Historical Value-Based Approach for Cost-cognizant Test Case Prioritization to Improve the Effectiveness of Regression Testing

Hyuncheol Park, Hoyeon Ryu, Jongmoon Baik
School of Engineering, Information and Communications University,
119 Munji-ro, Yuseong-Gu, Daejeon, Republic of Korea
{hcparker, hoyeon, jbaik}@icu.ac.kr

Abstract

Regression testing has been used to support software testing activities and assure the acquirement of appropriate quality through several versions of a software program. Regression testing, however, is too expensive because it requires many test case executions, and the number of test cases increases sharply as the software evolves. In this paper, we propose the Historical Value-Based Approach, which is based on the use of historical information, to estimate the current cost and fault severity for cost-cognizant test case prioritization. We also conducted a controlled experiment to validate the proposed approach, the results of which proved the proposed approach's usefulness. As a result of the proposed approach, software testers who perform regression testing are able to prioritize their test cases so that their effectiveness can be improved in terms of Average Percentage of Fault Detected per Cost.

1. Introduction

A software product, once developed, has a long life and evolves through numerous additions and modifications based on its faults, changes of user requirements, changes of environments, and so forth. With the evolution of a software product, assuring its quality is becoming more difficult because of numerous release versions. It is becoming much harder to manage the software itself. On the other hand, users hope that a new software version has better quality than before. However, sometimes The quality of a software becomes worse than before because the added or modified features create additional faults into the existing product as well as the newly modified version.

Regression testing has been used to support software-testing activities and assure acquiring an appropriate quality through several versions of a software product during its development and maintenance. Regression testing, however, is too expensive because it requires a lot of test case executions, and the number of test cases increases sharply as the software evolves [4, 16]. For this reason, several researches have been conducted to provide effective regression testing techniques.

However, the existing researches as to test case prioritization have had a critical weakness in that the most of them are based on the assumption that all factors in test case prioritization are considered equally: it is a value-neutral situation. For instance, the cost and fault severity of test cases are considered equivalently. In practice, however, those factors heavily affect the effectiveness of testing [20]. For this reason, some researchers have suggested a cost-cognizant test case prioritization technique [1, 2]. However, this cost-cognizant test case prioritization technique reveals a problem; the specific way to estimate cost and fault severity is not clarified even though such estimations are needed.

In this paper, we propose the Historical Value-Based Approach, which is based on the use of historical value to estimate the current cost and fault severity for a cost-cognizant test case prioritization technique. We also conducted a controlled experiment to validate the proposed approach by comparing it with an existing test case prioritization technique. The effectiveness of test case prioritization can be measured using the Average Percentage of Fault Detected (APFD) or Average Percentage of Fault Detected per Cost (APFDc). The result of the experiment proved the usefulness of the proposed approach for improving the effectiveness of test case prioritization in terms of the APFDc.

978-0-7695-3266-0/08 $25.00 © 2008 IEEE
DOI 10.1109/SSIRI.2008.52

2. Background and Related Works

In this section, we provide background information on regression testing, including its conceptual definition, test case prioritization techniques that are directly related to the proposed approach, and metrics to show the effectiveness of test case prioritization techniques.

2.1. Regression Testing

Regression testing is a kind of software testing that focuses on selective retesting through various versions of a software system [20]. The following is the formal definition of regression testing used by IEEE.

"Selective retesting of a system or component to verify that modifications have not caused unintended effects and that the system or component still complies with its specified requirements" [22]

In short, the basic idea of regression testing is the revalidation of a software system in order to figure out whether the modifications of the software system cause errors or not among the several versions of the software system. Because regression testing is highly expensive, several techniques have been researched for effective and efficient regression testing [14, 17, 19]. There are four major techniques for regression testing: retest-all [15], regression test selection [18], test suite reduction [13], and test case prioritization [4, 5]. Among them, test case prioritization has been perceived as one of the most effective and efficient techniques for regression testing [4].

2.2. Cost-cognizant Test Case Prioritization

Since test case prioritization was introduced, there has been an important weakness in the technique; there has been no consideration of test costs and fault severities. In practice, however, the cost of each test case and the severities of each fault are not equal. For this reason, test case prioritization techniques often produce no appropriate test orders in practice [2].

