Survey on Test Case Prioritization and Measuring Test Cases Using APFD

Han Moi Sim¹⁾, D. Sai Teja Reddy²⁾

Abstract

The software development is an evolutionary process in which a system is modified until every requirement is met. While testing the product after modification of the code, we need to check if there happen to be generation of new faults or reappearance of old faults in already tested code. So we use regression testing to check for change of effectiveness of code or arise of faults due to modification of code in properly working system. Regression testing of system is expensive and time taking. Test case prioritization is to prioritize test cases so that running of test cases in prioritized order improves certain measures like rate of fault detection. Software testers may prioritize their test cases to reduce the cost of regression testing. Various research papers related to test case prioritization, especially on APFD (Average Percentage of Faults Detected) will be downloaded, studied and reviews of the papers will be prepared. These reviews will be categorized and organized in a systematic way and finally the term paper report will be generated.

Keywords: prioritization, test case, detection, faults, systematic, APFD.

1. Introduction

We want to study various prioritization techniques and to measure the effectiveness of prioritization techniques using APFD.

1.1 Rationale:

The software development is an evolutionary process in which a system is modified until every requirement is met[1]. While testing the product after modification of the code, we need to check if there happen to be generation of new faults or reappearance of old faults in

Received(February 17, 2015), Review Result(1st: March 2, 2015, 2nd: March 30, 2015), Accepted(May 10, 2015)

¹Department of Drone, Seoul Hoseo technical College 420, Gangseo-ro, Gangseo-gu, Seoul
email: kmusic@naver.com

²(Corresponding Author) Vetri Vikas Educational Institutions, Mallur (Via), Rasipuram Taluk, Annamalaipatti, Tamil Nadu, India
email: saitejareddy007@gmail.com

already tested code[2]. So we use regression testing to check for change of effectiveness of code or arise of faults due to modification of code in properly working system. Regression testing of system is expensive and time taking[3]. If the user wants the system before execution of all the test cases, there is a chance of failure of the system because of undetected fault[4]. So there is a need for early detection of faults in the system. To increase the number of faults in early detection we need to check for all possible permutations of test cases[5-6]. But as the number of test cases increases the combinatorial explosion may occur, which makes it impossible. Instead we can prioritize the test cases using some factors. We want to study how many ways are there to prioritize the test cases.

1.2 Additional information

The test cases are prioritized to increase the effectiveness of the test suite. The effectiveness of the test suite represents the rate of fault detection which is a measure of how fast the faults are detected in early stage of execution. The effectiveness can be measured using metrics[7-9]. There are many metrics which measure the effectiveness of the test suite order. In this paper we show interest on APFD metric. The test suite consists of information about the test case to be performed. The test suite is taken in form of fault matrix in some algorithms to prioritize the test cases. The fault matrix consists of the information about the faults that can be detected by the run of test cases and test cases to be executed on the software product. The fault matrix is a matrix with faults as rows and test cases as columns.

2. Statement of purpose

Regression testing is a testing that is done to detect new bugs, or regressions, in a system after changes like improvisations, bug fixing or changes in configuration, have been done to them. The purpose of regression testing is to ensure that such changes have not introduced new faults. Often Regression test suites are tests that software engineers have developed previously, and that have been saved so that they can be used later to perform regression testing. The natural order of test cases to be performed on software product might take H machine hours to complete execution. But the client might not be able to spend that much in testing. In this case instead of a natural order we would prefer an order of test cases which gives early detection of faults. Even before the execution of all the test cases more number of faults should be detected so that more reliable product can be released. Let A, B, C are the

test cases in a test suite which detect (F2,F5), (F1,F2,F3F4), (F3) respectively. The natural order is A-B-C which is original order of test cases. The best order is the order which gives early detection of faults. The ordering of test cases such that early detection of faults takes place is called prioritization according to rate of fault detection. The prioritization can be done according to other factors like code coverage. To measure how early the faults are detected in test suite we use APFD a metric to measure the rate of fault detection. There are some other metrics but APFD (Average Percentage of Fault Detection) is the first and more commonly used metric. We want to study other prioritization techniques of the test cases and to measure how effective the techniques work on prioritized test cases. To measure the effectiveness we are using APFD.

