Failure History Data-based Test Case Prioritization for Effective Regression Test

Jeongho Kim
College of Software
Sungkyunkwan University
Republic of Korea
jeonghodot@skku.edu

Hohyeon Jeong
College of Software
Sungkyunkwan University
Republic of Korea
jeonghh89@skku.edu

Eunseok Lee
College of Software
Sungkyunkwan University
Republic of Korea
leees@skku.edu

ABSTRACT

For regression testing in the continuous integration environments, the time and cost should be considered; to satisfy these constraints, it is necessary to improve the test efficiency regarding the achievement of the test goal. It is especially important to identify the problem quickly by first executing a test case with a high probability of failure. This paper therefore proposes the FHD (Failure History Data)-Prioritization technique for the purpose of effective regression testing. This technique uses the failed test case history, the flipped result of method test case and the correlation data as the prioritization criteria, and the algorithm is designed to calculate the weight through the following two-step classification: 1) The FHD-Prioritization analyzes the failure history data statistically and sequentially arranges the test cases from the highest failure occurrence probability in the current session. 2) If a failure occurs during the test, the FHD-Prioritization reprioritizes in real time based on the correlation data. The performance of the FHD-Prioritization technique is evaluated with Tomcat and Camel, Apache open source software projects that were developed in the continuous integration environment. Because all of these projects are composed of the real faults of real-world projects, it is possible to practically evaluate the efficiency of the proposed approach. The FHD-Prioritization improved the efficiency of test case prioritization by about 5.62% and 2.17%, respectively, compared to ROCKET and AFSAC.

CCS Concepts

• **Software and its engineering → Software testing and debugging**

Keywords

test case prioritization; regression test; continuous integration environments; failure history data; failed test case history data; flipped result of test case data; correlation data

SAC 2017, April 03-07, 2017, Marrakech, Morocco

DOI: http://dx.doi.org/10.1145/3019612.3019831

1. INTRODUCTION

Regression testing is used to verify the modified source code, and it is a mandatory task during the maintenance process. The simplest way to do this is to run all of the regression test cases [1]. The drawback, however, is that the number of regression test cases to be executed increases sharply as the version of the software is heightened, so that even the number of regression test suites may be too high to run the normal regression testing. Therefore, the following three test case optimization techniques have been developed to satisfy the testing goal within the time limit [2]. Minimization [3] is a way to eliminate a part of a test case that duplicates or reduces the efficiency. Selection [4] is a technique for the selection of a part of a test case that can test the changed source code through the current and previous version comparisons. Prioritization [5] is a technique that sequentially sorts the probabilities of finding the test cases that result in failure. In other words, the minimization and selection techniques reduce the number of test cases, and prioritization is a technique that changes the order of the test cases.

In continuous integration environment, if a failure is first detected during test, testing system reports it immediately to the developer before testing is completed. In addition, it is often the case that the test result of the same source file flip due to other modified source files or dependency problems such as library, database. Thus, it is important to maintain the accuracy and improve the efficiency by ordering the entire test cases in the order of increasing probability of finding failure, rather than reducing the number of test cases [6].

Many researchers have proposed approaches regarding the prioritization of test cases for various objectives; in doing this, they have mainly considered the fault detection rate. Developers need to know when a regression fault is caused by the changes they have made; furthermore, they practically need to acquire the test data regarding the faults from the regression tests because of the subsequent debugging steps such as fault localization and the fix. The existing prioritization techniques for the improvement of the fault detection rate are based on the test information in the forms of the coverage [7], model [8], and requirements [9]. The historical data-based approaches for which the information from historical data are utilized form one of these prioritization techniques. The approaches that use the historical test data are used to identify the test cases that are error-prone or that comprise high fault-detection capabilities. In this respect, the historical data-based approaches represent a way to effectively obtain failure information.

Therefore, this paper proposes a test case prioritization method that is based on the failure history data. The proposed method performs

the analysis in two steps and prioritizes the test cases based on the results. The main contributions are as follows:

1. FHD-Prioritization
   This paper proposes a test case prioritization that is based on the failure history data of test cases. Step 1. A statistical analysis of the failure history data from previous versions calculates the weight of the individual test cases and sorts them in ascending order based on the results. Step 2. If a failure occurs during the regression test, a reprioritization is performed based on the correlation data. For the next session test, the failure information of the current session is updated in the correlation data.

