Volume 5, Issue 1, January 2015 ISSN: 2277 128X

International Journal of Advanced Research in Computer Science and Software Engineering

Research Paper

Available online at: www.ijarcsse.com

Test Case Selection and Prioritization Using Multiple Criteria

M. Suppriya, A. K. Ilavarasi
Department of Computer Science
Sona College of Technology,
Tamil Nadu, India

Abstract —Regression testing activities such as test case selection and test case prioritization are ordinarily based on the criteria which focused around code coverage, code modifications and test execution costs. The approach mainly based on the multiple criteria of code coverage which performs efficient selection of test case. The method mainly aims to maximize the coverage size by executing the test cases effectively. The selected test cases are then prioritized based on the priority which depends on the code coverage to achieve the desired results.

Keywords—Test case selection, Test case prioritization, Cod coverage, Jaccard distance, Greedy algorithm.

I. INTRODUCTION

Software testing is one of the important phases in the development of software systems. It evaluates the capability of the software program and reveals as many as errors in a given product for achieving the desired results. Sometimes adapting the new features or fixing of bugs introduces changes to the software. So, the Regression testing intends to ensure that changes to the software have not adversely affected it and have not introduced errors. Regression testing on the other hand repeats other tests in 'parallel' areas to ensure that the applied fix or a change of code have not introduced other errors or unexpected behavior. In regression testing, a good test case is the one that has a high probability of detecting an undiscovered defects and a necessary part of every test case is a description of the expected result. But, rerunning the existing test cases for regression testing is a potentially costly and time consuming task. Therefore test case selection and prioritization those are used to determine which test cases need to select and execute thereby making the results more effective. In test case selection, a predetermined number of test cases from an existing test suite are selected to determine which test cases are needed to execute based on some criteria. The important test cases are prioritized using a test case prioritization strategy for performing tests on time constrained conditions. Test case prioritization is the process of scheduling test cases to be executed in a particular order so that test cases with a higher priority are executed earlier in the test sequence. It prioritize the test cases so as to increase a test suite‘s rate of fault detection that is how quickly a test suite detects faults in the modified program to increase reliability. Studies reveal that relative performance of a technique varies across different programs. Therefore a regression
testing approach that is more consistent across different programs is required. A new regression test case selection and prioritization approach which uses multiple criteria that is consistent compared to existing techniques can be considered.

II. PROBLEM STATEMENT

For large software systems, there may be thousands of test cases available in its test suite. When a change is introduced into the system for next version, rerunning all the test cases is a costly and time consuming task. Therefore a need for selecting a subset of test cases from the original test suite is necessary. But the use of multiple criteria should increase the code coverage. So, an effective test case selection strategy is to be designed based on the code coverage. The Sum of coverage, min-max and max-max are the criteria considered for the selection of test cases. Prioritization is based on the optimized test cases derived from the selection process.

III. LITERATURE SURVEY

The goal of regression testing is to ensure that changes to the system have not introduced errors. One approach is to rerun all the test cases in the existing test suite and check for new faults. But rerunning the entire test suite is often too costly.

To make the execution of test cases more cost effective, two major approaches are made use of. They are the Regression Test Selection (RTS) and Regression Test Prioritization (RTP) techniques. Many RTS and RTP techniques consider a single criterion for optimization of test cases. But, the use of a single criterion severely limits the ability of the resulting regression test suite to locate faults. Harman et al., induce the need of multiple criteria [1] and provides a list of criteria with different weights. Mirarab et al., used multi-criteria optimization technique for test case selection strategy. The two criteria for selection are code coverage and sum coverage of the program [2][13]. Code coverage assumes that there exist test cases that effectively cover the changed area of code of the software. Sum coverage is a new approach that maximizes the minimum sum of coverage across all software elements. The selected test cases are prioritized using a greedy algorithm to maximize the minimum sum of coverage across all software elements. Another technique for regression test selection is Selective regression testing (SRT) [5]. It determines which test executions are potentially affected by a given modification to the system. The test cases that utilize code fragments are traced and their dependencies are extracted. The code fragments which are modified are indicated as affected test cases. Safe SRT technique [5] increases the number of selected test cases. Safe SRT approaches provide a guarantee that the set of selected test cases comprises all error-uncovering ones. Dejavu [6] is used for the safe selection test. It constructs the control flow graph of original and modified programs. Then it traverses these two graphs synchronously and identifies the modified nodes and edges. A capturing feature is required to record the results and execution profiles of tests. A simple execution profile called function call profile [6] is generated for each test. Regression test selection can be risk based or design based [7]. The matter of selecting which test cases need be included in the test plan. Length of test cases [3] can be considered a criterion for test case selection. It looks the number of steps in a test case. Next one is a count based criteria. It is based on the counts of the number of parameter values that they cover. Gurinder Singh [8] et al.,proposed a new approach to reduce the cost of regression testing by the concepts of Bee Colony Optimization and Genetic Algorithm.

