Int. J. Advance Soft Compu. Appl, Vol. 7, No. 2, July 2015
ISSN 2074-2827

Clustering Test Cases in Web Application Regression Testing using Self-Organizing Maps

Mojtaba Raeisi Nejad Dobuneh[1], Dayang N. A. Jawawi[1], Mojtaba Vahidi Asl[2], and Mohammad V. Malakooti[3]

[1]Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia (UTM), Skudai 81310, Johor, Malaysia
e-mail: rmojtaba2@live.utm.my, dayang@utm.my

[2]Faculty of Computer Engineering and Sciences, Shahid Beheshti University
e-mail: mo_vahidi@sbu.ac.ir

[3]Faculty of Computer Engineering, Islamic Azad University UAE Branch
e-mail: malakooti@iau.ae

Abstract

In the software testing domain, different techniques and approaches are used to support the process of regression testing in an effective way. Test case prioritization techniques improve the performance of regression testing, and arrange test cases in order to detect the faults in a reasonable amount of time. User-session is a unique feature of web applications that is useful in the process of regression testing as they comprise precious information about the application state before and after any change that is made to the software code. The main challenge is the effectiveness of average percentage fault detection rate and time constraint in the existing techniques. Thus, in this research the priority is given to the test cases, clustered according to some criteria. Using self-organizing map for clustering helps to obtain a higher fault detection rate in a relatively small time span.

Keywords*: Clustering test cases, Self-organizing map, Web application, Prioritization, Regression testing.*

1 Introduction

Regression tests are executed when some changes are made in the existing application in order to check the negative impact of the changes in the rest of the system or on the expected behavior of other parts of the software. It is a

complicated process for web applications based on modern architectures and technologies. In this paper, we make use of user session data for website testing. The real user session data, saved in server side, is beneficial for regression testing because the tester does not concern about underlying architecture and developing technology. The test cases that are generated by user sessions do not significantly depend on different technologies which are used to develop web based applications. Web applications supply usage logs to testers. The test cases extract from user data are called "user-session based testing". The user-sessions provide informative knowledge about the web interaction of users [1]. In the simplest form we can easily execute all the existing test cases in the test suite without any extra handling. Since, he the size of test suites gradually grow due to software modifications executing the entire test suite would be very expensive. This leads the software engineers to think about deploying efficient techniques to reduce the effort that is required for regression testing in different ways. During the development of an application, a new version of the application is released as a result of (a) requirements medications and (b) bug fixes [2], [3]. A large number of reusable test cases may be accumulated from different versions of application could be applied for testing the newer version of the application. However, running all the test cases may take a significant amount of time. For example, one may spend a few weeks on executing all test cases of earlier versions of a given application [4]. Regarding to the time restrictions, software testers need the selection and ordering of an optimal subset of test cases for execution.

The three major approaches for regression testing are test suite minimization, test case selection and test case prioritization [5]. Test case prioritization (TCP) helps us to find out the different optimal combinations of the test cases. A prioritization process is not associated with the selection process of test cases, and it is assumed that all test cases must be executed, but it tries to get the best schedule running of test cases in a way that if the test process is interrupted or early halted at an arbitrary point, the best result is achieved in which more faults are detected. TCP has been introduced by Wong et al. [7]. Structural coverage is the most commonly used metric for prioritization. The logic of this criterion is that faster structural coverage of the whole software code leads to maximum detection of faults in a limited time. Therefore, the aim of this approach is to achieve higher fault detection rates, with larger structural coverage [8] [9].

2 Background and Motivation

In general, the slight modifications on web applications increase the number of test cases, considerably. As a result, the effectiveness of conventional testing process would be decreased. In real world scenario the issue of scalability is challenging. For example, suppose there are 100,000 test cases need to be executed. Obviously it is unrealistic to expect a human tester to provide reliable responses for such a large number of test cases. Our approach using SOM cluster based prioritization reduces the number of test cases and can be very effective.

