User Session Based Test Case Generation for Website: A Review

Shruti N. Pardeshi[*], Vaishali Chourey[**]

[*] Research Scholar, Medicaps Group of Institutions, Indore
[**] Associate Professor, Medicaps Group of Institutions, Indore

Abstract- The existing testing approaches for web application testing can takes long time to generate test suites. So that the User Session Based Testing has introduced which is depends on capturing and replaying user sessions of a website. It is an effective testing methodology for web applications. Previous studies have exposed that the testing suites generated by user session based testing frequently find the faults neglected by other testing approaches. This paper addresses the problem and discussing different techniques and methodologies have been proposed for taking care of their issues. Genetic Algorithm (GA) is one such form of evolutionary algorithms which is the area of research. In this paper we discuss the algorithms based on genetic algorithm and in future we will try to resolve its problem.

Index Terms- Web application testing, User session based testing, Web server, Session log

I. Introduction

Web Application

Today web application is the most significant and fastest growing field of software systems. Web applications may support a wide range of activities including education, research, e-commerce, business, sales, medical etc. It is important for Web applications to be dependable, but recent studies conclude that they are not dependable. [1]

Below Figure 1.1 shows how a simple web application operates by taking example of online shopping. In it customer can send a request through a web browser by sending website address to web server. It can responds by sending the front page for the given URL that is static page of the web application. All these contents are taken form the markup language which is further interpreted by the browser to render a web page at the user site.

Some requests requires additional infrastructure to execute their complex task. For example, in an online shopping site as shown in the below figure a request may be of different type client may request for the product details to the web server as per the request from the client web server response with respective pages or information. If customer place request to order a particular product then first stock has to be verify for getting the availability of product.

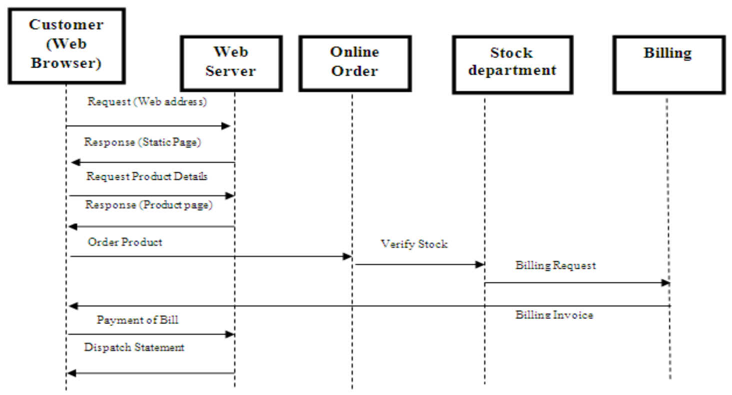


Figure 1.1: Sequence diagram of a web application

After checking availability bill is generated and then forwarded to the client in response to request from it. A set of name-value pairs (input fields' names and their values) is generated by translating information and it becomes part of the request.

Although the web server receives any request and it requires further elements to complete the process. According to the client requests and elements requires it generate pages by running its respective script. The newly generated page, generated at run time and depending on user's input, is called a dynamic web page [53].

Web Application Testing
Definition:

"Web application testing, is a software testing technique, exclusively implemented to test the applications that are hosted on web in which the application interfaces and other functionalities are tested". [16]

The aim of Web application testing is to execute application by trying different input values and state to determine failures. A failure is detected for finding the incapability of a system or for verifying the performance requirements of the application [16]. Generally, failure is nothing but the faults in the application itself .Fault may be in application or its running environment or in the interface between the environment and the application on which it runs.

Since a web applications execution is depends on it running environment it should be test differently to detect failure. To discover the various failures the different types of testing have to be executed [11]. Because of the functional requirement of the application the running environment affects the non-functional requirements of a Web application [1] [2].

User Session Based Testing

White box web application testing techniques has limitation such as Ricca and Tonella's technique has cost of finding inputs that exercise the system as desired. For the selection of such inputs is slow and must be accomplished manually. This problem can be solved by transparently collecting user session log and generating test cases from it [1].

Web application is a combination of static and dynamic web pages. Web server generates dynamic pages according to the request from the client. Changing user summaries and frequent minor maintenance changes complicate automated testing. Cost of creating test cases using white box testing is higher. It is easy and cheaper to collect user sessions and generates test cases from it.

