International Journal of Engineering and Management Research, Volume-3, Issue-3, June 2013
ISSN No.: 2250-0758
Pages: 1-4
www.ijemr.net

Web Applications Testing using White Box Method

Kanika Goswami.
M.Tech(Computer Science & Engineering)

ABSTRACT

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. Web testing is a promising technique to ensure the high quality of web application. The data flow information of the web application was captured using flow graphs. And the internal structure, design and implementation of the web application can be tested using white box testing in which the tester chooses inputs to exercise paths through the code and determines the appropriate outputs. In this flow graph web pages can be considered as node of a graph and links as input conditions provided at each node. With the help of basic path testing and graph matrices, the number of independent paths are generated for the flow graph and then finally test cases are derived and tested through mutation testing method.

Keywords—. White box testing; complexity; web testing; graph matrices; mutation testing.

I. INTRODUCTION

Testing is a process of exercising software and same fundamental philosophy is also valid for Web applications too. Web applications are a significant challenge for Web engineers. Because Web based applications reside on large networks and interact with many operating systems, browsers, hardware platforms, communication protocols, etc While doing web testing, Web engineers focus firstly on user-visible aspects of Web application and then they will proceed to tests that will check technology used in developing Web application. The testing process begins with exercising content and interface functionality of a Webapp because that is immediately visible to end users. After this the aspects of design architecture and navigation are exercised. Now the testing focus is shifted to exercising technological capabilities that are not always apparent to end-users which include Webapp infrastructure and installation or implementation issues. The work product for testing a web application is firstly a test plan will be produced. Then a suite of test cases will be developed testing each and every step and finally an archive of test results will be maintained for future use There are many quality dimensions that are relevant in any discussion of testing a Web application:

Content: It is assessed at both semantic and syntactic level. At syntactic level, punctuation, grammar, spellings are checked for text based applications. At semantic level, correctness, consistency and lack of ambiguity are all checked.

Function: Functions are tested to remove errors that indicate lack of conformance to customer requirements. In this each function of Web application is assessed for correctness, instability and conformance to implementation standards.[1]

Structure: Structure of a Web application is assessed to ensure proper delivery of its content and function. Its extensibility is also assessed so that new content and functionality can be added easily.

Usability: It is assessed to ensure that interface should support every category of user and also a user can learn and apply all navigation syntax and semantics through that interface.

Navigability: It is tested to uncover all type of navigation errors i.e. dead links, improper links, erroneous links, etc and also to ensure that all navigation syntax and semantics are properly exercised.

Performance: It is tested to ensure that the system is responsive to user interaction and can be able to handle loading without any operational degradation. Performance of a Web application is tested under a variety of operational conditions and configurations.

Compatibility: It can be tested by executing the Web application for a variety of host configurations on both server and client sides. In this we try to find out errors that are specific to a unique host configuration.[1]

Interoperability: It is tested to ensure that interfacing of a Web application with other applications and databases should be proper.

Security: Security failure may occur if the penetration attempts become successful. So security can be tested by assessing potential vulnerabilities and attempting to exploit each.

White -Box testing techniques

White-box testing, sometimes called glass-box testing is a test case design method that uses the control structure of the procedural design to derive test cases [2]. It is a security testing method that can be used to validate whether code implementation follows intended design, to validate implemented security functionality, and to uncover exploitable vulnerabilities. Two types of white-box testing technique used in this paper are: Basic Path testing and Graph matrices method.

Basis Path Testing

Basis path testing is proposed by Tom McCabe. The basis path method enables the test case designer to derive a logical complexity measure of a procedural design and use this measure as a guide for defining a basis set of execution paths. Test cases derived to exercise the basis set are guaranteed to execute every statement in the program at least one time during testing [2].

Graph Matrices

Graph matrix, a data structure, is the solution which can assist in developing a tool for automation of path tracing. Graph theorems can be proved easily with the help of graph matrices. So graph matrices are very useful understanding the testing theory [3]. A graph matrix is a square matrix whose rows and columns are equal to the number of nodes in the flow graph.

II. IMPLEMENTATION OF BASIC PATH TESTING

By taking the example of internet banking application of Oriental bank of Commerce we can implement the basic path testing approach on the real application. We use Finite State Model for internet banking because, Finite state machines defines what actions a process is allowed to take, which events it expects to happen, and how it will respond to those events Fig 2.1 shows the finite state model for internet banking application.[4] 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.

Flow graph of the internet banking application.

Fig 2.2 shows the flow graph of the whole program structure. In this total number of edges are 21, total number of nodes are 15.

