Different Approaches to White Box Testing Technique for Finding Errors

Mohd. Ehmer Khan
Department of Information Technology
Al Musanna College of Technology, Sultanate of Oman
ehmerkhan@gmail.com

Abstract

We can define software testing as a process or a series of processes, design to make sure that computer code does what it was actually design to do and it doesn't do anything unintended. Two of the most important software testing techniques are white box testing and black box testing. In my paper I have described one of the main software testing technique that is white box testing. We can define it as a test case design method that uses the control structure of the procedural design to derive test cases. I have also described briefly the working process of white box testing technique and some of its most frequently used techniques that are control flow testing, data flow testing, branch testing, basis path testing and loop testing.

Keywords: *Control Flow Testing, Data Flow Testing, Branch Testing, Basis Path Testing, Loop Testing*

1. Introduction

Software testing is a set of activities conducted with the intent of finding errors. It also ensures that the system is working according to the specification. The two important goals of software testing are to ensure system being developed is according to the customer requirement and to reveal bugs. An essential form of assurance is testing. Testing is laborious, expensive and time consuming job, so the choice of testing must be based on the risk to the system.

White box testing techniques is one of the most important and prevalent software testing techniques, it is typically very effective in validating design, decision, assumptions and finding programming errors and implementation errors in software.

White box testing based on analysis of internal workings and structure of a piece of software and it can uncover implementation errors such as poor key management. White box testing is performed based on the knowledge of how the system is implemented. White box testing is very effective and efficient in validating design decision and assumptions.

There are some advantages of white box testing such as it reveals error in hidden code, its side effects are beneficial and it helps in removing extra lines of code. [10][5]

The main disadvantage of white box testing is that, it is very expensive and some of the codes omitted in the code could be missed out. [10]

Some important types of white box testing are

- Control Flow Testing
- Branch Testing
- Basis Path Testing

- Data Flow Testing
- Loop Testing

Some synonyms of white box testing are [5]

1) Glass box testing
2) Clear box testing
3) Open box testing
4) Transparent box testing
5) Structural testing
6) Logic driven testing
7) Design based testing

2. Working Process of White Box Testing Technique

The general outlining of white box testing process is described below:

Step 1: Input ~~includes~~ → – Requirement
– Functional specification
– Detailed designing of documents
– Proper source code
– Security specifications are must [11]

Step 2: Processing Unit (Analyzing internal working only)
- Perform risk analysis to guide whole testing process
- Proper test plan
- Execute test cases and communicate results