Cost-cognizant test case prioritization incorporates test costs and fault severities into test case prioritization [1, 2]. In short, cost-cognizant test case prioritization considers the test cost and fault severity of each test case as important factors, and the test cost and fault severity are used for prioritizing test cases on the existing test case prioritization algorithms.

2.3. History-Based Test Prioritization

History-based test prioritization is based on the use of historical test execution data and a regression test selection technique. The keys of history-based test prioritization are the following two factors [3]:

- A procedure for test case prioritization
- How to set and assign the selection probabilities

The selection probabilities are calculated from the following formula [3]:

- $P_0 = h_1$
- $P_k = \alpha h_k + (1 - \alpha)P_{k-1}$ $(0 =< \alpha =< 1, k => 1)$

Let P_k be the selection probability of a test case at kth execution. Also, let h_k be a set of test execution times which is a time-ordered observation from 1st execution to kth execution. By changing α, the last probability of a test case and the previous probability of the test case affect the current probability of the test case. The historical information regarding each test case's execution is used to increase or decrease the selection probabilities at a current testing session.

2.4. Metrics of Test Effectiveness

There is a metric, APFD, to measure the prioritized order of test cases in a test suite in terms of the effectiveness of a test case prioritization technique [4, 5]. APFD focuses on increasing a test suite's rate of fault detection, how quickly the faults are detected during testing processes, in order to measure the average cumulative percentage of faults detected over the execution of test cases in a test suite's given order. Also, APFD quantifies the effectiveness and efficiency of the order of test cases and measures a test suite's rate of fault detection using two criteria, percentage of test suite executed and percentage of fault detected.

Basically, APFD is based on an assumption that all test cases have equal cost and all faults have equal severity. However, test costs and fault severities vary widely in practice. If the assumption is broken, APFD produces no appropriate results. For this reason, APFDc was introduced [1, 2]. APFDc adopts the considerations of test costs and fault severities into APFD in the view of Value-Based Software Engineering (VBSE).

In short, APFD is a metric used to show the effectiveness of test case prioritization, and it is value-neutral in terms of test cost and fault severity. In contrast, APFDc is a metric used to show the effectiveness of test case prioritization in terms of value, test cost, and fault severity.

3. Research Approach: Historical Value-Based Approach for Cost-cognizant Test Case Prioritization

In this section, we describe the proposed Historical Value-Based Approach for cost-cognizant test case prioritization. It includes an overview of the proposed approach, the contents of the approach with some examples, and a historical value model.

3.1. Overview of the approach

The proposed approach focuses on the use of historical information to determine the priority of given test cases. By using the historical information of the costs of the test cases and the fault severities of detected defects in a test suite, the historical value of the test cases is calculated and used for the basis of test case prioritization. Additionally, the historical value can be combined with not only a cost-cognizant test case prioritization technique, but also several existing test case prioritization techniques such as a coverage-based test case prioritization technique.

Namely, the historical value is calculated from the previous test costs and fault severities of detected defects in a test suite. Then, the historical value is used for the factor that affects the prioritization of test cases in a given test suite. The following figure shows the overall description of the proposed approach.

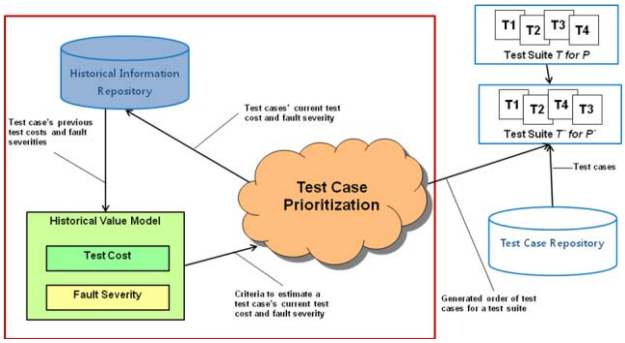

Figure 1. Overview of Historical Value-Based Approach

The following explains the above figure.

- *P* is a software system and *P`* is the modified version of *P*.
- To conduct regression testing for *P`*, a test suite is composed of the test cases from the test case repository.
- The cost of a test case and fault severity of the detected defects, which are the results from the execution of a test case, are stored in the historical information repository.
- When the prioritization is required, the historical value model uses the stored historical information, the test costs of the test cases and the fault severities of the detected defects, and calculates the historical value.
- The calculated historical value is used for the criterion of prioritizing test cases in a test suite.

