International Journal of Scientific Research in Computer Science, Engineering and Information Technology
© 2017 IJSRCSEIT | Volume 2 | Issue 6 | ISSN : 2456-3307

Covering All White Box Tests Using Basis Path Test

Dr. Ajay D. Shinde
Associate Professor, Department of Computer Studies, Chhatrapati Shahu Institute of Business Education and Research, Kolhapur, Maharashtra, India

ABSTRACT

Testing is process of executing a program with an intention to find out a yet undiscovered error. During software testing the aim should be to find out maximum number of error before release of software. This is possible only when each and every statement written in a program is executed at least once. The program structure is formed with control structures sequential, selection and iteration, how these statements are written defines the program structure. This paper presents use of basis path testing as a base for program structure testing; also it ensures that each and every statement will be executed at least once.

Keywords : Software Testing, White Box Testing, Basis Path Testing, Program Structure Testing, Testing techniques.

I. INTRODUCTION

Software testing is the procedure of executing a program or system with the intent of finding faults [5]. It is measured to be labor intensive and expensive, which accounts for > 50 % of the total cost of software development [6]. To test the software one should have knowledge about both the structure and functionality of the software. The technique based on functionality is known as black box testing whereas technique based on structure is known as white box testing. The white box testing technique is also known as [8]

- ✓ Glass box testing
- ✓ Clear box testing
- ✓ Open box testing
- ✓ Transparent box testing
- ✓ Structural testing
- ✓ Logic driven testing
- ✓ Design based testing

It includes various techniques such as basis path testing, Loop testing, Condition testing and Data flow testing. Here primary objective is to make sure that each statement is written in correct manner. The purpose of each technique is different, Basis path testing ensures that control flows correctly within the program, loop testing ensures that each loop is executed correctly within its bounds, condition testing ensures that each condition written contains correct expressions, relational and logical operators and brackets are placed correctly, Data flow testing ensures that correct data flows from one statement in a program to another statement in the same program. Now the question is, do the tester need to apply all white box testing techniques to the same program?. The answer for this question is NO, we don't need to apply all the techniques in fact only one technique is sufficient for all and it is basis path testing. In [10] Theresa Hunt has explained What is Basis Path Testing? According to [9] Basis path testing is a hybrid between path testing and branch testing:

Path Testing: Testing designed to execute all or selected paths through a computer program [IEEE610]
Branch Testing: Testing designed to execute each outcome of each decision point in a computer program [IEEE610]

Basis Path Testing: Testing that fulfills the requirements of branch testing & also tests all of the independent paths that could be used to construct any arbitrary path through the computer program [based on NIST]. Not every technique is full proof it will have certain advantages and disadvantages so is the white box.

There are certain advantages and disadvantages for white box testing.

The advantages for White-box testing:

- ✓ Side effects of having the knowledge of the source code are beneficial to thorough testing. [4]
- ✓ Optimization of code becomes easy as inconspicuous bottlenecks are exposed.[4]
- ✓ Gives the programmer introspection because developers carefully describe any new implementation.[4]
- ✓ Provides traceability of tests from the source, thereby allowing future chang the source to be easily captured in the newly added or modified tests.[3]
- ✓ Easy to automate.[2]
- ✓ Provides clear engineering-based rules for when to stop testing. [7][2]

Disadvantages:

- ✓ White-box testing brings complexity to testing because the tester must have knowledge of the program, including being a programmer. White-box testing requires a programmer with a high level of knowledge due to the complexity of the level of testing that needs to be done.[4]
- ✓ On some occasions, it is not realistic to be able to test every single existing condition of the application and some conditions will be untested.[4]
- ✓ The tests focus on the software as it exists, and missing functionality may not be discovered.

II. BASIS PATH TESTING

In [1] authors emphasize on importance of basis path testing by calling it as oldest structural testing technique. It is based on the control structure of the program and is one of the white boxes to testing technique. Basis path testing allows tester to test flow of control within the program. To test the program using basis path testing tester need to convert program to a graph showing flow of control known as control flow graph. Once the graph is drawn, independent paths are identified and for each path identified a test case need to be designed. This method was first proposed by Tom McCabe. To support number of independent paths identified cyclomatic complexity of the graph can also be calculated. The steps of implementation of basic path testing are given below:

- ✓ Convert your code/algorithm to a control flow graph.
- ✓ Find out independent path set
- ✓ Calculate the cyclomatic complexity of the control flow graph.
- ✓ Cyclomatic complexity and independent path set together define number of test cases need to designed, to execute each statement at least once.
- ✓ Finally design test cases to execute each path in independent path set by defining expected output for each input.