RTP techniques merely reorder the test cases based on the available time. In test case prioritization, test cases are ordered based on a certain criterion and test cases with highest priority are executed first to achieve a performance goal. RTP runs the test cases in the prioritized order until either the available time is exhausted or the faults on the changed area of code are detected.

A better understanding of the effects of time constraints could lead to improve prioritization techniques. When and when not to prioritize and the technique used for prioritization is very important [4]. Some techniques that are effective are 'total coverage' prioritization technique and 'additional block coverage' technique. Some other mechanisms include feedback and non-feedback techniques [4]. Gregg R.et. al. [9] describe several techniques for using test execution information to prioritize test cases for regression testing based on their total coverage of code components and their estimated ability reveal faults in the code components they cover. Rothermel et al., have researched and surveyed test case prioritization. For prioritizing a set of test cases [10] they considered nine approaches and the reported results measures the effectiveness of those approaches to improve the capability to reveal faults [13].

IV. PROPOSED WORK

The proposed system first generates test cases for the input program using test case generation technique. These test cases are executed and their execution profile is captured. Some parts of the input program are changed and this data is used to select some test cases from the overall test suite which comes in the limit of the changed code area.

Consider a program P and its set of test cases $T = \{T_{1,} T_{2\ldots} T_N\}$.

Program P is composed of M elements, $A = \{a_{1,} a_{2\ldots} a_m\}$.

A new version of the program P' is created by doing some changes to P. A function F is considered which assigns to a set of test cases, a scalar quantity that indicates how well it achieves a specific goal. The F function is ideally the number of detected faults. The sum of coverage and max-min criteria of the test cases determined. A S-function is used to define fault the detection capability. A S-function is defined as a measure of the code coverage provided by the elements of D. Assume that let $s_m(D)$ capture the fault detection capability of S as applied to $a_m \in A$.

$s_m(D) = \sum_{sn \in S} \delta m(sn)$

Where $\delta_m(s_n)$ captures the effectiveness of test case $s_n \in D$ as applied to element $a_m \in A$.

The S-functions for sum of coverage and max-min criteria are as follows.

$S_{sum}(D) = \sum_{am \in A} wm\ sm(D)$

And

$S_{min}(D) = \min w_m s_m(D)$

wherew$_m$ s are a set of weights emphasizing software elements that may be more important in the fault detection process. Ws$_m$ s account for changes since previous release. A third criterion is considered in the proposed system, i.e. **coverage size** of the test case.

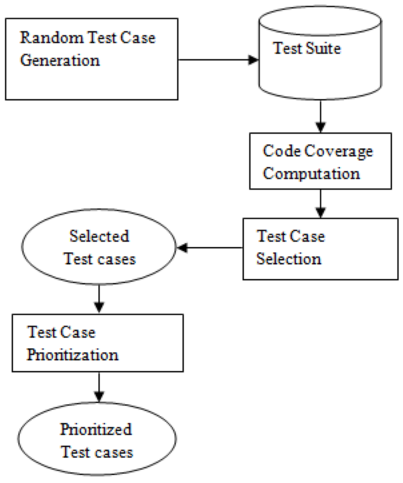

FIG. 1 ARCHITECTURE DESIGN

This criteria deal with the test cases whose coverage length of the code is small, but the area of code covered is critical to the whole program. Then using these three criteria an IP problem is formulated which is solved using linear programming and a solution set is obtained as the result. Using the jaccard distance the distance is measured with the coverage information. The next two criteria are maxavg and minavg. A voting scheme is performed in this solution set and the final solution is obtained. This final solution set is our selected test cases.