3.2. Historical Value-Based Approach for Cost-cognizant Test Case Prioritization

The Historical Value-Based Approach for cost-cognizant test case prioritization focuses on prioritizing test cases in terms of historical value. This means that the previous test costs of test cases and the fault severities of previously detected faults are used for a criterion of test case prioritization; however, the Historical Value-Based Approach has one assumption, that the test costs of the test cases and the fault severities of detected faults are not significantly changed from one release to a later one. We are concerned with this assumption, but it is rarely broken in general cases [3].

3.2.1. Test Cost and Fault Severity

Test costs are greatly diversified in software testing. Depending on the criteria, a test cost can be refined through several factors such as machine time, human time, test case execution time, monetary value of the test execution, and so forth [2].

Similarly, fault severity can also be refined by depending upon such criteria as test criticality (the criticality of the test case that detects a fault) and function criticality (the criticality of the function in the code that is covered by the test case).

In our approach, we refined test cost as the test case execution time of a test case. It is the most widely used definition to refine test costs in previous researches on test case prioritization [1, 2]. Fault severity is refined to test case criticality, which is designated to each test case by software testers.

3.2.2. Definitions used in the Historical Value-Based Approach

In our approach, some definitions are used for its designated meaning. In this part, we provide all of the definitions as follows.

- Let P be a software system.
- Let P` be the next version of software system P.
- Let n be the number of the test cases that consists of a test suite.
- Let T be a test suite, that is composed of the test cases from t_1 to t_n, for testing P.
- Let T` be a test suite, that is composed of the test cases from $t_1$ to $t_n$, for testing $P$`.
- Let t_i be a test case that is involved in test suite T.
- Let m be the number of the faults that can be detected by test suite T.
- Let $HV(t, i)$ be a historical value of test case t on an ith execution (from i-1th execution). ($i > 0$)
- Let $C(t, i)$ be the relative cost of test case t's ith execution by comparing the maximum cost among the cost of test cases in a test suite. (cf. C_0 = the mean value of the cost of all test cases in a test suite)
- Let $FS(t, i)$ be the relative total fault severities of test case t's ith execution, the sum of the fault severity of the faults that are revealed by test case t, by comparing the maximum fault severity revealed by test case t's ith execution. (cf. FS_0 = the mean value of the fault severity of all test cases in a test suite)
- Let wC_i be a weight factor of C with an ith execution of test cases.
- Let wFS_i be a weight factor of FS with an ith execution of test cases.
- Let $min(X_1, X_n)$ be a function that receives the minimum value between X_1 to X_n.

3.2.2. Historical Value Model

A historical value model is a model to quantify the historical value of a test case, $HV(t, i)$, in terms of previous test costs $C(t, i-1)$ and previous fault severity $FS(t, i-1)$. For test cost $C(t, i-1)$ and fault severity $FS(t, i-1)$, the relative values are used by comparing the maximum value among them. There are a software system P and five test cases each have their own cost as follows.

Table 1. Example to explain the relative cost

Test cases	Cost	Relative Cost
A	2	50
B	1	25
C	4	100
D	2	50
E	1	25

For instance, there are five test cases A, B, C, D, and E. Each of them has a cost from 1 to 4. Test case C has a maximum cost of 4, which is the maximum relative cost of 100 among the five test cases. By comparing the cost of C, which has the maximum value, the relative costs of the other test cases are each determined proportionally. Consequently, test case A has a relative cost of 50 because its cost is 2, half of the cost of C.

Fault severity is not same with cost. The following table shows an example of faults and their severities.

Table 2. Example to explain the total fault severities

Faults	Fault Severity
1	4
2	2
3	5
4	1
5	2
6	2
7	4
8	2
9	2
10	1

For instance, there are ten faults, from No. 1 to No. 10. Each of them has a fault severity from 1 to 5. If test case A can detect faults No. 1 and No. 4, test case A has a total of 6 fault severities. Assuming that each test case has a fault severity as in the following table, their relative fault severities are determined by depending on the maximum total fault severity among the test cases.

