Reusing Black Box Test Paths For White Box Testing of Websites

Rajiv Chopra
Computer Science Engg./ IT
GTBIT, GGSIPU DELHI
Delhi, India
raj_74chopra2004@yahoo.com

Sushila Madan
Computer Science Department
University of Delhi, Delhi
Delhi, India
sushila_lsr@yahoo.com

***Abstract*- As the numbers of web users are increasing exponentially, the software complexity is increasing exponentially and the malwares are increasing exponentially, so exhaustive and extensive testing of websites has become a necessity today. But testing of a website is not 100% exhaustive as the page explosion problem is also very usual. In this paper, we propose to reuse the basis test paths as obtained from the Page-Test-Trees (PTTs) for white box testing of websites. We traverse the same set of paths (obtained above) and test for the source code at these nodes. This saves significant amount of time required to generate test paths and hence test cases as compared to the existing approaches of white box testing. The cost and efforts are also minimized. The proposed technique ensures better website testing coverage as white box testing provides better results than black box testing. Then we validate the proposed reusability testing with two web navigational structures. The results show that doing regression testing can save several billion dollars. These test cases can be further minimized by using prioritization techniques of regression testing.**

Keywords- Website testing, Website-Under-Test (WUT), Page Flow Diagrams, Navigation, Page test Tree, Path Testing and Regression testing.

I. INTRODUCTION

Software testing is the process of executing the program with the intent of finding errors [1, 22]. Website testing, however, is more complex as the websites are becoming complex day by day. Software testing accounts for nearly 50% of the total development cost [10]. Exhaustive testing is not possible with websites. This is due to several reasons:-

- Numbers of Internet users are increasing exponentially.
- As websites become more complex, their cyclomatic complexity, V (G), also increases. A high V (G) implies lesser security [2]. The complexity explosion in software is exponential.
- Page explosion problem is also a serious issue.

From the website's navigational structure, we can draw its page-flow-diagram (PFD) and hence it's page-test-tree. From page-test-trees, test paths can be generated [3]. PFDs are used to compute the cyclomatic complexity, V (G). In this work , we compute V(G) of PFDs, reduce the number of paths based on all-path-coverage criteria of basis path testing approach of white box testing and finally reuse these paths and hence the test cases for white-box testing of websites. We have chosen path testing technique because path testing can alone detect almost 65% of the errors in the software [4]. Even if positive and negative tests are generated, yet there is no complete 100% of coverage. It is this white-box testing that helps to extend the test cases reasonably [5]. This work is an extension of earlier works on PFDs and PTTs of websites [3, 6].

In general, firstly black box testing is applied to test web components (Graphical User Interfaces) and white-box testing is done at later stages when source code starts creeping in. In this paper, the test-paths (and hence test cases) so generated after black-box testing are reused to do subsequent white-box testing using these paths only. This tests websites extensively. This has many significant benefits-

Test data is already available and also the test paths. Hence, we need to test source code at these points (nodes) only.

Branching nodes can be pruned; paths can be pruned, if required.

V (G) detects more security vulnerabilities and errors in website-under-test (WUT).

Black box testing covers more number of paths as compared to state based testing.

The main objective of present paper is to show the reusability of Black-box generated test paths for white-box testing of websites. Earlier the time taken to test website was the time taken to create the test paths for black box testing plus the time taken to get the paths for white box testing. But now the time taken to derive the test paths is only the time taken to derive the test paths from page test trees. In order to achieve this, we perform regression testing using these basis paths on two web navigational structures.

II RELATED WORK

Many researchers have proposed different testing approaches each of which has a different origin and pursuing different test goals for dealing with the unique characteristics of web applications. A few of related recent studies are stated below-

978-1-4673-4529-3/12/$31.00 ©2012 IEEE

In paper given by B.M. Subraya and S.V. Subramanya [11], a performance testing approach has been used to decompose the behavior of a website into testable components.

M. Benedikt [12] built a dynamic tool named as VeriWeb, to test web applications. VeriWeb's tool is based on graphs where nodes are Web pages and edges are explicit HTML links. But it never suggests using of these paths again for testing.

S. Elbaum uses the input data collected and remembered from previous user sessions and HTML forms. Better test cases are obtained [13].

The paper given by A. Andrews [14] proposes a method of deriving test cases from Finite State Machines. Even FSMs suffer from state explosion problem.