Instead of prioritizing individual test cases, we prioritize clusters of test cases using cluster based prioritization techniques [10] [11].

The most appropriate test cases for a web application are session-based because sessions best reflect real user patterns, making the testing process quite realistic [6]. The User-Session based techniques are new, useful lightweight mechanisms of testing. Automating the test process is more feasible and simpler in user-sessions when applied on web applications. In user-session approaches, the total interactions of users with the server are collected and the test cases are generated by using a suitable policy. The client's requests, transported as URLs and composed of page addresses and name value pairs, are the data to be captured. These data that can be found in the log files are stored in servers the data Collected from user sessions can be used to generate a set of http requests and converted into a real test case. The benefit of the approach is to generate the test cases without any awareness web application's internal structure. The test cases for web application comes from a log file which has some parameters such as date & time *%t*, http version *%H*, request method *%m*, resource address %U, session ID *%S*, response code *%s*. The log file format example is shown in Table 1.

Table 1: Log file collection

Mybookstore access log 2013-10-1					
%t-Date & Time	%H-Request Protocol	%m-Request Method	%U-Requested URL Path	%S-User Session ID	%s-Status Response Code
[01/Oct/2013:10:02:21]	HTTP/1.1	POST	/Login.jsp	[A6E460139A7AA2AB8EBEFD271D12B8EF]	[200]
[01/Oct/2013:10:02:23]	HTTP/1.1	POST	/Login.jsp	[65244F9450BA867695711B218482CEAC]	[200]
[01/Oct/2013:10:02:23]	HTTP/1.1	GET	/Login.jsp	[A069B16BA84989948130FE8BBFE8769B]	[200]
[01/Oct/2013:10:02:23]	HTTP/1.1	POST	/AdvSearch.jsp	[5F70911A64B3C61B7487D7F48C830026]	[200]
[01/Oct/2013:10:02:23]	HTTP/1.1	GET	/ShoppingCart.jsp	[91B043EF48A227E487B63833A72A37A1]	[200]
[01/Oct/2013:10:02:23]	HTTP/1.1	GET	/Default.jsp	[CCA757441C6509E05D80C9C4B8E22E1C]	[200]
[01/Oct/2013:10:02:24]	HTTP/1.1	POST	/Login.jsp	[6216DCD2456133F6BE6465BE2309CF92]	[302]
[01/Oct/2013:10:02:24]	HTTP/1.1	GET	/ShoppingCart.jsp	[6216DCD2456133F6BE6465BE2309CF92]	[200]
[01/Oct/2013:10:02:24]	HTTP/1.1	GET	/Login.jsp	[945E5DCC1FEE3A8F438380D29DA0B2C4]	[200]
[01/Oct/2013:10:02:14]	HTTP/1.1	GET	/BookDetail.jsp	[97D75C2AD932C2D2A6F49B49033E5DFB]	[200]

In the following, we briefly review the existing clustering methods based on the user session and the web application testing. Luo. et al. have developed a technique which clusters user sessions based on service profile and selects a set of representative user- sessions from each cluster. Subsequently, each selected user

session is tailored by augmentation with additional requests to cover the dependence relationships between web pages. A large number of studies have applied clustering for software testing [9]. Li and Xing presented an approach for adopted k-means algorithm for partitioning user session data into a reduced number of clusters. Each cluster represents similar scenarios of user interactions with a given web application [18]. Liu. et al. have suggested a user-session-based test cases optimization method based on agglutinate hierarchy clustering. This method firstly gives the function to calculate the distance between the user sessions, and then employs the bottom-up agglutinate hierarchical clustering algorithm to cluster the initial testing cases and produces different kinds of test suites [19]. Yoo et al. have combined clustering test case based on the dynamic runtime behavior. clustering test cases, based on their dynamic runtime behavior, can reduce the required number of pair-wise comparisons significantly [6].