Step 3: Output → Prepare final report

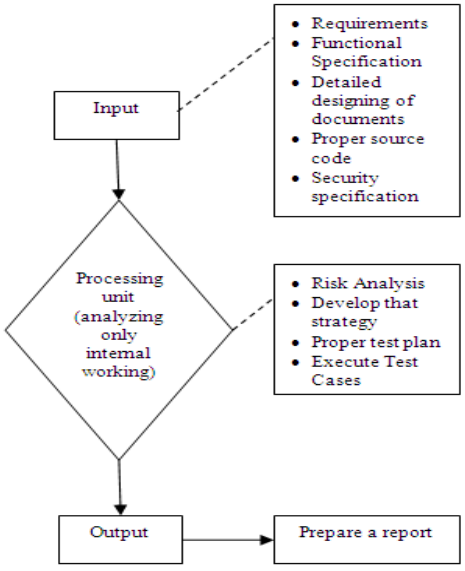

Figure 1 Represent Working Process of White Box Testing Techniques

3. Different Forms of White box Testing Techniques are

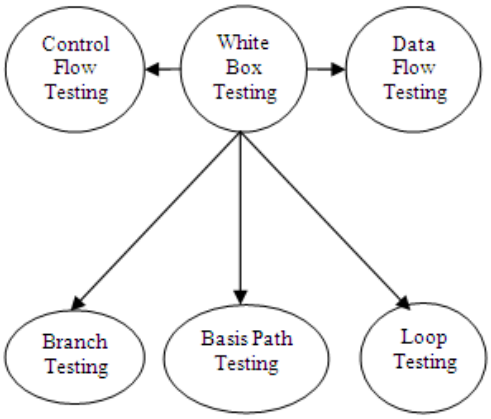

Figure 2 Represent Various Forms of White Box Testing Techniques

3.1 Loop Testing

Loop testing is another type of white box testing which exclusively focuses on the validity of loop construct. Loops are simple to test unless dependencies exist between the loops or among the loop and the code it contain. There are four classes of loops:

a) Simple Loop

b) Nested Loop

c) Concatenated Loop

d) Unstructured Loop

3.1.1 Simple Loop

The following sets of tests can be applied to simple loops, where n it's a maximum no. of allowable passes through loops. [13]

Step 1: Skip the loop entirely

Step 2: Only one pass through the loop

Step 3: Two passes through the loop

Step 4: m passes through the loop where n>m

Step 5: n-1, n, n+1 passes through the loops

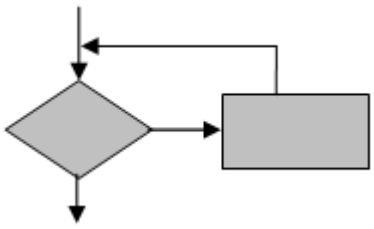

Figure 3 Represent Simple Loop

3.1.2 Nested Loop

The no. of possible test would grow geometrically as the level of nesting increases from simple to nested loops. [13]

Step 1: Set all the other loops to minimum value and start at the innermost loop

Step 2: Conduct simple loop test for the innermost loop and holding the outer loops at their minimum iteration parameter value

Step 3: Performing test for the next loop and work outward

Step 4: Continue until all the loops have been tested

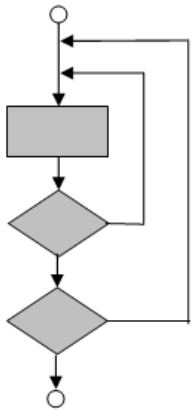

Figure 4 Represent Nested Loop

3.1.3 Concatenated Loop

If two loops are independent from each other then they are tested by using simple loop test. However, for two concatenated loops, if the loop counter for one loop is used as the initial value for the others, then the two loops are not independent. [13]

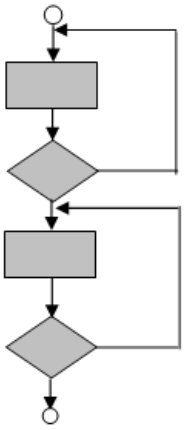

Figure 5 Represent Concatenated Loop

3.1.4 Unstructured Loop

This type of loop should be redesigned to reflect the use of structured programming construct.

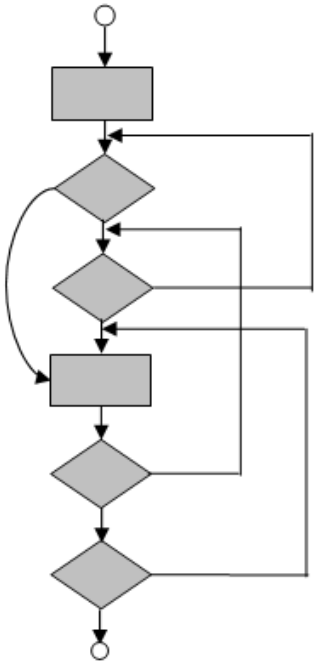

Figure 6 Represent Unstructured Loop

3.2 Branch Testing

The other synonym of branch testing are conditional testing or decision testing and comes under white box testing technique; it makes sure that each possible outcome from the condition is tested at least once.