Fillipo Ricca and Paolo Tonella stress that Web page is a central entity in any website. Webpages can be static or dynamic. They use white-box testing criteria like page testing, All-uses testing and All-paths testing criteria are also applied on websites. They discuss about path expressions and node-reduction algorithms. They have developed tools names as Reweb and TestWeb. The All-path testing criterion is achieved. It is restricted to independent paths [15].

Eric Y. K. Chan and Y. T. Yu used a classification tree method (CTM) as the black-box criterion and "same path" as white box criterion. CTM are actually the hierarchical trees. However, no web pages are considered [16].

The paper given by Jingxian Gu, Lei Xu, Baowen Xu, Hongji Yang, applied traditional MM-path based strategy to component based web application. So, they extended MM-path testing method. They stressed that normally, MM-path always starts from the main function and ends at the main function but paths in component-based web application can start from one page and end at another page without return paths [17].

Bo Song and Huaikou Miao developed navigation model to generate test. But it is a directed graphs and graphs have problems like cycle. So, EFSM is not easy to use. It will generate a set of test cases with duplicity. So, a FSM test tree (FSM-TT) is proposed. Shorter test sequences can be generated without loss of states. But this sequence of paths has not been utilized further except for generation of black-box paths for testing [18].

Hui-Zhong Shi, Bo Chen and Ling Yu integrated black box and white-box testing to propose a generic framework of web security evaluation. No consideration on Basis Path testing is considered [19].

Nicha Kosindrdecha, JiraPun Daengdej aims to derive tests from state chart diagrams. This is a UML based diagram [20].

Lixin Wang divides a program into many segments with small cyclomatic complexity. But for every segment the paths are derived again and the focus is on white box testing alone [24].

Du Qingfeng, Dong Xiao use basis path testing alone to show how this method can be used for testing codes having switch statements also in addition to the 'if statements'. They also used only white box testing approach [7].

Thus after a long literature survey, we find that much of the work has been done on black box and white box testing of websites while much less has been done on reusability of paths of black box testing.

The proposed methodology is as follows-

1. Construct PFD (Page Flow Diagram) of website under test.
2. From PFD draw its PTT (page Test Tree).
3. Get independent basis paths to test.
4. Consider these paths again and conduct white-box testing (basis path testing) and generate test samples to carry out targeted testing. Use only independent paths.
5. According to vulnerabilities generate test reports.

III. CASE STUDIES

3.1 CASE STUDY – I

Consider an OnlineShopping.com website navigational structure. A test-path generation approach followed above is used. The navigation structure of this website is as follows-

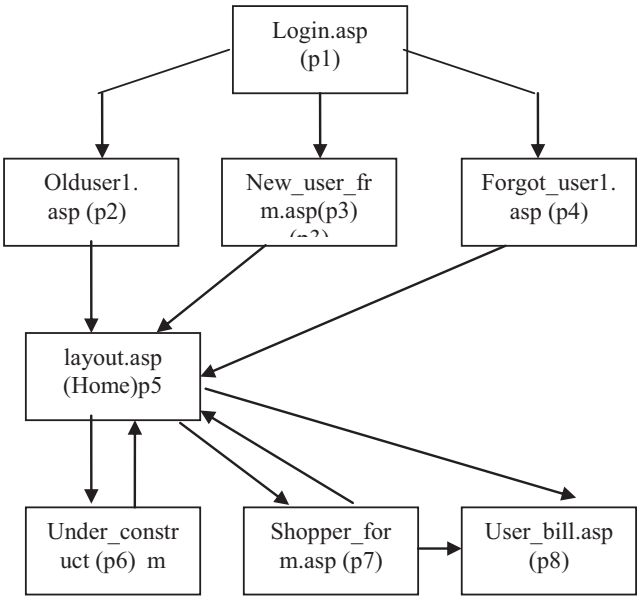

Fig. 1 A website navigation tree example

For this scenario, a Page Flow Diagram is firstly constructed.