The current researchers apply k-means algorithm for clustering or prioritization test suite for effectiveness of fault detection rate and time. Although the k-means algorithm involves some disadvantages [13], it is used in a wide variety of applications. The number of clusters, k, in the database using the k-means algorithm assumes to be known prior to the start of the process which is not realistic for real-world applications. Note they the k-means algorithm is an iterative technique in which the process is sensitive to the initial conditions (initial clusters and instance order). The k-means algorithm converges in a finite number of iterations to a local minimum. The running of the algorithm defines a deterministic mapping from the initial solution to the final one. A hard and fixed decision is provided by the vector quantization and especially the k-means algorithm which does not transmit enough information on the real observations [14].

It is suggested to improve the method by proposing a new technique that uses clustering and prioritization together with applied criteria. In this research, we have proposed a technique that further improved the effectiveness of regression testing by ordering a set of clustered test cases according to our defined criteria. Therefore, the problem is to propose a new technique which can be used to improve the effectiveness of fault detection rate and time.

The permutation of test cases in a way that leads to a faster detection of maximum available faults in a modified version of web application needs to find good criteria. The goal of this research is to propose a new technique which merges two approaches of prioritizing and the clustering test suite to improve one of the test case prioritization techniques called session-based technique in web application regression testing. The research aimed to improve the accuracy of existing test suites with respect to the effectiveness of time and the rate of fault detection.

User session used for web application testing and this approach has been accurate and adequate for the dynamic web application domain. Therefore, we conduct our research by using session based test case prioritization for web application testing.

The lines of code for large web applications are in the millions and debugging and error detection for all these lines are time consuming. Thus, there will be so many object interactions that also required the interactions of users significantly. The automated testing becomes complicated due to the continuous maintenance process and due to the changes that occurred in the profiles of the users [12].

The two techniques (clustering and priority test cases) have been used for the testing approach by different researchers [20]. The clustering techniques generate a set of test case which is smaller than the original while test cases set may be very large. Although, we can generate a smaller set of test cases as opposed to the original suite, but yet these smaller set of test cases can be so large that cannot be executed completely in terms of time constraint. In this paper, we have proposed a new technique that can be used for the clustering test suites to improve the effectiveness of testing by ordering the clustered set of test cases as well as to introduce the criteria for the ordering. Our technique can be useful for testers who encountered with limited time and resources but still want to complete the process of testing. The research goal is to investigate the optimized clustering test cases as a new technique of test case prioritization for the web application testing

3 Clustering and Prioritizing Test Cases

Priority test cases are one approach to schedule the test cases based on some criteria that increases the effectiveness in meeting the performance goal. To reduce the cost of regression testing and the time involved in it, software testers may prioritize their test cases so that those which are more important, by some measure, are run earlier in the regression testing process [5]. In previous study several prioritization criteria have been proposed which could be classified into five categories: General test case prioritization, Version-specific test case prioritization, Comparator Techniques, Statement Level Techniques, and Function Level techniques. In general test case prioritization, the test cases are ordered in descending order of the number of Parameters that are assigned values in each test case. In version-specific test case prioritization, the test cases are prioritized such that the resultant test suite could be most effective for a particular version of the software. In this technique, the test cases operate at a relatively fine granularity, including instrumentation, analysis, and prioritization at the level of source code statements. The comparator techniques use random ordering or the optimal ordering of the test cases. Statement Level Techniques include approached that prioritize test cases by considering the attributes of the program at the statement level. The function level techniques consist of the approaches that prioritize the test cases by considering the attributes of the program at the functional level.

A prioritization process is not associated with the selection process of test cases**.** In test case prioritization (TCP) all test cases will be executed, but to achieve the best results, some criteria need to be applied to prioritize the test cases. This effort

is helpful to find out the different optimal combinations of the test cases, and it tries to achieve best schedule running of test cases. It means that, if the test process is interrupted or early halted at an arbitrary point, the best result that is finding more faults is achieved. The prioritization criteria which used for clustering test cases are as follows:

1- Based on number of most common HTTP requests in pages.
2- Ordered dependency of HTTP requests helps to improve the rate of average percentage fault detection (APFD).
3- Ordered length of HTTP request chains to better APFD rate.