Table 3. Example to explain the relative total fault severity

Test cases	Total Fault Severity	Relative Total Fault Severity
A	6	60
B	4	40
C	10	100
D	3	30

E	2	20

Namely, test case A has a relative total fault severity of 40, by comparing the maximum relative total fault severity of 100, which is that of test case C. Similarly, the other test cases have a relative total fault severity with the proportion of their fault severities compared with the maximum value.

The weight factor wC_i at an ith execution is defined as follows:

$$wC_i = \frac{\overline{C_i}}{\overline{C_i} + \overline{FS_i}}$$

Similarly, the weight factor wFS_i at an ith execution is defined as follows:

$$wFS_i = \frac{\overline{FS_i}}{\overline{C_i} + \overline{FS_i}}$$

$\overline{C_i}$ is the mean value of C, the total relative costs, at an ith execution. Also, $\overline{FS_i}$ is the mean value of *FS*, the total relative fault severities, at an ith execution. Those weight factors are used for balancing two other kinds of values. This kind of use of weight factor is presented in the previous research with respect to requirement-based test case prioritization [21]. Because the historical value is basically a summation of *C* and *FS*, each weight factor, wC_i and wFS_i, represent the proportion of the each value in the historical value.

The historical value of test case t at an ith execution is defined as follows:

$$HV_{(t,i)} = [(100 + \min(C_{(t,1)}, C_{(t,n)}) - C_{(t,i-1)}] \times wC_{i-1} + FS_{(t,i-1)} \times wFS_{i-1}$$

HV(t, i) is the historical value of test case t's at an ith execution. It consists of two values: the relative cost, which is weighted by wC_i, and the total relative fault severities, the sum of the fault severities of the faults that are revealed by test case t, which is weighted by wFS_i. Because the actual value from the cost has an adverse relationship with cost, the maximum value of relative cost 100, the minimum cost among the test cases is added to the formula to inverse the value of cost. This kind of formula, a composition of a value with another value, is used in the previous research as to requirement-based test case prioritization [21]. We adopted a similar way in order to make a model to compose two different values, test cost and fault severity. Cost and fault severity, due to the use of relative values, can be considered independently, and the addition of relative cost and relative fault severity can be admitted in the calculation of the historical value.

The following figure shows the algorithm to calculate historical value *HV(t, i)*.

Input: Test suit *T* with *n* test cases (from t_1 to t_n), the current time of testing *i*, cost of *test case t* at *i-1* time $C_{(t,(i-1))}$, fault severity of *test case t* at *i-1* time $FS_{(t,(i-1))}$
Output: *Test case t's* historical value $HV_{(t,i)}$ at *i*th test execution
1: begin
2: set *T* empty
3: calculate wC_i
4: calculate wFS_i
5: for each test case $t \in T$ do
6: get $C_{(t,(i-1))}$ and $FS_{(t,(i-1))}$
7: calculate an historical value $HV_{(t,i)}$ of *t*
8: end for
9: end

Figure 2. An algorithm to calculate historical value *HV(t, i)*

4. Experiment

In this section, we explain the experiment to validate and prove the effectiveness of the proposed Historical Value-Based Approach for cost-cognizant test case prioritization. The environment used to conduct the experiment is depicted with the hypotheses, variables, and measures of the experiment. Last, we provide and explain an analysis of the experimental results.

4. 1. Hypotheses

Cost-cognizant test case prioritization with the Historical Value Based Approach produces better results than a coverage-based test case prioritization technique.

4. 2. Variables and measures

There are two variables of the experiment:

- Cost-cognizant test case prioritization with Historical Value-Based Approach
- Functional coverage test case prioritization

For a measure of the experiment, APFDc is used for comparison with two test case prioritization techniques.

4. 3. Experimental Environment

Some researchers have conducted their researches to prove the effectiveness of the test case prioritization by using an experimental environment [7, 8, 9, 10, 11, 12]. Those researches aim to provide for assessing test case prioritization techniques using hand-seeded faults based on the *JUnit* testing framework. Because it based on *JUnit*, the test execution is also performed on the *JUnit* environment. For the target of the experiment, an open-source Java software system, *ant*, is used. The testing objects, ant, whose faults are seeded by hand, and tools came from [10]. For a coverage analysis of the target, a dynamic Java analyzer, *Sofya*, is used [23].