So, we draw the PFD of the above web-project—Online Shopping.com :-

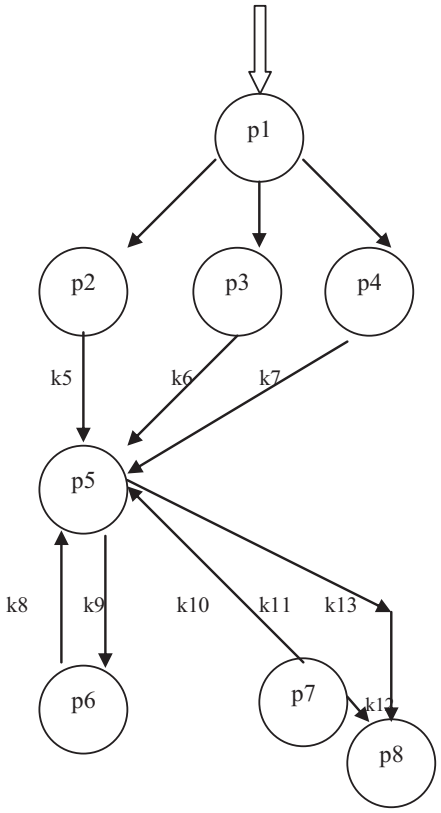

Fig. 2 Corresponding PFD of onlineShop website (block arrow shows first link entering into default page)

This PFD is essentially a graph. But graphs are sometimes problematic as they may contain cycles. This complicates the testing process. So, this PFD must be converted to a spanning tree. Hence, we get a Page Test Tree (PTT) based on our PFD. From PTT shorter paths can be generated. (than in PFD) without the loss of page and link coverage [3].

The construction algorithm (PFD-to-PTT) is as follows-

Consider two tables, T1 and T2 where-

a) First_table (T1): Has page identifiers (p1…….pn) which are unexplored or unmarked.

b) Second_table (T2): Has page identifiers (p1….pn) which have been explored from T1 and added to T2.

These are marked nodes.

Initially, both T1 and T2 are empty (Φ).

In our work, we apply the algorithm [3] to the project – OnlineShopping.com. For the PFD shown in Fig. 3, we analyze the contents of T1 and T2 as depicted in Table-1-

Table 1: Contents of T1 and T2 for case study-I

STEPS	T1	T2	COMMENTS
1	p1	Φ	Start with p1
2	p2,p3,p4	p1	p1 is explored
3	p3,p4,p5	p1,p2	p2 is explored
4	p4,p5,p5	p1,p2,p3	p3 explored
5	p5,p5,p5	P1,p2,p3,p4	p4 explored
6	p5,p5,p6,p7,p8	p1,p2,p3,p4,p5	p5 explored
7	p5,p6,p7,p8	P1,p2,p3,p4,p5	del p5 as it is in T2
8	p6,p7,p8	P1,p2,p3,p4,p5	del p5 as it is in T2
9	p7,p8,p5	p1,p2,p3,p4,p5,p6	p6 is explored
10	p7,p8	p1,p2,p3,p4,p5,p6	del p5 as it is in T2
11	p5,p8,p8	p1,p2,p3,p4,p5,p6,p7	p7 explored
12	p8,p8	p1,p2,p3,p4,p5,p6,p7	del p5 as it is in T2
13	p8	p1,p2,p3,p4,p5,p6,p7,p8	p8 explored
14	Φ	p1,p2,p3,p4,p5,p6,p7,p8	del p8 as it is in T2

