Research Paper

[Saini, 1(3): May, 2012]
ISSN: 2277-9655

IJESRT

INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

Comparative Study of Test Prioritization Testing Technique

Rimmi Saini[1], Deepa Gupta[2], Ajay Rana[3]

[1]Department of Computer Science Engg. Greater Noida Group of Institutions, Greater Noida, India
[2]Department of Computer Science Engg., Amity Institute of Information Technology, Noida, India
[3]Department of Computer Science Engg., Amity School of Engineering and Technology, Noida, India

rimmi_saini@yahoo.com

Abstract

Testing of software is an important means of assessing the software to determine its quality. Testing activities support quality assurance by gathering information about the nature of the software being studied. Software testing has been proven that testing, analysis, and debugging costs usually consume over 50% of the costs associated with the development of large software systems. These activities consist of designing test cases, executing the software with those test cases, and examining the results produced by those executions. Studies shows that testing consumes 40~50% of development efforts, and consumes more effort for systems that require higher levels of reliability. So it is a significant part of the software engineering. Various existing test prioritization techniques will be studied. After studying these techniques a comparison will be made which shows the various test prioritization techniques and their benefits and non-benefits in reducing the cost of the testing. This study will concentrates on a survey of test case prioritization techniques. Earlier some studies have classified and organized existing test case prioritization techniques researched into four categories: (a) customer requirement-based techniques (b) coverage-based techniques (c) cost effective-based techniques and (d) chronographic history-based techniques.

***Keywords*— software testing, test case, test prioritization, test prioritization testing technique.**

Introduction

Software Testing is an important process that is performed to support quality assurance or we can say testing of software is an important means of assessing the software to determine its quality. Testing activities support quality assurance by gathering information about the nature of the software being studied. These activities consist of designing test cases, executing the software with those test cases, and examining the results produced by those executions. Studies shows that testing consumes 40~50% of development efforts, and consumes more effort for systems that require higher levels of reliability. So it is a significant part of the software engineering.

With the development of Fourth generation languages (4GL), which speeds up the implementation process, the proportion of time devoted to testing is increased. As the amount of maintenance and upgrade of existing systems grow, significant amount of testing will also be needed to verify systems after changes are made [1].

Software testing is a very broad area, which involves many other technical and non-technical areas, such as specification, design and implementation, maintenance, process and management issues in software engineering. Software Testing is an important process that is performed to support quality assurance or we can say testing of software is an important means of assessing the software to determine its quality. Testing activities support quality assurance by gathering information about the nature of the software being studied. These activities consist of designing test cases, executing the software with those test cases, and examining the results produced by those executions. Studies shows that testing consumes 40~50% of development efforts, and consumes more effort for systems that require higher levels of reliability. So it is a significant part of the software engineering. With the development of Fourth generation languages (4GL),

which speeds up the implementation process, the proportion of time devoted to testing is increased. As the amount of maintenance and upgrade of existing systems grow, significant amount of testing will also be needed to verify systems after changes are made [1].

Software testing is a very broad area, which involves many other technical and non-technical areas, such as specification, design and implementation, maintenance, process and management issues in software engineering.

Literature survey

Software testing has been proven that testing, analysis, and debugging costs usually consume over 50% of the costs associated with the development of large software systems. Many researchers have found several approaches to schedule an order of test execution. Sometimes these techniques are insufficient to prioritize tests in large commercial systems. They incorrectly schedule tests and the cost is overrun during the prioritization process. Autonomous System Research Laboratory, Science and Technology, [4] reported that they proposes two new efficient prioritization methods to address the above issues. The first method aims to resolve the problem of many test cases assigned the same weight values. The second method is developed to effectively prioritize multiple suites [4].So in future existing test prioritization techniques given by various testing gurus will be compared and then analysis will be given.

Software maintenance is an important and costly activity of the software development lifecycle. To ensure proper maintenance the software undergoes regression testing. It is very inefficient to re execute every test case in regression testing for small changes. Hence test case prioritization is a technique to schedule the test case in an order that maximizes some objective function. A variety of objective functions are applicable, one such function involves rate of fault detection – a measure of how quickly faults are detected within the testing process [6]. Arup Abhinna Acharya et. al., in their paper propose a method to prioritize the test cases for testing component dependency in a Component Based Software Development (CBSD) environment using Greedy Approach [6].

According to Arup Abhinna Acharya et. al., the cost and time required for regression testing can be minimized by using the prioritization technique discussed in their paper. Here they have proposed a model based prioritization method by considering the number of Object Interactions per unit time as the objective function. Here more importance is given to number of inter component object interactions present because maximum faults are expected to be present when components interact with each other [6].

