I.J. Information Technology and Computer Science, **2014, 12, 61-66**
Published Online November 2014 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijitcs.2014.12.08

Event-Coverage and Weight based Method for Test Suite Prioritization

Neha Chaudhary
Research Scholar, GBU, Greater Noida, India
Email: neha.chaudhary@gmail.com

O.P. Sangwan
Guru Jambheshwer University of Science & Technology / Department of CSE, Hisar, 125001, India
Email: sangwan_op@yahoo.co.in

Richa Arora
InterGlobe Technologies, Gurgaon, 122001, India
Email: richaarora.arora3@gmail.com

Abstract — There are many challenges in testing of Graphical User Interface (GUI) applications due to its event driven nature and infinite input domain. Testing each and every possible combination of input require creating number of test cases to satisfy the adequacy criteria of GUI testing. It is not possible to test each and every test case within specified time frame. Therefore it is important to assign higher priority to test cases which have higher fault revealing capability than other test cases. Various methods are specified in literature for test suite prioritization of GUI based software and some of them are based on interaction coverage and weight of events. Weight based methods are defined namely fault prone weight based method, random weight based method and equal weight based method in which fault prone based method is most effective. In this paper we have proposed Event-Coverage and Weight based Method (EC-WBM) which prioritizes GUI test cases according to their event coverage and weight value. Weight value will be assigned based on unique event coverage and fault revealing capability of events. Event coverage based method is used to evaluate the adequacy of test cases. EC-WBM is evaluated for 2 applications one is Notepad and another is Calculator. Fault seeding method is used to create number of versions of application and these faults are evaluated using APFD (Average percentage of fault detection). APFD for prioritized test cases of Notepad is 98% and APFD for non-prioritized test cases is 62%.

Index Terms — Event coverage, GUI testing, Test-Suite Prioritization, Event-Coverage and Weight based Method (EC-WBM).

I. Introduction

Graphical User Interface (GUI) is composed of objects (buttons, menus, trash-can, recycling-bin) using metaphors familiar in real life. The software user interacts with the objects by performing events that manipulate the GUI objects as one would with real objects. Events cause deterministic changes to the state of software that may be reflected by a change in the appearance of one or more GUI object [1,2].

There are few important characteristics of GUI which include their graphical orientation, event-driven input, hierarchical structure, the objects they contain, and the properties (attributes) of those objects [3].

GUI Testing: As specified by Paul testing is known as a key Quality Assurance (QA) activity in the development process of software. During testing, test suits are generated and executed on an Application Under Test (AUT) [4].

Test-Case Prioritization: It is important to prioritize the test cases that uncover the most faults as fast as possible in the testing process. So prioritization of test suite is a challenging area. The Test-Case prioritization techniques aim at ordering the test cases from the highest priority of execution to the lowest priority and the test case prioritization is defined as given a test suite T, PT is the set of permutations of T, and f is a function from PT to real numbers [5]. The technique of prioritization is to find T'ϵ PT, such that (V T'')(T'' ϵPT)(T'' $\neq$T')[F(T')$\geq$ f(T'')].

GUI events are classified on the basis of their response to the system on selection and their classification is as follows:- Restricted-focus events, Unrestricted-focus events, Termination events, Menu-open events and System-interaction events [6,2]. Event-weight assignment for different types of events is shown in the Table1. The event type with high weight value (WV) is more important and may detect more number of faults. Thus, considering the system-interaction events, that directly interact with the underlying system codes, more faults may be detected when these event types are triggered. Therefore, the WV = 4 for the system-interaction event. A termination event is an event with medium importance, since it may have underlying codes to execute when it closes a window. Finally, a menu-open event or an unrestricted-focus event does not interact with the underlying software. Hence, the lowest weights are assigned to these two event types.

Events are categorized in five categories as specified in Table 1. Event weight is assigned according to their fault revealing capability defined in literature survey [6].

Event weight and event coverage will be used to prioritize test cases in high to low ordering [8]. If two test cases have same weight value, number of events will be

dominating factor for prioritization. If test cases have same event coverage and same weight value random tie breaking will be used.