Additionally, we developed the following module for the experiment:

- A module to store the historical information into the historical information repository
- A module to calculate the historical value by composing the cost and fault severity from the testing object and their historical information
- An interface module to integrate those objects, tools, and historical information.
- A module to calculate APFDc

The following figure describes the overall structure of the experimental environment.

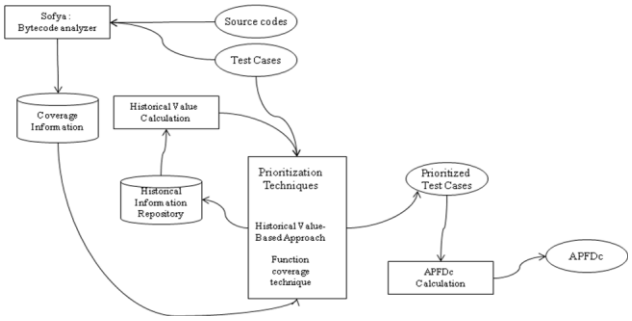

Figure 3. Overview of the experimental environment

From the execution of test cases in the target objects, their cost, execution time, and fault severity derived from the test criticality are stored in the historical information repository. The stored information is used for determining the historical value, and it continuously affects the Historical Value-Based Approach. By using *Sofya*, the coverage information is derived and stored in the coverage information repository.

For prioritizing the current test cases, two test case prioritization techniques are used, and they generate a prioritized order of test cases. Finally, the APFDc calculator calculates the effectiveness of the prioritized order of test cases.

4. 4. Analysis of the experimental result

The target of the experiment is comprised of several versions of *ant* with *JUnit* test cases. Those *ant* systems and test cases are provided by SIR and are fault seeded. The following table shows the overall target system.

Because the execution time of *JUnit* test cases are too short, each test case is executed 10,000 times, and the mean execution time of the overall execution is stored into the historical information repository for the costs of the test cases. Each *JUnit* test case has its test criticality and is used as the fault severity of each test case. Because there are lots of test suites by using the test cases of each version, we summarize the APFDc value of every test suite by using the mean value of the APFDc for each prioritized test suite.

The following Tables 4 and 5 show the experimental results by the above procedure. Table 4 describes the results from the first experiment, and Table 5 depicts the results from the second experiment with a correction of the Historical Value-Based Approach.

Table 4. Mean APFDc values from the experimental results when the intial value is 0

Version	Number of test cases	Mean APFDc for Functional coverage test case prioritization	Mean APFDc for Historical Value-Based Approach
0	34	62%	46%
1	34	60%	73%
2	52	58%	52%
3	52	61%	79%
4	101	62%	42%
5	104	66%	80%
6	105	64%	79%
7	150	62%	45%
8	151	65%	78%

The following figure shows the changes in the trend of mean APFDc value through *ant* versions in the first experiment. The black graph shows the trend of mean APFDc value when the Historical Value-Based Approach is used, and the pink graph shows the

trend of mean APFDc value of functional coverage test case prioritization. Sometimes, there are huge declinations of mean APFDc value when the Historical Value-Based Approach is used. The black graph even shows a worse mean APFDc value than functional coverage based test case prioritization.

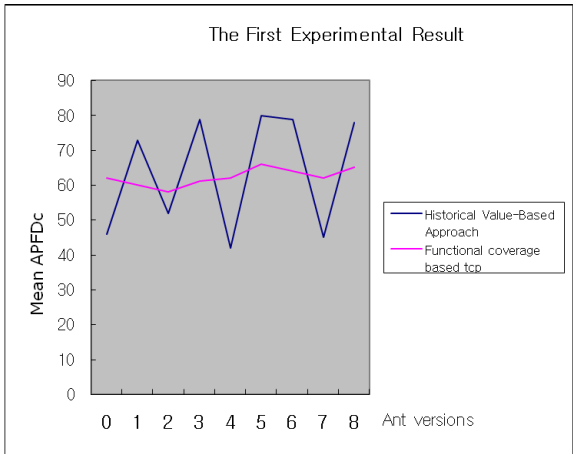

Figure 4. The change trends of mean APFDc values through ant versions in the first experiment

In the earlier part of our experiment, we used the initial value 0 for the newly added test cases. However, this caused a significant fall of mean APFDc value in the first experiment when the number of test cases is significantly changed.