2. Evaluation
   This paper evaluates the efficiency of the application of the proposed method to two open source projects. The subject projects are large-scale and real-world software programs without manually seeded faults. Accordingly, the practical results of the authors' experiments including the prioritization approaches that are proposed in this paper are shown.

The rest of this paper is organized as follows: In section 2, the authors introduce the related works. Section 3 explains the proposed FHD-Prioritization technique. Section 4 presents the experimentation including the details of the results and the analysis. In section 5, the conclusions are explained.

2. RELATED WORKS

2.1 Test Case Prioritization

The test case prioritization problem was formally defined by Rothermel et al. [5] as follows:

Definition 1. *The Test Case Prioritization Problem:*
Given: T, a test suite, PT, the set of permutations of T; f, a function from PT to the real numbers.
Problem: Find $T' \in PT$ *such that* $(\forall T'')\,(T'' \in PT)\,(T'' \neq T')\,[f(T') \geq f(T'')]$.

In this definition, *PT* denotes the set of all of the possible permutations of a given test suite *T*, and *f* denotes a function that yields an award value of the ordering for a certain objective. Prioritization techniques are used to determine the permutation of a test suite that satisfies a certain goal. Mainly, the goal of prioritization is the increase of the fault detection rate, thereby allowing one to detect faults more quickly while conducting a set of tests. Minimization and selection techniques are other optimization techniques that can be used for the regression tests; however, these exclude a subset of the test cases to reduce the size of the test suite. For those techniques, the ability that reveals some of the faults can consequently be lost. Alternatively, the prioritization techniques reorder and run all of the test cases to avoid this drawback; furthermore, by providing earlier feedback, the prioritization techniques can provide more time for testers and developers to cope with any of the faults that are detected by the regression test.

2.2 Historical Data-based Prioritization

The test data for the historical data-based prioritization include the bug report, requirement, failure history, code coverage, and test execution time. A number of studies have proposed prioritization techniques using the above test data. Di Nardo et al. [10] introduced a coverage-based test case prioritization that is applicable to real-world industrial systems for which the structural coverage information from the source code is used. K. Saha et al. [11] proposed the REPiR, which prioritizes the test cases using an IR (information retrieval) technique with the bug reports that are collected during the maintenance phase. Srikanth et al. [12] proposed a PORT that prioritizes the test cases by mapping them with the requirements. Cho et al. [13] proposed the AFSAC that improves the efficiency of the regression testing in continuous integration environments through the use of the failure history that is collected in the source code repository. Marijan et al. [14] proposed the ROCKET that prioritizes the test cases for continuous regression testing through the use of the failure history and the execution time of the test case. Basically, the main idea behind the equations and algorithms of those approaches is that the fault detection capabilities or error-proneness of the test cases can be found in certain forms of the historical data. Most of the studies that are based on the historical data conducted experiments and industrial case studies to demonstrate the improvement of the efficiency.

2.3 Continuous Integration Environments

The continuous integration environment indicates a connected environment in which a number of developers develop, integrate, and test through a shared repository. In these environments, the goal of the regression test is the prevention of the problems in the program integration. Effective testing is therefore essential because it is mandatory to check every integration problem each time with auto build, and to identify the problems as soon as possible. In some critical situations, the source code repository must be frozen and all of the integration processes must be stopped; this is the reason why it is important to conduct regression tests after every change. In the continuous integration environments, if faults are found by the regression test, their cause must be investigated. If the faults are severe, they are fixed immediately. For the regression tests in the continuous integration environments, many researchers try to solve the above problem by suggesting a test case prioritization technique.

In this paper, the focus is the rapid acquisition of the failure information. The explicit aim of this approach is the execution of the test cases in the sequential order of the failure probabilities; this should improve the fault detection rate and increase the amount of failure information that is used to manage the detected faults.

2.4 Correlation of Test Cases

Recently, the size and complexity of software has increased exponentially. Changes can therefore cause functional or non-functional regressions at points that are not expected by the data flow or the control flow, and this is especially the case in systems that contain numerous parallel processes, where it can be difficult to predict the result of one small incorrect change; in this case, the developer or tester generally predicts the outcome of the test case based on his or her heuristic experience and background.