2.1 Basic Description

Test case prioritization: Test case prioritization techniques prioritize test cases for execution in an order that tries to maximize some objective function. A variety of functions are applicable; one of such functions involves rate of fault detection. The rate of fault detection is a measure of how fast the faults are detected within the testing process. This technique is very useful to prioritize the test cases.

Regression testing: Regression testing is a part of the testing phase of software development where, when new modification or improvisation are added to the system and the added functionality is tested, then the previous functionality is re-tested to make sure that no new modification has corrupted the system.

APFD: APFD is the metric which analyzes the effectiveness of prioritization techniques. We use this metric to analyze the fault detection rate of test suite. In this metric we consider the weighted average of no. of faults contained in the program.

Test case: Test case is a document, which has a set of data about test like preconditions, expected results and post conditions, developed for a particular test scenario in order to verify compliance against a specific requirement. The set of test cases are called test suite.

While trying to find the good prioritized order of test cases we need to consider all possible permutations of test cases. The test suite is expressed as Fault Matrix. The Fault matrix consists of n test cases (T_i for $i=1,2,3,...,n$) which could detect m faults (F_i for $i=1,2,3,...,m$). The rows of

the fault matrix consist of the faults that are detected previously and the columns consist of the test cases used to test the software previously. The possible orders are permutations of the columns of the fault matrix. The permutations of the test cases in the matrix are the possible orders. The permutations of n test cases in a test suite give $n!$ Test suites each with different order of test cases. Test Case Prioritization is the process of arranging test cases in an order according to some criteria.

No prioritization is the order of untreated test suite that is used to compare the rate of fault detection of original test case order. The test suite results depend upon how the test cases are originally taken.

Random Prioritization is a prioritization technique in which test cases are ordered randomly in a test suite. It might be better than no prioritization. In some cases, no prioritization does not give good result at that time random prioritization is better.

Optimal Prioritization is a prioritization technique which gives the beat order according to criterion. But finding it among the all other for large sets of data is difficult. So, a greedy optimal prioritization technique is considered.

Total statement coverage prioritization is a prioritization technique in which the test cases are prioritized as per the number of their covered statements. The statements covered by test cases can be identified by using a program.

The test cases are arranged in descending order of count of covered statements. If two or more test cases have same number of statements then those test cases are arranged randomly.

Additional statement coverage prioritization is a prioritization technique in which test cases are scheduled according to order of total coverage achieved. If more than one test case covers same statements, when once the statements are covered the count in other test cases will be reduced.

Total branch coverage prioritization is a prioritization technique which is similar to total statement coverage prioritization. This technique uses coverage of branches rather than coverage of statements. This technique is not as efficient as the total statement coverage prioritization.

Additional branch coverage prioritization is a prioritization technique which is similar to additional statement coverage. This technique use test coverage in terms of branches of program. In this technique we need a method to prioritize the left over test cases after total coverage.

Total fault-exposing-potential (FEP) prioritization is a prioritization technique that prioritizes the test cases first which has high ability to expose fault. The branch and statement may ignore the test cases which can reveal particular faults. This technique focuses on potential of a

test case in exposing faults.

Additional fault-exposing-potential (FEP) prioritization is a prioritization technique which is extension to branch or statement coverage prioritization and to additional branch or statement coverage prioritization and total FEP prioritization. The additional execution of test cases is unnecessary. So as the test cases are executed the weighted values of test cases will be reduced.

Total Function coverage prioritization is a prioritization technique which works at function level; it prioritizes test cases by ordering the test cases basing on number of functions executed by the test cases.

Additional function coverage prioritization is a prioritization technique includes feedback to total function coverage prioritization. It prioritizes test cases basing on number of functions they cover excess.

Total FEP (function level) prioritization is a prioritization technique similar to total FEP prioritization technique. To make an interpretation of that system to function level, a capacity level estimation of fault exposing potential is required.

Additional FEP (function level) prioritization is a prioritization technique which is similar to total FEP (statement level). The technique adds feedback to total FEP(function level) prioritization technique.

Faults generally exist in irregular number of faults in functions. The technique finds and uses the number of faults detected in functions to prioritize the test cases.

Additional fault-index (FI) prioritization is a technique which is similar to additional function coverage prioritization by summing feedback to total fault index coverage prioritization. As the test cases are run the fault index values are changed.

Total FI with FEP coverage prioritization is a technique which uses estimation of fault exposing capacity and estimation of proneness of fault. It might be able to get high rate of fault detection.