For this reason, we modified our approach to take the mean value of the cost and fault severity of all test cases for the initial historical information of the newly added test cases and then conducted a second experiment. Table 5 shows the experimental result of the second experiment, where the decline of the mean APFDc value is greatly decreased.

Table 5. Mean APFDc values from the experimental results when the intial value is the mean value of the prior test cases' historical information

Version	Number of test cases	Mean APFDc for Functional coverage test case prioritization	Mean APFDc for Historical Value-Based Approach
0	34	62%	46%
1	34	60%	73%
2	52	58%	65%
3	52	61%	79%
4	101	62%	68%
5	104	66%	80%
6	105	63%	79%
7	150	62%	70%
8	151	65%	78%

The following figure shows the changes in trend of mean APFDc value through ant versions in the second experiment. The black graph shows the trend of mean APFDc value when the Historical Value-Based Approach is used, and the pink graph shows the trend of mean APFDc value for functional coverage test case prioritization. Also, there are falls in the black graph, and this shows the declination of mean APFDc value when the Historical Value-Based Approach is used. However, the declination of the mean APFDc value is largely decreased when the Historical Value-Based Approach is used, and the declined mean APFDc values are greater than the mean APFDc value, which comes from the functional coverage based test case prioritization. Consequently, the black graph always shows a better mean APFDc value than functional coverage based test case prioritization.

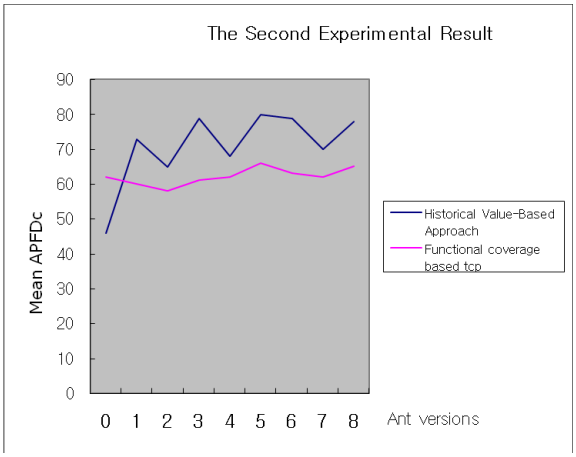

Figure 5. The change trends of mean APFDc values through *ant* versions in the first experiment

5. Conclusions and Further Works

In this paper, we suggest the Historical Value-Based Approach for a cost-cognizant test case prioritization technique that includes an estimation of the trends of cost and fault severity by using historical information. We also conducted a controlled experiment to validate the proposed approach, the results of which proved its usefulness and effectiveness. As a result of the experiment, the Historical Value-

Based Approach for cost-cognizant test case prioritization produces better results, in terms of APFDc, than functional coverage-based test case prioritization techniques.

The major contributions of this research are the following two points. First, it provides a way to estimate the cost and fault severity of the current test case by using historical information. Second, the proposed approach can complement other test case prioritization techniques because it can be combined with other test case prioritizations.

We have many future works for enhancing the proposed approach and providing the best test case prioritization technique. First, more sufficient experimental data are required because we only consider two techniques, the Historical Value-Based Approach and functional coverage test case prioritization. Second, we only used a linear scale for fault severity. For instance, we can use many kinds of scale for fault severity such as exponential or logarithmic scales. Third, for cost and fault severity, one of them may have greater priority than the other one. To support this situation, we should provide a way to give more priority to one of them, cost or fault severity. Finally, we are working on appending a consideration of other factors that affect the prioritization of the test cases such as the number of not yet detected defects in a test case.

Acknowledgement

This research was supported by the MIC(Ministry of Information and Communication), Korea, under the ITRC(Information Technology Research Center) support program supervised by the IITA(Institute of Information Technology Advancement) (IITA-2008-(C1090-0801-0032))

References

[1] Sebastian Elbaum, Alexey Malishevsky, and Gregg Rothermel, "Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization", Proceedings of the Internal Conference on Software Engineering (ICSE`01), May 2001.

[2] Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel, and Sebastian Elbaum, "Cost-cognizant Test Case Prioritization", Technical Report TR-UNL-CSE-2006-0004, University of Nebraska-Lincoln, March 2006.