Table 1. Event weight assignment [7]

Event type	WVs
Restricted-focus event	5
System-interaction event	4
Termination event	3
Menu-open event	2
Unrestricted-focus event	1

In this paper we have proposed a technique which consider weight value of each event & number of unique event that test case is covering (event coverage) as factors for test suite prioritization.

This paper is organized as follows: Section II describes the related work previous work. Section III demonstrates proposed method for prioritization i.e. "Event-Coverage & Weight based Method" and also includes experimental results. Section IV covers threat to validity. Finally, conclusion and future work are presented in section V.

II. Related Work

This section covers various methods for test case prioritizations for GUI based software. In our recent work we proposed multiple factors for test suite prioritization using fuzzy logic [9].

Renee C. Bryce and Atif M. Memon proposed test suite prioritization using interaction coverage. Test suite for GUI based program is prioritized by t-way interaction coverage and rate of fault detection is compared with fault detection by other prioritization criteria. Experimental results shows that test suits with the highest event interaction coverage benefit the most and test suits that has less interaction coverage does not benefit using this prioritization technique [5].

Atif M Memon and Renee C Bryce provided a single abstract model for GUI and web application testing for test case prioritization. In this approach test cases are prioritized by set of count based criteria, set of usage-based frequency and set of interaction based criteria. The results shows that prioritization by 2-way (interaction based criteria) and PV-LtoS (Parameter count based criteria) has provided better improvement in the rate of fault detection for GUI based software[10].

Authors Xun Yuan et al. proposed combinatorial interaction testing. In this paper authors proposed unique criteria that incorporate context in terms of event combination strength, sequence length, and it includes all possible positions for each event. Authors have included case studies on eight different applications which shows that when event combination strength is increased, starting and ending position of events are to be controlled that will be able to detect large number of undetected faults. These criteria proved effective and efficient. The problems of state-based testing domain also exist in this strategy like if there are infeasible path they will generate infeasible sequences. To remove these infeasible paths manual methods are used [11].

Renee C. Bryce et al. included cost of test case in the prioritization technique based on interaction coverage. Cost-based combinatorial interaction coverage metric as 2way interaction coverage and cost-based 2way interaction coverage was proposed by the authors. According to experimental results the difference in APFDC between 2way and cost-based 2way for CPM was less than 3%. APFDC was slightly less effective [12].

Sebastian et al. provide an analysis of fault detection rates that result from applying several different prioritization techniques to several programs and modified versions. This analysis can be used to determine the prioritization techniques appropriate to other workloads [13].

Yuen Tak and Man Fai proposed fault-based prioritization of test cases which directly utilizes the theoretical knowledge of their fault-detecting ability and the relationships among the test cases and the faults in the prescribed fault model, based on which the test cases are generated [14].

Luay et al. present and evaluate two model-based selective methods and a dependence-based method of test prioritization. These models utilize the state-based model of the system under test. The existing test suite is executed on the system model and information about this execution is used to prioritize tests[15].

Sreedevi Sampath et al. formulate three hybrid combinations based on Rank, Merge, and Choice. They have suggested that hybrid criteria of others can be described using Merge and Rank formulations, and hybrid criteria they have developed most often outperformed individual criteria[16].

Another method for test suite prioritization is proposed by Huang Chin is cost-cognizant test case prioritization which based on the use of historical records. In this paper authors have used genetic algorithm to determine the most effective order [17].

III. Event-Coverage & Weight based Method (EC-WBM)

In this paper we have proposed EC-WBM based approach for test suite prioritization. In this approach test cases are created using QTP tool for GUI based application. Different versions of application are created using manual fault seeding method and fault matrix is generated with the help of QTP tool. Module component extractor takes GUI based application as input and extracts events from the application. Number of events of application and test case are input for prioritization algorithm.

Prioritization algorithm is assigning a weight value for each test case. According to this weight value a new order of test cases is generated. For the comparison purpose we have considered random order of test cases and prioritized order of test cases and their APFD are compared.