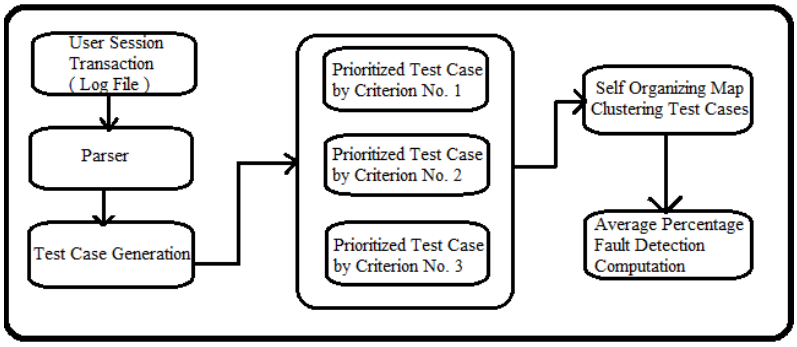

Fig. 1 Proposed solution model

The proposed model generate test cases, prioritizes them according to the criteria and clusters test cases with self organizing map (SOM) algorithm, shown in Fig.1. The model comprise of steps define by the following details:

1- Collecting log file of user session in server.
2- Parsing log file by change user session parameters.
3- Generating test cases from user session.
4- Prioritizing test cases based on criteria
5- Using self organizing map for clustering test suites.
6- Calculating average percentage fault detection.

The test case clustering techniques could be categorized into two modes of partitioned or hierarchical. The partitions of the data are constructed by a partitioned cluster algorithm that cluster test cases based on the minimal sum of squared distances from the mean to obtain the optimum result for each cluster.

As Partitioned clustering enumerate itemizes all possible groupings and tries to find the global optimum, the complexity of the system will be huge. Even for a small number of objects there exist a large number of partitions. Hence for this reason, the common solutions are usually with an initial, random partition start.

Then, it continues with the refinement and partitioning algorithms for different sets of initial points, (as representatives intended) to search and find out whether all solution led to the same partition. The value of similarity or distance can be calculated from the partition clustering algorithm and the optimum results will be chosen. Therefore, most of them would be recognized as greedy like algorithm.

A hierarchical decomposition of the objects is created by hierarchical algorithms. They are (top-down) divisive or (bottom-up) agglomerative: firstly Divisive algorithm approach begins with a single group of all objects and then be divided into smaller groups until each of the objects are within a cluster. Secondly Agglomerative algorithms follow the opposite strategy. They start with those objects that are within a cluster, then the groups merged based on distance and similarity of each other. If the objects are in a one group or at any other point the user wants, the algorithm will stop. This technique followed a greedy like bottom up merging [16]

3.1 Self Organizing Map

Self-organizing map (SOM) is defined as a neural technique for clustering. It is demonstrating the two spatial spaces of link among clusters. SOM has been able to present the data points that are in one or three-dimensional space, that provided by SOM capabilities. Moreover, due to the easy of visualization and the trade-off between information content two dimensional spaces have been used more often [15].

In this research Kohonen's algorithm is used for clustering by SOM. The test cases organize into a two dimensional map, according to user session ID and Resource Address. It is aimed to provide an interactive tool so they can retrieve information more effectively and efficiently. Our inputs consist of a set of test cases. The desired output is a two-dimensional map of M nodes.

We modified and match the SOM algorithm for applying to the clustering test case for web application testing. The SOM algorithm is presented in detail as follows:

1 – Select two parameters from the log file which are Session ID and resource address, to put in the array.

2 - Data parameter values are normalized and given in the form of a numeric matrix.