James A Jones et al. [7] have presented an algorithm for test suite prioritization that incorporates the complexities of modified condition or decision coverage. Boris [8], claimed that software testing should take around 40-70% of the time and cost of the software development process. Many approaches have been proposed to reduce time and cost during software testing process, including test case Prioritization techniques and test case reduction techniques.

Numerous test prioritization testing techniques have been described in the research literature (Elbaum et al., 2001b, 2002[9][10]; Jones and Harrold, 2001; Rothermel et al., 2001[11][12]; Wong et al., 1997) [13].

In recent years, several researchers have addressed the test case prioritization problem and presented Techniques for the same. Research has shown that at least 50% of the total cost of software development consists of testing activities [14]. It has been tested experimentally in [15] that to optimize the time and cost spent on testing, prioritization of the test cases in a test suite can be beneficial. Code coverage based TCP techniques proposed in [16], involve ranking test cases based on the code coverage. For prioritized statement coverage, test cases are ranked based on the number of statements covered by the test case.

Rothermel et al. (Rothermel et al., 2001) done the test case prioritization problem and describe several issues relevant to its solution. Their paper reviews the portions of that material that are necessary to understand this article [17]. In Rothermel et al., paper the Test Case Prioritization Problem is stated as follows:

Given: *T, a test suite; PT, the set of permutations of T; and f, a function from PT to the real numbers.*

Problem: Find T0 2 PT such that (8T00) (T00 2 PT) (T00 6= T0) [f(T0) ¸ f(T00)].

Here, PT represents the set of all possible prioritizations (orderings) of T, and f is a function that, applied to any such ordering, yields an award value for that ordering. There are many possible goals for prioritization [17]. According to Erlbaum et al., Test case prioritization techniques help engineers execute regression tests in an order that achieves testing objectives earlier in the testing process. In their paper they discussed that one testing objective involves *rate of fault detection* {a measure of how quickly a test order detects faults}. An improved rate of fault detection can provide earlier feedback on the system under test, enable earlier debugging, and increase the likelihood that, if testing is prematurely halted, those test cases that offer the greatest fault detection ability in the available testing time will have been executed [18].

In their paper Elbaum et al., says that these early indications of potential are encouraging; however, studies have also shown that the rates of fault detection produced by prioritization techniques can vary significantly with several factors related to program attributes, change attributes, and test suite characteristics (Erlbaum et al., 2001a, 2003) [18][19]. As described by Elbaum et al.,[22][23]; Rothermel et al., [20][21]in their literature and investigation, for the target prioritization techniques they chosen four heuristics. In the literature and investigated in empirical studies, that could easily be (or have already been) implemented by practitioners, and that allow them to examine two of the key dimensions of differences among techniques: the uses of feedback and information on modifications. (For simplicity and to facilitate comparison, Elbaum et al., Rothermel et al., restricted their attention to function-coverage-based techniques). The four techniques were:

Total function coverage prioritization (total),
Additional function coverage prioritization (addtl),
Total binary-dif function coverage prioritization,
Additional binary-di function coverage prioritization [21].

Zheng Li et al. have tested experimentally that genetic algorithms perform well for test case prioritization [24]. Kim and Porter [25] in their paper present a technique, which they refer to as a "history-based prioritization" that utilizes information from previous testing cycles to select the test cases that must be executed for a new version of the program. This technique is not, however, a "prioritization technique" in the sense done in the literature because it imposes no ordering on test cases {the characteristic essential to the definition of prioritization}. Rather, the approach selects a subset of a test suite, using history information to determine which test cases should be selected, and is more accurately described as a "regression test selection technique" by Rothermel and Harrold in 1996 [26].

Gregg Rothermel [27] has proven that prioritizing and scheduling test cases are one of the most critical tasks during the software testing process. He referred to the industrial collaborators reports, which shows that there are approximately 20,000 lines of code, running the entire test cases requires seven weeks. In this situation, test engineers may want to prioritize and schedule those test cases in order that those test cases with higher priority are executed first. Additionally, he [28], [29] stated that test case prioritization methods and process are required, because: (a) the regression testing phase consumes a lot of time and cost to run, And (b) there is not enough time or resources to run the entire test suite (c) there is a need to decide which test cases to run first [30]. According to A. G. Malishevsky et al., Test case prioritization has been primarily applied to improve regression testing effects as mentioned in their paper [31].

In [32], Rothermel et al. pointed that the potential goal of prioritization is to increase a test suite's rate of fault detection earlier in the software process. It has been tested experimentally in [33] that some of the biggest causes for project failures are lack of user input and changing or incomplete requirements. Software engineers save the test cases and re-run the test cases as regression test in later versions.