In this paper, the correlation data of the test cases that are derived from the test history are used. Carlson et al. [15] proposed an approach for which the similarity information that is based on the code coverage is utilized as the correlation data; this is a clustering approach that helps in the improvement of the efficiency of the test case prioritization. Test cases that have a low code-coverage similarity with the executed test cases are prioritized to detect the faults earlier. The authors use a different prioritization technique with the correlation data to effectively obtain the failure information. In section 3.2, the authors explain the way that the correlation data are acquired and used.

3. APPROACH

This paper proposes an FHD-Prioritization method to improve the efficiency of the regression test in the continuous integration environments. The criteria for the prioritization are the failed test case history, the flipped result of method test case and the correlation data. If the software is developed in compliance with the general software development process, all of this information can be obtained easily. In other words, unlike approaches based on bug report, requirement, history, code coverage, and test execution time, FHD-Prioritization does not utilize the additionally acquired information.

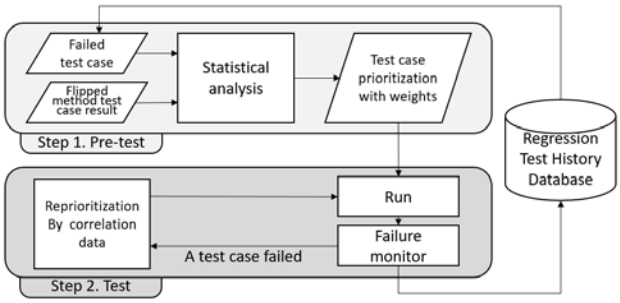

Figure 1: Overview of FHD-Prioritization

FHD-Prioritization are divided into pre-test and test phases depending on the analysis step, as shown in Figure 1.

Step 1. The pre-test statistically analyzes the failed test case history, the flipped result of method test case and then assigns the weight of the test case. Based on the results of this analysis, the execution order of the test cases is determined in an ascending order.

Step 2. The test is performed in the order of the test case priority that is determined in Step 1, and a monitoring of whether the failure occurs or not is conducted. If a failure occurs, a reprioritization is performed based on the correlation data. The observed-failure information is stored in the Regression Test History Database so that it can be used as the input data for the application of the prioritization technique in the next test session.

3.1 Prioritization based on the Statistical Analysis of Failure History Data (Step 1)

The ROCKET motivated the FHD-Prioritization study, as it utilizes the failure history and the execution time as historical data. Like the ROCKET, it is important to perform as many failure test case candidates as possible within the time limit, considering the execution time. It is more important, however, to accurately predict the test case in which the failure occurs, and to reduce the unnecessary test load; therefore, the FHD-Prioritization improves the analysis accuracy and efficiency by using the flipped result of method test case instead of the execution time as additional historical data.

This paper classifies test cases into general test cases and method test cases to help explain this technique. A test case is defined as a set of method test cases that can test different methods in the same source file. In other words, one test case consists of several method test cases. Therefore, the result of all method test cases must be pass in order to determine the result of one test case as pass. Otherwise, it is fail.

In addition, the flipped result of method test case is defined as follows. The flipped result of method test case consists of F to F, P to F, F to P and P to P. For example, F to F means that the method test case fails in both τ-2 and τ-1 sessions (τ = current session). FHD-Prioritization utilizes the number of flipped result of method test case as information which assign weight of test case. F to F indicates the number of method test cases that continuously fail in τ-2 and τ-1 sessions. If there are a large number of method test cases, the probability of failure of the test case in the next session is relatively high. P to F indicates the number of method test cases that passed in the τ-2 session but failed in the τ-1 session. If there are a large number of method test cases, the probability that the test cases fail in the next session is not relatively low. This is because the test will fail again if there are any unresolved parts in current commit among the failed method test cases in the previous session. F to P indicates the number of method test cases that failed in the τ-2 session but passed in the τ-1 session. If there are a large number of method test cases, the probability that the test cases fail in the next session is not relatively high. The large number of F to P means that the developer solved many problems through bug fixes in the previous commit. Therefore, if the developer does not create a new bug in the current commit, the probability to fail again in the method test cases becomes lower. P to P indicates the number of method test cases that continuously pass in τ-2 and τ-1 sessions. If there are a large number of the method test case, the probability of pass of the test case is relatively high in the next session. However, the number of pass test cases is generally much higher than the number of failed test cases. In other words, because the number of P to P is too many, the performance of this technique deteriorates. Therefore, this factor is not considered in this paper.