3 – The $m \times n$ matrix is as input of the SOM algorithm

4 – Set the learning rate and neighborhood distance with iteration number for determining the clusters of test cases by running the SOM algorithm. In this case examined $\sigma^2(0) = 1$ and $\alpha(0) = 0.2, 0.5, 0.7, 0.9$

5 - Using the distance function to check the similarity degree between test cases

$$d_j = \|x - w_j\| = \sqrt{\Sigma_{i=1}^{n}(x_i - w_{ij})^2} \quad (1)$$

Where x is input sample and w is the weight vector of ith node.

6 - The winner of the competition between nodes in a network node with the minimum distance is selected

$$w_{ij}(t+1) = w_{ij}(t) + c\,[x_i - w_{ij}(t)] \quad (2)$$

For all nodes j in $N_m(t)$, $c = \alpha(t)\exp(-\|r_i - r_m\|/\sigma^2(t))$

The distance value between each node and the winning node is $r_i - r_m$

7 - The weights to all nodes within a topological distance updated by repeat step 6 for all entries of the matrix

8 – The output is provided test suite with similarity test cases in the same groups.

After the network is trained through repeated presentations of all test cases (each test case is presented for Ω epochs), present unit input vectors of every test case to the trained network and assign the winning node the session ID and resource address for related test case. Update the number by labeling the node as the number of test cases allocated to the node.

4 Case Study

It is an open source web application online bookstore and is selected as a case study available on www.gotocode.com. The test cases are generated from the real user session transactions for the web application testing in server-side and each log file is converted to the test cases. Online book store is a portal for shopping books. Users are allowed to register on the bookstore, logging in, searching for books by keyword, browse for books, add books to a shopping cart, rate the books, update personal information, and log out. Bookstore comprises classes use Java scripts for its front end and a create database with MySQL for the back end. Bookstore application has introduced to undergraduate students of Universiti Teknologi Malaysia (UTM) to collect log files on server side. The log files have been collected for 60 days.

Consider an example to implement the algorithm for clustering test cases of the selected web application named as online book store. The normalized matrix S comes from conversion of the two-parameters of the log file, (session ID and resource address) as input for SOM clustering. The session ID is based on hexadecimal unique values for the request that is changed into the decimal respected value in advance, then normalize the input matrix S as entry of SOM clustering.

The second parameter mapped from the requested URL's path related to any transaction between user and bookstore web application. The mapping code is given as follows:

Switch (temp)
{Case”Defualt.jsp": Temp=1; break;
Case”Registration.jsp": temp=2; break;
Case”Login.jsp": temp=3; break;
Case”Advsearch.jsp": temp=4; break;
Case”Shoppingcart.jsp": temp=5; break ;
Case”Books.jsp":temp=6;break;}.

$$S = \begin{bmatrix} 0.26 & 0.26 & 0.26 & 0.26 & 0.15 & 0.15 & 0.15 & 0.15 & 0.15 & 0.15 \\ 0.3 & 0.4 & 0.1 & 0.3 & 0.5 & 0.6 & 0.3 & 0.5 & 0.1 & 0.2 \end{bmatrix}$$

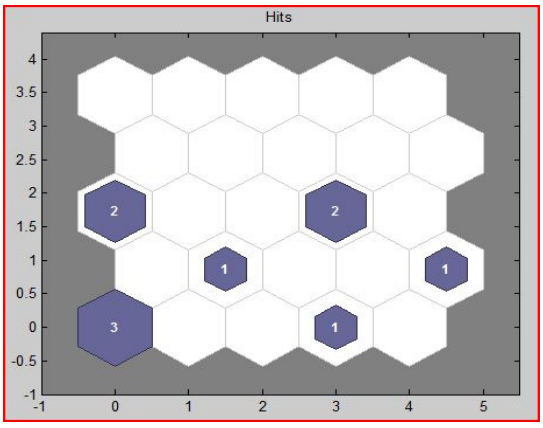

Fig. 2 Plot SOM sample hits

The clustering output is depicted in Fig. 2 and it is a SOM layer plot that shows the number of input vectors classified with each neuron. It shows the test cases in separate clusters based on the similarity and minimum distance. Each neuron has shown by the number of vectors via size of a colored label.
Fig. 3 shows the plot SOM neighbour weight, distance for the SOM layer showing neurons as gray-blue labels and their direct neighbor relations with red lines. The neighbor labels are colored from black to yellow to show how close each neurons weight vector is to its neighbors.