Comparison

Software maintenance is an important and costly activity of the software development lifecycle. To ensure proper maintenance the software undergoes regression testing. It is very inefficient to re execute every test case in regression testing for small changes. Hence test case prioritization is a technique to schedule the test case in an order that maximizes

some objective function. Various Test Prioritization Testing Techniques are studied. After studying these techniques a comparison is made on the basis of benefits and limitations of the various Test Prioritization Testing Techniques. Every researcher has given some characteristics about their Prioritization Technique like techniques should prioritize and schedule test cases in an order that attempts to maximize some objective function, should achieve code coverage at the fastest rate possible, exercises features in order of expected frequency of use, or exercises sub systems in an order that reflects their historical propensity to fail.

But the main emphasis will be given on the techniques which can reduce the cost of testing. Because cost is the factor which affect the software development the most. If we can predict the set of test suites or test cases at the early stages of the software development, then cost of the software testing can be reduced. When the time required to execute all test cases in a test suite is short, test case prioritization may not be cost effective - it may be most expedient simply to schedule test cases in any order [28], [29]. When the time required to run all test cases in the test suite is sufficiently long, the benefits offered by test case prioritization methods become more significant.

A variety of objective functions are applicable, one such function involves rate of fault detection – a measure of how quickly faults are detected within the testing process [6]. Arup Abhinna Acharya et.al. [6] in their paper propose a method to prioritize the test cases for testing component dependency in a Component Based Software Development (CBSD) environment using Greedy Approach [6].

The cost and time required for regression testing can be minimized by using the prioritization technique discussed in this paper. Here they have proposed a model based prioritization method by considering the number of Object Interactions per unit time as the objective function. Here more importance is given to number of inter component object interactions present because maximum faults are expected to be present when components interact with each other [6].

To facilitate regression testing by optimizing the time and cost, Arup Abhinna Acharya et.al. propose a method to prioritize the test cases by using model based prioritization method by extracting the benefits of Unified Modelling Language(UML). The proposed model found to be very effective as it increases the Average Percentage of Fault Detection (APFD) when it is applied to few of the projects developed in Java by java 45%-50%. This approach is mainly applicable to test the component composition in case of component based software maintenance [6].

The Benefits of using this model is

- it is applicable to test the component composition in case of component based software maintenance

S. Elbaum et. al. have presented a study of test case prioritization techniques applied across eight systems. Their results regarding the effectiveness of the techniques confirm previous findings, among

According to S. Elbaum et.al. test case prioritization techniques help engineers execute regression tests in an order that achieves testing objectives earlier in the testing process. Here the author also involved the same testing objective which is the *rate of fault detection* {a measure of how quickly a test order detects faults}.

An improved rate of fault detection can provide earlier feedback on the system under test, enable earlier debugging, and increase the likelihood that, if testing is prematurely halted, those test cases that offer the greatest fault detection ability in the available testing time will have been executed [18]. These early indications of potential are encouraging; however, studies have also shown that the rates of fault detection produced by prioritization techniques can vary significantly with several factors related to program attributes, change attributes, and test suite characteristics [18].

To date, most proposed techniques have been code-based, relying on information relating test cases to coverage of code elements, and a first dimension along which techniques can be distinguished is in terms of the type of code elements they consider.

them the fact that the performance of test case prioritization techniques varies significantly with program attributes, change attributes, test suite characteristics, and their interaction. These results

support the search for strategies by which practitioners could choose appropriate prioritization techniques for their particular testing scenarios.

They have proposed two such strategies.

1. The basic instance-and-threshold strategy, recommends the technique that has been successful in the largest proportion of instances in the past, accounting for cost-benefit thresholds.
2. The enhanced instance-and-threshold strategy, adds into consideration the attributes of a particular testing scenario, using metrics to characterize scenarios, and employing classification trees to improve the likelihood of recommending the proper technique for each particular case.

The Benefits of using this model is that it

- results regarding the effectiveness of the techniques confirm previous findings
- results support the search for strategies by which practitioners could choose appropriate prioritization techniques

The Limitations of using this model is that

- researchers or practitioners wishing to evaluate new techniques or scenarios should begin by considering the basic instance-and-threshold strategy
- Results show that some of the expectations we might have about technique effectiveness (e.g. adding modification information will improve rate of fault detection") can be incorrect [18].

The literature review reveals that there are many outstanding research issues in the test case prioritization area, such as poor performance of prioritization algorithms, non-practical weight factors and noncommercial prioritization methods [4].