Hence, we get the following PTT for our onlineShopping.com website-

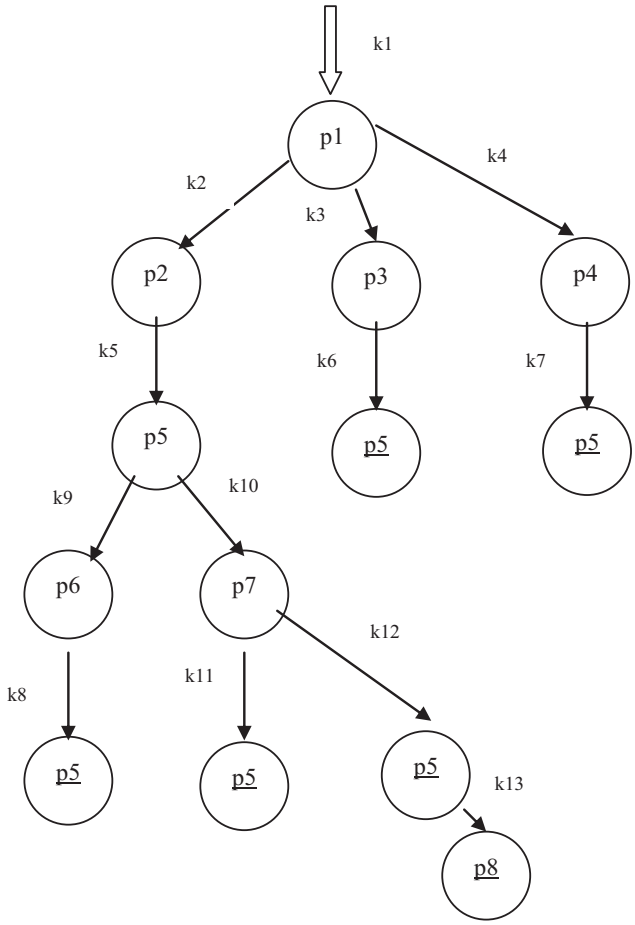

Fig. 3: PTT for PFD of fig.2 .

There are 13 links in Fig. 4 and 14 steps are needed for conversion from PFD to PTT.

A path is a possible sequence of links starting from the root-link and ending in any link during navigation [3]. A path is denoted by

k1->k2->k3->k4->…….kn; where ki is a link such that 1<=i<=n and n>0.

Hence, the test path is a path from the root link of the tree to a tail link. The objective is to find these paths. Path expressions directly represent all test paths in a tree; they can be used to generate sequences of links which satisfy any of the coverage criteria. To satisfy a given criterion, determine the minimum number of test paths from a path expression. A PTT is used to establish path expressions [21] where a path expression is defined as an algebraic representation of paths in a tree. Variables in the path expression are links. They can be combined using operators like + and *, that denote selection and loop respectively. Brackets may used to group sub path-expressions. For example,

k1->k2->k5->(k9+k10) is a path expression. In this example, a selection is encountered at k5 with

k9 = (p5, p6)

& k10 = (p5, p7)

In [21], B. Beizer has given a Node-Reduction algorithm for computing path expressions for a web application. These path expressions are used to get test-paths automatically and hence the test cases.

Since there are five (5) tail links in PTT of figure 4, so there are 5 test paths in this PTT. These test paths are as follows-

Path-1: k1->k2->k5->k9->k8
Path-2: k1->k2->k5->k10->k11
Path-3: k1->k2->k5->k10->k11->k13
Path-4: k1->k3->k6
Path-5: k1->k4->k7

The entire path expression for PTT of Fig. 4 is as follows—

k1->(k2->k5->k9->k8 + k10->(k11 + k12->(k13+k3--->(k6 + (k4->k7)))))

Since these five paths (from root-link to tail-link) cover all the links, so it is All-links-coverage strategy.

Also, since all the nodes underlined and the corresponding edges pointing to them are pruned (removed) from the tree, so we get the minimum number of test paths that cover all pages under consideration**.** This is All-pages-coverage strategy**.** Obviously, the test paths of All-links-coverage pass the test paths of All-pages-coverage[3].

But surprisingly, this website is a smaller one. Now –a-days, websites are very complex as they have numerous pages and links [22]. In such a scenario, it is difficult to proceed as the PFDs and PTTs will be very complex, large and unmanageable. So, exhaustive exploration is difficult to achieve. Hence, a divide-and-conquer strategy may be followed to divide a website into sub-websites and recursively applying this model of testing. For each sub-website, a corresponding PFD and PTT will be there. If there are n sub-websites then we will get {PFD_1 , PFD_2 , ……..PFD_n } and { PTT_1 , PTT_2 , PFD_n }, page flow diagrams and page test trees respectively. So, sub-websites can be conquered respectively [3]. Many other approaches exist in literature to solve page explosion problem- Pairwise coverage, Heuristic webpage testing, Using web crawlers.

The graph (PFD) of Fig. 3 is a sort of Control Flow Graph (CFG) that is not strongly connected. But they become strongly connected when a '*virtual edge*' is added connecting exit node to the entry node [21]. Now, we need to convert PFD of fig. 2 to a strongly connected graph as it is only then we can find cyclomatic complexity V(G) of this graph and hence go for white-box or non-functional or structural testing. Consider fig. 3 again. Observe that the entry node (p1) has an outdegree of 3. But this is problematic as it violates the definition of predicate nodes (with outdegree as 2). To adjust for this, a predicate node can be split into two subnodes as shown in Fig. 4.