$$FHD_{v_i} = \sum_{i=1}^{n} T_{r_i} \times \omega + \sum_{j=n+1}^{m-1} T_{r_j} \times W_{m-j} + \left(\sum_{k=1}^{l} (MT_{r_k} \times \sigma) \right) / l \qquad (1)$$

(1)-a (1)-b (1)-c

Formula (1) FHD_{v_i} represents a formula for the calculation of the weight of individual test suites, and Figures 2 and 3 show examples.

Formula (1)-a is a formula that determines how far the past data (range) will be reflected from the current test session and how much data (degree) will be reflected. Compared with the ROCKET, the weight of the previous versions is lower, and the weight of the recent versions is higher because the recent test results are more important in this technique. The ROCKET assigns the weights of 0.7, 0.2, and 0.1 according to the test results of the previous session, and the FHD-Prioritization assigns 0.5, 0.3, 0.1, and 0.001 for each of the previous sessions. The FHD-Prioritization multiplies the ω by calculating the number of the failure occurrences 1 to 18 ($T_{r_i} = 11$) in test suite 1 via Formula (3).

[Formula (1)-a = - 11 × 0.001 = - 0.011]

$$W_k = \begin{cases} \alpha = 0.5, & k < Fr_{min} \\ \beta = 0.3, & Fr_{min} \le k < Fr_{avg} \\ \gamma = 0.1, & Fr_{avg} \le k < Fr_{max} \\ \delta = 0.001, & Fr_{max} \le k \end{cases} \qquad (2)$$

$$\omega = 0.001$$

$$T_{r_i} = \begin{cases} -1, & Test\ case\ i\ failed \\ 0, & Otherwise \end{cases} \qquad (3)$$

$$MT_{r_i} = \begin{cases} -1, & Method\ test\ case\ i\ failed \\ 1, & Otherwise \end{cases} \qquad (4)$$

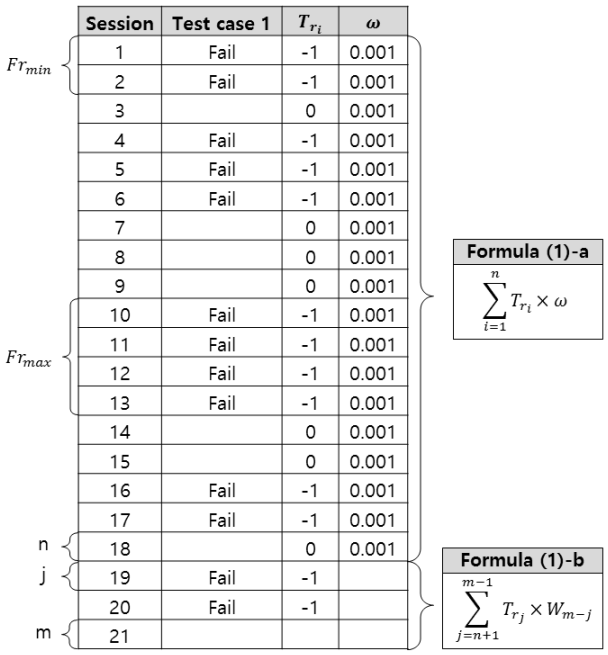

Figure 2: Example of weight calculation with failed test case history

Formula (1)-b is a formula that assigns the weight differently in consideration of the number of consecutive failure sessions in the historical data. The number of failures ($T_{r_j} = 2$) from the last passed session (session 18) to the current session in test suite 1 is calculated via Formula (3). This number is substituted into Formula (2) to calculate W_k ($W_k = \beta = 0.3$) . ($Fr_{min} = 2$, $Fr_{max} = 4$, Fr_{avg}= 3((2 + 4)/2))

[Formula (1)-b = - 2 × 0.3 = - 0.6]

