Test Cases Prioritization in User Session based Testing for Improving Fault Detection Rate

Hsu Mon Maung, Kay Thi Win
University of Computer Studies, Mandalay
hsumon77@gmail.com

Abstract

Web application testing has been used in finding various faults in order to improve the quality of reliable web services. Among test cases generation approaches, user session based testing is an approach to create test cases with real user data. However, real user data usage is extremely large and executing all the test cases can be time consuming in practice. Executing all the tests in a reduced test suite can still be time consuming in practice. This paper describes the test cases prioritization method to schedule the test cases in order to improve the rate of fault detection. This criterion is based on two factors, frequency (Feq) and dependent count (Dept) of requests. The average percent of fault detected (*APFD) metric is used to reveal the permutation of test cases in a way may lead to faster detection available faults in a modified version of web application.*

Keywords*:* user session based testing, test cases prioritization, web application testing

1. Introduction

As the web application usage has been dramatically increased and most daily activities rely on the services provided by them, the qualities of these applications are central role. Efficient and effective testing of web application is crucial for reliable services. Testing must be performed completely in time without service interruption. Testing, designing and generating test cases are challenging tasks because web application is complex and changeable.

User session based test cases generation has been recently researched to generate test cases by the use of user session data. For web application system, field data has the additional advantage because the usage data is independent of the underlying implementation and server technologies [1]. A user session based test case is a sequence of base requests and parameter name value pairs. The advantages of user session based testing are less dependent on heterogeneous system and can generate test cases that reflect actual user behavior. But, there is a considerable issue that is collecting, analyzing and replaying the large amount of test cases generated from user session data [2]. Many researchers presented various reduction and prioritization techniques to solve these issues.

Even the reduced test suites can be large to execute in some commercial system. In this paper, dependency and frequency based test cases prioritization criterion is proposed in which the test cases are ordered based on occurrence and dependent count of single request contained in each test case. The purpose of test cases prioritization lies in ordering test cases based on a particular technique [3]. There are two main parts in the system: (1) criterion to prioritize user session based test cases and (2) evaluate proposed criterion by using fault detected rate and by comparing results of test length based criterion proposed by [4]. Section 2 presents related works concerned with user session based test cases prioritization techniques. In section 3, background theory in user session based testing and test cases prioritization. Section 4 describes proposed system and experimental results by comparing proposed system and another

prioritization criterion. The conclusion of proposed system is described in section 5.

2. Related Works

S. Roongruangsuwan and J. Daengdej [5] proposed two new efficient prioritization methods to address the problem of failing multiple test cases prioritization and same priority cases. This study proposed four prioritization factors to schedule test cases.

S. Sampath and R. Bryce [6] described several approaches to order reduced test suites using experimentally verified prioritization criteria for the domain of web applications. This approach prioritizes the reduced test suites by applying different verified prioritization criteria. Moreover, Mod_APFD_C which is a modification of the traditional prioritization effectiveness measure is also developed to enable comparison between different sizes of test suites.

Sampath et al. [2] explored the possibility of using concept analysis for achieving reduction and scalability in user session based testing of web applications. The method only considers the base request. The studies showed the low coverage of the base requirement, including statement coverage, fault coverage and base request coverage. The authors also admitted the importance of request data and ordering.

Another method presented by A. K. Upadhyay and A. K. Misra [7], is prioritization test cases using clustering approach in software testing. This study applied dendrogram methods for prioritization method and also showed the significant improvement in fault detection rate by prioritized test cases.

The purpose of test case prioritization lies in ordering test cases based on a particular technique [8]. This approach mainly focused on possibility of revealing faults earlier in the testing process. In this system, K-means cluster based prioritization is used to reduce the number of comparisons and effectiveness. K-means clustering methods produce clusters from a set of objects based upon the squared error objective functions.

3. Background Theory

This section describes about the user session based testing and type of test requirements for this testing. Test case prioritizing is also briefly described in this section.

3.1. User Session based Testing

User session based testing is a type of black box testing. A user session based test case is a sequence of user requests in the form of base requests and parameter name value pairs (eg, form field data). User session based testing makes use of field data to create test case, which has the great potential to effectively generate test case that can effectively detect residual faults [6]. The key advantage is the minimal configuration changes that need to be made to the web server to collect user requests [2]. Sample process of user session data collecting is shown in figure 1.