As shown in Fig.1, we have designed an experimental set up for test suite prioritization using proposed approach EC-WBM

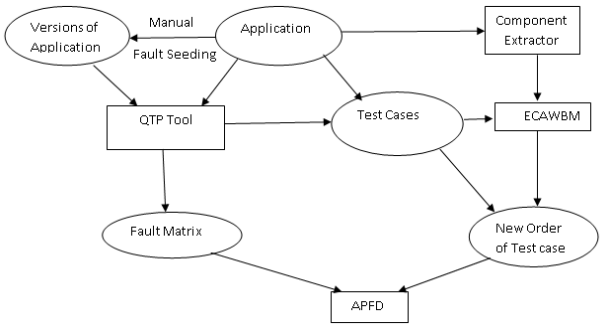

Fig. 1. Experimental Design for test suite prioritization

Independent and Dependent Variables: In this study the independent variables are test suite created by QTP tool and seeded faults. Dependent variables are average percentage of fault detection and prioritized sequence.

The method required to implements the approach is specified in following steps:

Step 1: Generation and Identification of GUI based application

In our experiment we have selected 2 different applications Notepad and Calculator that perform basic arithmetic operations.

Step 2: Generation of test cases using QTP

In this experiment test cases are generated using HP-QTP version 11 [18]. This is Capture and Replay tool. We have generated different set of test cases for both applications. Component extractor is created to extract events from test log generated by testing tool. This will take test log file as input and provide list of events as output.

Further different versions are created for the application by manual fault seeding method and fault matrix is created for both applications.

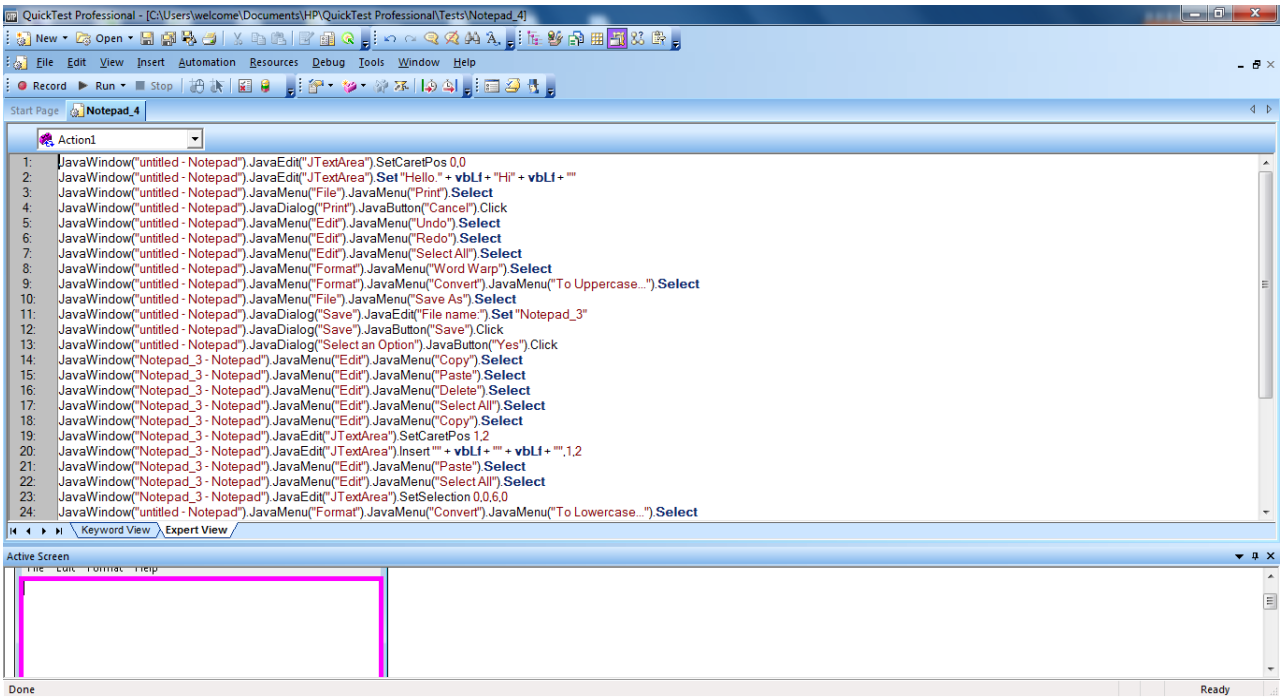

Fig. 2. QTP Log Window

Fault matrix for Calculator is shown in Fig. 3 and for Notepad is shown in Fig. 4.

