ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

Metric Based Approach to Identify Test Case Orderings

Palem Naresh
M.Tech Scholar in Department of CSE
JNTUA College of Engineering, Ananthapuram, India

Abstract - **In Testing we will evaluate a system or its components with the purpose to find whether the specified requirements are satisfied or not. The goal of Regression Testing is to make certain that improvements, patches or configuration changes have not introduced new faults in the source code. In regression testing, running each test case requires more time and assets. Test Case Prioritization (TCP) endeavors to plan test cases to accomplish objectives with higher scope or quicker fault recognition. Here, the investigation of utilization of Information Retrieval (IR) methods to enhance the viability of TCP, especially to test rarely tried code. Our approach considers the recurrence at which components have been tried, in extra to conventional scope data, adjusting these factors utilizing straight relapse displaying.**

Index Terms: Regression Testing, Test Case Prioritization, Information Retrieval

1. INTRODUCTION

The software testing is connected with the history of software engineering which helps to understand how the practice of testing has evolved. Testing focused on hardware, the program checkout and the debugging were seen as one and the same. Software Testing is an essential stage in software development. The superior purpose of software testing is to identify the quality of the software developed. It contains a set of activities mainly used to find errors in software. The errors need to be corrected before the product is released.

Regression Testing that takes care about the software changes. It mainly ensures to not affect any software changes in developed software. Regression Testing is the most critical processes in software testing because where it predicts the changes in the developed software. While the Regression Testing is going on than no new test cases are created, instead of that previously created test cases are executed.

Each of the Test Case Prioritization (TCP) is executed in a way that is being saves the cost and time effectively. Test Case Prioritization is more effectively used by the testers because of its advantages and easily understandable to make decisions on software. Most researchers stick to various methods for TCP in regression testing at any mode of employment during the execution of process.

The big advantage over the software developed programs that early fault detection happens when we introduced TCP. In Regression testing the bugs would found prior then debugging could be start earlier and hence bugs could be fixed faster. Many of the testers wish to increase their confidential ability of the system under test at a faster rate by archiving minimization of testing costs. Identification of high risk defects earlier and increases reliability of the software. This is the reason behind using the TCP plans over the software program.

II. LITERATURE REVIEW

In [1], the authors present a prioritization technique that achieves modified code coverage at the fastest rate possible. The advantage of Test Case Prioritization is, i t can utilize the information about the changed code. Further it could be prioritized for the test cases.

In [2], the authors presents algorithm is used to execute the modified lines of code with minimum number of test cases and the priority associated with the modified lines. The priority value of modified lines is given by the tester according to some rules. These rules can be change according to tester and as well as requirement of the program which is to be under regression testing.

In [3,4] the authors present the regression testing to ensure bug fixing. Efficient methods of selecting small subsets of regression test sets.

In [5], the authors proposed the effective cluster analysis for execution of profiles which helps to find the failures in test cases. The clustering metrics provides the huge range

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

of industrial profiles features which are most effective to utilize.

In [6], the authors present tools to generate automatic test suites with help of one-test-at-a-time greedy algorithms. They used an Walk-Through algorithm.

In [7], the authors proposed a Test Case Prioritization techniques schedule for test cases which to maximize the fault detection rate. Each of the test case the catalog of prioritization techniques are available for effective results and quick execution.

In [8], the authors proposed to attempt the improvement in regression testing. They tried to improve the fault detection rate with the help of test suites and based on total coverage they are taken ordered test cases which are very important for execution.

In [9], proposes a novel test suite reduction based on set theory and data mining technique clustering. In this proposed technique, determine the intersection between requirements of branch coverage criteria for the set of test cases. This approach reduces the time wasted in testing unnecessary test cases.

In [10], the authors proposed a modified condition/decision coverage verification technique which to uncover safety faults for future. A very new kind of algorithms is used for test suite reduction and prioritization that leads to cost effective.

To improve the test suites fault detection rate we use the metric called Average percentage of fault detection (APFD) was developed to measure the weighted average of the test suite. Generally, the APFD values ranges from 0 -100, higher the numbers imply faster fault detection rates are accounted at the time of execution.

