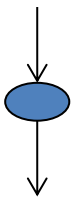
$$\mathbf{V(G) = e - n + 2}$$

It also corresponds to the number of linearly independent paths in a program.

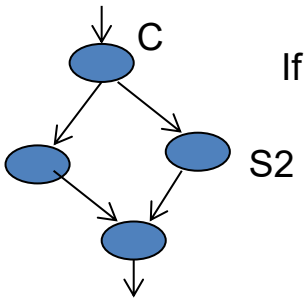
Program Flow Graph (Control Flow Diagram)

Flow graph notation defines the program using nodes connected by edges. The following shows some examples for flow graph:

Sequential statement block where the statements are executed sequentially.

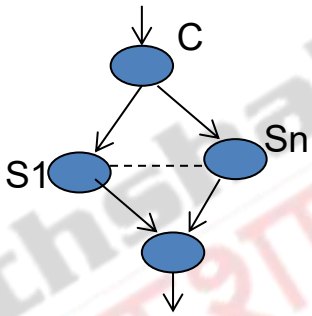


If else block – defines an if and else condition



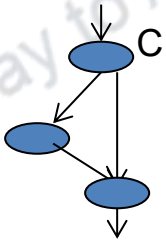
If C Then S1 else S2;

Switch case block



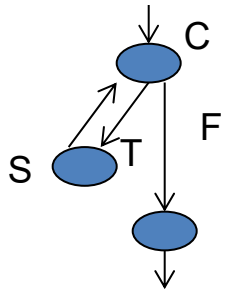
Case C of
L1: S1;
L2: S2;
…
Kn: Sn;
end;

If- then block - The code is executed only if a particular test evaluates to true.



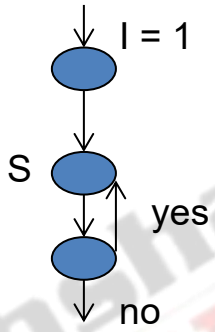
If C Then S1;

While block - executes a block of statements continuously while the given condition is true.



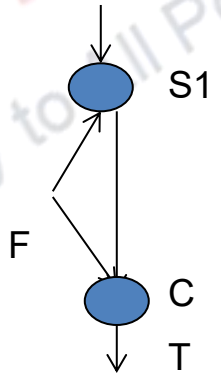
While C do S;

Loop – a block of code is executed continually until a certain condition is reached.



For loop:
for I = 1 to n do S;

Do-loop - block of instructions are continually executed while a Boolean condition is true or until the condition becomes true.



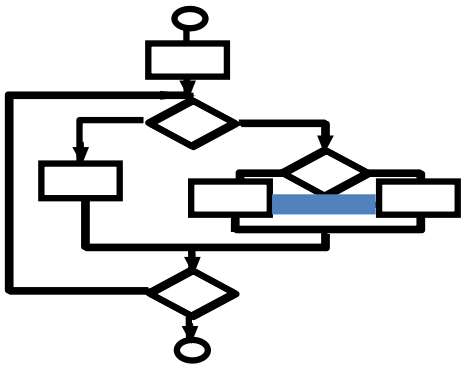
Do loop:
do S1 until C;

Computing the cyclomatic complexity

Cyclomatic complexity of graph G equals →#edges - #nodes + 2

V(G) = e – n + 2

Computing the cyclomatic complexity involves **number of simple decisions + 1 (or) number of enclosed areas + 1**. In the following case, V (G) = 4



Examples for graph complexity: