International Journal of Scientific Research in Science, Engineering and Technology
Print ISSN - 2395-1990
Online ISSN : 2394-4099
Available Online at : www.ijsrset.com
doi : https://doi.org/10.32628/IJSRSET

Implementation of Test Case Optimization and Prioritization Techniques in Software Testing

Ankit Sharma[1], **Ms Shubhangi Sharma**[2]
[1]Research Scholar, Bharat Institute of Technology, Meerut, India
[2]Assistant Professor, Bharat Institute of Technology, Meerut, India

ARTICLEINFO

Article History :
Accepted: 10 Sep 2023
Published: 20 Sep 2023

Publication Issue :
Volume 10, Issue 5
September-October-2023
Page Number :
210-215

ABSTRACT

Statistics gathered in past research show that testing, analysis and debugging costs usually consume over 50% of the costs associated with the development of large software systems. Specifically, regression testing has been shown to be a critically important phase of software testing and many techniques have been proposed that reduce effort, time and cost of testing, such as test case prioritization techniques, regression selection techniques and test case reduction methods. This study concentrates on a survey of test case prioritization techniques. This study classifies and organizes existing test case prioritization techniques researched since 1998 into four categories: (a) customer requirement-based techniques (b) coverage-based techniques (c) cost effective-based techniques and (d) chronographic history-based techniques. Also, this study resolves the following research problems: (a) ignoring practical weight factors (b) inefficient test case prioritization methods and (c) ignoring the size of test cases.
Keywords: Software Testing, Customer Requirement-Based Techniques, Coverage-Based Techniques, Cost Effective-Based Techniques, Chronographic History-Based Techniques. Also, This Study Resolves The Following Research Problems, Ignoring Practical Weight Factors, Inefficient Test Case Prioritization Methods

I. INTRODUCTION

Software testing is a comprehensive set of activities conducted with the intent of finding errors in software. It is one activity in the software development process aimed at evaluating a software item, such as system, subsystem and features (e.g., functionality, performance and security) against a given set of system requirements. Also, software testing is the process of validating and verifying that a program functions properly. Many researchers have proven that software testing is one of the most critically important phases of the software development life cycle and consumes significant resources in terms of effort, time and cost.

Test case prioritization techniques prioritize and schedule test cases in an order that attempts to maximize some objective function. For example, software test engineers might wish to schedule test cases in an order that achieves code coverage at the fastest rate possible, exercises features in order of expected frequency of use, or exercises subsystems in an order that reflects their historical propensity to fail. 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 (Rothermel et al., 2002). 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.

Although, test case prioritization methods have great benefits for software test engineers, there are still outstanding major research issues that should be addressed. The examples of major research issues are: (a) existing test case prioritization methods ignore the practical weight factors in their ranking algorithm (b) existing techniques have an inefficient weight algorithm and (c) those techniques are lacking automation during the prioritization process.

Software testing has been widely used as a way to help engineers develop high-quality systems. Testing is an important process that is performed to support quality assurance by gathering information about the nature of the software being studied (Harrold, 2000). These activities consist of designing test cases, executing the software with those test cases and examining the results produced by those executions (Beizer, 1990) indicates that more than fifty percent of the cost of software development is devoted to testing with the percentage for testing critical software being even higher. As software becomes more pervasive and is used more often to perform critical tasks, the importance of its quality will remain high. Unless engineers can find efficient ways to perform effective testing, the percentage of development costs devoted to testing may increase significantly.

II. LITERATURE REVIEW

Software Testing is an empirical investigation conducted to provide stakeholders with information about the quality of the product or service under test (Kaner, 2006), with respect to the context in which it is intended to operate. Software Testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks of implementation of the software. Test techniques include the process of executing a program or application with the intent of finding software bugs. It can also be stated as the process of validating and verifying that software meets the business and technical requirements that guided its design and development, so that it works as expected. Software Testing can be implemented at any time in the development process; however, the most test effort is employed after the requirements have been defined and coding process has been completed.

The next sections present techniques to reduce effort, time and cost during the software testing phase, including test case prioritization and test case reduction techniques to help testers reduce the time and cost required for running test cases.

Software engineers generally save test suites that they develop so that they can easily reuse those suites later as the software evolves. Reusing test cases in regression testing process is pervasive in the software industry (Onoma et al., 1998) and can save as much as one-half of the cost of software maintenance (Beizer, 1990) However, executing a set of test cases in an existing test suite consume a huge amount of time.

Rothermel et al. (1999-2001a) gave an interesting example as follows: one of the industrial collaborators reports that for one of its products that contains approximately 20,000 lines of code, running the entire test suite requires seven weeks. In such cases, testers may want to order their test cases so that those test cases with the highest priority, according to some criterion, are run first. This has proven that prioritizing and scheduling test cases are one of the most important tasks during regression testing process.