Paper [4] highlights two critical outstanding research issues, which are:

(a) Existing prioritization methods ignores prioritizing multiple test suites and

(b) Existing techniques randomly prioritize all test cases with the same weight values, without any systematic algorithm [4]. In this paper it is clear that existing prioritization methods ignores prioritizing multiple test suites. This can lead to the following difficult situations in the commercial industry. In the commercial systems, there is always more than a single test suite, during a system integration testing.

The Limitations of using this model is that

- The existing prioritize techniques are not applicable if there are many test suites.
- It may take longer time to individually prioritize each test suite with existing methods.
- As a result, it may rapidly increase an amount of time and cost to prioritize test suites [4].

The second issue in this paper [4] is that the existing techniques randomly prioritize all test cases with the same weight values, without any systematic algorithm. This issue may cause many limitations.

The Limitations of using this model are

- A poor performance problem while prioritization process.
- There is a high probability that a significant number of test cases can be assigned to the same weight values.
- With those test cases, the existing methods randomly prioritize without any systematic algorithm and weight factor.
- The prioritized test cases may not be accurate with the right weight values [4].

Conclusion

Software testing is an expensive and time consuming activity that is often restricted by limited project budgets. Accordingly, the National Institute for Standards and Technology (NIST) reports that software defects cost the U.S. economy close to $60 billion a year [34]. They suggest that approximately $22 billion can be saved through more effective testing. There is a need for advanced software testing techniques that offer a solid cost-benefit ratio in identifying defects. Previous empirical studies have shown that several prioritization techniques can significantly improve rate of fault detection, but these studies have also shown that the effectiveness of these techniques varies considerably across various attributes of the program, test suites, and modifications being considered.

In present work we have performed a detailed study of various Test Prioritization Testing Techniques. After studying these Testing Techniques, we compared benefits and limitations of different test prioritization testing techniques. Our main aim was to

predict the cost of testing, because cost is the factor which affects the software development the most. There are a variety of objective functions, one such function involves rate of fault detection a measure of how quickly faults are detected within the testing process. The results of the study suggest if the fault is detected quickly at the earlier stages of development then the cost of software testing will be less.

References

[1] J. J. Marciniak, "Encyclopedia of software engineering", Volume 2, New York, NY: Wiley, 1994, pp. 1327-1358, [Marciniak94].

[2] E. F. Miller, "Introduction to Software Testing Technology," *Tutorial: Software Testing & Validation Techniques*, Second Edition, IEEE Catalog No. EHO 180-0, pp. 4-16
[Miller81]

[3] [Myers79] Myers, Glenford J., *The art of software testing*, Publication info: New York : Wiley, c1979. ISBN: 0471043281 Physical description: xi, 177 p. : ill. ; 24 cm.

[4] Journal of Theoretical and Applied Information Technology, © 2005 - 2010 JATIT & LLS. All rights reserved, www.jatit.org,

[5] TEST CASE PRIORITIZATION TECHNIQUES, SIRIPONG ROONGRUANGSUWAN, 2JIRAPUN DAENGDEJ, Autonomous System Research Laboratory, Science and Technology, Assumption University, Thailand.

[6] Arup Abhinna Acharya, Durga Prasad Mohapatra, And Namita Panda," Model Based Test Case Prioritization for Testing Component Dependency in CBSD Using UMLSequence Diagram", *(IJACSA) International Journal of Advanced Computer Science and Applications,Vol. 1, No. 6, December 2010.*

[7] H. Leung and L. White. Insights into regression testing. In Proceedings of the *International Conference on Software Maintenance*, Miami, Florida, U.S.A., pages 60-69, Oct 1989.

[8] Jefferson Offutt, Jie Pan and Jeffery M. Voas, "Procedures for Reducing the Size of Coveragebased Test Sets", 1995.

[9] S. Elbaum, A. Malishevsky, and G. Rothermel. Incorporating varying test costs and fault severities into test case prioritization. In *International Conference on Software Engineering*, pages 329{338, May 2001b.

[10] S. Elbaum, A. Malishevsky, and G. Rothermel. Test case prioritization: A family of empirical studies. *IEEE Transactions of Software Engineering*, 28(2):159{182, February 2002.

[11] J. Jones and M. Harrold. Test-suite reduction and prioritization for modi¯ed condition/ In *Proceedings of the International Conference on Software Maintenance*, Nov. 2001.

[12] G. Rothermel, R. Untch, C. Chu, and M. Harrold. Test case prioritization. *IEEE Transactions on Software Engineering*, 27(10):929{948, October 2001.