For more effectiveness, these test cases are prioritized using greedy algorithm which uses coverage data and priority measure for software elements. First it finds the set of bottleneck elements among all of the elements covered by at least one of the test cases. For each remaining test case, it finds the number of bottleneck elements that it covers. The all the software elements are sorted on ascending current coverage. In the next step, it finds a test case that covers the most number of bottle neck elements based on the earlier computation. In case of a tie, it finds the test case that provides the largest coverage for the first element in the ordering computed earlier. In case the tie does not fully break, it moves to the second software element in the ordering, and so forth until tie breaks. If there is still a tie after considering all of the elements, the test case with the highest index is selected. The current coverage values are updated to reflect the coverage of the selected test case. The test case is then added to the result set. These steps are repeated until all test cases are added to the result set.

In this, the system proposed first generates test cases for the input program using test case generation technique. These test cases are executed and their execution profile is captured. Some parts of the input program are changed and this data is used to select some test cases from the overall test suite which comes in the limit of the changed code area. From the generated test cases, the subsets of test cases are selected using the multiple criteria. In order to make the result effective, more than two criteria are considered. The criteria such as min-max, sum of coverage, coverage size, maximum average and minimum average are considered to increase the coverage of the code. Using the coverage information, the distance is measured by the Jaccard distance which significantly reduces the number of test cases. The selected test cases are then prioritized using the greedy algorithm which executes the highest priority first to make the results more efficient.

V. CONCLUSION

Test case selection and prioritization is a strategy of prioritizing and scheduling test cases with the help of which highest priority run in an order to minimizing the testing effort, time and cost. For efficient test case selection more than two criteria are considered. Test cases selected covered all the statements within the documents. So our result shows the effective priority of finding the faults by covering all the criteria's together. Our result helps software testers in their practice to test more test cases in an effective manner.

REFERENCES

[1] M. Harman, "Making the Case for MORTO: Multi Objective Regression Test Optimization," Proc. First Int'l Workshop Regression Testing, pp. 111-114, 2011.

[2] SiavashMirarab, SoroushAkhlaghi, LadanTahvildari, "Size-constrained Regression Test Case Selection Using Multi-Criteria Optimization". IEEE Transactions on Software Engineering, June 2011.

[3] Rene´e C. Bryce, SreedeviSampath, Atif M. Memon, "Developing a Single Model andTest Prioritization Strategies for Event-Driven Software", IEEE Transactions on Software Engineering, Volume: 37, Jan.-Feb. 2011.

[4] Hyunsook Do, SiavashMirarab, LadanTahvildari, Gregg Rothermel, "The Effects of Time Constraints on Test Case Prioritization: A Series of Controlled Experiments". IEEE Transactions on Software Engineering, volume 36, Sept.-Oct. 2010.

[5] ElmarJuergens, Benjamin Hummel, Florian Deissenboeck, Martin Feilkas, Christian Schlögel, Andreas Wübbeke, "Regression Test Selection of Manual System Tests in Practice", 15th European Conference on Software Maintenance and Reengineering, March 2011.

[6] Songyu Chen, Zhenyu Chen, Zhihong Zhao, BaowenXu, Yang Feng, "Using Semi-Supervised Clustering to Improve Regression Test Selection Techniques", Fourth IEEE International Conference on Software Testing, Verification and Validation, March 2011

[7] JussiKasurinen, OssiTaipale and Kari Smolander, "Test Case Selection and Prioritization: Risk-Based or Design-Based?", ESEM '10 Proceedings of the 2010 ACM-IEEE International Symposium on

[8] Empirical Software Engineering and Measurement Article No. 10.

[9] Gurinder singh, Dinesh gupta. 2013. An Intergarted to Test suite Selection Using ACO and Genetic Algorithm. IJARCSSE (Issue 6), pp. 1770-1778.

[10] Gregg Rothermel, Roland H. Untch, ChengyunChuand Mary Jean Harrold, "Prioritizing Test Cases for Regression Testing", IEEE Transactions on Software Engineering, 2001.

[11] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold, "Prioritizing Test Cases for Regression Testing," IEEE Trans. Software Eng., Oct. 2001.

[12] Hyunsook Do and Gregg Rothermel, "A Controlled Experiment Assessing Test Case PrioritizationTechniques via Mutation Faults", Proceedings of the IEEE International Conference on Software Maintenance, 2005.

[13] Gregg Rothermel and Mary Jean Harrold, "A Safe, Efficient Regression Test Selection Technique", ACM Transactions on Softw. Eng. And Methodology, 6(2): 173-210, 1997.