[3] Jung-Min Kim and Adam Porter, "A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environments," Proceedings of the International Conference on Software Engineering (ICSE`02), May 2002.

[4] Sebastian Elbaum, Alexey G. Malishevsky, and Gregg Rothermel, "Test Case Prioritization: A Family of Empirical Studies", IEEE Transactions on Software Engineering, vol. 28, no. 2, pp. 159-182, Feburary 2002.

[5] Gregg Rothermel, R. Untch, C. Chu, and Mary J. Harrold, "Prioritizing Test Cases for Regression Testing," IEEE Transactions on Software Engineering, vol. 27, no. 10, pp. 929-948, October 2001.

[6] .

[7] Hyunsook Do and Gregg Rothermel, "A Controlled Experiment Assessing Test Case Prioritization Techniques via Mutation Faults", Proceedings of the International Conference on Software Maintenance (ICSM`05), 2005.

[8] Hyunsook Do and Gregg Rothermel, "On the Use of Mutation Faults in Empirical Assessments of Test Case Prioritization Techniques", IEEE Transactions on Software Engineering, vol. 32, no. 9, September 2006.

[9] Hyunsook Do, Sebastian Elbaum, and Gregg Rothermel, "Supporting Controlled Experimentation with Testing Techniques: An Infrastructure and Its Potential Impact", Empirical Software Engineering, vol. 10, no. 4, pp. 405-435, 2005.

[10] Software-artifact Infrastructure Repository, http://sir.unl.edu

[11] Hyunsook Do, Gregg Rothermel, and A. Kinneer, "Prioritizing JUnit Test Cases: An Empirical Assessment and Cost-Benefits Analysis," Empirical Software Engineering, vol. 11, no.1, pp. 33-70, Mar. 2006.

[12] Hyunsook Do, Gregg Rothermel, and Alex Kinneer, "Empirical Studies of Test Case Prioritization in a JUnit Testing Environment", Proceedings of the International Symposium on Software Reliability Engineering (ISSRE'04), November 2004.

[13] Sebastian Elbaum, D. Gable, and Gregg Rothermel, "Understanding and Measuring the Sources of Variation in the Prioritization of Regression Test Suites," Proceedings of the International Software Metrics Symposium, April 2001.

[14] Jung-Min Kim, Adam Porter, and Gregg Rothermel, "An Empirical Study of Regression Test Application Frequency," Proceedings of the International Conference on Software Engineering (ICSE`00), June 2000.

[15] Hareton K. N. Leung and Lee White, "Insights into Regression Testing", Proceedings of the International Conference on Software Maintenance (ICSM'89), October 1989.

[16] K. Onoma, W-T. Tsai, M. Poonawala, and H. Suganuma, "Regression Testing in an Industrial Environment," Communications of ACM, vol. 41, no. 5, pp. 81-86, May 1988.

[17] Gregg Rothermel, Sebastian Elbaum, A.G. Malishevsky, P. Kallakuri, and X. Qiu, "On Test Suite Composition and Cost-Effective Regression Testing," ACM Transactions on Software Engineering and Methodology, vol. 13, no. 3, pp. 227-331, July 2004.

[18] Gregg Rothermel and M.J. Harrold, "Analyzing Regression Test Selection Techniques," IEEE Transactions on Software Engineering, vol. 22, no. 8, pp. 529-551, August 1996.

[19] A. Srivastava and J. Thiagarajan, "Effectively Prioritizing Tests in Development Environment," Proceedings of the International Symposium on Software Testing and Analysis, July 2002.

[20] W.E. Wong, J.R. Horgan, S. London, and H. Agrawal, "A Study of Effective Regression Testing in Practice," Proceedings of the International Symposium on Software Reliability Engineering, November 1997.

[21] Hema Srikanth and Laurie Williams, "On the Economics of Requirements-Based Test Case Prioritization", Proceedings of the International Workshop on Economics-Driven Software Engineering Research (EDSER'05), July 2005.

[22] Institute of Electrical and Electronics Engineers(IEEE). *IEEE Standard Computer Dictionary: A Compilation of IEEE Standard Computer Glossaries*. New York, NY: 1990.

[23] Alex Keener, *Sofya – dynamic program analysis for Java software*, http://sofya.unl.edu