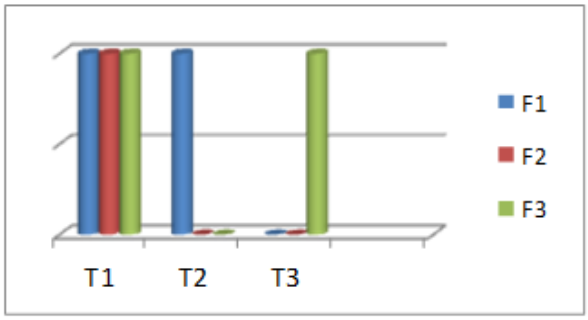

Fig. 3. Fault Chart for Calculator

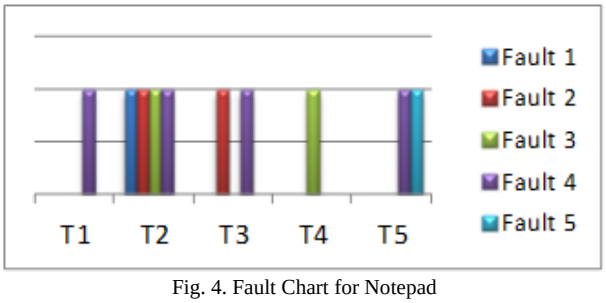

Fig. 4. Fault Chart for Notepad

Step 3: Coverage evaluation of Test Cases

This procedure will take total number of events in the application as input and unique events list as input and provide coverage of test case using (1):

CT[i]= n[i]/Tn (1)

Where CT[i] is event coverage of testCase i, n[i] is number of unique event in testCase i and Tn is total number of events in Application under test.

Table 2. Unique Event Coverage

Application	T1	T2	T3	T4	T5
Notepad	64.2%	60.7%	60.7%	57.1%	50%
Calculator	35%	55%	65%	-	-

Initial unique event coverage for both applications are provided in Table 2.

Step 4: Prioritization of Test Cases

After fetching all the uncovered components into a single excel file implement the prioritization technique to assign some priority on the basis of Fault-prone weight-base method. Prioritization is done on the test cases according to the weight assigned to the components.

In this paper, we have proposed an algorithm that will prioritize test cases according to weight value of events and event coverage. Random ordering of test cases has been used for comparison of proposed method. According to our approach each event will be assigned weight value according to fault revealing capability of that type of event. Then weight of each event will be summed up and multiplied with the coverage of test case. Coverage will be computed by counting number of events in the test case divided by total number of events in the application. For example, suppose there are two test cases t1 and t2 for AUT (Application under Test) with 6 events. Test cases t1 includes events E1, E2 & E3 and t2 includes E1, E4, E5 & E6. According to weight of events we assume E1=2, E2=5, E3=4, E4=2, E5=1 and E6= 3. Weight of t1 will be 7 and priority will be 11* 3/6 =5.5 and weight of t2 will be 8*4/6= 2.67, so t1 will be assigned higher priority than t2, so rate of fault detection capability of t1 will be higher than that of t2. For this work we have developed prioritization algorithm that will assign priority for each test case.

In our algorithm weight of test case will be calculated according to following formula:

$$W_{TC} = n[i] / Tn * \sum_{j=1}^{n} W_j \quad (2)$$

Where W_{TC} is Weight of test case, Wj is the j^{th} event weight, n is the number of events in test case and Tn is the total number of events in AUT.

Algorithm for Test case Prioritization using Event-Coverage & Weight based Method

```
Input computeWeight()
n[i]= number of unique events in test case i
Tn= total number of events in Application
WTC [i] = weight value of i^th test case
CT[i]= event coverage of testCase i
eventWeight[]=weight array of events in bestTest
1.  testcount=1;
2.  hightWTC =0;
3.  for i→1 to totalTestCount
4.  hightWTC = WTC [i];
5.  CT[i]= n[i]/Tn;
6.  WTC [i]= Wi * CT[i];
7.  for j→i+1 to totalTestCount
8.  if(WTC [j]> hightWTC)
9.  hightWTC= WTC [j];
10. bestTest=T[j];
11. if j!= totalTestCount;
12. T[j]=T[j+1];
13. end if
14. end if
15. end for
16. end for
17. end computeWeight
18. while(testCount!=0)
19. while(eventCount in test case i != 0)
20. for j→1 to number of events in bestTest
21. if eventWeight[eventCount]>0 && event of testCase[eventCount]=event of bestTest[j]
22. CT[eventCount]=CT[eventCount]-eventWeight[eventWeight];
23. end for
24. eventCount--;
25. testCount--;
26. end while
27. end while
28. call computeWeight( )
```