Additional FI with FEP coverage prioritization is a prioritization technique which adds feedback to above technique. The technique changes the weight age at the time of run time as the test cases are executed.

Total DIFF prioritization is a prioritization technique which is similar to FI-based technique. It is used for estimation of fault inclination. This technique requires summation of many metrics. It is done using "diff" tools. This technique uses changed data from diff tool.

Additional DIFF prioritization is a prioritization technique which is similar to additional FI prioritization technique. Unlike additional FI prioritization technique it uses data from diff tool to prioritize test cases.

Total DIFF with FEP prioritization is a prioritization technique which is similar to total FI with FEP prioritization. In addition to that, this technique uses changed information from diff tool to prioritize the test cases.

Code-based prioritization: The supposition is that the augmentation of auxiliary scope will build the possibility of the increase of detection of fault. For instance, if an (experiment A) covers more explanations, fundamental pieces, or routines contrasted with another (experiment B), more faults can be recognized with experiment An as a

result of the more noteworthy number of proclamations, essential blocks of code, or techniques worked out.

Distributed based approach: It organizes experiments in basis of the dispersion of the profiles of test cases. Grouping test cases demonstrate that comparative test cases are repetitive and detached bunches may bring about failures.

Human-based approach: A human analyzer is utilized as a part of this sort of methodology. Prioritization depends on the examinations made by the human analyzer.

Probabilistic approach: Probabilistic hypothesis is applied. Determination likelihood is relegated to every test case in a test suite, and the execution history of every test case is considered. Bayesian system based methodologies are appeared in this classification.

History based approach: Historical data, for example, history of execution or change data are utilized as a part of this methodology.

Requirement based approach: Properties of requirements are utilized. For instance, variables, like, client allocated priority on prerequisites, necessity instability, and designer view execution many-sided quality of necessities can be applied.

Model based approach: Distinctive models, for example, UML (Unified Modeling Language) succession or activity diagram are utilized rather than code pieces. The UML is the standard visualization dialect for displaying the product.

Cost aware approach: The expenses of test cases are considered in basis of the fact that the expenses of all the experiments can't be equivalent.

Other approaches: In this class, the authors demonstrated different methods, like interface-contract change, relevant cuts, and call-tree ways.

3. Conclusion

Test case prioritization techniques are used to prioritize the test case of a test suite in an order to improve a criterion. The techniques are studied by many researchers and many models and algorithms are proposed to improve the criterion. The models are published in many papers. The IEEE, Springer, ACM papers on Test case prioritization are collected, studied, understood and is categorized. The reviews of the papers are categorized according to techniques to make better understandings of Test case prioritization techniques and to measure the effectiveness of the techniques using APFD metric.

References

- [1] S. Elbaum, D. Gable, and G. Rothermel, Understanding and measuring the sources of variation in the prioritization of regression test suites, Software Metrics Symposium, 2001. METRICS 2001. Proceedings. Seventh International, (2001), pp.169-179.
- [2] X. Qu, Configuration aware prioritization techniques in regression testing, software Engineering-Companion, Vol.2009. ICSE-Companion 2009, 31st International Conference on, (2009), pp.375-378.
- [3] J. M. Kim and A. Porter. A history-based test prioritization technique for regression testing in resource constrained environments, Software Engineering, 2002. ICSE 2002. Proceedings of the 24rd International Conference on, (2002), pp.119-129.
- [4] G. Rothermel, R. H. Untch, C. Chu, and M. J. Harrold, Prioritizing test cases for regression testing, Software Engineering, IEEE Transactions on 27, (2001), No.10, pp.929-948.
- [5] M. Sherriff, M. Lake, and L. Williams, Prioritization of regression tests using singular value decomposition with empirical change records, Software Reliability, ISSRE'07. The 18th IEEE International Symposium on, (2007), pp.81-90.
- [6] R. Khan, S. Ur, S. P. Lee, R. M. Parizi, and M. Elahi, A code coverage-based test suite reduction and prioritization framework, Information and Communication Technologies (WICT), 2014 Fourth World Congress, (2014), pp.229-234.
- [7] <https://software.intel.com/>, March 11 (2014).
- [8] <http://www.dis.anl.gov/projects/ri.html>, February 23 (2012).
- [9] R. Albert and A. L. Barabasi, Statistical mechanics of complex networks, Reviews of Modern Physics, (2002), Vol.74, No.1, pp.47-97.