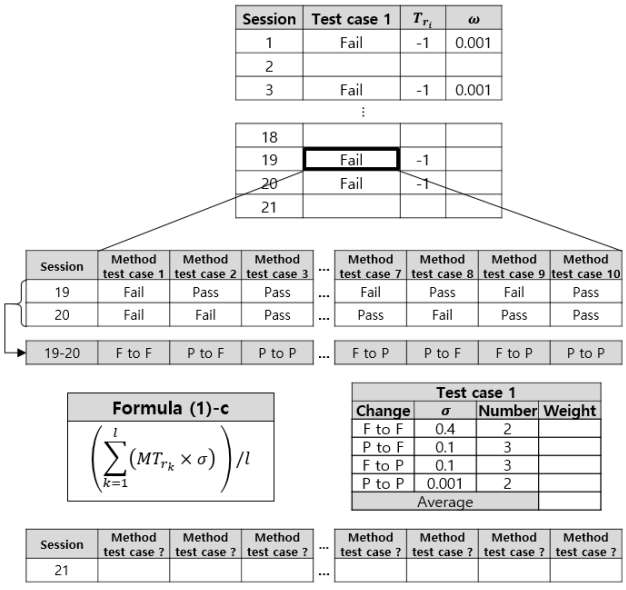

Figure 3: Example of weight calculation with flipped result of method test case

Formula (1)-c is a formula to calculate the weight to be reflected in the current session by analyzing the change of the test result of the previous one-step and two-steps session. In Figure 3, the numbers of F to F, P to F, F to P and P to P are 2, 3, 3 and 2 respectively in the session 19 and 20. The number of flipped results is multiplied by the weight (σ) of the method test case. Finally, the weight of the total test case is calculated using the cumulative average of all method test cases.

[((-2×0.4)+(-3×0.1)+(3×0.1)+ (2×0.001)) /10= -0.0798]

Formula (1)-a + (1)-b + (1)-c = -0.011 + -0.6 + -0.0798= -0.6908, so the total weight value of test suite 1 becomes -0.6908. In the same way, it calculates the weight values of all of the test suites and tests them in an ascending order.

3.2 Prioritization based on the Analysis of the Correlation Data (Step 2)

If a failure occurs during the regression test according to the priority determined in Step 1, it is immediately reprioritized using the correlation data. The fact that the results of a test case in one commit are flipped identically means that the test cases are related to each other; therefore, the correlation data are used as the criteria for the reprioritization.

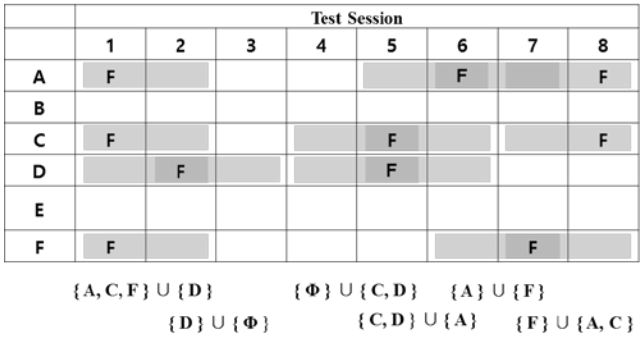

Figure 4: An example of the correlation data

Figure 4 depicts the definition of the correlations among the test cases in this manner. In test sessions 1 and 2, the test cases A, C, F, and D are correlated to each other; in the same manner, based on the results of test sessions 5 and 6, the test cases C, D, and A are correlated. After the termination of every regression test, a correlation matrix is formed so that the correlation data can be used in the next test session. The correlation matrix is composed of the values that reflect the accumulated number of the flipped results.

Algorithm 1. Prioritization based on the correlation data

Parameters:
Test suite *TS*, includes *n* number of test cases *TC*,
Weight value W_c,
Correlation matrix *CM*, includes frequency value, $Cf_{i,j}$

1: **for all** $TC_i \in TS$ $(1 \le i \le n)$ **do**
2: Run TC_i (*TC* in position *i*)
3: **if** TC_i failed **then**
4: store TC_i to buffer B_i
5: **for all** *j* $(i < j \le n)$ **do**
6: **if** TC_j is correlated with TC_i **then**
7: $Pv_j \leftarrow Pv_j + (W_c$ * negative of $Cf_{i,j})$
8: **end if**
9: **end for**
10: Sort remaining TC_j $(i < j \le n)$ by priority
11: **end if**
12: **end for**
13: update CM with data of B_i and B_{i-1}

Algorithm 1 describes the way that the correlation data are used and obtained for the reordering of the remaining test cases during the regression test. When a test case fails, the priorities of the remaining test cases are changed by the correlation data and are reordered. After the termination of the test, the correlation matrix is updated with the data that are related to the flipped or failed test cases.