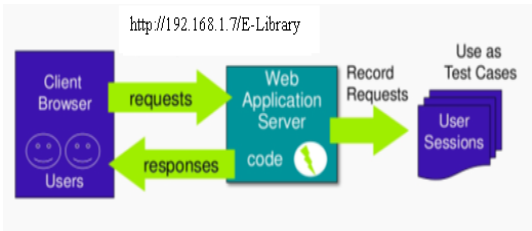

Figure 1. User Session based Test Cases

The user session based test cases can be viewed the perspective of the different test requirements. The test requirement data can be extracted from collected log data. The summarized test requirements are described as follows [6].

- base: Form of a base request
- Seq: Form of base sequences of size 2
- Name: Form of base request and parameter name

- Name_value: Form of base request, parameter name and parameter value
- Seq_name: Form of size 2 sequences of base request and parameter name

3.2. Test Cases Prioritization

Test cases prioritization techniques become more significant when the time to replay all the tests is insufficient under test. These techniques prioritize and schedule test cases in order that attempts to maximize some objective functions. [5]. Rothermel et al.[9] define test cases prioritization problem as follows:

Given: T, a test suite;

PT, the set of permutations of T;

F, a function from PT to the real numbers;

Problem: Find T′ € PT such that (∀T″) (T″€PT) (T″ ≠ T′) [f (T′) ≥ f (T″)]. Here, PT is set of all possible prioritizations of T and f is a function that, applied to any such ordering. In prioritization process, test cases are executed according to some criterion to satisfy some performance goal. Some prioritization criteria are test length based, code based, frequency based and other possible criteria. Most criteria focus on the goal of increasing the rate of fault detection earlier in the testing process.

4. Proposed System

This paper mainly focuses on test suite prioritization process. Our proposed system consists of three main phases: generating test cases, reducing test cases and prioritizing reduced test cases for subject application. The overview framework of proposed system is shown in Figure 2.

4.1. Test Cases Generation

In test cases generation process, the daily usage logs are collected from specific web application. The user session data is parsed to extract necessary information such as IP address, GET, POST methods, time stamp and cookie information. The unwanted and redundant data are removed. The user access log is converted into test cases in the form of http requests that can be sent to the web server.

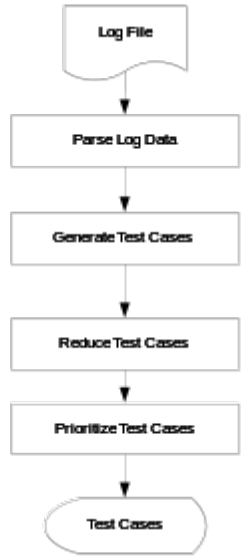

Figure 2. Overall Process of Proposed System

4.2. Test Cases Reduction

When a request from a new IP address arrives at the server, a user session is identified as initial and when the user leaves or session time out, the user session is identified as the end. The 30 minutes is taken to identify the user session.

In this phase, Shanon's entropy gain theory is applied to reduce test cases [10].

$$E(H) = -\sum_{i=0}^{n} P_i Log_n P_i \qquad (1)$$

where P_i is probability of $link_i$ that are accessed by users and n is the number of links of web site. This process is implemented in our previous work to reduce test cases that are not only small in size but also equivalent in effectiveness to an original test suite. The concept in this reduction method is that the

higher entropy value may lead to more URLs covered [11].

4.3 Process of Proposed Prioritization Method

Many criteria have been proposed how test cases are ordered to get the efficient fault detection quality. In reduction techniques, the criteria create smaller test suite than original suite but test cases are in no specific order. In this system, the new prioritization criterion is proposed to be more efficient in fault finding of reduced test suites.

The proposed criterion considers two factors dependent count (Dept) and frequency (Feq) of single request in ordering multiple test cases. In some cases, although the frequency of base request is high, the accessed requests in this test case may be initial pages or start pages (e.g. home.php or index.php).

The procedures of proposed method are described in Table 1:

Table 1. Procedure of Proposed Method

Input : set of reduced test cases T set of request S Output : prioritized test cases PT
Step 1: The dependency table of base request is constructed to order the test suites based on dependent count d of base request sequences Si. Step 2: The frequency table of base request is constructed to order the test suites based on dependent count feq of base request sequences Si. Step 3: Order the sequences (S_0, S_1,...,S_i) in decreasing order of both d and feq. Step 4: Replay the prioritized test cases according to this criterion to server.

Firstly, the request is selected to sort the dependency and frequency of it. The frequency table is constructed in decreasing order of occurrence of this request in the test suite. The second factor (Dept) dependency of request is the important factor for prioritization because this may increase the fault detection rate.