In the form of URLs and name-values pair this technique captures and stores the client's requests, and then applies policies to create test cases. Generally, a user session starts when a request from a new IP address reaches the server and finished when the user leaves the web sites or when the session times out. Every logged request of the user session is changed into HTTP request that can be sent to a web server and it transforms into test case. A set of HTTP requests that are associated with each user session is contained within the test case. For collected user session several approaches are applied to generate test cases.

Generalized Framework

Below Figure1.2 shows the generalized framework for User Session Based Testing which is further described below.

1. User Request

User can use the services by sending request via various links provided on website. The various request types explained below:

i. GET

For the specified resource or service it will send request. By using GET request only data can be retrieve nothing else.

ii. POST

It requests to the server that it accept the entity enclosed in the request as a new subordinate of the web resource recognized by the URI. And the posted data might be, for example, an explanation for existing resources, a message for a newsgroup, mailing list and a block of data that is the result of submitting a web form to a data handling process or an item to add to a database.

iii. CONNECT

It converts the request connection to a transparent TCP/IP tunnel, usually to facilitate SSL-encrypted communication (HTTPS) through an unencrypted HTTP proxy.

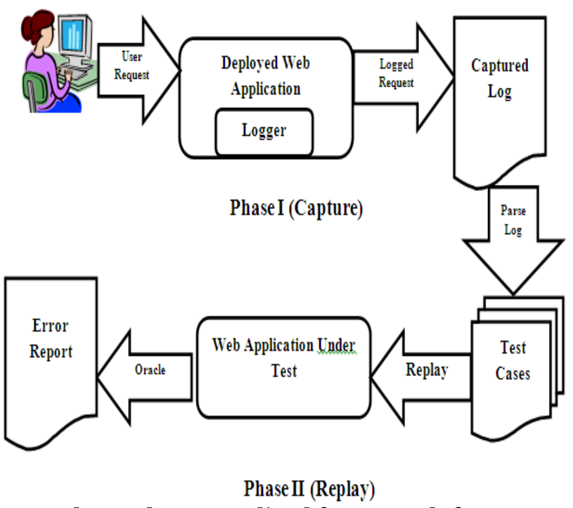


Figure1.2 shows the generalized framework for User Session Based Testing

II. Deployed Web Application

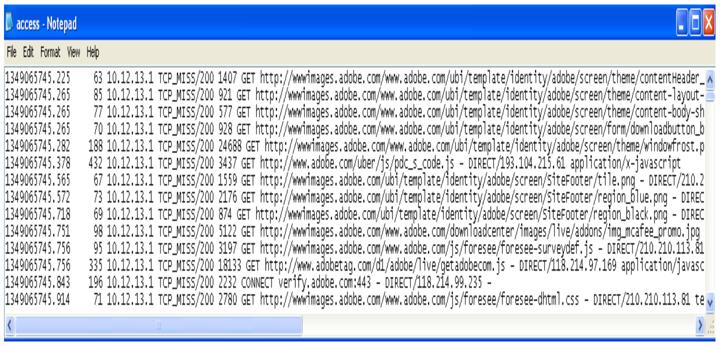
To make all of the activities available for use the web application is deployed on the web server. Some correlated activities are involved in deployment process with possible transitions between them. These activities can occur at the producer side or at the consumer side or both sides.

III. Captured Log

A request for a file made by a user is called as Hit. Web browsers and search engine indexing programs are included by user. Every time a user view a webpage it requests the separate files that make up the page from the web server where the website is stored. In the web server the record of the hits received is automatically created and saved and it is called as web server log. Log can be treated and the resulting statistics interpreted to build a representation of how people might be using a website

Each time a person views a webpage a request is sent from their computer to the computer where the website is stored (server). These requests are for the individual files that make-up the webpage. The web server log is a record of these requests. Figure 1.3 shows the sample access.log file. Statistics included in Server log:

- The type of browser used to view the website
- The details of the service the person uses to connect to the internet, including an IP address
- Their country of origin
- If the person linked to the website from a search engine, the words they searched for (*query*)
- The webpages viewed
- The time spent on the website
- Sample path is the order in which webpages were viewed



```
1349065745.225    63 10.12.13.1 TCP_MISS/200 1407 GET http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen/theme/contentHeader_
1349065745.265    85 10.12.13.1 TCP_MISS/200 921 GET http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen/theme/content-layout-
1349065745.265    77 10.12.13.1 TCP_MISS/200 577 GET http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen/theme/content-body-sh
1349065745.265    70 10.12.13.1 TCP_MISS/200 928 GET http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen/form/downloadbutton_b
1349065745.282   188 10.12.13.1 TCP_MISS/200 24688 GET http://wwwimages.adobe.com/www.adobe.com/ubi/template/identity/adobe/screen/theme/windowfrost.p
1349065745.378   432 10.12.13.1 TCP_MISS/200 3437 GET http://www.adobe.com/uber/js/pdc_s_code.js - DIRECT/193.104.215.61 application/x-javascript
1349065745.565    67 10.12.13.1 TCP_MISS/200 1559 GET http://wwwimages.adobe.com/ubi/template/identity/adobe/screen/SiteFooter/tile.png - DIRECT/210.2
1349065745.572    73 10.12.13.1 TCP_MISS/200 2176 GET http://wwwimages.adobe.com/ubi/template/identity/adobe/screen/SiteFooter/region_blue.png - DIREC
1349065745.718    69 10.12.13.1 TCP_MISS/200 874 GET http://wwwimages.adobe.com/ubi/template/identity/adobe/screen/SiteFooter/region_black.png - DIREC
1349065745.751    98 10.12.13.1 TCP_MISS/200 5122 GET http://wwwimages.adobe.com/www.adobe.com/downloadcenter/images/live/addons/img_mcafee_promo.jpg
1349065745.756    95 10.12.13.1 TCP_MISS/200 3197 GET http://wwwimages.adobe.com/www.adobe.com/js/foresee/foresee-surveydef.js - DIRECT/210.210.113.81
1349065745.756   335 10.12.13.1 TCP_MISS/200 18133 GET http://www.adobetag.com/d1/adobe/live/getadobecom.js - DIRECT/118.214.97.169 application/javasc
1349065745.843   196 10.12.13.1 TCP_MISS/200 2232 CONNECT verify.adobe.com:443 - DIRECT/118.214.99.235 -
1349065745.914    71 10.12.13.1 TCP_MISS/200 2780 GET http://wwwimages.adobe.com/www.adobe.com/js/foresee/foresee-dhtml.css - DIRECT/210.210.113.81 te
```

Figure 1.3 : Access.log

IV. Test Cases

Server maintains the log of all the users accessing the website. These sessions are parsed into test cases. There are several algorithms available to generate test case. That we will study in next chapter.

V. Web Application Under Test

In this phase the generated test cases are tested for its Functionality test, Coverage test, Block test.

VI. Error Report

According to the result of above block the report is generated in the form of tables and graphs.

Test case generation by User Session Based Testing

Generalized Algorithm

1. Access Record {User IP address, Request time, Request mode (GET, POST, CONNECT), URL, Data Transport Protocol (HTTP), Status code, No. of bytes transferred and the type of client}.
2. Firstly, purge irrelevant data including the records whose status codes are erroneous which contains error status code like (Status Codes =200 for success and 400 for error).
3. Create user sessions by scanning the log from web servers.
4. For every new IP address, a new user session is created.
5. Requests form same IP address is appended to r respective session.
6. Test Cases are generated for all the requests from all the users.
7. Test cases are reduced by applying reduction techniques.

VII. Related Work

S. Elbaum & et.al [1][2] explores a user session based techniques to test a Web application and also comparative study the effectiveness of white box and user session techniques. They also introduce three test case generation methods that are US-1, US-2 and US-3. Along with the effectiveness of white box and user session techniques in terms of capability of recognizing the faults, even that techniques presented it was possible to find different kinds of faults these algorithms are comparable. This research also exposed that the effectiveness of user session techniques recovers as the number of collected user sessions increases.

Jessica Sant & et.al [3] has proposed methods that builds model of web application by means of user session log and on the web server it is stored. The captured user sessions log data shows the dynamic actions of an application that can be useful for addressing the web applications testing problems. Their systems are able to generate test cases that achieve high coverage and in which user behavior is mapped correctly. A preliminary estimation that demonstrates the coverage found using the randomly generated test suite along with the accuracy of the replayed test cases is also introduced by them.

