A Systematic Review of Techniques for Test Case Prioritization

Aman Jatain
Assistant Professor
ITM University, Gurgaon-122017

Garima Sharma
ITM University,Gurgaon-122017

ABSTRACT

In software development life cycle, maintenance phase is an important phase as it deals with the activities like error correction, upgradation, deletion and optimization of software capabilities. For this reason, regression testing is required in order to revalidate the modifications in the software. It is an expensive process. Various techniques of performing regression testing are available. Software testers can select the technique that suit their requirement as well as optimize the basic cost and time factors. This paper mainly discusses various test case prioritization techniques for regression testing presented by various researchers and the various search algorithms used in the test case prioritisation process.

General Terms

Test Case prioritization techniques, Search Algorithm.

Keywords

Regression testing, Test case prioritization, algorithms, Requirement Based Test Case Prioritisation, Chronographic. Hybrid.

1. INTRODUCTION

Regression testing is used to uncover any new software bugs or error in the existing system after changes, such as enhancements that have been made to them. Let any program P and P' be its modified version T on P' along with the new test cases needed to effectively test the newly added code or functionality in producing. T be the test suite for P, then regression testing aims at reusing P'. Various techniques for regression techniques are [1]:

a. Retest all: It is one of the conventional methods which performs regression testing by rerunning all the test cases in the test suite and is therefore very expensive as compared to others. This requires more time as well as budget to be performed.

b. Regression test selection: In order to reduce the cost of running all the tests again we use RTS which selects a portion of test suite to rerun so that the cost of running selected tests is less than running the test cases that RTS allows us to omit. RTS may add new tests in order to cover the areas that are not covered by existing tests. Various researchers have given various techniques of RTS[1] for instance, modified non-core function technique [2], modification focussed minimization technique[3],coverage focussed minimization technique[4] etc.

c. Test case prioritization: Test case prioritization techniques [5] aim to schedule test cases in an execution order according to some criteria in order to meet some objectives which could be the likelihood of revealing faults earlier or increase the rate of fault detection, locating the high risk faults earlier, increasing the likelihood of revealing the errors related to specific code changes earlier, or to increase the confidence in reliability of system. Various prioritization criteria can be applied to a test suite according to the objective that needs to be met. Section 2 describes various techniques given by researchers in order to perform test case prioritization.

d. Hybrid approach: The hybrid approach is based on both the selection as well as prioritization of the test cases. Researchers working on this have proposed many algorithms [1], for instance, Test Selection algorithm by K.K.Aggrawal et al[6], Hybrid technique by Wong et al[7].

2. TEST CASE PRIORITIZATION TECHNIQUES

Rothermel [8], mentioned that the process of test case prioritization is needed in software testing because: (a) the regression testing consumes a lot of time and cost, (b) time or resources to run the entire test suite are not available, therefore (c) there is a need to identify which test cases should be run first.

"4C" classification of existing test case prioritisation techniques was introduced in [8] based on the prioritization algorithm characteristics. This classification is discussed below:

Table1. Test case prioritization techniques catalogue

TECHNIQUES	DESCRIPTION
Customer requirement - based techniques	Customer requirement factors are taken into account and provided some weights and based of these values test case weight for requirement is evaluated. Test cases with high weight value are executed first following the ones with lower value.

Coverage - based techniques	It is based on code coverage analysis or the quantity of code covered by a test case. Various coverage criterions are taken into account and the amount of coverage is evaluated and used to prioritize the test cases.
Cost Effective - based techniques	This technique prioritizes the test cases based on costs factors, such as cost of running test cases, cost of analysis etc.
Chronographic history-based techniques	This technique prioritizes the test cases based on test case's prior executions in order to increase or decrease the likelihood that it will be taken into account in current test execution.