Branch testing however, has the objective to test every option (either true or false) on every control statement which also includes compound decision (when the second decision depends upon the previous decision). [17]

In branch testing, test cases are designed to exercise control flow branches or decision points in a unit. All branches within the branch are tested at least once. [1]

Example of Branch Testing:

Read Length (l)
Read Breadth (b)
IF l > 4 THEN
Print "invalid length"
ENDIF
IF b > 5 THEN
Print "invalid breadth"
ENDIF
*AREA= (l*b)*
Print ("Area="Area)

IF Area>12 THEN
Print "False Area"
ELSE
IF Area<=12 and Area>=1 THEN
Print "True Area"
ELSE
IF Area<1 THEN
Print "Area Not Defined"
ENDIF

Another way to look at branch testing is to draw a diagram that shows how the code is flowing.

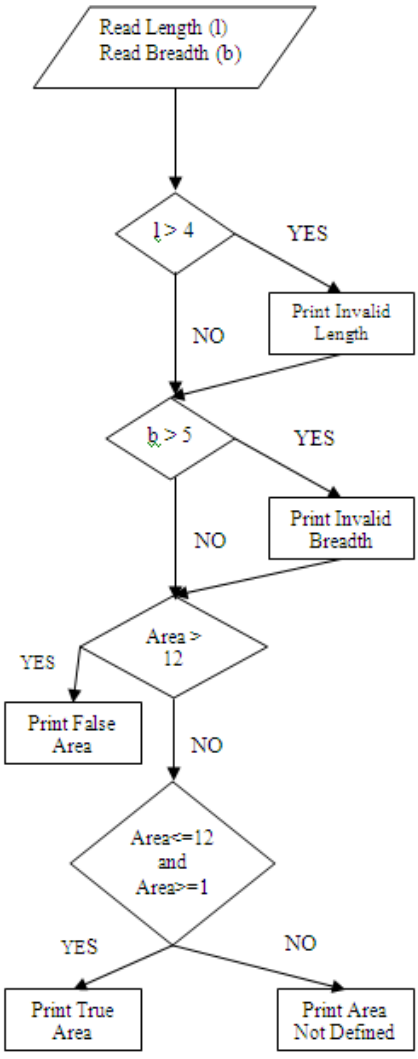

Figure 7 Branch Testing

3.3 Data Flow Testing

Data flow testing is another type of white box testing which looks at how data moves within a program. In data flow testing the control flow graph is annoted with the information about how the program variables are defined and used. [9] (Control flow graph is a diagrammatic representation of a program and its execution). [2] The figure below represents Simplified Control Flow Graph.

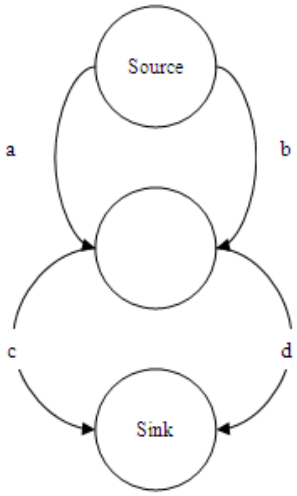

Figure 8 Simplified Control Flow Graph

We can also define data flow testing as testing techniques which is based on the observation that values associated with variables can effect program execution. [8] Data flow testing picks enough paths to assure that:

1. Every data object has been initialized prior to its use.
2. All defined objects have been used atleast once.

Some of the important points of data flow testing are: [15]

1. All data flow anomalies are resolved.
2. Avoid integration problems by doing all data flow operation on a variable within the same routine.
3. When possible use explicit (rather than implicit) declaration of data.

A number of data flow testing strategies have been studied and compared i.e. [NTA88], [FRA93].

Data flow testing tends to uncover bugs like variable used but not initialized or declared but not used, so on.

3.4 Control Flow Testing

Another white box testing technique to find bug is control flow testing. Control flow testing applies to almost all software and is effective for most software. It is a structural

testing strategy that uses the program's control flow as a model control flow testing favour more but simpler paths over complicated but fewer paths.