[13] W.Wong, J. Horgan, S. London, and H. Agrawal. A study of e®ective regression in practice. In *Proceedings of the Eighth International Symposium on Software Reliability Engineering*, pages 230{238, November 1997.

[14] M. J. Harold. Testing: A Roadmap, *In Proceedings of the International Conference on Software Engineering, Limerick, Ireland*, pages 61-72, 2000.

[15] D. Jeffrey and N. Gupta. Test Case Prioritization Using Relevant Slices. In Proceedings of the *30th Annual International Computer Software and Applications Conference*, pages 411-420, 2006.

[16] J. M. Kim and A. Porter. History-Based Test Prioritization Technique for Regression Testing in Resource Constrained Environments. In Proceedings of the 24th *International Conference on Software Engineering*, pages 119-129, May 2002.

[17] G. Rothermel, R. Untch, C. Chu, and M. Harrold. Test case prioritization. *IEEE Transactions on Software Engineering*, 27(10):929{948, October 2001.

[18] S. Elbaum, D. Gable, and G. Rothermel. Understanding and measuring the sources of variation in the priori- tization of regression test suites. In *Proceedings of the Seventh International Software Metrics Symposium*. Institute of Electrical and Electronics Engineers, Inc., April 2001a.

[19] S. Elbaum, K. Kallakuri, A. G. Malishevsky, G. Rothermel, and S. Kanduri. Understanding the e®ects of changes on the cost-e®ectiveness of regression testing techniques. *Journal of Software Testing, Veri¯cation, and Reliability*, 13(2):65{83, June 2003.

[20] G. Rothermel, R. Untch, C. Chu, and M. Harrold. Test case prioritization: An empirical study. In *Proceedings of the International Conference on Software Maintenance*, pages 179{188, 1999.

[21] G. Rothermel, R. Untch, C. Chu, and M. Harrold. Test case prioritization. *IEEE Transactions on Software Engineering*, 27(10):929{948, October 2001.

[22] S. Elbaum, A. Malishevsky, and G. Rothermel. Incorporating varying test costs and fault severities into test case prioritization. In *International Conference on Software Engineering*, pages 329{338, May 2001b.

[23] S. Elbaum, A. Malishevsky, and G. Rothermel. Test case prioritization: A family of empirical studies. *IEEE Transactions of Software Engineering*, 28(2):159{182, February 2002.

[24] G.Rothermel, R. H. Untch, C. Chu and M. J. Harrold. *Prioritizing test cases for regression testing. IEEE Transactions of Software Engineering*, 27(10): 929–948, 2001.

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

[26]G. Rothermel and M. Harrold. Analyzing regression test selection techniques. *IEEE Transactions on Software Engineering*, 22(8):529{551, Aug. 1996.

[27] David Leon and Andy Podgurski, "A Comparison of Coverage-Based and Distribution-Based Techniques for Filtering and Prioritizing Test Cases", *Proc. Int'l Symp. Software Reliability Eng.*, pp. 442-453, 2003.

[28] B. Korel and J. Laski, "Algorithmic software fault localization", Annual Hawaii International Conference on System Sciences, pages 246–252, 1991.

[29] Dennis Jeffrey and Neelam Gupta, "Test Case Prioritization Using Relevant Slices", In *Proceedings of the 30th Annual International Computer Software and Applications Conference*, Volume 01, 2006, pages 411-420, 2006.

[30] Journal of Theoretical and Applied Information Technology, © 2005 - 2010 JATIT & LLS. All rights reserved, www.jatit.org, TEST CASE PRIORITIZATION TECHNIQUES, SIRIPONG ROONGRUANGSUWAN, 2JIRAPUN DAENGDEJ, Autonomous System Research Laboratory, Science and Technology, Assumption University, Thailand.

[31] A. G. Malishevsky, J. Ruthruff, G. Rothermel, and S. Elbaum, Cost-cognizant Test Case Prioritization, *Technical Report TR-UNL-CSE-2006-0004*, Department of Computer Science and Engineering, University of Nebraska - Lincoln, March, 2006.

[32] Z.Li, M. Harman and R.M. Hierons. Search Algorithms for Regression Test Case Prioritization.*IEEE Transactions on Software Engineering*, 33(4):225-237, April 2007.

[33] S. Elbaum, G. Rothermel, S. Kanduri and A. G. Malishevsky. Selecting a Cost-Effective Test Case Prioritization Technique. *Software Quality Journal*, 12(3): 185-210, 2004

[34] National Institute of Standards and Technology. *The Economic Impacts of Inadequate Infrastructure for Software Testing.* U.S. Department of Commerce, May 2002.