Calculate cyclomatic complexity:

1. The total number of regions in a given flow graph is 8.
2. V(G) = 21 edges - 15 nodes + 2 = 6+2 = 8.
3. V(G) = 7 predicate nodes + 1 = 8. The predicate node is the one where we need to take the decision which path we have to follow to reach the final node of the graph because from the predicate node there exists various paths. Predicate nodes are: node 4, node 5, node 6, node 7, node 8, node 11, and node 15.

Total number of independent paths:

1) Path1: 1-2-3-4-6-14-15.
2) Path2: 1-2-3-4-7-11-13-14-15.
3) Path3: 1-2-3-4-7-12-14-15.
4) Path4: 1-2-3-4-5-9-4-6-14-15.
5) Path5: 1-2-3-4-9-4-7-11-13-14-15.
6) Path6: 1-2-3-4-5-9-4-7-12-14-15.
7) Path7: 1-2-3-4-8-10-4-6-14-15.
8) Path8: 1-2-3-4-8-10-4-7-11-13-14-15.
9) Path9: 1-2-3-4-8-10-4-7-12-14-15.

Number of test cases by applying following input test sequences:

1) For Path 1, the input test sequences are: {abcenp, abcfnp}.
2) For Path 2, the input test sequences are: {abcgkonp, abcglonp}.
3) For Path 3, the input test sequences are: {abcgjnp, abcgmnp}.
4) For Path 4, the input test sequences are: {abcdiqenp, abcdiqfnp}.
5) For Path 5, the input test sequences are: {abcdiqghonp, abcdiqglonp}.
6) For Path 6, the input test sequences are: {abcdiqgjnp, abcdiqgmnp}.
7) For Path 7, the input test sequences are: {abchjqenp, abchjqfnp}.
8) For Path 8, the input test sequences are: {abchjqgkonp, abchjqglonp}.
9) For Path 9, the input test sequences are: {abchjqgjnp, abchjqgmnp}.

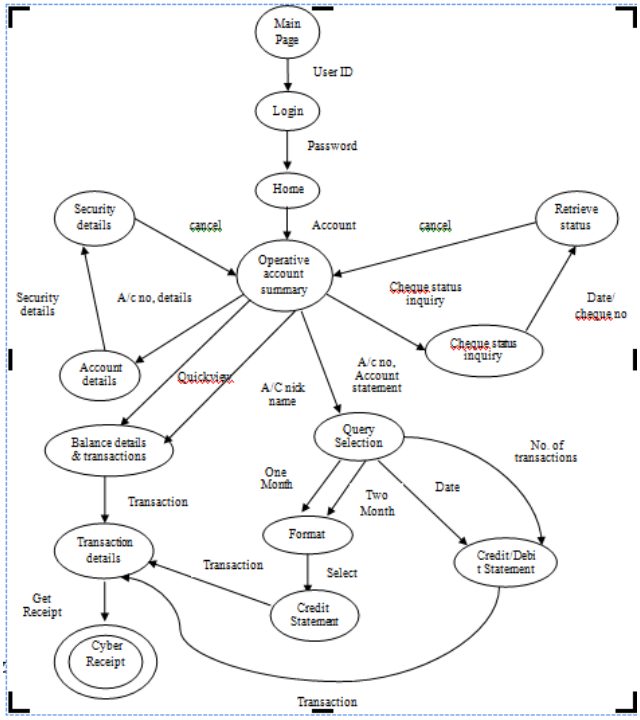

Fig 2.1: Finite state model of Net Banking Application.