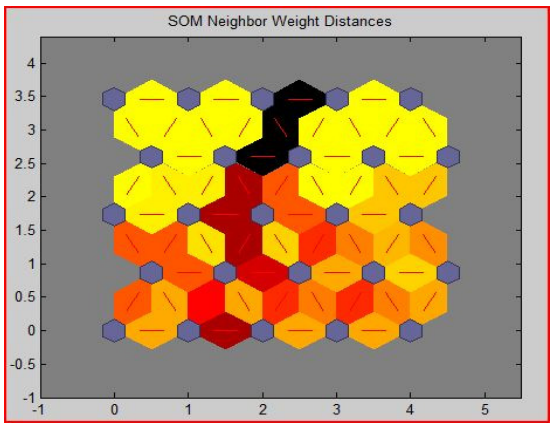

Fig. 3 SOM output plots

5 Results and Discussion

This section is comprised of verification and validation of the proposed new technique for web based application. The fault detection rate is defined as the total number of faults which detected in a shown subset of the ordered test case priority. For 100% detection of faults, the time used by each prioritized suit is measured properly. The best possible option to calculate the detection of the total faults is to detect them in earlier stages of the tests.

The case study is selected in order to verify it by applying criteria to test cases of web application. The results of applied criteria to the Bookstore are shown in Table 2. Moreover, the results show the ordered test cases, with SOM clustered, have detected more faults at early stage of running the test cases. Thus, the rate of fault finding remains constant until all test cases are executed. Test cases executed by random to demonstrate effectiveness of the test order with APFD comparable to the other criteria.

The result of applied criteria to the Bookstore benchmark is shown in Table 2. The effectiveness of fault detection rate in SOM cluster is increased while it can find the majority of faults with running half of all test cases. As it can be seen, the SOM cluster orders test cases for fault detection process result in finding 28 out of 30 faults at early stage of running test cases.

The third criterion which orders test cases based on dependency HTTP requests enables the proposed technique to detect faults at early stages of running test cases by 50 % of whole test cases, detecting 28 out of 30 faults. The 28 of total faults have been detected by first criterion (number of most common HTTP requests in pages) by executing 60% of existing test cases. The length of HTTP request prioritization criteria as a second criterion, detects faults by run more than 80 % of test cases. The test cases executed randomly shows a low fault detection rate for web application testing.

Table 2: Results for self organizing map test case prioritization

Test case execution percentage	**Average number of fault detection for Bookstore**				
	fault detected by numbers of common http request	***fault detected by length of http request***	***fault detected by dependency http request***	***fault detected by clustering SOM***	***fault detected by random***
10%	23 faults	21 faults	23 faults	25 faults	16 faults
20%	25 faults	24 faults	23 faults	25 faults	16 faults
30%	25 faults	26 faults	25faults	26 faults	20 faults
40%	25 faults	26 faults	25 faults	26 faults	24 faults
50%	26 faults	26 faults	28 faults	28 faults	24 faults
60%	28 faults	26 faults	28 faults	28 faults	24 faults
70%	28 faults	26 faults	28 faults	28 faults	24 faults

Test case execution percentage	Average number of fault detection for Bookstore				
	fault detected by numbers of common http request	*fault detected by length of http request*	*fault detected by dependency http request*	*fault detected by clustering SOM*	*fault detected by random*
80%	28 faults	28 faults	28 faults	28 faults	26 faults
90%	28 faults	28 faults	28 faults	28 faults	28 faults
100%	28 faults	28 faults	28 faults	28 faults	28 faults