The test case with the highest W_{TC} value is selected, which is T1. The current sequence of non-prioritized test cases of application Notepad is:

<T1, T2, T3, T4, T5>

Eliminate all the components which are duplicates of the test cases having high SW, and therefore the sequence of test case will change with the value of summarized weight SW for all the four test cases and ordering also changes and the same will be implemented for all the iterations and the final prioritized sequence is as follows:

<T2, T3, T1, T5, T4>

Step 5: Evaluation Metric:

After prioritizing test cases, and detecting faults in GUI software, the fault detection percentage for test cases will be evaluated using a metric APFD (Average Percentage of Fault Detection). APFD is defined as [19]:

$$APFD = 1 - \frac{TF_1 + TF_2 + \cdots TF_m}{nm} + \frac{1}{2n} \quad (3)$$

Step 6: Experimental Results:

In Notepad application, 5 faults are detected by running the application in the QTP Tool. After prioritizing the test cases using the Event-Coverage & Weight based Method approach prioritized sequence that is obtained from the prioritization approach is:

{T2, T3, T1, T5, T4}

While before the prioritization the sequence of the test cases was:

{T1, T2, T3, T4, T5}

Following are the values of *n* that are total number of test cases and *m* is the total number of faults detected in the notepad application and the value of *n = 5* and *m = 5* should be almost same for calculating the average percentage of fault detection, so putting the value in the APFD equation, the average percentage of faults are detected for Calculator and notepad are specified in Table 3for prioritized sequence. Table 4 specifies APFD value for non-prioritized sequence.

Table 3. APFD for Prioritized Sequence (Notepad and Calculator)

GUI Application	APFD of Prioritized Test Sequence obtained for GUI Application
Notepad	66%
Calculator	98%

In the given Fig. 5 the APFD for the prioritized sequence of Calculator application is represented in which 98% average percentage of fault detection effectiveness is calculated using the APFD method for the prioritized sequence of test cases in Calculator.

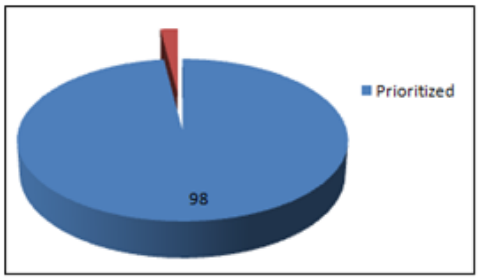

Fig. 5. APFD for Prioritized Calculator Application

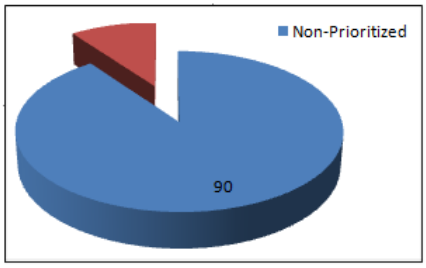

Fig. 6. APFD for Non-Prioritized Calculator Application

Fig.6 depicts APFD for prioritized and non-prioritized order of test cases for Application Calculator.

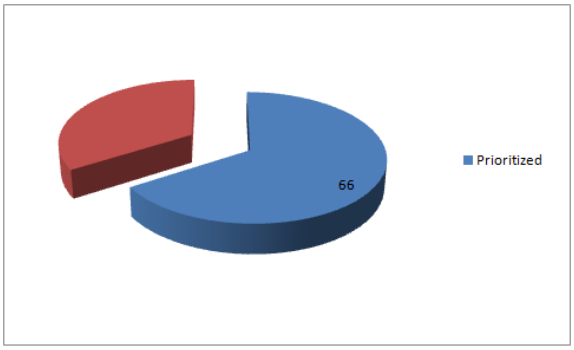

Fig. 7. APFD for Prioritized Notepad application

Fig.7 shows that the APFD of prioritized order of test cases for Application Notepad is comparatively greater than the APFD of the non-prioritized sequence for the same.