III. PROPOSED METHOD

The aim of the proposed method is to enhance the fault detection rate by using coverage-based TCP techniques. Towards this, we combine traditional coverage scores with similarity scores produced using IR. We adopt the linear regression model with two features to automatically set weights on the coverage and similarity scores of test cases. The regression model adjusts the weight of each feature such that a feature having little correlation with faults in a program has little weight. In Fig.1 shows an overall framework for building a Linear Regression Model with IR and coverage information (IRCOV). The process consists of three phases: validation phase, training phase and test phases. In the validation and training phases, the faults that were detected in previous versions can be used as validation and training data.

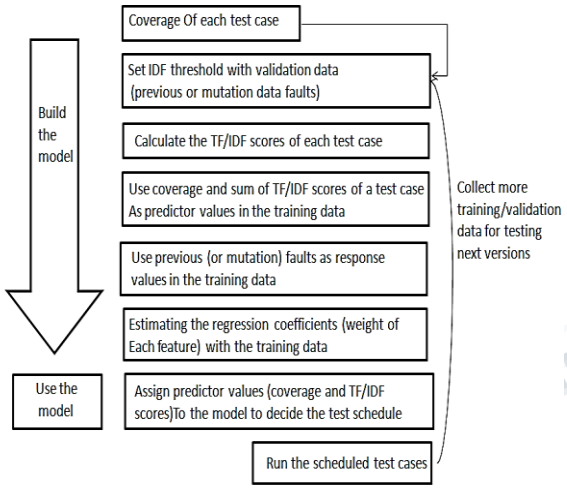

Fig.1 Proposed Framework with overall process for building a linear regression model with Information Retrieval (IR) and coverage information (IRCOV).

In the validation phase the IDF threshold could be determined easily. Next, in the training phase, weight values are calculated in a manner that minimizes the learning error (difference between estimated and real values). In the testing stage, the scheduled test cases are executed in a way that the fault detection effectiveness of the test cases is measured.

An IDF threshold is the maximum number of test cases considered when assigning an IDF score to a code element. This property is required because a test case having high coverage can cover many small IDF elements. Since, the similarity scores of an each test case is the sum of the (TF × IDF) scores of code elements, the test case may have a large similarity score. This is not relevant to the objectives, as focus is on detecting faults in less tested elements faster.

To avoid the foregoing problem, an IDF threshold is set. The IDF threshold is decided by the validation data. The validation data consist of faults that existed in previous

ISSN (Online) 2394-2320

International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)
Vol 4, Issue 9, September 2017

versions and information about which test cases detected the faults. For a ratio from 0.1 to 1.0, the threshold is set by the number of test cases × the ratio. With each threshold, the regression model is trained with the validation data. Only code elements whose DF is not above the threshold can obtain an IDF value. After the models with each ratio are built, their learning errors are compared. The ratio that leads to the minimum learning error is selected for the IDF threshold. When there are multiple candidate ratios, the approach selects the smallest ratio.

The model is trained with validation data in the same manner as with training data. The DF score of a word in IR is the number of documents containing the word. In the context of TCP, the DF score of a covered code element is related to the number of test cases exercising the element. As the number of test cases covering an element increases, the DF score of the element increases. IDF is the inverse of DF. Thus, IDF scores can be used to find unique code elements.

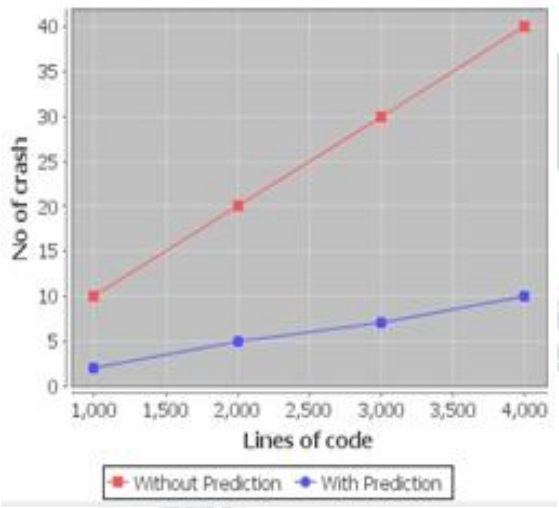

Fig.2 Performance Comparison Graph

Test cases that cover many high IDF elements may have high capacity to find defects in the program because they cover many unique elements. The similarity score between a test case (document) and changed elements (query) can be calculated by the sum of the TF-IDF scores of the common elements between covered elements of the test case and elements in the query. The similarity score indicates how much the test case is related to the modifications.