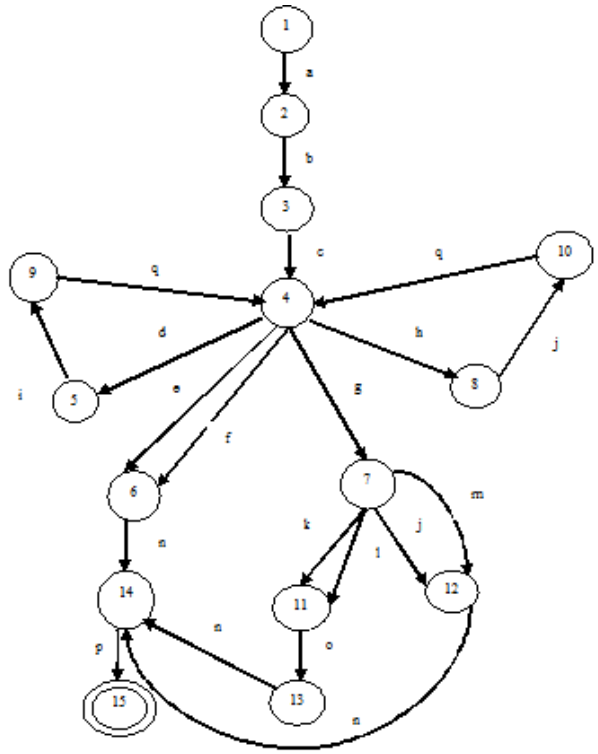

Fig 2.2 **Flow graph of net banking application.**

III. IMPLEMENTATION OF GRAPH MATRICES

Representation of Flow Graph into graph matrix:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1		a													
2			B												
3				c											
4					d	e,f	g	h							
5									i						
6														n	
7											k,l	j,m			
8										j					
9				q											
10				q											
11													o		
12														n	
13														n	
14															p
15															

Fig 7.2 Graph Matrix

The main objective is to use matrix operations to obtain the set of all paths between all nodes. It can be generalized that for *n* number of nodes, we can get the set of all paths of (*n-1*) links length with the use of matrix operations. [4] So for 15 nodes, we get 14 link paths.

Number of input test sequences after applying matrix multiplication:

1) From node 1, the input test sequences are: { **abc((di+hj)q)^{2}g(k+l)onp, abc((di+hj)q)^{2}g(j+m)np, abc((di+hj)q)^2(e+f)np** }.
2) From node 2, the input test sequences are: { **bc((di+hj)q)^3(e+f)np, bc((di+hj)q)^{2}g(k+l)onp, bc((di+hj)q)^{2}g(j+m)np** }.
3) From node 3, the input test sequences are: { **c((di+hj)q)^{3}g(j+m)np, c((di+hj)q)^3(e+f)np, c((di+hj)q)^{2}g(k+l)onp** }.
4) From node 4, the input test sequences are: {**((di+hj)q)^{3}g(k+l)onp, ((di+hj)q)^{3}g(j+m)np, ((di+hj)q)^3(e+f)np** }.
5) From node 5, the input test sequences are: { **iq((di+hj)q)^3(e+f)np, iq((di+hj)q)^{2}g(k+l)onp, iq((di+hj)q)^{2}g(j+m)np** }.
6) From node 6, the input test sequences are: {**np**}
7) From node 7, the input test sequences are: { **(j+m)np, (k+l)onp**}
8) From node 8, the input test sequences are: { **jq((di+hj)q)^3(e+f)np, jq((di+hj)q)^{2}g(k+l)onp, jq((di+hj)q)^{2}g(j+m)np** }.
9) From node 9, the input test sequences are: { **q((di+hj)q)^{3}g(j+m)np, q((di+hj)q)^3(e+f)np, q((di+hj)q)^{2}g(k+l)onp** }.
10) From node 10, the input test sequences are: { **q((di+hj)q)^{3}g(j+m)np, q((di+hj)q)^3(e+f)np, q((di+hj)q)^{2}g(k+l)onp** }.

IV. OUTPUT

Fig 4.1: Get Cyber Receipt of transaction details

V. COMPARISON OF BASIC PATH TESTING AND GRAPH MATRICES METHOD

1. Basic path testing is a manula technique of finding the test sequences while graph matrix method is the automated tool for finding set of all the paths.
2. Number of test sequences is less in basic path testing while number of test sequences are more in graph matrix method.
3. Basic path testing is less efficient method than graph matrix technique.
4. Basic path testing is a fast method while graph matrix is a time-consuming technique.
5. Probability of errors are more in graph matrix and less in basic path tesing.

VI. CONCLUSION AND FUTURE SCOPE

White box testing is a method in which the internal structure/design/implementation of the item being tested is known to the tester. By using the concept of basic path testing technique, firstly a computational model of web application can be built. In this model, web pages can be considered as states of finite state model and links can be considered as input conditions provided at each state. 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. We can insert the mutants or wrong input conditions to check the validity and security of the web applications. If the mutant is alive after execution then we upgrade our test cases otherwise if the mutant is killed then our web application is authenticated and running properly and its testing are done successfully.

REFERENCES

[1]. Roger S. Pressman , Sixth Edition, International Edition 2005, Mc Graw Hill, "Software engineering : A practitioner's approach", ISBN**:** 007-124083-7, pp. 595-600.

[2]. http://www.codeproject.com/Articles/37111/White-Box-Testing-Technique

[3]. Naresh Chauhan, Oxford University Press, Edition: Paperback, "SOFTWARE TESTING Principles and Practices", EAN: 9780198061847.

[4]. https://www.**obc**online.co.in/