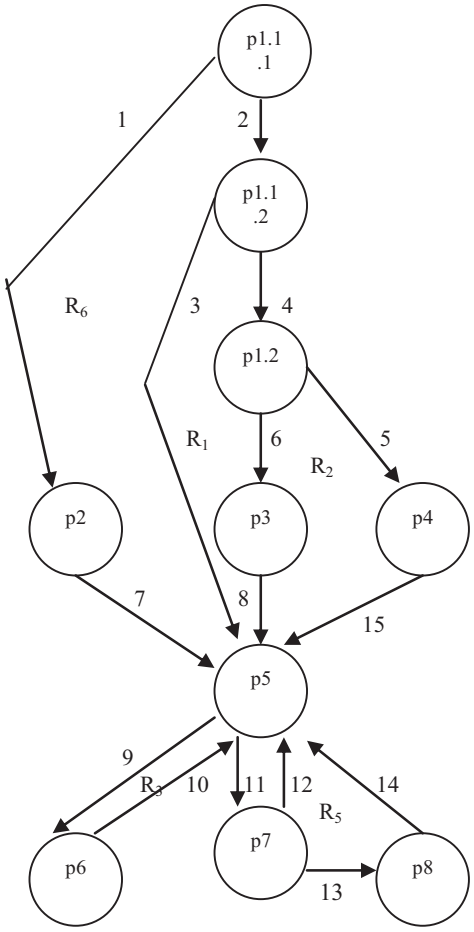

Fig. 4: Division of node p1

Now, the cyclomatic complexity, V (G) of this graph is:-

a) V(G) =e-n+1 = 13-9+1 = 5

b) V(G) = Number of enclosed regions + 1 =7

c) V(G) = P +1 = 4 +1 =5 [p1.1,p1.2,p5 and p7 are predicate nodes]

Now, this is incorrect V(G) as V(G) must be same by all the three methods. Even if we add a new edge (virtual edge), its addition also makes node p1.1 problematic. So, we divide it into sub nodes again as shown in Fig. 5.

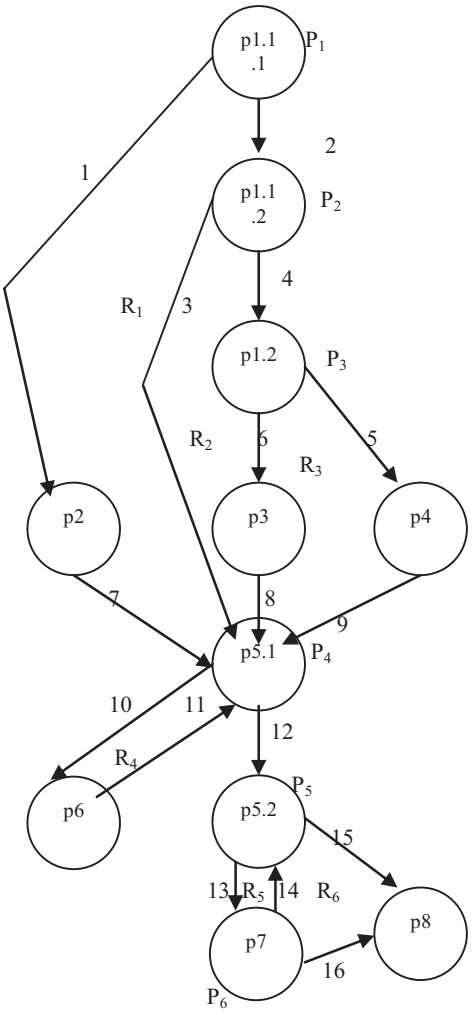

Fig. 5 Division of p5 node

Therefore, we find V (G) by all the three methods again:-

a) V(G) = e-n+2 = 16-11+2 = 5+2=7

b) V(G) = P+1 = 6+1 = 7 [Nodes P_1 ,P_2 ,P_3 ,P_4 P_5 and P_6]

c) V(G) = Number of enclosed regions +1=6+1=7

Hence, V (G) is same by all the three methods.