2.1 Customer Requirement-based techniques

Hema[9] proposed the requirement based test case prioritisation technique based on 3 factors associated with the customer requirements and proposed to use those factors to assign weight to test cases. Those factors are: (1) customer-assigned priority on requirements(CP) is the priority value between 1-10 i.e assigned by the customer based on the importance of the requirement , (2) requirement complexity(RC), is the value between 1-10 assigned by the developer based on the implementation complexity (3) requirements volatility(RV) is the value based on the how many times consumer is modifying the project requirements during the software development cycle .Higher the values of these factors (*PFvalue*), more is the need of prioritization required. The development team assigns weight to each of these factors (*PFweight*).Based on these, the following equation is used to calculate the weighted prioritization (WP) factor that measures the importance of testing a requirement earlier.

$$WP = \Sigma(PFvalue * PFweight);\ PF=1 \text{ to } n$$

According to the WP values, test cases are order and the ones with higher values are executed first. Thereafter, Hema[10] presented a new value driven approach named PORT (Prioritization of Requirements for Testing) which aims to prioritize tests based on four requirement assess factors: requirements volatility (*RV*), customer priority(*CP*), implementation complexity (*IC*) and fault proneness of the requirements (*FP*).

$$PFVi = \Sigma 4j=1(FactorValueij * FactorWeightj)$$

In the above equation *PFVi is* the prioritization factor value for any requirement *I*, *FactorValueij* is the value for factor *j* for requirement *i and FactorWeightj* is the factor weight for the *j*th factor for a particular product.

Various research has been done in this field of test case prioritisation using the factors mentioned in[3] and [10]. Ashraf et al[11] presented a value based prioritisation technique in 2 levels i.e on requirement level in which is provided by the stakeholders based on requirement factors and further on second level prioritisation, the development team provided grades to the test cases according to the respected requirements. These test cases were scored according to some pre-defined factors. (1) customer priority (2) implementation complexity (3) requirement volatility (4) requirements traceability (5) execution time and (6) fault impact of requirement. Values obtained from these two levels were calculated to get the net value and based on these values test cases were ordered for execution. Also[12] proposed some similar requirement factors to prioritise test cases on the basis of fault severity and further in [13] they proposed a genetic algorithm using a fitness function and analysed that this approach gave better and effective set of test cases as compared to random prioritisation technique. In [14]changing requirements and the impact of these changes on other modules is considered, two requirements based factors are considered i.e *Rfactors and* requirement priority (*RPriority)* computed from those two *Reactor*'s *R-value and Reweight*. Rfactors include Requirement Modification Impact Localization (*RMIL*) *value* calculated by dividing the number of changes for any requirement R divided by the maximum number of changes to be made as a result of change in that requirement R among entire project requirements; and Degree of Coupling (*DCP*) value which is calculated according to the levels mentioned in[14].

2.2 Coverage - based techniques

The coverage-based technique is a white-box testing technique i.e a method that tests internal structures of a software. Unlike black-box testing, this tests the program behavior against requirements specifications. The coverage based techniques uses the coverage factors like branch testing, statement level testing, function level testing, requirement coverage.

Gaurav[1] presented a grouping of various coverage based techniques based on statement level and function level. The various prioritization techniques are implemented based on code coverage information include (i) Statement based coverage which prioritises test cases based on no. of statements covered by test cases, (ii) Branch based coverage which prioritises test cases based on the no. of branches executed, (iii) Loop based coverage which prioritises based on no. of loops executed and (iv) Condition based coverage which prioritises based on coverage measured in terms of numbers of basic Boolean terms executed.[8] Jeffrey presented a new approach for the Prioritisation of test cases that is based the number of statements executed that influence or have the potential to influence the output produced by the test case. The set of such statements corresponds to the relevant slice, that is computed based on the output of the program when executed by the test case [8]. The factors they used in their approach to prioritize test cases are (a) the number of statements in the relevant slice of output for the test case as any change should necessarily affect some computation in the relevant slice to be able to change the output for this test case and (b) the number of statements executed by the test case but not in the relevant slice of the output. Jeffrey [8] determined the weight of test cases according to the formula