The notations used to draw a flow graph are as given below.

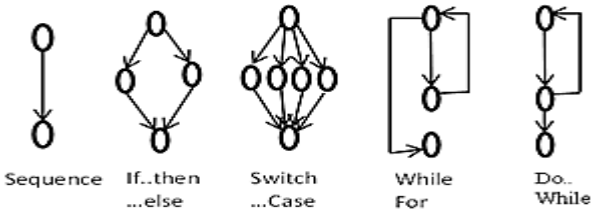

Figure 1. Notations used to draw a controlflow graph

III. HOW DO DRAW A CONTROL FLOWGRAPH

Let us see the process of converting a program/algorithm to a control flowgraph with the help of an example.

Consider the following algorithm, we will consider an algorithm for calculating the sum of even and odd numbers separately.

```
set lower_bound to 1, even_sum, odd_sum to 0
accept value for upper_bound
while lower_bound <= upper_bound
        if (lower_bound mod 2 = 0)
                even_sum = even_sum + lower_bound
        else
                odd_sum = odd_sum + lower_bound
        endif
        increment lower_bound by 1
endwhile
print even_sum, odd_sum
END
```

Figure 2. Algorith to calculate sum of even and odd numbers seperately

A. Convert algorithm to control flow graph

Assign numbers to each statement in algorithm the rules are

- ✓ every looping and selection statement should have a separate number
- ✓ statements written between if and else should have a separate number
- ✓ the statements from else to before endif should have a separate number.
- ✓ end of every control structure should have a separate number
- ✓ END indicates end of algorithm should have separate number (statements written before end of control structure and end of algorithm can be grouped together)

By following above rules

```
1.  Set lower_bound to 1, oddsum to 0, evensum to 0
    Accept value for upper_bound
2.  While lower_bound <= upper_bound
3.          If (lower_bound mod 2 = 0)
4.                  Evensum = evensum + lower_bound
5.          Else
                    Oddsum = oddsum + lower_bound
6.          Endif
            Increment lower_bound by 1
7.  Endwhile
    Print evensum, oddsum
8.  END
```

Figure 3. Assigning statement numbers following the rules given above

Use the statement numbers in algorithm as nodes in the graph and depending upon the flow of control draw arch's between them

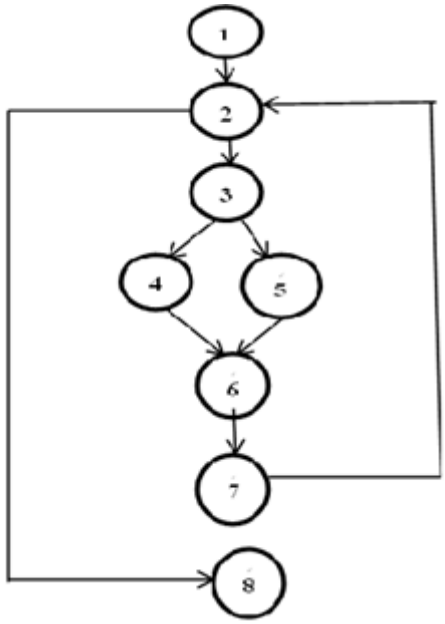

Figure 4. controlflow fraph for algorithm in Figure 3

B. Identifying independent path set

Independent path is the path to at least one new node

Path 1 : 1 – 8

Path 2 : 1-2-3-4-6-7-2-8

Path 3 : 1-2-3-5-6-7-2-8

We need to design three test cases for the given algorithm. To support this claim we must use another technique, to calculate the cyclomatic complexity of the graph.

C. Calculating cyclomatic complexity of the graph

Cyclomatic complexity = (Number of edges – number of nodes) + 2

= (9 – 8) + 2

= 3

Cyclomatic complexity = Number of nodes with predicate logic + 1

= 2 + 1

= 3

Note : Node 2 and 3 are nodes with predicate logic where flow of control decided based on condition.

Cyclomatic complexity = Number of regions formed in graph = 3

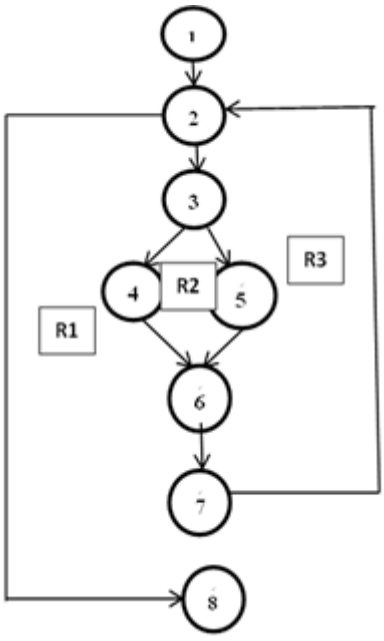

Figure 5. Graph showing number of regions

Now the total numbers of paths match with the cyclomatic complexity of the graph, so we need to design three test cases for the algorithm.