This implies that there must be 7 independent paths which is not the minimum number of paths and they are as follows:-

Path 1: p1->p2->p5->p6->p5

Path 2: p1->p3->p5->p7->p5

Path 3: p1->p3->p5->p8

Path 4: p1->p3->p5->p7->p5->p8

Path 5: p1->p4->p5->p8

Path 6: p1->p4->p5->p7->p5->p8

Path 7: p1->p3->p5->p7->p8

But we know that every path must introduce a new node. So, path-5 and path-7 are not required as they do not introduce any new node. That is, five paths are sufficient to test this website thoroughly. Hence, five test cases must be derived.

This result is same as that of PTT based approach (i.e. V (G) =5). This implies that we can reuse the paths obtained from PTTs for white-box testing directly.

3.2 CASE STUDY-II

To further verify this mixed approach of testing, consider another website structure. Its Page Flow Diagram (PFD) is a shown below-

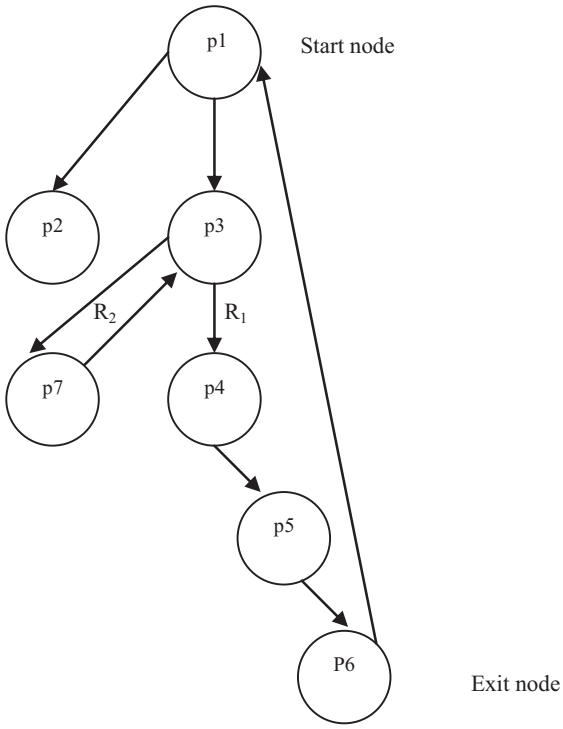

Fig. 6: Another PFD example

This PFD is converted to a Page Test Tree (PTT) using steps given earlier. Hence, we get the following PTT—

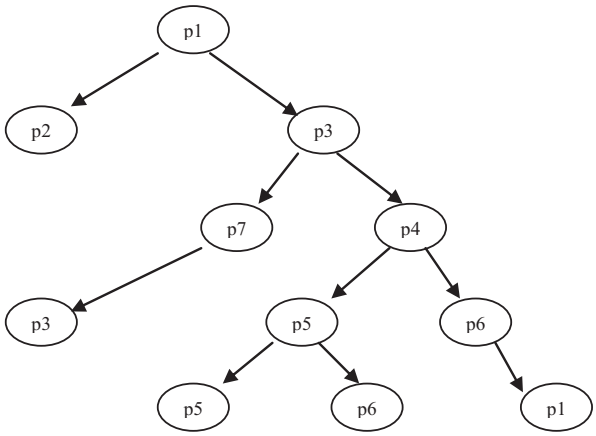

Fig. 7 PTT for case study-II

So, there are five test paths and are as follows:-

Path 1: p1->p2

Path 2: p1->p3->p7->p3

Path 3: p1->p3->p4->p5->p5

Path 4: p1->p3->p4->p5->p6->p1

Path 5: p1->p3->p4->p6->p1

But here path-3 and path-5 are not required to be traversed as they are already covered in other paths. So, three (3) paths are sufficient to test this website.

From PFD above,

a) V(G) = e-n+2 = 8-7+2 = 1+2 = 3

b) V(G) = P+1 = 2+1 = 3

c) V(G) = Number of enclosed regions + 1 =2+1 = 3

Therefore, V (G) is same by all the three methods. This is also equal to the paths obtained from PFD. This means that we can directly use the paths obtained from PTTs for white-box testing too.

V CONCLUSIONS

In this paper, we have shown that the test-paths derived from black-box testing of websites can be further utilized to do white-box testing of any website under test. All earlier works do not utilize these paths so extensively. So, in this paper we extend the same black box paths for structural testing using basis path analysis technique. This saves cost, time and manpower. There are several directions for future works. Using this methodology for AJAX based applications, test case generations, test case reduction and test driven development based on the test paths are some of them.