$$TW = ReqSlice + ReqExercise$$

Where *TW* is a weight prioritization calculated for each test case, *ReqSlice* is the no. of requirements presented in the relevant slice of output for each test case and *ReqExercise* is the no. of requirements exercised by the test case.Prakash and Rangaswamy[15] multiple criterion based merging technique for tests case prioritization method where the test cases are prioritized based on more than one coverage criteria such as fault coverage, statement coverage, path coverage, function coverage, etc and it was seen that the proposed method considerably improves the rate of coverage criteria.

[19] proposed the technique to build reusable cluster-based test cases during the framework domain engineering stage. It introduced a methodology that experimentally estimated the possible coverage of the reusable cluster-based test cases that are generated using the framework hook descriptions. The study results showed that the coverage results of the reusable class-based test cases are better than the coverage results of the reusable cluster-based test cases which proved that hook descriptions can be used to build reusable test cases.

In [20], version specific test case prioritisation was considered which deals with prioritisation of test cases based for a particular version of software and effectiveness of this technique is studied over the general prioritisation and moreover prioritisation techniques are divided into three groups namely: control level, statement level and function level and the results are shown that show statistical differences in various coverage prioritisation techniques. Also [21] proposed a test case selection as well as version specific prioritisation technique where all the changes that are made in a software are available and the prioritisation focuses only on the modified parts and aims to execute all modified lines of source code with a minimum number of selected test cases. This works by executing the modified lines of source code at least once and Executing the lines of source code after deletion of deleted lines from the execution history of the test case and that are not redundant.

2.3 Cost Effective - based techniques:

Cost effective-based techniques are prioritization techniques in which test cases are ordered for execution based on costs, such as cost of analysis and cost of prioritization. [8]Leung and White presented a cost model that incorporates various costs of regression testing i.e. the cost of executing and validating test cases, and the cost analysing the test selection. This model divided the costs into direct cost including test selection, test execution and result analysis; and indirect costs including overhead cost and tool development cost. This model also had a disadvantage that it ignored the cost of undetected faults [17]. Alexey Malishevsky [8] used the cost factors like cost of analysis, Ca(T) and cost of the prioritization algorithm, Cp(T) to calculate the weight of a test case and then arrange them accordingly.Also Malishevsky divided testing process in prilimnary and critical phase where the prilimnary phase activities had different costs than critical phase as there may be greater changes in the critical phase for things like release time of the software. The cost of a test case depends upon the resources required to execute the test case as well as validate it. These resources can be machine or human time, hardware cost for execution of test case, wages etc.APFD(average percentage of faults detected) metric measures the average percentage of faults detected while executing the test cases in a test suite in a given order. [18]The APFD metric is based on two assumptions: (1) all faults have equal costs (fault severities), and (2) all test cases have equal costs(test costs).The cost-cognizant prioritization techniques are used when these assumptions do not hold. It requires an estimate of the severity of each fault that can be revealed by a test case as it reflects the cost if a fault persists and affect the users using the software and also if it doesn't reach the users. In [8], some additional cost factors are proposed like execution cost, cost of analysis, cost of data preparation and cost of validation.[23] used a cost and coverage factors and presents a metric for assessing the rate of fault detection of prioritized test cases, APFDc, that incorporates varying test cases and fault costs.