4. EXPERIMENTATION

To evaluate the efficiency of our approach, we designed a prioritizing system and conducted experiments. In this section, we describe the details of our experiments and present the experimental results. We also discuss the efficiency of FHD-Prioritization by answering the following research questions:

1) RQ1: Can FHD-Prioritization improve the efficiency of regression tests? Is FHD-Prioritization more effective than other techniques?
2) RQ2: Can the weighting method using the statistical analysis of FHD-Prioritization improve the efficiency?

4.1 Subject Programs

We use Apache open source software projects in our experiments such as Tomcat and Camel [16]. The details of these subjects are presented in Table 1. The subjects are licensed under Apache license v2.0. These projects have continuously been evolved by numerous developers. Hence, it is appropriate to conduct experiments on these projects to assess our approach in continuous integration environments. Except for industrial case studies, many researchers have used software projects with manually seeded faults as the subjects of their experiments. Alternatively, we use real faults in real-world projects for our evaluation. Using these projects helps practical evaluation of the efficiency of our approach.

Table 1: Subject projects

Subject	LoC	Size of test suite	Faulty versions / Total versions
Tomcat	205,176	624	1596 / 2334
Camel	562,089	3970	2896 / 3608

4.2 Experiment Environment

We ran the tests of both projects, constructed on the JUnit test framework for regression testing. We then conducted tests and collected the results of every test session. These results were used to prioritize test cases for the next test session. In tests of Tomcat, there are three different Tomcat connectors. We omitted APR connector because it needs an external library binary for testing.

Table 2: Prioritization techniques for experiments

Technique	Description	Abbr.
Untreated	No prioritization	Unt
Random	Random execution order	Rnd
Failure window-n	Adopt the prioritization technique using the failure window in the post-submit phase [17]. If the test case failed in the previous n-testing session, it has priority but is not weighted.	FW-n
ROCKET	ROCKET algorithm without considering the execution time.	RKT
AFSAC	Prioritize during two stages by statistical analysis and correlation data	AFSAC
FHD	The proposed technique in this paper	FHD

Table 2 shows the comparison prioritization technique and brief description. The names of these techniques are abbreviated as in column 3.

4.3 Evaluation Metrics

Many researchers use APFD (Average Percentage of Fault Detection) to evaluate the fault detection rate. The values of APFD range from 0 to 1. A higher value implies that the first test cases that detect each fault are executed earlier. Formula (5) shows how to calculate APFD [18].

$$APFD = 1 - \frac{TF_1 + TF_2 + TF_3 + \cdots + TF_m}{nm} + \frac{1}{2n} \qquad (5)$$

The test suite has n test cases and detects m number of faults. TF_i is ranking of the first test case that find out fault i. We selected the APFD metric to measure the efficiency of our approach.

4.4 Results and Analysis

Our results are shown in the forms of a boxplot and average APFD values. We also discuss the results by answering the two research questions mentioned in section 4.

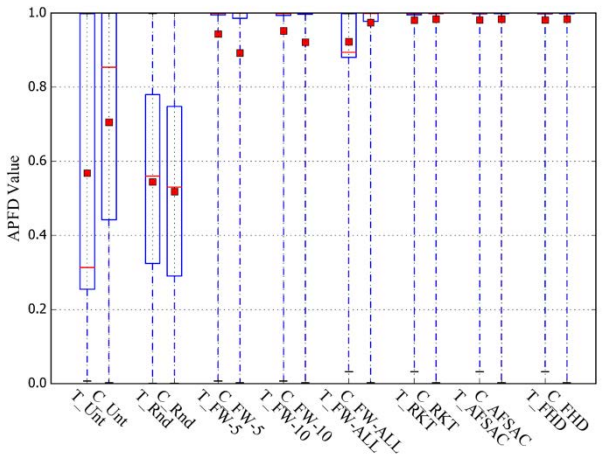

Figure 5: APFD boxplots of the prioritizing techniques used for Tomcat (T) and Camel (C)