Researches shows that control flow testing catches about 50% of all bugs during unit testing (unit testing is dominated by control flow testing). Control flow testing is more effective for unstructured code rather than structured code. Most bugs can result in control flow errors and therefore misbehaviour that could be caught by control flow testing. Control flow bugs are not as common as they used to be as they are minimize because of structured programming languages. [16]

Some of the limitations of control flow testing are [4]

1. Control flow testing cannot catch all initialization mistakes.
2. Specification mistake are not caught by control flow testing.
3. It's unlikely to find missing paths and features if the program and the model on which the tests are based are done by the same person.

The adequacy of the test cases measured with a metric called coverage (coverage is a measure of the completeness of the set of test cases). [6]

Now, we will define various coverage methods:

3.4.1 Statement Coverage

Statement coverage is a measure of the percentage of statements that have been executed by test cases. Anything less than 100% statement coverage means that not all lines of code have been executed. [2] We can achieve statement coverage by identifying cyclomatic number an executing this minimum set of test cases. Measure benefit of statement coverage is that it is greatly able to isolate the portion of code, which could not be executed. Since it tends to become expensive, the developer chose a better testing technique called branch coverage or decision coverage. [3] [6]

3.4.2 Branch Coverage

A stronger logic coverage criterion is known as branch coverage or decision coverage. Branch coverage or decision coverage is a measure of the percentage of the decision point of the program have been evaluated as both true and false in test cases. Examples of branch coverage-DO statements, IF statements and multiway GOTO statements. Branch coverage is usually shown to satisfy statement coverage. By 100% branch coverage we mean that every control flow graph is traversed. [2] [6]

3.4.3 Condition Coverage

A criterion which is stronger than decision coverage is condition coverage. It is a measure of percentage of Boolean sub-expressions of the program that have been evaluated as both true and false outcome in test cases. [6]

3.5 Basis Path Testing

Basis path is a white box testing techniques which is first proposed by Tom McCabe and it allows the test case designer to produce a logical complexity measure of procedural design and use this measure as an approach for outlining a basic set of execution path (basic set is the set of all the execution of a procedure). [14] These are test cases that exercise basic set will

execute every statement atleast once. Basic path testing makes sure that each independent path through the code is taken in a predetermined order.

3.5.1 Flow Graph Notation

The simple notation used for representing control flow is called flow graph. The flow graph Represent logical control flow which is used to depict the program control structure.

- Each circle in a flow graph is called flow graph node.
- The arrows are called edges or links (represent flow of control).
- Area bounded by nodes and edges are called regions.

Below are the individual constructs combine together to produce the flow graph for any particular procedure. [12]

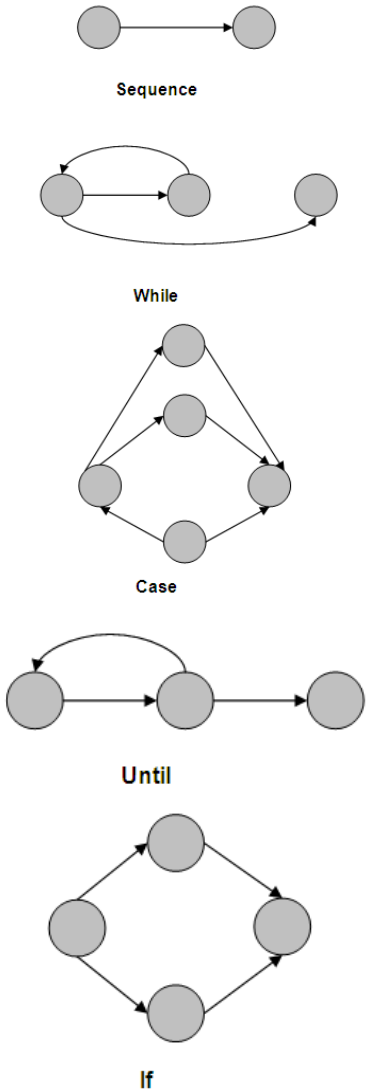

Figure 9 Represent Different Types of Flow Graphs

3.5.2 Cyclomatic Complexity

A cyclomatic complexity is a software metric which gives a quantitative measure of 4th logical complexity. When used in the context of the basis path testing method, it defines the number of independent paths (any path through the program that introduces atleast one new set of processing statement or a new condition) in the basis set of program and provides upper bound for number of tests required to guarantee coverage of all program statements. [7]