REFERENCES

[1] Glenford J. Myers, "The Art of Software Testing", Second Edition, Wiley India Pvt. Ltd., 2009.

[2] www.mccabe.com/ Path Insensitive Insecurity.

[3] Zhongsheng Qian, Huaikou Miao, Hongwei Zeng, "Apractical web testing model for web application testing", Third International IEEE conference on Signal-Image Technologies nd Internet-Based System, 2008.

[4] B. W. Kernighan, P. J. Plauger, "the elements of Programming Style", Mc Graw-Hill, Inc., New York, NY,USA,1982.

[5] Andre Takeshi Endo, Michael Linschulte, Adenilso da Silva Simao and Simone do Rocio Senger de Souza, "Event and Coverage-based testing of web services", Fourth IEEE International Conference on Secure Software Integration and Reliability Improvement Companion, 2010.

[6] Huaikou Miao, Zhongsheng Qian, Bo Song, "Towards Automatically Generating Test Paths for Web Applications", IEEE Symposium on Theoretical Aspects of Software Engineering, 2008.

[7] Du Quingfeng, Dong Xiao, "An Improved Algorithm for basis Path Testing" , IEEE 2011.

[8] Anoj Kumar, Shailesh Tiwari, K K Mishra and A K Mishra, " Generation of Efficient test Data using Path Selection Strategy with Elitist GA in Regression testing", IEEE 2010.

[9] Irman Hermadi, Chris Lokan and Rahul Sarker, "Genetic Algorithm based Path Testing: challenges and Key parameters", IEEE, Second WRI World Congress on Software Engineering, 2010.

[10] Sangeeta Sabharwal, Ritu Sibal, Chayanika Sharma, "A Genetic Algorithm based Approach for Prioritization of test case scenarios in static testing", International Conference on Computer and Communication Technology(ICCCT)-2011 (IEEE).

[11] B. M. Subraya , S.V. Subrahmanya, "Object driven Performance testing of Web Applications", The first Asia-Pacific Conference on Quality software, HongKong, China, Oct. 2000.

[12] M. Benedikt, J. Freire and P. Godefroid, " VeriWeb: automatically testing Dynamic Websites", In Proceedings of 11th International WWW Conference, Honolulu, HI, May 2002.

[13] S. Elbaum, S. Karre and G. Rothermal, " Improving Web Application with User Session Data" in the proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, May 2003, p-p 49-59.

[14] A. Andrews, J. Offut and R. Alexender , "Testing web Applications by Modeling with FSMs", Software and Systems Modeling, 2004.

[15] Fillippo Ricca and Paolo Tonella, "Analysis and Testing of Web Applications", ITC-irst, Centro per la Ricerca Scientificae technologica , Italy, IEEE 2001.

[16] Eric Y. K. Chan and Y. T. Yu, "Evaluating several Path-based Partial Dynamic Analysis Methods for selecting Black-Box generated Test Cases", Fourth International Conference on Quality Software ,IEEE 2004.

[17] Jingxian Gu, Lei Xu, Baowen Xu, Hongji Yang, "An Extended MM-path Approach to Component-based Web Application Testing", 12th IEEE International Workshop on Future Trends of Distributed Computing Systems, IEEE 2008.

[18] Bo Song, HuaiKou Miao, "Modelling Web Applications and Generating Tests: A combination and Interactions-guided Approach", 3rd IEEE International Symposium on Theoretical Aspects of Software Engineering, IEEE 2009.

[19] Hui-Zhong Shi, Bo Chen and Ling Yu , "Analysis of Web Security Comprehensive Evaluation Tools", 2nd International Conference on Network Security, Wireless Communications and Trusted Computing, IEEE 2010.

[20] Nicha K, JiraPun Daengdej . "A Black box test case generation", International Journal of Computer Science and Information Security (IJCSIS), September 2010.

[21] B. Beizer, "Software Testing Techniques", Second Edition, dreamtech Press, 2009.

[22] Rajiv chopra, "Software Testing-A Practical Approach", Fourth Edition, Katsons, 2013.

[23] K. Mustafa, R. A. Khan, "Software Testing – Concepts and Practices", Narosa Publishing House, 2007.

[24] Lixin Wang, "A Program Segmentation Method for Testing Data Generating Based on Path Coverage", IEEE, 2010.