1) RQ1: Efficiency of FHD-Prioritization and Comparing the results of the different techniques:

In Figure 5, the X-axis indicates test case prioritization techniques and Y-axis indicates the APFD value. In addition, T and C as the first letter of the name of these techniques indicate the result value of Tomcat and Camel project, respectively. As a result of applying two subject projects, the average APFD values of Untreated, Random, Failure window-5, Failure window-10, Failure window-All, ROCKET, AFSAC and FHD are 0.636601957, 0.531336047, 0.917785947, 0.936877442, 0.948315115, 0.981984187, 0.982497057 and 0.982524789, respectively. Although the APFD absolute value is not significantly different, the APFD value of the FHD technique is the highest among all the techniques. Therefore, FHD is more effective than other techniques.

The boxplot in Figure 5 indicates that the APFD values of Untreated, Random, and Failure window-5, 10, and All do not perform significantly better than other techniques. Therefore, Figure 6 compares only the ROCKET, AFSAC, and FHD techniques in detail through the pair-wise comparison analysis.

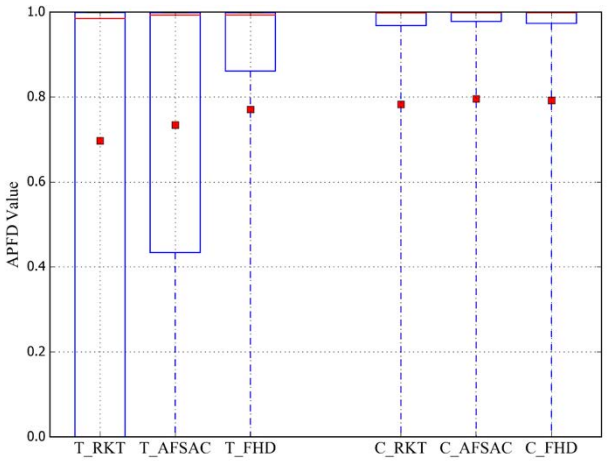

Figure 6: Pair-wise comparison of ROCKET, AFSAC and FHD

2) RQ2: Efficiency of the weighting method using statistical analysis:

In Figure 6, the X-axis indicates test case prioritization techniques and Y-axis indicates the APFD value. In addition, T and C as the first letter of the name of these techniques indicate the result value of Tomcat and Camel project, respectively. As a result of applying two subject projects, the average APFD values of ROCKET, AFSAC and FHD are 0.739922101, 0.764890171 and 0.781485618, respectively. The proposed FHD technique improved the efficiency of test case prioritization by about 5.62% and 2.17%, respectively, compared to ROCKET and AFSAC. Thus, the weighting method using the statistical analysis is effective.

4.5 Threats to Validity

We initially expected to conduct experiments with a greater number of projects in order to be convinced that our approach is effective for prioritizing test cases. However, we used two open source projects as benchmarks to evaluate the approach. Although the number of target projects is small, it is a large and famous open source project. We are also concerned by the number of approaches that we compared with our approach. Many existing studies for test case prioritization in the literature considered the coverage or source code and aimed to increase the fault detection rate. Those are not suitable for comparison with our approach in terms of the criteria and aims. In the future, we will perform experiments with more projects and comparing other approaches in order to further verify our results.

Our approach is based on historical data and focuses on continuous integration environments. Therefore, the efficiency is dependent on the amount of data and the conditions of the subject projects. Thus, our experiments used historical data that was sufficiently long-term in order to provide reliability in our experimental results.

5. CONCLUSION

This paper proposed a test case prioritization method that is based on the historical data. The proposed method performs the analysis in two steps and prioritizes the test cases based on the results. Step 1. A statistical analysis of the historical data from previous versions calculates the weight of the individual test cases and sorts them in ascending order based on the results. Step 2. If a failure occurs during the regression test, a reprioritization is performed based on the correlation data. For the next session test, the failure information of the current session is updated in the correlation data. This paper evaluated the efficiency of the application of the FHD-Prioritization to two open source projects. The subject projects are large-scale and real-world software programs without manually seeded faults. As a result of applying two subject projects, the average APFD values of ROCKET, AFSAC and FHD are 0.74, 0.76 and 0.78, respectively. The FHD-Prioritization improved the efficiency of test case prioritization by about 5.62% and 2.17%, respectively, compared to ROCKET and AFSAC.

