ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

A Comparative Study of Basis Path Testing and Graph Matrices

Aakanksha Rana, Ajmer Singh
Department of Computer Science & Engineering,
Deenbandhu Chhotu Ram University of Science & Technology, Murthal, Haryana, India
aakanksha8822@gmail.com

Abstract

Web engineers and stake holders are more concerned about competition in web application testing. Therefore various technical activities are carried out in the process of testing a web application. Basis Path testing and graph matrices are the two test case generation techniques of white box testing. So the idea is to generate and compare the test cases using the flow graphs of the basis path testing and the graph matrices. It is not easy to find out all the test cases in the program. This fundamental problem in testing thus throws an open question, as to what would be the strategy we should adopt to find the test cases for testing. In this paper, we have compared the test cases, generated by basis path testing and graph matrices.

Keywords: testing, test cases, path testing, graph matrices, white box testing, basis path testing, control flow graph.

Introduction

TESTING is a process of exercising software and same philosophy is also valid for Web Application too. Web application testing is a collection of related activities with the aim and objective of uncovering the errors in web application content, function, usability, navigability, performance, capacity and security. For testing, one should not wait for its whole development, instead of it the testing should be started at the time when you write one line of code. With the development of the software industry, software testing gradually takes a more important role in order to assure the software quality. White box testing, which is also called structural testing or logic-driven testing, mainly focus on the internal logic test of the code. Its objective is to achieve specific logical coverage indicator, such as statement coverage, condition coverage, branch coverage, basis path coverage and so on. Test cases are always designed according to the Control Flow Graphs (CFGs) [1].

A well tested software system will be validated by the customer before acceptance. The effectiveness of this validation and verification depends on number of errors found which in turns depend on the quality of test case generated.

The paper is structured as follows: Section II a review of basis path testing, Section III a review of graph matrices, Section IV a review of conclusions.

Basis path testing

Basis path testing is the oldest structural testing technique. The technique is based on the control structure of the program. Basis path testing is a white box testing technique that is used to test the code based on control flow. The method uses a control flowchart and a control flow graph to convert the code into a model and then derive independent test paths from it. Basis path testing is a white-box testing technique first proposed by Tom McCabe. Basis path testing was proposed by Thomas McCabe in the eighties of last century. Based on cyclomatic complexity measure to CFG and specific algorithm, independent basis paths can be created and test cases can be design according to these paths. The steps of implementation of basic path testing are given below:

1) Make the flow graph of the whole program structure.
2) Calculate the cyclomatic complexity of the given flow graph.
3) The cyclomatic number gives the total number of independent paths that we need to execute at least once.
4) Finally make the test cases by applying input conditions so as to check the output of the program structure.

Implementation of Basis Path Testing

Figure 1 shows the flow graph of the program structure. In this, the total numbers of edges are 10, and total numbers of nodes of are 8.

CFG describes the logic structure of software components. Each CFG consists of nodes and edges. The nodes represent computational statements or expressions and the nodes are denoted by a circle. These nodes are numbered or labeled where as the edges represent transfer of control between nodes, this is denoted by an arrow on the edge. A node with more than one arrow leaving it is called a decision node. Each possible execution path of a software module has a corresponding path from the entry to the exit node of the module's control flow graph. This correspondence is the foundation for the structured testing methodology. Areas bounded by edges and nodes are called regions. When counting regions, we include the area outside the graph as a region. Each node that contains a condition is called a predicate node and is characterized by two or more edges emanating from it [2].

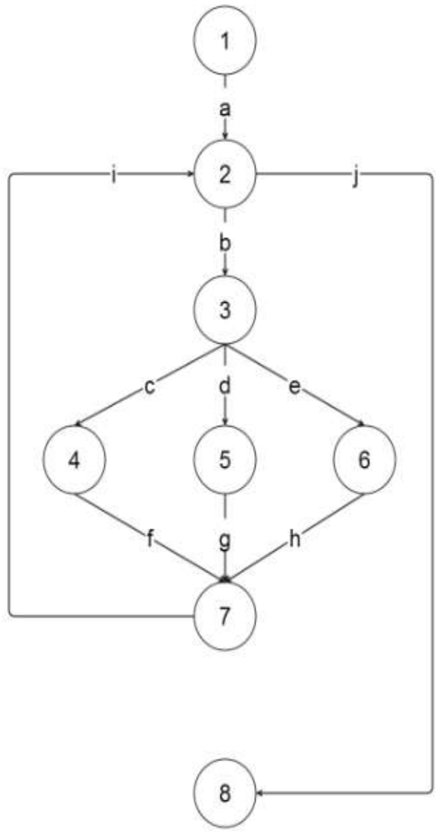

Figure1: Control Flow graph of program

1) Calculate Cyclomatic Complexity:

Cyclomatic complexity can be used to measure complexity of judge structure of a module. McCabe was also given calculation formula of complexity of a program structure. Cyclomatic complexity is also known as V (G) [3], where v refers to the cyclomatic number in graph theory and G indicates that the complexity is a function of the graph. According to graph theory, in a strongly connected directed graph G, the cyclomatic number is defined as V (G) = m – n + p, where m is number of arcs in the graph G, n is number of nodes, and p is number of strongly connected components. For a program that has a single entry and exit point, the entire module has only one program [3], then p = 1. Program control flow graphs are not strongly connected, but they become strongly connected when a "virtual edge" is added connecting the exit node to the entry node, thus the complexity number of the module is e – n + 1. Without the "virtual edge", control flow graphs can be served as undirected graphs, thus the cyclomatic number can be calculated like this: V (G) = e – n + 2 [3]. Cyclomatic complexity for figure 1 is calculated below:

a. The total number of regions in above flow graph is 4.
b. V (G) = 10 edges - 8 nodes + 2 = 2+2 = 4.
c. V (G) = 3 predicate nodes + 1 = 4.

2) Total number of independent paths:

a. Path1: 1-2-8
b. Path2: 1-2-3-4-7-2-8
c. Path3: 1-2-3-5-7-2-8
d. Path4: 1-2-3-6-7-2-8

Test Case Generated

Test case generated from the list of independent paths:

a. For Path 1, the input test sequences are: {aj}
b. For Path 2, the input test sequences are: {abcfij}
c. For Path 3, the input test sequences are: {abdgij}
d. For Path 4, the input test sequences are: {abehij}

Graph matrices

Flow graph is an effective method in path test testing however, path tracing with the use of flow graphs may be time consuming activity. Graph matrix, a data structure, is the solution which can assist in developing a tool for automation of path tracing. 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 [8]. 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).

ISSN: 2277-9655
Scientific Journal Impact Factor: 3.449
(ISRA), Impact Factor: 1.852

The following point describes a graph matrix:

1) Each cell in the matrix can be direct connection or link between one node to another node.
2) If there is a connection from node '*a*' to node '*b*', then it does not mean that there is connection from node '*b*' to node '*a*'.
3) Conventionally, to represent a graph matrix, digits are used for nodes and letter symbols for edges or connections.

A. Implementation of Graph Matrices

A graph matrix is a square matrix whose rows and columns are equal to the number of nodes in the flow graph. Each row and column identifies a particular node and matrix entries represent a connection between the nodes [8]. Following table 1 is the representation of flow graph into graph matrices:

TABLE1

	1	2	3	4	5	6	7	8
1		a						
2			b					j
3				c	d	e		
4							f	
5							g	
6							h	
7		i						
8								

1) Connection Matrix

Graph matrix is a tabular representation and does not provide any useful information. If we add link weights to each cell entry, then graph matrix can be used as a powerful tool in testing. The links between two nodes are assigned a link weight which becomes the entry in the cell of matrix. The link weight provides information about control flow. In the simplest form, when the connection exists [8], then the link weight is 1, otherwise 0 (But 0 is not entered in the cell entry of matrix to reduce the complexity). A matrix defined with link weights is called a connection matrix. The connection matrix for above example is shown below. The connection matrix for the table 1 is:

TABLE2

	1	2	3	4	5	6	7	8
1		1						
2			1					1
3				1	1	1		
4							1	
5							1	
6							1	
7		1						
8								

2) Use of Connection Matrix in finding Cyclomatic Complexity Number

Connection matrix is used to see the control flow of the program. Further, it is used to find the cyclomatic complexity number of the flow graph. Given below is the procedure to find the cyclomatic number from the connection matrix.

Step 1: For each row, count the number of 1s and write it in front of that row.

Step 2: Subtract 1 from that count. Ignore the blank rows, if any.

Step 3: Add the final count of each row.

Step 4: Add 1 to the sum calculated in step 3.

Step 5: The final sum in step 4 is the cyclomatic number of the graph.

The cyclomatic number calculated from the connection matrix of table 2 is shown below:

TABLE3

	1	2	3	4	5	6	7	8	
1		1							1-1=0
2			1					1	2-1=1
3				1	1	1			3-1=2
4							1		1-1=0
5							1		1-1=0
6							1		1-1=0
7		1							1-1=0
8									0-1=0
Cyclomatic number = 3+1=4									

3) Use of Graph Matrix for finding Set of all Paths

Another purpose of developing graph matrices is to produce a set of all paths between all nodes. It may be

of interest in path tracing to find *k*-link paths from one node. For example, how many 2-link paths are there from one node to another node? This process is done for every node resulting in the set of all paths. This set can be obtained with the help of matrix operations. The main objective is to use matrix operations to obtain the set of all paths between all nodes.
Next, we find 6-link set of paths of the above graph matrix as shown below:
TABLE4