2.4 Chronographic history based prioritisation techniques

This technique takes into account test execution history in order to prioritise the test cases. Only a few researchers have researched in this area. Some research ideas like exponential weighted moving average and exponential smoothing have taken the base from statistical quality control and exponential smoothing respectively [8].Also another approached also listed in [8] is the use selection probabilities of each test case at some particular time based upon time-ordered observations taken by executing the test case again and again and a smoothing constant used to weight individual historical observations. The higher the value of the probability the recent is the observation. For black box testing when source code is not available, one way to prioritise test cases for execution is to initialize the test suite based history of test, and then adjusting the rest of the test cases based on run-time information available to us by forming a matrix and recording the values[8]. This can be done by selecting a subset of test cases and ordering them according to available historical information and then forming a matrix based on available information, then running a test case from the selected test case and reordering the rest test cases using run-time information and information in the matrix that is formed.[22] proposed a technique cost cognizant prioritisation based on the historical test data. The test cost list, detected fault list and fault severity list values are obtained from historical information repository and then genetic algorithm is proposed to obtain an efficient order and the results are again stored in the repository. Experiments performed to study the effectiveness of the technique indicated that this technique has effective fault detection.

3. ALGORITHMS FOR TEST CASE PRIORITISATION

Many search algorithms are being used as basis in test case prioritisation process. 5 main algorithms explained in [16] are:

3.1 Greedy algorithm

It uses the "next best" search terminology in which the element with the the maximum weight is taken first followed by the next maximum weight and so on. This search aims to minimize the estimated cost to reach some goal. It is inexpensive in implementation as well as execution time and thus advantageous. For example, if there are 5 test cases say A, B, C, D where A covers 7 statements, B covers 6 statements, C and D both cover 5 each. If greedy algorithm is applied, the test case A is selected first since it covers maximum statements i.e 7,. Test case B is selected next which covers 6 statements, Now since Test cases C and D cover the same number of Statements, the Greedy Algorithm could return either A; B;C;D or A; B; D;C, depending upon the order in which test cases are considered. The cost of prioritisation for greedy algorithm is O(mn) where m is the number of statement in program and n is the no. of test cases in the test suite.

3.2 Additional Greedy Algorithm

It is a type of greedy algorithm with a slightly different approach. This algorithm works by using feedbacks from previous selections. It is more efficient as it selects the maximum weight element from the space that that is not yet a part of previously selected elements. For example when applying additional greedy algorithm, if some test case A is selected first from 4 test cases namely A, B, C and D (as it covers maximum no. of statements) leaving statements 5 and 6 uncovered. Test case B will be skipped if it covers neither statement 6 nor statement 5. Then if Test cases C and D cover statements 5 and 6,respectively.Additional Greedy would return either A;C; D;B or A; D;C;B. The cost of prioritisation for this algorithm is given as O(mn^2) as cost of selecting a test case and readjusting the test cases again is O(mn) and the process of readjustment is done O(n) times.

3.3 2-Optimal Algorithm

The 2-Optimal (Greedy) Algorithm[16] is an instantiation of the K-Optimal Greedy Approach when value of K is 2.The [16] Traveling Salesman Problem (TSP) which is defined as “find the cycle of minimum cost that visits each of the vertices of a weighted graph G at least once” is an example of using 2-optimal algorithm. The cost of prioritisation for this algorithm is given as O(mn^3) as cost of selecting a pair of test case and readjusting the test cases again is O(mn^2) and the process of readjustment is done O(n) times.

3.4 Hill Climbing Algorithm

It is also known as local search algorithm.There are two basic variations to this search algorithm i.e Steepest ascent and next best ascent. It uses the concept of neighbourhood [16] which is defined as any new ordering of a test suite that can be obtained by exchanging the position of the first test case and any other test case. It is easy and cheap to use. However it [1] has a con of dividing the O(n^2) neighbours and is unlikely to scale.

3.5 Genetic Algorithm

It is based on Darwins Theory of Survival of the Fittesst[1]. The population initialy is a set of randomly generated individuals where Each individual is represented as a sequence of variables/parameters known as the chromosome[16].The steps involved in this search algorithm are: Encoding, Selection, Crossover and Mutation.

It was formulated in [16] that the Additional Greedy and 2-Optimal Algorithms are the best approaches overall; Additional Greedy, 2-Optimal, and Genetic Algorithms always outperform the Greedy Algorithm. In [1], PORT version 1.1 is also listed as a prioritisation algorithm for Requirement based prioritisation of test cases.