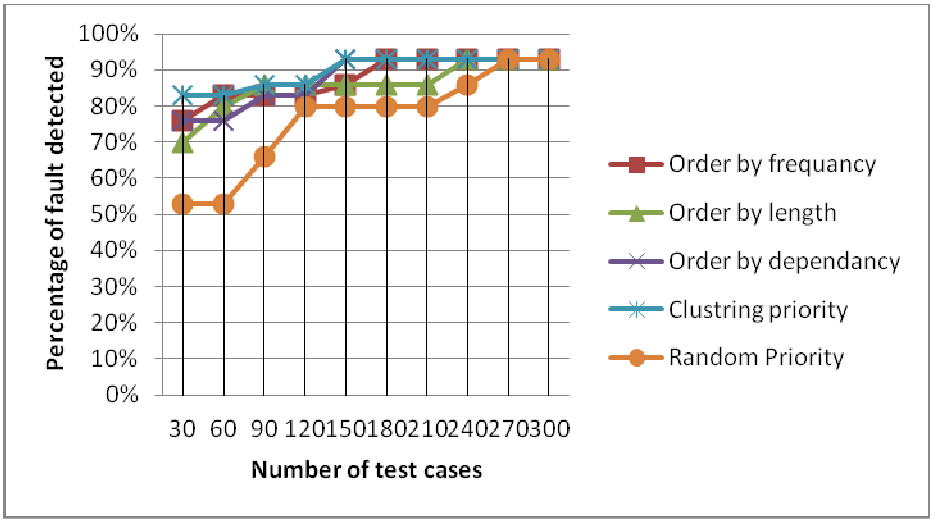

Fig. 4 Results of APFD

Fig. 4 shows the APFD results of the proposed technique and prioritization criteria according to Table 2. As could be seen different test case prioritization criteria have the same APFD. The performance of the proposed clustering technique is 93.33% in terms of fault detection at an early stage as compared to the other techniques.

Despite the same APFD for different criteria, considering the running time of test cases, reveal that the proposed technique based on SOM cluster criterion is more effective comparing to other criteria.

The test suite contains 291 test cases, and the average time for executing 100% of test cases is 2.48 seconds. The average time of executing prioritized test cases is shown in Table 3. As could be seen, prioritization by SOM cluster criteria has the best running time of test cases with a minimum average time, 0.62 seconds.

Table 3: Test case execution time

	Average time of fault detection for Bookstore				
	by numbers of common http request	by length of http request	by dependency http request	by clustering SOM	*by random*
Time	1.9 Sec	1.5 Sec	1.2 Sec	0.62 Sec	2.2 Sec

6 Conclusion

In this paper, we have proposed a new web application regression testing based on real user sessions The proposed technique prioritizes test cases by applying SOM clustering technique The results of this research show that clustering with SOM algorithm has performed effectively to prioritize test cases according to user sessions. This aims to accurately cluster the test cases into groups based on the similarity parameters that are used in the early stages of the process. The use of self-organizing map and new prioritization criteria reduce the time of test suite execution and obtain a higher fault detection rate comparing with random technique.

References

[1] Elbaum, S., Malishevsky, A. G. and Rothermel, G. 2002. Test case prioritization: A family of empirical studies, *Software Engineering, IEEE Transactions* on. Vol.28, No.2, 159-182.

[2] Pertet, S. and Narasimhan, P. 2005. Causes of Failure in Web Applications (CMU-PDL-05-109). *Parallel Data Laboratory*. 48.

[3] Sampath, S., Bryce, R. C., Viswanath, G., Kandimalla, V. and Koru, A. G. 2008. Prioritizing user-session-based test cases for web applications testing, *Proceedings of the 2008 Software Testing, Verification, and Validation*, 2008 1st International Conference on: IEEE, 141-150.