	1	2	3	4	5	6	7	8
1			$ab^2i(cf+dg+eh)$					
2				$b^2ic(cf+dg+eh)$	$b^2id(cf+dg+eh)$	$b^2ie(cf+dg+eh)$		
3							$ib(cf+dg+eh)^2$	
4		$fbi^2(cf+dg+eh)$						
5		$gbi^2(cf+dg+eh)$						
6		$hbi^2(cf+dg+eh)$						
7			$i^2b^2(cf+dg+eh)$					
8								

B. *Test Case Generated*
Number of test cases by applying following input test sequences:

1. From node 1, the input test sequences are: {a, ab, aj, abc, abd, abe, ab(cf+dg+eh), abi(cf+dg+eh), ab^2i(cf+dg+eh)}.
2. From node 2, the input test sequences are: {b, j, bc, bd, be, b(cf+dg+eh), bi(cf+dg+eh), b^2i(cf+dg+eh), b^2ic(cf+dg+eh), b^2id(cf+dg+eh), b^2ie(cf+dg+eh)}.
3. From node 3, the input test sequences are: {c, d, e, (cf+dg+eh), i(cf+dg+eh), ib(cf+dg+eh), ij(cf+dg+eh), bci(cf+dg+eh), bdi(cf+dg+eh), bei(cf+dg+eh), ib$(cf+dg+eh)^2$}
4. From node 4, the input test sequences are: {f, if, ifb, ifj, fbci, fbdi, fbei, fbi(cf+dg+eh), fbi^2(cf+dg+eh)}.
5. From node 5, the input test sequences are: {g, ig, igb, igj, gbci, gbdi, gbei, ibg(cf+dg+eh), gbi^2(cf+dg+eh)}.
6. From node 6, the input test sequences are: {h, ih, ihb, ihj, hbci, hbdi, hbei, ibh(cf+dg+eh), hbi^2(cf+dg+eh)}
7. From node 7, the input test sequences are: {i, ib, ij, bci, bdi, bei, ib(cf+dg+eh), i^2b(cf+dg+eh), i^2b^2(cf+dg+eh)}.
8. From node 8, the input test sequences are: {0}.

Conclusion

By using the concept of basic path testing technique and graph matrices method, firstly the computational models of web application can be built. In this graph, web page's coding statements can be considered as nodes of control flow graph and links can be considered as input conditions provided at each node. And then with the help of basic path testing and graph matrices, total number of independent paths can be determined. Not only can this, but complexity of the program structure also be found out using this method. Next step is to generate test cases for the normal execution of the program.
In this paper, we have carried out, doing depth analysis of path testing, that no matter graph matrix method is time consuming but it covers more number of test sequences as compare to basis path testing. Graph matrices cover all the possible set of independent paths between all nodes. The set of all paths between all nodes is easily expressed in terms of matrix operations.
Next, graph matrix is the automated tool for finding set of all paths whereas basis path testing is a manual technique of finding the test sequences. Basis path testing is a fast method while graph matrix is a time consuming technique. But, the probability of errors is more in graph matrix than in basis path testing.

References

1. *Du Qingfeng, Dong Xiao, " An Improved Algorithm for Basis Path Testing", Shanghai, China, pp. 175-178, ©2011 IEEE*

2. *Arthur H. Watson and Thomas J. McCabe, "Structured testing: a testing methodology using the cyclomatic complexity metric," NIST Special Publication, September 1996.*
3. *Zhang Zhonglin, Mei Lingxia," An Improved Method of Acquiring Basis Path for Software Testing", in 5th International Conference on Computer Science & Education, Hefei, China. August 24–27, 2010, pp. 1891-1894, ©2010 IEEE*
4. *Rajiv Chopra, "Software Testing (A Practical Approach) Software Verification Software validation", Third Edition 2010, Katson Books, ISBN: 978-81-89757908, pp.113-114.*
5. *Beizer, B.,"Software Testing Techniques", New York: Van Nostrand Reinhold, 1983, pp. 37-73.*
6. *Du Qingfeng and Li Na, "White box test basic path algorithm," Computer Engineering, vol. 35, Augest 2009, pp. 100–102,123.*
7. *Zhongsheng Qian, Huaikou Miao & Hongwei Zeng, "A practical web testing model for web applications testing", Signal-Image Technologies and Internet-Based System, 2007, Third International IEEE Conference on 16-18 Dec. (2007), Shanghai , pp.434-441.*
8. *Roger S. Pressman, "Software engineering: A practitioner's approach", Sixth Edition, International Edition 2005, Mc Graw Hill, ISBN: 007-124083-7, pp. 595-600.*