The Fig.8 depicts APFD for non-prioritized sequence of GUI application Notepad. Value of APFD for non-prioritized sequence is 62%.

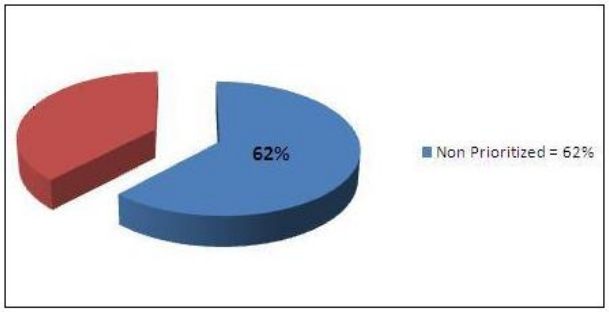

Fig. 8. APFD for Non-Prioritized Notepad Application

Thus, comparing the APFD for both prioritized and non-prioritized test cases the Average percentage of fault detection rate for prioritized sequence is higher than non-prioritized test sequences for both applications thus the rate of fault detection is improved after prioritization.

IV. Threats to Validity

Threats to validity are factors that may impact ability to generalized results to other circumstances. The first threat is the validation of the method fault prone weight based prioritization. For the validation two different applications are considered and these two applications are entirely different in terms of their GUI. These applications are deliberately chosen for the generalization of results. But further experiments should be done for other type of applications. Second threat to validity is the size of application in terms of number of menu and number of events. Both standard applications are considered test case generation in which Notepad application have 28 events and components. Third threat to validity is there may be different cost associated with every test case execution uniform cost of execution is considered in the thesis for evaluation.

V. Conclusion and Future Work

From the analysis of APFD computed for two different applications it is concluded that Notepad application is showing total 4% improvement and Calculator application is showing 8% improvement. Experimental result shows that when prioritization is done using fault prone weight based method there is significant improvement for test cases generated using capture replay tool. In future work we may consider other costs of tests, including scaffolding costs, test execution time, and the time that it takes testers to examine test cases.

REFERENCES

[1] Memon Atif ,"Automatically Repairing Event Sequence-Based GUI Test Suites for Regression Testing," ACM Transaction on Software Engineering and Method, Volume 18, Issue 2, 2008.

[2] Ishan Banerjee , Bao Nguyen, Vahid Garousi, Atif Memon, " Graphical user interface (GUI) testing: Systematic mapping and repository", in the Journal of Information and Software Technology, vol. 55, pp. 1679–1694, March 2013.

[3] Memon Atif, Soffa Lou Mary, Martha E. Pollack, " Coverage Criteria for GUI Testing", Proc.of the 8th European Software Engineering conference held jointly with 9th ACM SIGSOFT international symposium on Foundations of Software Engineering, pp. 256-267, 2001.

[4] Gerrard Paul, "Testing GUI Applications", EuroSTAR, Edinburgh UK, 1997.

[5] Bryce Renee C., Memon Atif ," Test Suite Prioritization by Interaction Coverage", Domain-Specific Approaches to Software Test Automation Workshop, Dubrovnik, Croatia, 2007.

[6] Huang Chin-Yu, Chang Jun-Ru and Chang Yung-Hsin, "Design and analysis of GUI test-case prioritization using weight-based methods," in the Journal of Systems and Software vol. 83, pp. 646-659, 2010.

[7] Memon Atif, Lou Soffa Mary, E. Pollock Martha, "Coverage criteria for GUI testing," in the proceeding of 21st International conference on software engineering, ACM press, pp 257-266, 1999.

[8] Izzat Alsmadi, Sascha Alda,"Test Cases Reduction and Selection Optimization in Testing Web Services," published in the International Journal of Information Engineering and Electronic Business (IJIEEB), Vol.4, No.5, October 2012

[9] Chaudhary Neha, Sangwan O.P., Singh Yogesh, "Test Case Prioritization Using Fuzzy Logic for GUI based Software", International Journal of Advanced Computer Science and Applications, 2012.

[10] Bryce Renee C., Sampath Sreedevi , Memon Atif, "Developing a single model and Test Prioritization Station for Event- Driven Software", IEEE Transaction on Software Engineering, 2010.

