Value based Regression Test Case Prioritization

E. Ashraf, A. Rauf, and K. Mahmood

***Abstract*—Test case prioritization organizes test cases in a way to accomplish some performance goals efficiently. Rate of fault detection is one of most important performance objective. The test cases must run in an order that raises the opportunity of fault detection in the earlier life cycle of testing. Test case prioritization techniques have proven to be beneficial in improving regression testing activities. In this paper, we have proposed an algorithm which prioritizes the system test cases based on the six factors: customer priority, changes in requirement, implementation complexity, requirement traceability, execution time and fault impact of requirement. We conducted a controlled experiment on two industrial data sets to compare the proposed value based test case prioritization algorithm with random prioritization for early rate of fault detection. Average percentage of fault detection metrics has been used to measure the efficiency of proposed and random prioritization and it shows that the proposed value based algorithm is more efficient than random prioritization to generate sequence of test cases for early rate of fault detection.**

***Index Terms*—Test case prioritization (TCP), test case factors, Average percentage of fault detection**

I. Introduction

Software testing requires resources and consumes 30-50% of the total cost of development. Testing is often done in time to market pressure and is supposed to test whole software in a systematic manner to achieve quality as much as possible. Testing also includes many other expectations such as delivering error free versions and checking thorough software iteration in available time and other resources. In this exercise it may not be possible for tester to provide quality product free of bugs to customers so it ultimately raises the possibility of potential risks in software, while on the other hand time slippage occurs for delivering the satisfactory quality assessment of software [1].

Testing has been traditionally performed in value neutral approach in which all software parts are given same testing resources to test but this eventually does not satisfy the end customer as approximate 36% of software functions are only often used [2].Therefore it is meaningless to test the whole software in this way.

Manuscript received Jul 23, 2012; revised Aug 09, 2012. This work was supported by Air University, Pakistan.

E. Ashraf. Author is with the Air University, Islamabad, Pakistan (e-mail: erum.ashraf2011@gmail.com).

A. Rauf. Author is with the College of Computer and Information Sciences, Al-Imam Muhammad ibn Saud Islamic University, Riyadh, KSA (e-mail: rauf.malik@ccis.imamu.edu.sa).

K. Mahmood. is with the Department of industrial Engineering and Management Sciences, University of Engineering and Technology, Taxila, Pakistan (e-mail: khurrum85@hotmail.com).

One type of testing is regression testing in which software is tested after making some changes to it. Regression testing is considered to be very expensive due to repeated execution of existing test cases. Regression testing involves execution of large number of test cases and is time consuming [3]. It is impractical to repeatedly test the software by executing complete set of test cases under resource constraints. Therefore, it is suitable to select partial test cases to test the software for fault identification at early stage. There are many selection techniques for this such as test all or random selection techniques. But selection of test cases according to some pre-defined criteria for early rate of fault detection might be somehow risky technique as there could be many other unselected test cases, which can results in more fault identification. When very high quality software is desired, it might not be wise to discard test cases as in test minimization. Whereas, we can prioritize the test cases by following some criteria such as code coverage, execution time or early fault detection to execute the test cases in that order to achieve the particular goal.

We can take benefit of test case prioritization if we combine it with value based approach. The concept of value has been introduced by Barry Boehm. In value based software approach, more value is awarded to software parts; those are more critical or important to our stakeholders. As objective of software or any business is to increase the return of investment (ROI) so by introducing value in testing, testers can focus on more concerned modules of software to satisfy stakeholders. These modules are identified with the help of stakeholders. Stakeholders give some value to modules and according to some predefined criteria; these modules are assigned some value and tested accordingly. This concept is known as value based test case prioritization [2].

There are many algorithms used for test case prioritization such as greedy, additional greedy, hill climbing etc. Artificial intelligence is very vast field and many of its algorithms are using in software testing such as genetic algorithm or particle swarm optimization.

There is need to optimize the testing resources in a way to provide quality software. In this paper, we present value based test case prioritization algorithm in which test cases are prioritized while considering the worth of value in testing process. Our goal of value based test prioritization is for early fault detection. An experiment has been performed to compare effectiveness of proposed and existing technique. The results imply that the proposed AI value based test prioritization may be a good solution for prioritization problem than random technique.