Test case prioritization techniques prioritize and schedule test cases in an order that attempts to maximize some objective function. For example, software test engineers might wish to schedule test cases in an order that achieves code coverage at the fastest rate possible, exercises features in order of expected frequency of use, or exercises subsystems in an order that reflects their historical propensity to fail. 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. 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.

Test case prioritization techniques provide a way to schedule and run test cases, which have the highest priority earliest in order to provide earlier feedback to software testing engineers and earlier detect faults. This study presents numerous techniques developed, between 2002 and 2008, that can improve a test suite's rate of fault detection.

III. PROPOSED METHODS

Here, proposes a new 2R-2S-3R continuous test case prioritization process. Also, this section discusses a proposed method that resolves the above research problems. The proposed method aims to: (a) include practical weight prioritization factors (b) improve the ability to rank and schedule test cases during the prioritization process and (c) reserve a large number of test cases with high priority.

Test Case Prioritization Process

This section introduces a new 2R-2S-3R continuous process to prioritize and schedule test cases introduced by using the above literature review and previous works (Kosindrdecha and Roongruangsuwan, 2007; Roongruangsuwan and Daengdej, 2009). Also, the new process includes a re-prioritization sub-process in order to ensure that the result of prioritization is satisfied.

Figure 1 shows a proposed test case prioritization process. The proposed process is a continuous process that allows users to continuously prioritize test cases until they are satisfied with the result. However, it starts with a large number of test cases needed to be prioritized. The process begins with a requisite process and follows with a reordering process. It simply prioritizes test cases based on given prioritization technique, coverage factors, weight value and priority value.

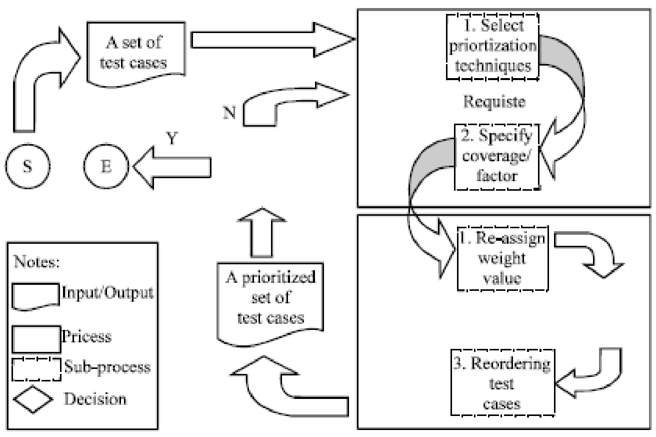
A set of test cases
1. Select priortization techniques
Requiste
2. Specify coverage/ factor
N
Y
S
E
1. Re-assign weight value
Notes:
Input/Output
Pricess
Sub-process
Decision
A prioritized set of test cases
3. Reordering test cases

Fig. 1: A 2R-2S-3R test case prioritization process

However, this study proposes to include a re-prioritize process in case that the result of prioritization is not satisfied. All existing test case prioritization techniques do not include that process. Those techniques assume explicitly that the result is always satisfied.

There are six approaches to assign weights for each factors: (a) balance oriented (b) cost oriented (c) time oriented (d) defect oriented (e) complex oriented and (f) customization.

Balance oriented model assigns 25 points for each group (e.g., cost, time, defect and complex). Cost oriented model focuses on only cost factors. Time oriented model assigns 100 points for all time factors. Defect oriented model also assigns 100 points for each defect factor. Complex oriented model gives 100 points for complex factors as well. The last model allows users to customize and give their own points for each group. Assign a value for each test case. Due to the fact that auto-assigned value algorithm is very complex and beyond the scope of this study, the following assigned-value model is proposed.

This is because high priority test cases have higher priority value more than lower priority test cases. Therefore, the high percentage of high priority reserve effectiveness is desirable. This metric can be calculated as the following formula:

% HPRE = (# of Reserved / # of Total)*100

Size of Acceptable Test Cases

This metric is the number of acceptable test cases, expressed as a percentage, as follows:

% Size = (# Size / # of Total Size)*100

Total Prioritization Time

This is the total number of times the prioritization methods are run in the experiment. This metric is related to the time used during pre-process and post-process of test case prioritization. Therefore, less time is desirable. It can be calculated as the following formula:

TPT = ComT + CalT + RPMT

IV. RESULTS AND DISCUSSION

Here, discusses an evaluation result of the above experiment. This section presents a graph that compares the above proposed method to other three existing test case prioritization techniques, based on the following measurements: (a) high priority reserve effectiveness (b) size of acceptable priority and (c) total time. Those three techniques are: (a) random approach (b) Hema's method and (c) Alexey's method. There are two dimensions in the following graph: (a) horizontal and (b) vertical axis. The horizontal represents three measurements whereas the vertical axis represents the percentage value.

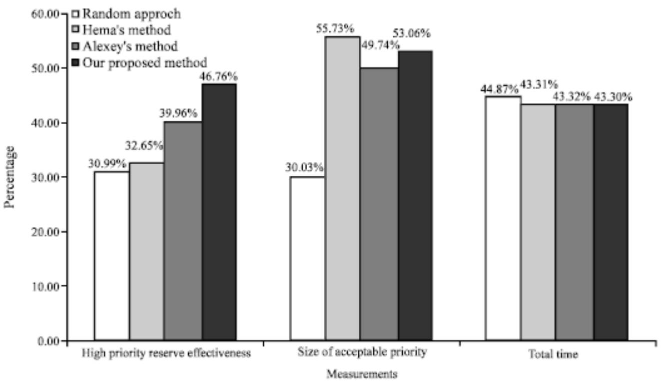

Fig. 4: An evaluation result of test case prioritization methods

Figure 4 represents an evaluation result of comparing an effectiveness of high priority reservation, a number of acceptable priority cases and total prioritization time. The above graph showed that the above proposed method generated the highest high priority reserve effectiveness. It was calculated as 46.76% where as the other techniques was computed less than 40%. Those techniques reserved the less number of test cases with high priority. Also, the graph showed that the proposed method consumes the least total time during a prioritization process, comparing to other techniques. It used only 43.30%, which is slightly less than others. Finally, the graph presented that the proposed method is the second best technique to reserve the acceptable priority test cases.

Figure this study determines and ranks the above comparative methods into five ranking: 5-Excellent, 4-Very good, 3-Good, 2-Normal and 1-Poor. This study uses a maximum and minimum value to find an interval value for ranking those methods.

For an effectiveness of high priority test cases reservation, the maximum and minimum percentage is 46.76% and 30.99%. The different between maximum and minimum value is 15.77%. An interval value is equal to a result of dividing the different values by 5. As a result, the interval value is 3.154. Thus, it can be determined as follows: 5-Excellent (since 43.606 to

46.76%), 4-Very good (between 40.452 and 43.605%), 3-Good (between 37.298 and 40.451%), 2-Normal (between 34.144 and 37.2988%) and 1-Poor (from 30.99 to 34.143%).

Table 4: A comparison of test case reduction methods

Algorithm	High priority reserve effectiveness	No. of acceptable test cases	Total time
Random method	1	1	1
Hema's method	1	5	5
Alexey's method	3	4	5
Our proposed method	5	5	5

For a number of acceptable test cases, the maximum and minimum percentage is 55.73 and 30.03%. The different value is 25.7%. The interval value is 5.14. Therefore, it can be determined as follows: 5-Excellent (since 50.59 to 55.73%), 4-Very good (between 45.45 and 50.58%), 3-Good (between 40.31 and 45.44%), 2-Normal (between 35.17 and 40.30%) and 1-Poor (from 30.03 to 35.16%).

For a total prioritization time, the maximum and minimum percentage is 44.87 and 43.30%. The different between maximum and minimum value is 1.57%. An interval value is equal to a result of dividing the different values by 5. As a result, the interval value is 0.314. Thus, it can be determined as follows: 5-Excellent (since 43.3 to 43.614%), 4-Very good (between 43.614 and 43.928%), 3-Good (between 43.928 and 44.242%), 2-Normal (between 44.242 and 44.556%) and 1-Poor (from 44.556 to 44.87%).

Therefore, the experiment result of those four comparative methods can be shown in Table 4.

The above result suggests that our proposed method is perfectly suitable for a scenario that concentrates on reserving a large number of high priority test cases, preserving acceptable cases and minimizing total prioritization time. Our proposed method is by far better than other three methods in term of high priority reserve effectiveness. Hema's method and our method are top two excellent prioritization methods for reserving medium priority test cases. Finally, the random approach consumes the greatest prioritization time comparing to other three methods.

V. CONCLUSION AND FUTURE WORK

This study proposes a new test case prioritization process, called 2R-2S-3R. The new process contains two processes, named 2R: (a) requisite and (b) reordering. The first process consists of two sub-processes, called 2S, which are: (a) select test case prioritization technique and (b) specify coverage or factors. The second process is composed of three sub-processes, called 3R, included as follows: (a) re-assign weight value (b) re-calculate priority value and (c) re-order test cases. This study reveals that there are many research challenges and gaps in the test case prioritization area. However, this study focus on solving the following research issues: (a) a lack of practical weight factors (b) an inefficient ranking algorithm used in the prioritization process and (c) ignore to reserve the high priority test cases. This study introduces a new practical set of weight factors used in the test case prioritization process. The new set is composed of four groups: (a) cost (b) time (c) defect and (e) complex. Also, this study proposes to improve the ability to weight and rank test cases with practical factors. This study compares the proposed method to other existing test case prioritization methods, which are: (a) random approach (b) Hema's technique and (c) Alexey's work. Consequently, this study reveals that the proposed method is the most recommended method to reserve the large number of high priority test cases with the least total time, during a prioritization process. However, there is an improvement to maintain and reserve the acceptable numbers of test cases, carried out in the future works.