Sara sprenkle & et.al [4] for the automated web application they make the replay and oracle modules, which pose problems beyond those in the corresponding testing steps for traditional web applications. It involves implement and assess the set of automated replay tool and oracle comparators for user session based testing of web applications. Giuseppe & et.al [6] have discussed about functional and non-functional testing of web application. They conclude their exploration as all the testing aspects that are directly dependent on the implementation technologies have to be deeply adapted to the heterogeneous and dynamic nature of the web applications and the other features may be used again with a reduced adaptation effort.

Sara Sprenkle & et.al [8] propose automatic test case generation on various levels of multi user interaction and also state the dependencies by leveraging web application data field. Izzat Alsmadi & et.al [6] discussed the utilization of user sessions for test case generation and execution of those test cases. Their method utilizes the user sessions which are expected to improve from being copied and reused in the framework, by this it will become difficult to edit and translate into a format that can be used and utilized in different platforms and applications. They have also discussed the utilization of user session test automation for GUI. They have developed a system to create user session information in a format that is independent of the tool that collects them.

T.Deenadayalan & et.al [7] explores the clusters user session based approach. This method has the clusters user sessions depends on the service profile and selects a set of demonstrative user sessions from each cluster and personalized by augmentation with further requests to cover the relationships of web page dependencies. Furthermore, through attaching the requests with dependence relationships with the requests contains in user sessions generated during user events, it permits the coverage of the structure of the implementation.

The approach proposed by Xuan Peng & et.al [10] is depends on grey box testing. The concept of named transition relation in the form of page-request-page to represent the transition relationship between pages and requests for the pages has been introduced by them. In proposed approach, first the structural analysis was made with the application and then RDG is constructed according to the request dependence relationship between pages of web application. The relations are extracted according to RDG and other examination of data dependence and link dependence was made to recognize the significance of each transition relation. In the last phase Genetic Algorithm empirical was proposed to generate test cases with combination of different user sessions so as to cover as many fault sensitive transition relations as possible.

Comparison of Algorithms

Table 1.2: Description and drawbacks of Test Case Generation Techniques

Sr.NO	Technique	Description	Drawbacks
1	US-1	Direct Reuse of User Session	Privacy Problem caused by the more intensive instrumentation used by capture replay tool.
2	US-2	Combining different user session	Lowest coverage than others.
3	US-3	Reusing User Sessions with form modification.	No multi user session interaction.
4	Random Test Case Generation	Select User Sessions Randomly.(Using Random Traversal)	Lack of Accuracy and fault detection.
5	Fixed Time Blocks	Multi-user interaction, Variable sized test cases.	Requires smart timeout: likely to split user sessions across test cases.
6	Server Inactive	Multi-user interaction, Variable sized test cases.	Requires smart threshold: can split user sessions across test cases.
7	Augmented User Session	Represent logical user sessions, multi user interaction.	Larger test cases than user sessions, higher cost to generate test cases.
8	User Session Clustering	Uses Clusters of user sessions.	No identification for obsolete test cases and test case prioritization.
9	ReduceUSession	It combines test case creation and reduction.	It only consider Test coverage ratio. (Running time, time to save test state should be considered)
10	Test Case generation using genetic Algorithm(US-RDG)	User Session log is considered as Initial population of GA.	Only those sessions are covered which are covered in transition relation.

Table 1.1: Comparison of algorithms

From the above study we have map the comparison report of all user session based test case generation approaches shown in below table 1.2

Year	Technique	No. of Requests	Test Suite Size/no. of test cases	Block Coverage	Function Coverage	Faults Detected
2003	US-1	1975	85	263	65	23
	US-2	1919	84	255	64	23
2005	US-1	1975	85	263	65	23
	US-2	1919	84	255	64	23
	US-3	2742	407	288	65	26
	Random Test Case Generation	-	-	-	-	-

2006	Fixed Time Blocks	Min	16275	8447	-	12,270(45.2)	-
		Hour	16275	1769	-	15,713(57.9)	-
	Server Inactive		16275	1814	-	17,745(65.4)	-
	Augmented User Session		184656	1342	-	N/A	-
2010	User Session Clustering		-	-	-	-	-
	ReduceUSession		-	-	-	-	-