4. PROPOSED APPROACH AND CHALLENGES

Requirements prioritization is an essential area of research in the field of test case prioritisation. It aims to maximize the software value delivered to the clients and also consider changing requirements. The order in which requirements are implemented in a system affects the value of software that is being delivered to the final users. The basic challenge that developer faces are: a) to rank the requirements so as to trade off user priorities and implementation constraints, such that the technical dependencies among requirements and necessarily limited resources are allocated to the software and the highest priority requirements are implemented first, b) As requirements change frequently there is a need of a flexible approach to facilitate requirements change management. c) Also the developer needs to make the prioritisation process more interactive as the opinions of different developers may not be the same about assigning the priority to the requirements. So, there must be a mechanism which takes the perspective of different users to prioritise the requirements rather than simply relying on the single developer. This would make the prioritisation process more efficient and interactive.

Despite the clear need to prioritize requirements in software projects, finding a practical method for requirements prioritization is a difficult task. Existing requirements prioritization methods that provide the most consistent results are complex, and therefore the difficult to implement. So, there is a need of more informal methods that save time and are easier to apply, but may not be suitable for practical scenarios because they lack the structure and consistency required to properly analyse the requirements.

The possible approach to overcome the challenges that are discussed above could be something that attempts to quantify the quality of requirements to provide a measurement that is representative of all quality criteria identified for a specific software project. A quality measurement metric can be formed using some new requirement quality attributes or even using some existing ones as per the requirement of the software developing team. This metric can be used as the main measure for requirements prioritization and based on the results of these ,etric a function can be calculated that ranks the requirements based on its values.

Requirement analysts possess relevant knowledge about the relative importance of requirements. After prioritizing requirements according to above metric, to further make the process interactive, an algorithm can be further introduced into this approach which considers second opinion from various experts to produce a requirement ordering which complies with the existing priorities, satisfies the technical constraints and takes into account the relative preferences elicited from the user. After considering the opinions from various analyst, a genetic algorithm can be applied where these opinions can be considered as the populations and a fitness function is calculated so as to obtain a final set of prioritised requirements.

After the set of prioritised requirements is obtained, test cases can be ranked based on the degree how a particular test case meets the requirements, the test cases that meet the requirements that appear early in the above obtained prioritised sequence are runned first following the ones that cover the requirements appearing lately in the requirement prioritisation sequence.

5. CONCLUSION

Various regression testing techniques are discussed in this paper, mainly focussing on test case prioritisation. The various techniques of test case prioritisation are explained in detail to make the researches understand the scope of working various techniques. Also, various search algorithms used in the process of test case prioritisation are listed along with explanation. This paper also lists various challenges associated with the requirement based prioritisation and also proposes an approach to overcome these challenges.

6. REFERENCES

[1] Gaurav Duggal, Bharti Suri [2008],"Understanding regression testing techniques" Proceedings of 2nd National Conference on Challenges and Opportunities in Information Technology.

[2] Y. Chen, D. Rosenblum, and K. Vo. Test Tube, “A system for selective regression testing,” In Proceedings of the 16th International Conference on Software Engineering, pages 211-220, May 1994.

[3] K. Fischer, F. Raji, and A. Chruscicki, "A methodology for retesting modified software," In Proceedings of the National Telecommunications Conference B-6-3, pages 1-6, Nov. 1981.

[4] R. Gupta, M. J. Harrold, and M. Soffa, "An approach to regression testing using slicing," In Proceedings of the Conference on Software Maintenance, pages 299-308, Nov. 1992

[5] S. Elbaum, A. G. Malishevsky and G. Rothermel, "Test Case Prioritization: A Family of Empirical Studies," IEEE Transactions Software Engineering, Vol. 28, No. 2, 2002, pp.159-182.