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

The rest of paper is organized as follows: Section 2 provides briefly the related work. Section 3 presents the problem analysis of test case prioritization. Section 4 is about proposed methodology and experimentation. In Section 5 conclusions and future research is discussed.

II. Related Work

There has been using different criterion for test case prioritization in literature such as code coverage [4, 5, 6, 7] or non-code coverage techniques [8, 9] etc. Li et al. [4] has performed an experiment to compare three greedy algorithms and two meta-heuristics algorithms for test case prioritization under code coverage criteria on six programs. Results showed that it is the size of search space that affects complexity of test case prioritization; not the size of test suite itself. Global search techniques and additional greedy algorithm performed better than local search techniques and greedy algorithms respectively. In another paper seven fault versions of C programs are used in an experiment to compare different statement level techniques by manually seeding faults in programs. Results showed that optimal prioritization greatly improved rate of fault detection [15]. Rothermel et al. has also used code coverage criteria for an experiment on seven fault versions of programs written in C language for early fault detection The results specify that test case prioritization can considerably improve the rate of fault detection of test suites [5].

Artificial intelligence techniques are getting popular to work for test prioritization. Walcott et al. has used genetic algorithm for test case reordering for early fault detection criteria by using coverage information [6]. A controlled experiment along with two case studies has been conducted to evaluate effectiveness of parameterized genetic algorithms. Results were compared with other heuristics including initial, reverse of initial test suit ordering, random and fault aware prioritization. It was required for each test case to be independent from other test cases to maximize the fault detection ability. Results reveal genetic algorithm as more promising than any other technique under time constraint environment.

Fayoumi et al. [7] has used ant colony optimization (ACO) and rough set theory concepts to find best quality test case of unit test for object oriented source code. This approach used method call, passing arguments and control flow dependency graphs. A hybrid novel framework was proposed by inspiring the natural ant. Circulation and exploring best test case value had been done through Ant colony pheromone matrix Rough set is used as stopping criteria rule in proposed model. Bayesian network (BN) approach has also been used to prioritize the test cases [8]. An empirical study on five java objects indicates the effectiveness of feedback mechanism of BN approach in terms of early fault detection. Particle swarm optimization (PSO) is optimization technique of swarm intelligence paradigm. In [9] author has used test case coverage and used PSO to assess the best possible position of test cases in search swarm in modified software units. Existing test case priorities and fitness of test cases were used as parameters for new priorities of test cases. PSO was found to be more effective in term of time and cost than greedy algorithm.

Test case prioritization has also been used to reduce the quality assurance cost as well as for minimizing the fault detection effort. The problem with reducing fault detection effort was that it may cause the information loss, as a result of which debugging cost gets increase. So it was a big challenge to reduce the quality assurance cost which includes both the testing and debugging cost while minimizing the loss of diagnostic fault information. Author has proposed the on-line greedy diagnostic prioritization approach that uses the observed test outcome to determine the next test case. In this approach high utility tests were those tests which maximize the reduction of diagnostic cost at each step on average [10].

III. Problem Analysis

Faster detection of faults and early satisfaction of stakeholders are two objectives which can be fulfilled by a value based testing. Traditionally random prioritization is being used to meet these objectives but it is not yielding the desired results all the times. So there is a need of better technique for overcoming this issue. For this purpose we are proposing value based TCP algorithm to reorder the test cases to produce more optimize results. Our proposed TCP algorithm consider stakeholder's value against requirement and test cases and generate ordering that detects more faults at earlier stages.

IV. Proposed Methodology

We have used value based approach at two levels 1) on requirement level 2) and on testing level. On first level requirement priority is being used to incorporate test case value. This requirement priority is provided by the concerned stakeholders. Stakeholders did ranking of these requirement from 1- 10 by categorizing those as catastrophic, medium complexity and less important. Requirements having more value were more important to stakeholders. On second level development team was requested to grade the test cases by following the respected requirements. These test cases were scored according to some pre-defined factors. These factors are discussed below. Values from these two levels were computed to get the net value. Test cases were prioritized according to produced results for earlier fault detection.

To determine the factors involved in our proposed prioritization algorithm, we consider the following factors from literature in our proposed algorithm: (1) customer priority (2) implementation complexity (3) requirement volatility (4) requirements traceability (5) execution time and (6) fault impact of requirement. We discuss these factors here under.