2011	TestCase Generation Using Genetic Algorithm(US-RDG)		1415	• Unique Requests
				• Data dependence transition relations
				• Link dependence transition relations
				• Transition relations
						10(100%)
						15(100%)
						26(76.47%)
						41(83.67%)	

Table 1.2: Comparison report of test case generation approaches

VIII. CONCLUSION

In this paper we discussed about web application and its execution. Also explore the Web application testing strategies. Generating test cases of high quality is the premise of web application testing. The various approaches we have discussed in this paper. The US-RDG is the existing method used for test case generation by using Genetic Algorithm. The limitation of US-RDG is, only those sessions are covered which are covered in transition relation. We are trying to resolve this problem in future.

REFERENCES

[1] Elbaum S, Karre S, Rothermel G (2003), Improving web application testing with user session data. In: ICSE'03: Proc. 25th Int. Conf. Software Eng. Portland, Oregon, USA, pp 49-59.

[2] Elbaum S, Rothermel G, Karre S (2005b), Leveraging user-session data to support web application testing. IEEE Tran. Softw. Eng., 31(3):187-202.

[3] J.Sant, A.Souter, and L. Greenwald, An exploration of statistical models of automated test case generation, In Int Work on Dyn Anal (WODA), May 2005.

[4] Sara Sprenkle, Emily Gibson, Sreedevi Sampath, and Lori Pollock, Automated Replay and Failure Detection for Web Application, ASE'05, November 7–11, 2005, Long Beach, California, USA.Copyright 2005 ACM 1581139934/05/0011

[5] Sara Sprenkle, Emily Gibson, Sreedevi Sampath, and Lori Pollock , A case study of Automatically Creating Test Suites from Web Application Field Data

[6] Izzat Alsmadi, and Kenneth Magel , Using User Sessions for Test Case Generation and Execution. Proceedings of the International MultiConference of Engineers and Computer Scientists 2008 Vol I IMECS 2008, 19-21 March, 2008, Hong Kong

[7] Mr.T.Deenadayalan,Dr.V.Kavitha, Mrs.S.Rajarajeshwari, Examining Web Application by Clumping and Orienting User Session Data, Journal of Computing, Volume 2, Issue 6, June 2010, ISSN 2151-9617

[8] Qian Zhongsheng , Test Case Generation and Optimization for User Session Based Web Application Testing, JOURNAL OF COMPUTERS, VOL. 5, NO. 11, NOVEMBER 2010

[9] Zhongsheng Qian , User Session-Based Test Case Generation and Optimization Using Genetic Algorithm*, J. Software Engineering & Applications, 2010, 3, 541-547 doi:10.4236/jsea.2010.36062 Published Online June 2010

[10] Xuan Peng* and Lu Lu, User Session-Based automatic Test Case Generation and Optimization Using GA, International Journal of the Physical Sciences Vol. 6(13), pp. 3232-3245, 4 July, 2011

[11] B. Michael, F. Juliana, and G. Patrice. Veriweb: automatically testing dynamic web sites. In Proceedings of 11th International WWW Conference, Honululu, May 2002.

[12] F. Ricca and P. Tonella. Analysis and testing of web applications.In Proceedings of the International Conference on Software Engineering, pages 25–34, May 2001.

[13] S. Manley and M. Seltzer. Web facts and fantasy. In Proceedings of the 1997 Usenix Symposium on Internet Technologies and Systems, Monterey, CA, 1997.

[14] C. Liu, D. Kung, P. Hsia, and C. Hsu. Structural testing of web applications. In Proceedings of the 11th IEEE International Symposium on Software Reliability Engineering, pages 84–96, Oct. 2000.

[15] E. Kirda, M. Jazayeri, C. Kerer, and M. Schranz. Experiences in Engineering Flexible Web Services. IEEE Multi-Media, 8(1):58–65, Jan. 2001.

[16] http://www.tutorialspoint.com/software_testing_dictionary/web_application_testing.

AUTHORS

First Author – Shruti N. Pardeshi, Research Scholar, Medicaps Group of Institutions, Indore, shruti1.pardeshi@gmail.com
Second Author – Vaishali Chourey, Associate Professor, Medicaps Group of Institutions, Indore, vaishalichourey@yahoo.com