For the test suite in the experiment, there are 118 requests in the reduced test suite to prioritize. In proposed prioritization process, firstly one request is selected from total 118 requests. The dependent count (Dept) and frequency (Feq) of the selected request are calculated and then test cases are ordered according to these values. The order of test cases shown in sample Table 2 is prioritized with the request ucsm.edu.mm/index.php/component/users/?view=remind. In this test prioritizing, the user (119.31.123.207) is the first test case to replay because the frequency and dependent count of this test case are highest. The value zero in the table means that the selected url does not contain in the test cases.

4.4 Evaluation of the Proposed System

Fault detection technique can be used in evaluating the efficiency of prioritized test cases. For detection experiment, naturally occurring faults that were discovered by users during application deployment are used to analyze the effectiveness of the test suites from the test requirements. The 21 faults are used in analyzing fault detection effectiveness. Table 3 shows that fault identification of each test case in our experiment. Eight test cases can find all 21 faults which are link faults in our experiment.

Table 2. Frequency Table of Access Sequence

User IP Address	Frequency (Feq)	Dependency (Dept)
119.31.123.207	8	4
203.81.93.18	0	0
203.81.88.190	0	0
203.81.93.76	0	0
203.81.69.87	0	0

From Table 3, it is clear that test case t1 detects 11 faults; t2 one fault and so on. To analyze the fault detection results of proposed approach, the average percent of fault detected (APFD) is calculated by using reduced test cases of proposed reduction method.

To evaluate the APFD results of proposed criterion, the value of fault detection rates is analyzed based on single request in test suite. There are 118 requests in the reduced test suite. In this experiment, 40 requests of total requests (118) in the reduced test suite are used for calculating dependent count and frequency. The test order with highest APFD value is selected to replay the test cases to the server. The APFD values of prioritized test cases can be calculated by using equation 2.

$$APFD = 1 - \frac{tf1 + tf2 + tf3 + \ldots + tfn}{mn} + \frac{1}{2n} \quad (2)$$

According to the APFD metric, total number of test cases in a given test suite is n and F is a set of m faults detected by T. Let TFi be the position of the first test case t in T′, where T′ is an ordering of T, that detects fault i. The APFD values are between 0 and 1 and higher numbers imply faster (better) fault detection rates. The prior knowledge of the faults detected by the test suite is available to the testers.

Table 3. Detected Faults in Each Test Case

Test Case	User IP	Faults
t1	203.81.93.18	f1, f2, f3, f4, f5, f6, f7, f8, f10, f11
t2	203.81.88.190	f12
t3	203.81.69.87	f13
t4	203.81.93.76	f14
t6	203.81.94.61	f15, f16
t7	203.81.93.73	f17
t11	203.81.88.191	f18
t16	119.31.123.207	f19-f20-f21

4.5 Experiment and Analysis

In empirical studies, the official website of the University of Computer Studies, Mandalay (UCSM) is used. It is now deployed and maintained by PhD students at the UCSM. On this website, Users can view the proceedings and workshop announcement, exam results, lecture invitation, and activities and other related information in this site. The 55 access log files are collected using UCSM web site during the 2013-2014 and 2014-2015 academic years at the UCSM. To observe the effectiveness of prioritizing test cases in our experiments 25 reduced test cases form 745 user sessions are used. These reduced test cases are ordered by using two methods, test length based ordering, and proposed prioritization criterion.

4.5.1 Test Length based Prioritization

Many criteria have been proposed how test cases are ordered to get the efficient fault detection quality. In this system, the test length based criteria proposed by [4] is used to compare the results of proposed test prioritized criterion. Ordering test cases based on their requests length can affect the fault detection of the ordered test suite [12]. In test length based criterion, the number of requests in each test case is counted and test cases are ordered in descending (Request Long to Short) order of test length and ascending order (Request Short to Long), where the length of test case is the number of requests in this test case. For the example test suite in Table 4 test cases that have maximum length of these requests are selected for execution before other test cases in the test suite. In the same length case; one of them is selected at random.

According to the test length based prioritizing criterion, the test cases are ordered by the request length long to short. The percent of faults detected versus the fraction of the test suite are

shown in Figure 3. The area under the curve represents the weighted average of the percentage of faults detected over the life of the test suite.