Approximately 36% of the software functions are only constantly used, while 19% only used often and while the

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

rest percentage is not used at all i.e. 45 % used [2]. Frequent failures are caused by the fault that is situated along the course of regular execution, and greater effort should be made to detect such kind of faults [11, 12]. The perceived customer value and satisfaction can be increased, by giving priority to customer needs for development [13,2]. To address this problem, customers were asked to rate the requirement from 1 to 10 by considering the importance of requirement. Highest customer priority is denoted by 10.

Implementation complexity refers to individual measure of amount of difficulty perceived by the developers of the requirement by the development team. Amland carried out an investigation to determine that the functions with greater McCabe complexity are those with high number of faults [14]. From the total system 20 % modules of the system resulted in 80% of the faults [15, 16, 17]. Approximately 50% of the total faults discovered in a project comprise of those errors that are introduced in the requirement phase [24].

Rigorous defects that deliver to the customer costs hundred times greater averagely to resolve as compared to resolving the same problem in the requirements time [18].Requirements volatility is measured as the number of times the development cycle of a requirement has been changed with respect to when the requirement was first introduced. It is basically a judgment of the requirements change with respect to its start date. It also ranges from 1 to 10[19, 17].

The relationship between various artifacts in a software development process such as requirements, design and test cases is known as traceability [23]. Literature shows that software quality can be improved by taking into account the requirements traceability [21].

In literature many authors have considered execution time of test case as cost of test case [19, 16, 22, 9]. Test case costs should have a large impact on the test case prioritization. In terms of test case cost, it can be related to the resources, such as execution time of test case, hardware costs or even engineers' salaries. In our case test case cost is the execution time of test case.

Fault proneness (FP) of requirements is the identification of the requirements that have the most failures in the previous version by the development team [16].As proved from literature the test efficiency can be improved by concentrating on features that have the greatest number of faults [20] [19].

A. Experimental Setup

We have performed a controlled experiment on two industrial projects to measure the effectiveness of our proposed prioritization algorithm. We have been provided the documentation of these projects. We picked use cases and test cases and asked the customers to rate these use cases. A value based requirement prioritization tool was used to rank these requirements [13]. The particular tool was designed to rate the requirement at stakeholder and expert level. Project managers did the job of expert role in these projects. Microsoft excel 2009 was used as requirement-test case traceability tool. MATLAB 9.0 was used to implement this algorithm.

We have used random number generation to produce 20 different orders of test cases. Test cases are executed in these orders. No of test cases executed to find the faults are calculated. For each project mean value of results are computed. The results of fault detection in both the cases are compared to strengthen the effectiveness of the proposed test case prioritization.

Following table describe details of the selected projects.

TABLE I
PROJECTS DESCRIPTION

Attribute Description	**Project 1**	**Project 2**
Nature of Project	Web based	Web based
No. Of modules	10	10
No. Of Test Cases	20	20
Complexity level	Medium	Medium
Team size	9	6

There were five stakeholders involved in this process. Their role in the process is defined below:

Customer was responsible to provide system requirements, requirement's priority and field failure.

Developers were asked to rank the requirement according to its development complexity level.

Requirement Analyst /Business analyst records the requirements, their priorities and any modifications in the requirements.

Maintenance Engineer resolves the field failures defects and links the failure back to the requirements impacted.

Tester provides test cases for each requirement, map the requirement to its test case, and executes the test cases.

B. Proposed Algorithm

Following is the proposed algorithm of value based PSO.

1. *Get test case factor values for all test cases.*
2. *Compare values factor wise.*
3. *Give highest score to maximum and lowest score to minimum value.*
4. *Assign scores to remaining values by counting the number of terms to which the particular value is greater.*
5. *Repeat 2 to 4 for all test cases against every factor value.*

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

6. *Add these factor values for all test cases.*
7. *If same value is obtained for more than one test case then decision is taken by comparing requirement value of these test cases.*
8. *If use case rating also becomes equal or more than one test cases of same value belong to same use case, then test case will be execute on first come first serve base.*

In our work we have predicted the effectiveness of our proposed algorithm for early fault detection by using APFD metric. APFD metric is first developed by Elbaum et al. [4, 5].This metric has been used to measures the average rate of fault detection per percentage of test suite execution. The APFD is calculated by taking the weighted average of the number of faults detected during the run of the test suite. APFD can be calculated using a notation:

Let T be the test suite under evaluation, m be the number of faults contained in the program under test P and n is total number of test cases while TFi denotes the position of the first test in T that exposes fault i.