In Fig.2 shows the comparison graph between the crash prediction and without crash prediction results. The crash reduction graph is drawn based on lines of codes blue color on x- axis and the no of crash occurred red color on the y- axis. This graph is generated between number of crash and lines of code.

IV. CONCLUSION AND FUTURE WORK

The main contribution of this work is to order the test cases to improve the performance of regression testing. The proposed methods are, first the clustering method and then the multiple modular based approach as second with the use of TCP. Clustering method requires co-ordination of developers with testing team, but modular based approach is independent of developers. In the clustering method, the test cases are taken for different coverage's like fault coverage, code coverage and requirement coverage.

The condition coverage are considered, prioritized and clustered. In the modular based approach, all the test cases are equally divided as a set called module and prioritized using greedy algorithm. After this all test cases are merged and then prioritization is done by using 2-optimal greedy algorithm.

In order to implement the proposed method, a "Student Academic Program" which includes course registration, continuous assessment marks, end semester test marks and grading etc, was developed using PHP, MYSQL and executed in quick test professional software testing tool. While executing the performance of clustering approach is compared with non-clustering scheme and the performance of multiple modular based approaches is compared with non-modular based scheme.

The TCP using clustering approach executes test cases in 20 seconds with the help of automation testing tool. So the fault detection rate is high with minimized time when comparing with non- clustering test case prioritization. The non-clustering test case prioritization approach takes 45 seconds to execute the same number of test case list. So the clustering approach yields good results when compare with non-clustering method.

The multiple modular based Test Case Prioritization method yields good results with increasing fault detection rate when compare with non-modular TCP method. The entire process is evaluated with the help of Average Percentage of Fault Detection (APFD) which the faults are detected in that particular cluster and the execution time.

The APFD for multiple modular approaches is 82% and the execution time is 27 seconds. When compared to non-modular based approach of the APFD is 70% and execution time is 40 seconds. By watching the results it's clearly reveals that the use of Test Case Prioritization using clustering approach provides the effective prioritization in terms of higher rate of fault detection and takes less execution time when compared to the multiple modular based approach.

REFERENCES

[1] Aggrawal, Yogesh Singh, Kaur "code coverage based technique for prioritizing test cases for regression testing "on ACM SIGSOFT Software Engineering Notes, September 2004, Volume 29.

[2] Amrit Jothi, Yogesh Kumar, Sharma, Ashish Bagla, Pandey "Recent Priority Algorithm In Regression Testing " at International journal of technology and knowledge management on December 2010,volume 2,pp.391-394.

[3] Agrawal H, Horgan JR, Krauser EW, London SA. "Incremental regression testing". Proceedings of the International Conference on Software Maintenance (ICSM 1993), IEEE Computer Society, 1993, pp. 348–357.

[4] Eric Wong, Joseph R. Horgan, Saul London, and Hira A. Bellcore. "A study of e ective regression testing in practice". In Proceedings on IEEE International Symposium on Software Reliability Engineering (ISSRE), pages 264–274, 1997.

[5] Dickinson, D. Leon, and A. Podgurski. "Finding failures by cluster analysis of execution profiles". In ICSE '01, pages 339 - 348, Washington, DC, USA, 2001.

[6] Bryce and C. Colbourn. Test prioritization for pairwise interaction coverage. In A-MOST '05, pages 1 - 7, 2005.

[7] Gregg Rothermel, Roland H. Untch, Chengyun Chu, Jean Harrold "Test Case Prioritization: An Empirical Study" in Proceedings of the International Conference on Software Maintenance, Oxford, UK, September, 1999.

[8] Gregg Rothermel, Roland H. Untch, Chengyun Chu, Mary Jean Harrold "Prioritizing Test Cases For Regression Testing " in IEEE transactions on software engineering, vol. 27, 10, october 2001.

[9] Jyothi kameswari, vinod kumar reddy, amudala sai kiran, sri varun "Novel Techniques for Test suite reduction" international journal of science and advanced technology (ISSN 2221-8386) ,volume 1 octobar 2011.

[10] Jones, M.Harrold. "Test-suite reduction and prioritization for modified condition/decision coverage". In Proceedings of the International Conference on Software Maintenance, Nov 2001.