Table 4. Frequency Table of Access Sequence

User IP Address	No. of Requests
203.81.93.18	69
203.81.88.190	56
203.81.69.87	35
203.81.93.76	35
203.81.94.89	33
203.81.94.61	33
203.81.93.73	33
203.81.93.78	33
203.81.88.188	31
203.81.93.74	29

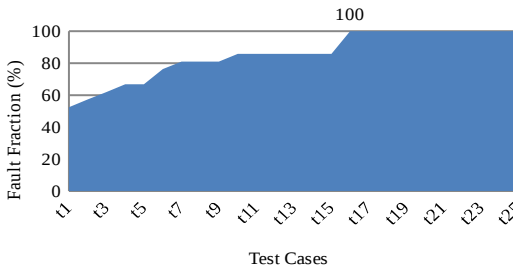

Figure 3. Fault Fraction of Request L to Request S

On the test length based order (Figure 3), the first test case (t1) detects 11 of 21 faults are detected and thus 52.38% of the faults have been detected after t1 has been executed. After running test case t2, one more fault is detected and thus 57.14% of fault detects have been detected. The test order detects 100% of faults after executing t16. Therefore, the resulting APFD for this test length based order (Req L to Req S) is 79.33%.

Analyzing APFD values on another order (Req Short to Req Long) is described in Figure 4. The first fault f1 has been detected by the test case t1 with the position of 25. Therefore, only 14.29% of the faults are detected after replaying t16. In this ordering, the 100% of faults fraction is obtained after the last test case t1 has been executed. The APFD value of this prioritization (Req S to Req L) is very low, 12.67%. Therefore, the experimental results show that the prioritization results of Request L to Request S performed better than the Request S to Request L.

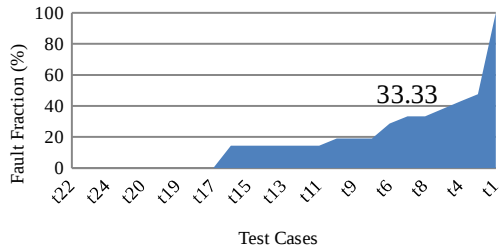

Figure 4. Fault Fraction of Request S to Request L

4.5.2 Proposed Prioritization Method

To evaluate the APFD results of proposed criterion, the value of fault detection rates are analyzed based on single request in test suite. There are 118 requests in the reduced test suite. In this experiment, 40 requests of total requests (118) in the reduced test suite are used for calculating dependent count and frequency. The test order with highest APFD value is selected to replay the test cases to the server.

Table 5. APFD Values of Sixteen Ordering

URLs	APFD (%)	Feq	Dept
R1	81.42	63	2
R2	85.05	43	2
R3	79.71	94	2
R4	85.81	50	2
R5	80.29	16	3
R6	79.71	72	3
R7	84.48	8	4
R8	84.48	1	4
R9	79.71	126	3
R10	80.29	23	3
R11	79.52	99	2

R12	79.71	60	2
R13	79.52	6	2
R14	84.48	8	1
R15	84.48	2	2
R16	82.57	1	4

According to the empirical results, 16 prioritizations of total 40 requests performed better than the test length based criterion. The APFD values of 16 prioritizations with frequency and dependency are shown in Table 5.

The APFD results of ten prioritizations based on proposed criterion are equal results with test length based criterion (Long to Short) and fourteen orderings performed lower APFD values than test length based ordering. These values with associated frequency and dependency are described in Table 6 and Table 7.

Table 6. APFD Values of Ten Ordering

URLs	APFD (%)	Feq	Dept
R17	79.33	7	3
R18	79.33	28	3
R19	79.33	15	3
R20	79.33	36	3
R21	79.33	31	2
R22	79.33	47	3
R23	79.33	3	3
R24	79.33	36	3
R25	79.33	23	3
R26	79.33	5	3

The highest APFD value is obtained by ordering test cases based on R4. The dependent count of R4 is 2 and occurrence numbers are 50. By inspection, it is clear that proposed criterion results in the earliest detection of the most faults and illustrates an optimal order, with APFD 85.81%. Figure 5 shows the test suite fraction of R 4. Analyzing the percent of fault fraction on R4, over 90% of the faults are detected after replaying t6. In this ordering, the 100% of faults fraction is obtained after the test case t11 has been executed.

General assumption is that the factor frequency (feq) is important to prioritize the test cases because the percentage of faults involving may be high in user frequently accessed requests. In the empirical results shown in Table 7, although the frequency is high, the APFD value is low. One of findings is that 75.90% APFD value is obtained on the request R27 with the maximum frequency (3686).