APFD= 1-TF1+TF2+ …..+ TFM /NM +1/2N .... (1)

C. Experimental Results

Our algorithm is based on analysis of the percentage of test cases executed to detect the faults and on APFD metric's results. Observing the percentage of executed test cases in earlier fault detection is important as some times regression testing ends without executing all test cases.

Results show that our algorithms can also achieve better performance in this case. For example, in first project if only 75% test cases could be run due to resource constraint, random strategy could detect about 66% faults; while our proposed algorithm detects about 88% faults. In second project if we take 30% test cases to execute; then random strategy could detect about 27% faults; while our proposed algorithm detects about 40% faults. This shows clear evidence that our proposed algorithm is much better in earlier fault detection than random technique. Graphical representation of these results is given below.

We had also validated our results with the help of standard APFD metric. We can observe the improved fault detection rate in earlier testing stage through our algorithm which is the proof of our algorithm; as more efficient and beneficial in earlier fault detection goal; whereas gradual improvement in APFD results are obtained by random strategy later in testing phase.

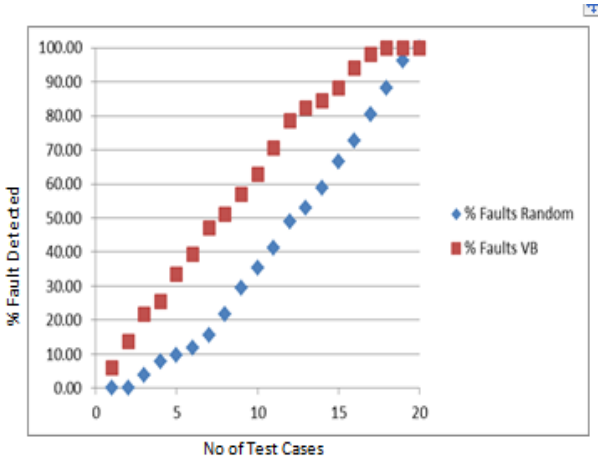

Fig. 1. Project 1 Results

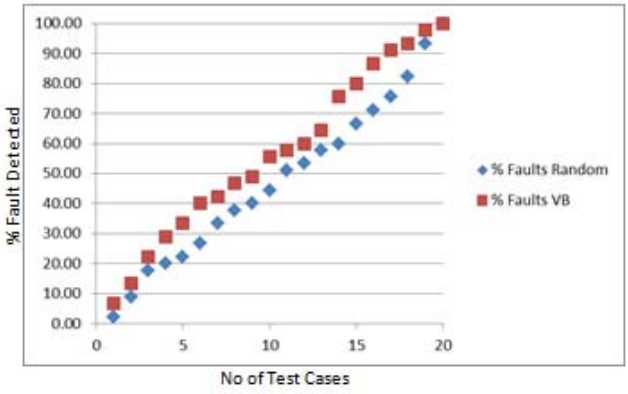

Fig. 2. Project 2 Results

Following tables describe the details of obtained results.

TABLE II
PROJECT 1 RESULTS

Projects	Strategy	APFD results	Executed TC (%)
P1	Random	55%	66%
P1	TCP	78%	88%

TABLE III
PROJECT II RESULTS

Projects	Strategy	APFD results	Executed TC (%)
P2	Random	50%	27%
P2	TCP	67%	40%

In first project our proposed algorithm has detected 78% faults while random ordering produces 55% of faults, in second project our proposed algorithm has detected 67% faults while random ordering produces 50% of faults which again shows significance of our findings.

V. CONCLUSION

The proposed algorithm is novel approach which

ISBN: 978-988-19251-6-9
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

introduced various test case prioritization factors in terms of value of stakeholders. The proposed algorithm works at two levels 1) requirement level and 2) testing level. Six test case factors were being used to rate the test cases while use cases were prioritized by using the existing value based application. Value based algorithm was compared with random prioritization technique on two industrial projects and it showed the effectiveness of value based algorithm for early rate of fault detection.

We are currently working to see the effect of proposed algorithm with evolutionary techniques. Additionally the proposed algorithm is tested on limited data set. It can be validated by taking large size projects having huge pool of use cases and test cases.