VI. REFERENCES

[1]. Malishevsky, A., G. Rothermel and S. Elbaum, 2002. Modeling the cost-benefits tradeoffs for

regression testing techniques. Proceedings of the International Conference on Software Maintenance, Oct. 03-06, IEEE Computer Society, Washington, DC., USA., pp: 204-204.

[2]. Offutt, A.J., J. Pan and J.M. Voas, 1995. Procedures for reducing the size of coverage-based test sets. Proceedings of the 12th International Conference on Testing Computer Software, June 1995, Washington, DC., USA., pp: 111-123.

[3]. Beizer, B., 1990. Software Testing Techniques. 2nd Edn., Van Nostrand Reinhold, New York, ISBN: 0-442-20672-0, Pages: 550.

[4]. Qu, B., C. Nie, B. Xu and X. Zhang, 2007. Test case prioritization for black box testing. Proc. 31st Ann. Int. Comp. Software Appl. Conf., 10: 465-474.

[5]. Korel, B. and J. Laski, 1991. Algorithmic software fault localization. Proc. 24th Ann. Hawaii Int. Conf. Syst. Sci., 20: 246-252.

[6]. Kaner, J.D.C., 2006. Exploratory testing. Proceeding of the Quality Assurance Institute Worldwide Annual Software Testing Conference, Nov. 17, Orlando, FL., pp: 1-47.

[7]. Cadar, C. and D. Engler, 2005. Execution generated test cases: How to make systems code crash itself. Proceeding of the 20th ACM Symposium on Operating Systems Principles, March 25, Stanford University, USA., pp: 1-14.

[8]. Leon, D. and A. Podgurski, 2003. A comparison of coverage-based and distribution-based techniques for filtering and prioritizing test cases. Proceedings of the 14th International Symposium on Software Reliability Engineering, November 17-21, 2003, IEEE Computer Society Washington, DC, USA., pp: 442-453.

[9]. Jeffrey, D. and N. Gupta, 2006. Test case prioritization using relevant slices. Proc. 30th Ann. Int. Comp. Software Appl. Conf., 1: 411-420.

[10]. Hoffman, D., 1999. Cost benefits analysis of test automation. Proceedings of the SoftwareTesting Analysis and Review Conference, (STARC'99) Orlando, FL., USA., pp:1-14.

[11]. Aranha, E. and P. Borba, 2006. Measuring test execution complexity. Proceedings of the International Workshop on Predictor Models in Software Engineering, (PMSE'06) Informatics Center Federal University of Pernambuco, pp: 1-2.

[12]. Rothermel, G., R.H. Untch, C. Chu and M.J. Harrold, 2001. Prioritizing test cases for regression testing. IEEE Trans. Software Eng., 27: 929-948.

[13]. Rothermel, G., R.H. Untch, C. Chu and M.J. Harrold, 1999. Test case prioritization: An empirical study. In Proceedings of the 15th IEEE International Conference on Software Maintenance, August 30-September 3, 1999, Oxford, UK., pp: 179-188.

[14]. Rothermel, G., M.J. Harrold, J. Ostrin and C. Hong, 1998. An empirical study of the effects of minimization on the fault detection capabilities of test suites. Proceedings of the 14th IEEE International Test Conference on Software Maintenance, November 16-20, 1998, Bethesda, Maryland, pp: 34-43.

[15]. Rothermel, G., M.J. Harrold, J. von Ronne and C. Hong, 2002. Empirical studies of test-suite reduction. Software Testing Verif. Reliab.

[16]. Rothermel, G. and M.J. Harrold, 1997. A Safe, efficient regression test selection technique. ACM Trans. Softw. Eng. Methodol., 6: 173-210.

Cite this article as :

Ankit Sharma, Ms Shubhangi Sharma, "Implementation of Test Case Optimization and Prioritization Techniques in Software Testing", International Journal of Scientific Research in Science, Engineering and Technology (IJSRSET), Online ISSN : 2394-4099, Print ISSN : 2395-1990, Volume 10 Issue 5, pp. 210-215, September-October 2023.
Journal URL : https://ijsrset.com/IJSRSET2310519