We can compute cyclomatic complexity for a flow graph in any of the given ways. [12]

1. Cyclomatic complexity, V(G), for a graph flow G is defined as
   V(G) = P + 1
   Where P is the number of predicate nodes

2. The number of regions of the flow graph corresponds to the cyclomatic complexity.

3. Cyclomatic Complexity, V(G), for a flow graph G is defined as
   V(G) = E – N + 2
   Where N is the number of nodes in a flow graph and E is the number of edges.

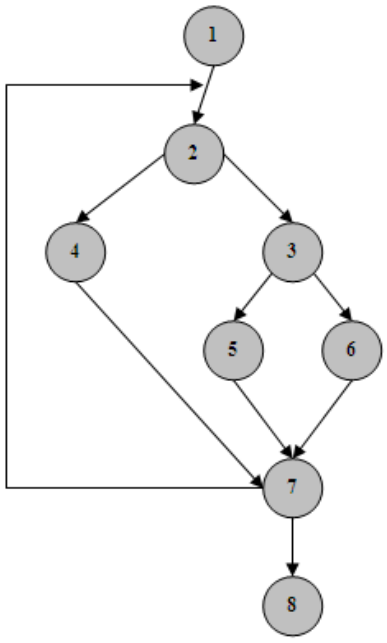

Figure 10 Represent Cyclomatic Complexity Graph

As set of independent paths for flow graph illustrated in Figure 10 is:

Path 1: 1-2-4-7-8
Path 2: 1-2-3-5-7-8
Path 3: 1-2-3-6-7-8
Path 4: 1-2-4-7-2-4…..-7-8

To measure cyclomatic complexity
Region, R = 4

Number of Nodes = 8
Number of edges = 10
Number of Predicate Nodes = 3

Cyclomatic Complexity
V(G) = R= 4

Or

V(G) = Predicate Nodes + 1
= 3 + 1 = 4

Or

V(G) = E – N + 2
= 10 – 8 + 2
= 4

3.5.3 Deriving Test Cases

The main objective of basis path testing is to derive the test cases for the procedure under test. Basis path testing can be seen as a set of steps

Step 1: Draw appropriate flow graph from the given design or source code.

Step 2: Calculate the cyclomatic complexity of this flow graph by using any of the three formulas.

Step 3: Determine a basis set of linear independent path.

Step 4: Prepare test cases that will force execution of each path in the basis set.

After completing all test cases tester make sure that all statements in the program are executed atleast once. [12]

3.5.4 Graph Matrices

Graph matrix is two dimensional matrix that helps in determining the basic set. It has columns and rows each equal to number of nodes in a flow graph. To distinguish from each other each node is represented by some letter. Each edge is provided with some link weight (0-for no connection, 1-if there is connection). [12]

A basic flow graph and its associated graph matrix is shown below in figure 11:

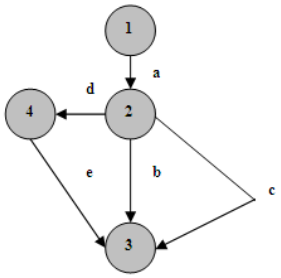

Figure 11 Example of Graph Matrix

Table 1: Graph Matrix

Node	1	2	3	4
1		a		
2			b, c	d
3				
4			e	

Represented in the below form the graph matrix is called connection matrix

Table 2: Connection Matrix

Node	1	2	3	4	Connections
1		1			1-1=0
2			1, 1	1	3-1=2
3					0
4			1		1-1=0

Cyclometic Complexity is 2+0=2

4. Conclusion

White box testing is verification and validation technique which software engineers can use to examine their code works as expected. In my paper I have described the working process and the various white box testing techniques such as Control Flow Testing, Data Flow Testing, Branch Testing, Basis Path Testing and Loop Testing.