Table 7. APFD Values of Fourteen Ordering

URLs	APFD (%)	Feq	Dept
R27	75.90	3686	0
R28	78.19	10	3
R29	78.57	1	3
R30	78.57	1	1
R31	78.95	17	3
R32	75.90	48	2
R33	75.71	39	2
R34	78.57	140	2
R35	78.00	27	2
R36	78.00	74	2
R37	77.24	43	2
R38	79.14	7	2
R39	78.19	5	2
R40	77.23	72	2

From results of dependency table, the requests R7, R8 and R16 have highest dependent counts (4). It is observed that prioritizations on these requests performed the higher percent of APFD values than the test length based criteria, although the frequency values of R7, R8 and R16 are low counts which are 8, 1 and 1 respectively. Therefore, the dependent count of each request is also important factor for deciding to prioritize the test cases.

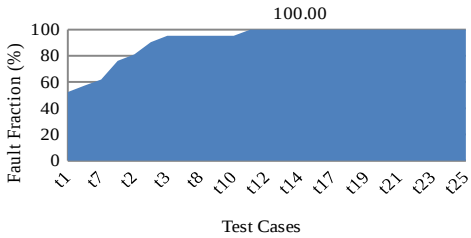

Figure 5. Fault Fraction of Request R4

The experimental results reveal that the important factor is not only frequency but also dependent count of each request in prioritization process. But, there may be some deviation according to the experimental results because of limitation of APFD metric.

5. Conclusion

Large amount of user session based test cases in web application testing is not practical within time constraint. In this paper, a new prioritization method is presented to make more efficient the fault detection rate of reduced test cases. According to the analyzed results, the proposed prioritization method yields higher APFD values than the test length based criterion because they are prioritized based on frequency (Feq) and dependency (Dept) of each request. From our experience, faults can be detected faster in replaying test cases by considering not only frequency but also dependent count of each request. The APFD values of proposed criterion are ranging between 75.71%, to maximum 85.81% respectively and the order of test cases that has the maximum value 85.81% is selected. From the experimental results on another prioritization method (test length based criterion), the maximum APFD value is 79.33%. It is found that ordering the fault detection quality of prioritized test case using the proposed method can perform better results for testability of user session data. Our findings also show that prioritizing test cases based on both frequency and dependency can support finding faults as early as possible under testing process. Therefore, the prioritizing test cases in user session data can help software testers for improving the test results of web based applications.

References

[1] S. Sprenkle, E. Gibson, S. Sampth, and L. Pollock, “A case study of automatically creating test suites from web application field data”, Portland, Maine, USA, TAVWEB06 (Jul. 2006).

[2] S. Sampth, V. Mihaylov, A. Souter, and L. Pollock, “A Scalable Approach to User Session based Testing of Web Applications through Concept Analysis”, Proc.19th Int. Conf. Automated Sofw. Eng. Washington DC, USA, 2006, pp. 132-141.

[3] S. G. Elbaum, A. G. Malishevsky, and G. Rothermel, “Test Case Prioritization: A Family of Empirical Studies”, IEEE Transactions on Software Engineering, 28(2): 2002.

[4] M. R. Keyvanpour, H. Homayoun and H. shirazee, “Automatic Software Test Cases Generation:An Analytical Classification Framework”, International Journal of Software Engineering and Its Applications, Vol. 6, No. 4, October, 2012.

[5] S. Roongruangsuwan and J. Daengdej, “Test Cases Prioritization Techniques”, Journal of Theoretical and Applied Information Technology, JATIT 2010.

[6] S. Sampath and R. C. Bryce, “Improving the Effectiveness of Test Suite Reduction for User Session based Testing of Web Application”, Information and Software Technology 54 (2012), 724-738.

[7] A. K. Upadhyay and A. K. Misra “Prioritizing Test Suites using Clustering Approach in Software Testing”. ISSN: 2231-2307, volume-2, Issue-4, September, 2012.

[8] S. Elbaum, A. G. Malishevsky, and G. Rothermel, “Test Case Prioritization: A Family of Empirical Studies”, IEEE Transactions on Software Engineering, Vol. 28, No. 2, February 2002.

[9] S. Elbaum, A. G. Malishevsky and G. Rothermel , “Prioritizing Test Cases for Regression Testing”, International Symposium of Software Testing and Analysis, August 2000, p. 102-112.

[10] K. D. Bailey, “Entropy Systems Theory”, December, 2013.

[11] H. M. Maung, “Test Cases Reduction Approach in User Session based Testing for Web Application”, ICCA, (2014).
[12] S. Sampathy, R. C. Brycez, G. Viswanathy, V. Kandimallaz, and A. G. Koru, "Prioritizing User Session based Test Cases for Web Applications Testing", Proceedings of 1st International Conference on Software Testing, Verification, and Validation, IEEE,2008.