[4] Onoma, A. K., Tsai, W.-T., Poonawala, M. and Suganuma, H. 1998 Regression testing in an industrial environment, *Communications of the ACM*. Vol.41, No.5, 81-86.

[5] Rothermel, G., Untch, R. H., Chu, C. and Harrold, M. J. 2001. Prioritizing test cases for regression testing, *Software Engineering, IEEE Transactions* on. Vol.27, No.10, 929-948.

[6] Yoo, S. and Harman, M. 2012. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability*, Vol.22, No.2, 67-120.

[7] Wong, W. E., Horgan, J. R., Mathur, A. P. and Pasquini, A. 1999. Test set size minimization and fault detection effectiveness: A case study in a space application, *Journal of Systems and Software*. Vol.48, No.2, 79-89.

[8] Leon, D. and Podgurski, A. 2003. A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases, *Proceedings of the 2003 Software Reliability Engineering: IEEE*, 442-453.

[9] Xingmin Luo, Fan Ping, and Mei-Hwa Chen. 2009. Clustering and tailoring user session data for testing web applications, *International Conference on Software Testing Verification and Validation*.

[10] Srikanth, H., Williams, L. and Osborne, J. 2005. System test case prioritization of new and regression test cases, *Proceedings of the 2005 Empirical Software Engineering, International Symposium on: IEEE*, 10.

[11] Tonella, P., Avesani, P. and Susi, A. 2006. Using the case-based ranking methodology for test case prioritization, *Proceedings of the 2006 Software Maintenance, 22nd IEEE International Conference on: IEEE*, 123-133.

[12] Kirda, E., Jazayeri, M., Kerer, C. and Schranz, M. 2001. Experiences in engineering flexible web services, *Multimedia, IEEE*. Vol.8, No.1, 58-65.

[13] Arvind Kumar Upadhyay, A. K. Misra. 2012. Prioritizing Test Suites Using Clustering Approach in Software Testing. *International Journal of Soft Computing and Engineering (IJSCE)*, Vol.2, No.4.

[14] Lazli, L., Mounir, B., Chebira, A., Madani, K. and Laskri, M.T. 2011. Connectionist probability estimators in HMM using genetic clustering application for speech recognition and medical diagnosis, *International Journal of Digital Information and Wireless Communications* Vol.1, No.1, 14-31.

[15] Kohonen, T. 2001. Self-organizing maps of massive databases. *International Journal of Engineering Intelligent Systems for Electrical Engineering and Communications*, Vol.9, No.4, 179–185.

[16] Raeisi Nejad Dobuneh. M, Jawawi. D. N. A, Malakooti. V. M. 2013. An Effectiveness Test Case Prioritization Technique for Web Application Testing. *International Journal of Digital Information and Wireless Communications*. Vol.3, No.4, 117-125

[17] Raeisi Nejad Dobuneh. M, Jawawi. D. N. A, Masitah Ghazali, Malakooti. V. M. 2014 . Development Test Case Prioritization Technique in Regression Testing Based on Hybrid Criteria. *8th Malaysian Software Engineering Conference (MySEC)*, 301-305.

[18] Jin-hua Li and Dan-dan Xing. 2011. User Session Data Based Web Applications Test With Cluster Analysis, *Springer-Verlag Berlin Heidelberg CSIE*, 415–421.

[19] Yue Liu, Kang Wang, Wang Wei, Bofeng Zhang, Hailin Zhong. 2011. User-session-based Test Cases Optimization Method based on Agglutinate

Hierarchy Clustering, *IEEE International Conferences on Internet of Things, and Cyber, Physical and Social Computing*.

[20] Yuan Fang Li, Paramjit K.Das, David L.Dowe. 2014. Two Decades of Web Application Testing: A Survey of Recent Advances, *Information Systems*, Vol.43, (2014), 20–54.