[6] K. K. Aggrawal, Yogesh Singh, A. Kaur, " Code coverage based technique for prioritizing test cases for regression testing," ACM SIGSOFT Software Engineering Notes, vol 29 Issue 5 September 2004.

[7] W. E.Wong, J. R. Horgan, S. London and H.Agrawal, "A study of effective regression testing in practice," In Proceedings of the 8th IEEE International Symposium on Software Reliability Engineering (ISSRE' 97), pages 264-274, November 1997.

[8] Siripong Roongruangsuwan, Jjirapun Daengdej, "test case prioritization techniques", Journal of Theoretical and Applied Information Technology, © 2005 - 2010 Autonomous System Research Laboratory, Science and Technology, Assumption University, Thailand.

[9] Hema Srikanth, Laurie Williams," Requirements-Based Test Case Prioritization".

[10] Hema Srikanth, Laurie Williams, Jason Osborne," System test case prioritisation of new and regression test cases," Proceedings of the seventh international workshop on Economics driven software engineering research, pages 64-73, May 2005.

[11] E. Ashraf, A. Rauf, and K. Mahmood, "Value Based Regression Test case Prioritisation", Proceedings of the World Congress on Engineering and Computer Science 2012 Vol I WCECS 2012, October 24-26, 2012, San Francisco, USA

[12] Varun Kumar; Mohit Kumar, "Test Case Prioritization Using Fault Severity". International Journal of computer science and technology, Vol. 1, No. 1,p 67-71.

[13] Sujata, Mohit Kumar, Dr.Varun Kumar, "Requirements based Test Case Prioritization using Genetic Algorithm". International Journal of computer science and technology Vo l. 1, IS Su e 2, December 2010

[14] Aseem Kumar, Sahil Gupta, Himanshi Reparia, Harshpreet Singh, " An approach for test case prioritization based upon varying requirements". International Journal of Computer Science, Engineering and Applications (IJCSEA) Vol.2, No.3, June 2012

[15] N.Prakash, T.R.Rangaswamy, "Multiple Criteria Based Test Case Prioritization for Regression Testing" European Journal of Scientific Research,ISSN 1450-216X Vol.84 No.1 (2012), pp.36 – 45

[16] Zheng Li, Mark Harman, and Robert M. Hierons,"Search algorithms for regression test case prioritisation," IEEE trans. On Software Engineering, vol 33, no.4, April 2007

[17] J u n g -Mi n Kim , A d a m P o r t e r," A History-Based Test Prioritization Technique for Regression Testing in Resource Constrained "

[18] Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel, Sebastian Elbaum" Cost-cognizant Test Case Prioritization" Technical Report TRUNL-CSE-2006-0004, Department of Computer Science and Engineering, University of Nebraska –Lincoln, 2006.

[19] Jehad Al Dallal, Paul Sorenson," Estimating the Coverage of the Framework Application Reusable Cluster-Based Test Cases" . Information & Software Technology 50(6): 595-604 (2008).

[20] G. Rothermel, R.H. Untch, C. Chu, and M.J.Harrold, "Prioritizing Test Cases for Regression Testing," IEEE Trans. Software Eng., vol. 27, no. 10, pp. 929-948, Oct. 2001.

[21] Ruchika Malhotra, Arvinder Kaur and Yogesh Singh," A Regression Test Selection and Prioritization Technique", Journal of Information Processing Systems, Vol.6, No.2, June 2010

[22] Yu-Chi Huang' , Kuan-Li Peng, Chin-Yu Huang," A history based cost cognizant test case prioritisation technique" Journal of Systems and Software Volume 85, Issue 3, March 2012, Pages 626‑637.

[23] A.Askarunisa, L.Shanmugapriya, Dr.N.Ramaraj,"Cost and Coverage Metrics for Measuring the Effectiveness of Test Case Prioritization Techniques", INFOCOMP Journal of Computer Science.