6. ACKNOWLEDGMENTS

This research was supported by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea(NRF)(No.2012M3C4A7033347) funded by the Ministry of Education, Science and Technology and Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Education(No.2016R1D1A1B03934610) and MISP(Ministry of Science, ICT & Future Planning), Korea, under National program for Excellence in Software(R2215-16-1005) supervised by the IITP(Institute for information & communications Technology Promotion).

REFERENCES

[1] Wong, W. E., Horgan, J. R., London, S., and Agrawal, H. A study of effective regression testing in practice. *In Proceedings of International Symposium Software Reliability Engineering*, 1997, 264-274.

[2] Yoo, S., and Harman, M. Regression testing minimization, selection and prioritization: a survey. *Software Testing, Verification and Reliability*, 2012, 22(2), 67-120.

[3] Wong, W. E., Horgan, J. R., Mathur, A. P., and Pasquini, A. Test set size minimization and fault detection effectiveness: A case study in a space application. *In Proceedings of Computer Software and Applications Conference*, 1997, 522-528.

[4] Rothermel, G. and Harrold, M. J. A safe, efficient regression test selection technique. *IEEE Transactions on Software Engineering and Methodology*, 1997, 6(2), 173-210.

[5] Rothermel, G., Untch, R. H., Chu, C. and Harrold, M. J., Prioritizing test cases for regression testing. *IEEE Transactions on software engineering*, 2001, 2, 7(10), 929-948.

[6] Kim, J. M., and Porter, A. A history-based test prioritization technique for regression testing in resource constrained environments. *In Proceedings of International Conference on Software Engineering*, 2002, 119-129.

[7] Aggrawal, K. K., Singh, Y., and Kaur, A. Code coverage based technique for prioritizing test cases for regression testing. *ACM SIGSOFT Software Engineering Notes*, 2004, 29(5), 1-4.

[8] Korel, B., Tahat, L. H., and Harman, M. Test prioritization using system models. *In Proceedings of International Conference on Software Maintenance*, 2005, 559-568.

[9] Arafeen, M. J., and Do, H. Test case prioritization using requirements-based clustering. *In Proceedings of International Conference on Software Testing, Verification and Validation*, 2013, 312-321.

[10] Di Nardo, D., Alshahwan, N., Briand, L., and Labiche, Y. Coverage-based test case prioritisation: An industrial case study. *In Proceedings of International Conference on Software Testing, Verification and Validation*, 2013, 302-311.

[11] Saha, R. K., Zhang, L., Khurshid, S., and Perry, D. E. An information retrieval approach for regression test prioritization based on program changes. *In Proceedings of International Conference on Software Engineering*-Volume 1, 2015, 268-279.

[12] Srikanth, H., and Williams, L. On the economics of requirements-based test case prioritization. *In ACM SIGSOFT Software Engineering Notes*, 2005, Vol. 30, No. 4, 1-3.

[13] Cho, Y., Kim, J., and Lee, E. History-based Test Case Prioritization for Failure Information. I*n Proceedings of Asia-Pacific Software Engineering Conference*. 2016 (Poster session in press)

[14] Marijan, D., Gotlieb, A., and Sen, S. Test case prioritization for continuous regression testing: An industrial case study. *In Proceedings of International Conference on Software Maintenance*, 2013, 540-543.

[15] Carlson, R., Do, H., and Denton, A. A clustering approach to improving test case prioritization: An industrial case study. *In Proceedings of International Conference on Software Maintenance*, 2011, 382-391.

[16] Apache Software Foundation Projects Directory, URL: https://projects.apache.org

[17] Elbaum, S., Rothermel, G., and Penix, J. Techniques for improving regression testing in continuous integration development environments. *In Proceedings of the International Symposium on Foundations of Software Engineering*, 2014, 235-245.

[18] Elbaum, S., Malishevsky, A. G., and Rothermel, G. Test case prioritization: A family of empirical studies. *IEEE transactions on software engineering*, 2002, 28(2), 159-182.