IV. DESGINING TESTS CASES TO COVER ALL TESTS UNDER WHITE BOX TESTING

Test case 1 : for path 1

Input : upper_bound value less than lower_bound

Expected Output : Program should terminate with the value of even_sum and odd_sum equal to zero.

At the time of execution of this test case, following white box tests can be covered

i. Statement 2 contains a condition the test is to be conducted for verifying and validating the condition written in statement this will cover condition testing part of white box testing.
ii. After execution of statement 1 control will flow to statement 2, we can ensure that correct data is passed from statement 1 to statement 2 this will cover data flow test.
iii. It will also cover loop testing technique as statement 2 implements a loop we can cover 1st part of loop testing that ensures output to be obtained when the loop is skipped completely.

Test case 2 : for path 2

Input : Value of lower_bound is equal to upper_bound and it is even number

Expected output : It should print the value of lower_bound(upper_bound) as even_sum and the value of odd_sum should be zero.

The second test case will cover following tests from white box testing technique

i. The second tests case will also ensure that the condition written in statement 2 is correct. This covers condition testing.
ii. Second test case also ensures that correct flows from statement 1 to statement 2.
iii. Statement 3 implements selection structure which implements a condition this test can be used to ensure that condition written in statement three is correct.
iv. This test will cover second step in loop testing it will ensure the output to be obtained when the loop is executed exactly once.

Test case 3 : for path 3

Input : The value of lower_bound is equal to upper_bound and it is odd number

Expected output : It should print the value of lower_bound(upper_bound) as odd_sum and the value of even_sum should be zero.

At the time of third test case execution following tests can be covered

i. Condition written in statement 2 can verified and validated
ii. Flow of data from statement 1 to statement 2
iii. Condition written in statement three
iv. Execution of looping statement within its bounds (Executing loop exactly once)

Test cases designed above will exercise conditions written in statement 2 and 3 to both true and false side. In addition to this test cases 2 and 3can be extended by giving different values for lower_bound and upper_bound, where lower_bound < upper_bound , this will become part of loop testing. It will check results when the loop is executed twice, thrice,n times and so on.

V. CONCLUSIONS

Theoretically, white box testing techniques may have different techniques for different purposes, but it is not necessary to conduct tests separately for each technique. The basis path testing technique which is one of the white box testing techniques is sufficient to carry out entire white box testing. As each path formed in the algorithm/program contains certain control structures and control structures are formed with the help of conditions and looping and selection statements. The test cases designed to execute each path formed in algorithm/program will ensure that it will cover condition testing, dataflow testing and loop testing. If path set identifies error handling paths the basis path testing will ensure error handling paths also. This

implies that if basis path test is conducted there is no need to conduct other tests under white box testing. Only one technique will be sufficient for all tests, there is no need to spend extra time and effort for conducting all tests under white box testing technique.

VI. REFERENCES

[1]. Aakanksha Rana and Ajmer Singh, A Comparative Study of Basis Path Testing and Graph Matrices, INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY, ISSN: 2277-9655, Publication Date: Jul 30, 2014, PP 746-750

[2]. Ammann, Paul; Offutt, Jeff (2008).Introduction to Software Testing.Cambridge University Press.ISBN 9780521880381.

[3]. Binder, Bob (2000).Testing Object-oriented Systems.Addison-Wesley Publishing Company Inc.

[4]. Ehmer Khan, Mohd (May 2010)."Different Forms of Software Testing Techniques for Finding Errors" (PDF).IJCSI International Journal of Computer Science Issues.7 (3): 12.Retrieved 12 February 2013.

[5]. G.J.Myers, T.Badgett, T.M.Thomas, and C.Sandler, The Art of Software Testing.John Wiley & Sons, 2004.

[6]. M.Sharma and B.S.Chandra, "Automatic Generation of Test Suites from Decision Table-Theory and Implementation," in Software Engineering Advances (ICSEA), 2010 Fifth International Conference on, 2010, pp.459-464.

[7]. Myers, Glenford (1979).The Art of Software Testing.John Wiley and Sons.

[8]. Software Testing by Cognizant Technology Solutions

[9]. www.westfallteam.com/sites/default/files/papers/Basis_Path_Testing_Paper.pdf