Using white box testing techniques outlined in my paper, a software engineer can design test cases that

- Exercise internal data structure to ensure their validity.
- Exercise logical decisions on both their true and false side.
- Exercise independent path within a module.
- Execute loops at their boundaries and within their operational bounds.

Tester need to understand the white box techniques that are available to make educated decisions about their use for the specific system we are currently, and in future, will be testing. Properly planned with explicit input output combinations white box testing is a controlled V & V technique.

References:

[1] Branch Testing and Condition Testing available at http://softwaretestinginterviewfaqs.wordpress.com/category/software-testing-basics-interview-faq/page/10/

[2] Control Flow Testing available at http://www.computingstudents.com/notes/software_analysis/control_flow_testing.php

[3] All about Code Coverage - A White Box Testing Technique by Yogindernath Gupta available at http://ezinearticles.com/?All-About-Code-Coverage---A-White-Box-Testing-Technique&id=1796983
[4] Control Flow Testing notes taken from Boris Beizer books available at http://testdesigners.com/testingstyles/ControlFlowTesting.html
[5] Software Testing by Cognizant Technology Solutions
[6] White-Box Testing by Laurie Williams published in 2006
[7] White Box Testing available at http://www.eduresourcecollection.com/testing_whitebox.php
[8] Data Flow Testing available at http://www.allbusiness.com/computing-information-technology/software-testing/11983363-1.html
[9] White Box Testing available at http://agile.csc.ncsu.edu/SEMaterials/WhiteBox.pdf
[10] White Box Testing available at http://www.testingbrain.com/WHITEBOX/WHITE_BOX_Testing.html
[11] White Box Testing notes from Microsoft Corporation in January 2005 available http://msdn.microsoft.com/en-us/library/ff649503.aspx
[12] White Box Testing available at http://www.freetutes.com/systemanalysis/sa9-white-box-testing.html
[13] Types of Loop Testing available at http://productdevelop.blogspot.com/2010/10/loop-
[14] Basis Path Testing by Robin Roy available at http://www.codeproject.com/KB/testing/white-box-testing.aspx
[15] Topics in Software Dynamic White-box Testing Part 2: Data-flow Testing [Reading assignment: Chapter 7, pp. 105-122] available at http://www.cs.drexel.edu/~spiros/teaching/SE320/slides/dataflow-testing.pdf
[16] Topics in Software Dynamic White-box Testing Part 1: Control-flow Testing [Reading assignment: Chapter 7, pp. 105-122] available at http://www.cs.drexel.edu/~spiros/teaching/SE320/slides/controlflow-testing.pdf
[17] White Box Testing by Sharon Robson in August 2009 published by STANZ 2009

Author

Mohd. Ehmer Khan I completed my B.Sc in 1997 and M.C.A. in 2001 from Aligarh Muslim University, Aligarh, India, and pursuing Ph.D (Computer Science) from Singhania University, Jhunjhunu, India. I have worked as a lecturer at Aligarh College Engineering & Management, Aligarh, India from 1999 to 2003. From 2003 to 2005 worked as a lecturer at Institute of Foreign Trade & Management, Moradabad, India. From 2006 to present working as a lecturer in the Department of Information Technology, Al Musanna College of Technology, Ministry of Manpower, Sultanate of Oman. I am recipient of PG Merit Scholarship in MCA. My research area is software engineering with special interest in driving and monitoring program executions to find bugs, using various software testing techniques.