ACKNOWLEDGMENT

E.Ashraf author thanks her supervisor, colleagues, friends, family and all well-wishers who supported her to carry out this research.

REFERENCES

[1] R. Ramler, S. Biffl and P. Grunbacher “Value-Based Management of Software Testing”, Book Chapter
[2] B. Boehm and L. Huang, "Value-Based Software Engineering: A Case Study," IEEE Computer, vol. 36, pp. 33-41, Mar 2003
[3] L. Zhang, S. S. Hou, C. Guo, T. Xie and H. Mei “Time Aware Test-Case Prioritization using Integer Linear Programming”, ISSTA’09 , Chicago, Illinois, USA, Jul 2009
[4] Z. Li, M. Harman and R. M. Hierons “Search Algorithms for Regression Test Case Prioritization”,IEEE Trans. on Software Engineering, vol. 33, no. 4, Apr 2007
[5] G. Rothermel, R. Untch, C. Chu and M. Harrold, "Test Case Prioritization: An Empirical Study" Int. Conf. on Software Maintenance, Oxford, UK, pp. 179 - 188, Sep 1999
[6] K. R. Walcott and M. L. Soffa “Time Aware Test Suite Prioritization”, ISSTA’06, Portland, Maine, USA, Jul 2006
[7] M. Fayoumi, P. Mahanti and S. Banerjee: OptiTest: “Optimizing Test Case Using Hybrid Intelligence” World Congress on Engineering 2007
[8] S. Mirarab and L.Tahvildari “An Empirical Study on Bayesian Network-based Approach for Test Case Prioritization” Int. Conf. on Software Testing, Verification, and Validation 2008
[9] K. H. S Hla, Y. Choi and J. S. Park “Applying Particle Swarm Optimization to Prioritizing Test Cases for Embedded Real Time Software Retesting”, 8th IEEE Int. Conf. on Computer and Information Technology Workshops 2008
[10] A. G. Sanchez “Prioritizing Tests for Software Fault Localization” 10th Int. Conf. on Quality Software, 2010
[11] J. C. Munson and S. Elbaum, "Software reliability as a function of user execution patterns and practice," 32nd Annual Hawaii Int. Conf. of System Sciences, Maui, HI, pp. 255-285, 1999
[12] J. Musa, “Software Reliability Engineering”. New York, NY: McGraw-Hill, 1999
[13] B. Boehm, "Value-Based Software Engineering," ACM Software Engineering Notes, vol. 28, pp. 1-12, Mar 2003.
[14] S. Amland, "Risk Based Testing and Metrics," 5th Int. Conf. EuroSTAR ’99, Barcelona, Spain, pp. 1-20, 1999.
[15] R. Krishnamoorthi, S.A. Sahaaya and Arul Mary “Incorporating varying Requirement Priorities and Costs in Test Case Prioritization for New and Regression testing”, 2008
[16] X. Zhang, C.Nie, B. Xu and B.Qu “Test Case Prioritization based on Varying Testing Requirement Priorities and Test Case Costs”, 2007
[17] H. Srikanth, L. Williams and J. Osborne “System Test Case Prioritization of New and Regression Test Cases”, 2005
[18] F. Shull, V. Basili, B. Boehm, W. Brown, P. Costa, M. Lindvall, D. Port, I. Rus, R. Tesoriero and M. Zelkowitz, "What We Learned about Fighting Defects," IEEE Symposium on Software Metrics, Ottawa, Canada, pp. 249-258, Jun 2002.
[19] R. Krishnamoorthi, S.A. Sahaaya and Arul Mary “Incorporating varying Requirement Priorities and Costs in Test Case Prioritization for New and Regression testing”, 2008
[20] T. Ostrand, E. Weyuker and R. Bell, "Where the Bugs Are," Proceedings of the ACM SIGSOFT International Symposium on Software Testing and Analysis, Boston, MA, pp. 86-96, Jul 2004
[21] A. Ahmed, “Software Testing as a Service” Auerbach Publications, New York: 2009
[22] A. M. Smith and G. M. Kapfhammer “An Empirical Study of Incorporating Cost into Test Suite Reduction and Prioritization”, 2009
[23] R. Krishnamoorthi and S.A. Mary “Factor oriented requirement coverage based system test case prioritization of new and regression test cases”,2009
[24] Standish.Group,"CHAOS." http://www.standishgroup.com/chaos.htm.