[11] Xun Yuan, Myra B. Cohen. And Atif M. Memon, "GUI Interaction Testing: Incorporating Event Context" in IEEE Transactions on Software Engineering, vol. 37, no. 4, pp. 559-574, 2011.

[12] Bryce Renee C., Sampath Sreedevi, Pedersen Jan B., Manchester Schuyler, "Test suite prioritization by cost-based combinatorial interaction coverage", Published in International Journal of System Assurance Engineering and Management vol 2, Issue 2, pp 126-134, 2011.

[13] Sebastian Elbaum, Gregg Rothermel, Satya Kanduri, and Alexey G. Malishevsky, "Selecting a Cost-Effective Test Case Prioritization Technique", *Software Quality Control* 12, pp. 185-210, September 2004.

[14] Yuen Tak Yu and Man Fai Lau., "Fault-based test suite prioritization for specification-based testing", *Inf. Softw. Technol.* 54, pp. 179-202, February 2012.

[15] Luay Tahat, Bogdan Korel, Mark Harman and Hasan Ural, "Regression test suite prioritization using system models", *Softw. Test. Verif. Reliab.* 22, pp. 481-5067, November 2012.

[16] Sreedevi Sampath, Renee Bryce, and Atif Memon, "A Uniform Representation of Hybrid Criteria for Regression Testing", *IEEE Trans. Softw. Eng.* 39, October 2013.

[17] Huang Chin-Yu, Peng Kuan-Li, and Huang Yu-Chi, " A history-based cost-cognizant test case prioritization technique in regression testing," Elsevier journal of The Journal of Systems and Software, 2011.

[18] Kaur and Kumari, HP QuickTest Professional version 11. 2010. HP – QTP version 11, Comparative study of Automated Testing Tools: Test Complete and QuickTest Pro, Punjab University, 2011.

[19] Rothermel G., Untch R.,H. Chu C., Harrold M. J., "Prioritizing test cases for regression testing", IEEE Transactions on Software Engineering, vol. 27 (10), pp. 102-112, 2001.

Authors' Profiles

Dr. Om Prakash Sangwan received his PhD in Computer Science & Engineering and Master of Technology (M.Tech) degree in Computer Science & Engineering with distinction in Research Work from Guru Jambheshwar University of Science & Technology, Hisar, Haryana. He is also CISCO Certified Network Associate (CCNA) and CISCO Certified Academic Instructor (CCAI). His area of research is Software Engineering focusing on Planning, Designing, Testing, Metrics and application of Neural Networks, Fuzzy Logic and Neuro-Fuzzy. He has numbers of publications in International / National Journals and Conferences. He is presently (on EOL from Department of Computer Science & Engineering, School of Information & Communication Technology, Gautam Buddha University, Greater Noida, Uttar Pradesh) working as Associate Professor with Department of Computer Science & Engineering, Guru Jambheshwer University of Science & Technology, Hisar, Haryana,. Before joining the current assignment Dr. Sangwan has worked as Dy. Director with Amity Resource Centre for Information Technology (ARCIT), and LMC & Head, CISCO Regional Networking Academy, Amity Institute of Information Technology, Amity University, Uttar Pradesh. He is also Member of Computer Science Teacher Association (CSTA), New York, USA, International Association of Engineers (IAENG), Hong Kong, IACSIT (International Association of Computer Science and Information Technology, USA, professional member Association of Computing Machinery, USA, IEEE, and Life member, Computer Society of India, India. He has also published a book on Soft Computing Techniques in Software Engineering co-authored by Prof. Yogesh Singh, Hon'ble Vice Chancellor, M.S. University, Baroda, Gujarat.

Neha Chaudhary holds a Masters of Technology and a Bachelor of Engineering degree in Computer Science & Engineering. She is pursuing her Ph.D from Gautam Buddha University. Her research interest includes Web Testing, Software Testing, GUI Testing and metrics development. She has many publications in international journals and conferences to her credit.

Mrs. Richa Arora holds a Masters of Technology in Computer Science & Engineering and a Bachelor of Technology degree in Information Technology. She is currently working as a Technical Writer at Interglobe Technologies, Gurgaon. She has done her research on e-learning, where she and her team created an e-learning portal for online tutorials.