Testlist Automation Technique for Quality Enhancement of Allegro Design Workbench Tool

Sekhada Jayadipkumar Kantilal[1]

[1]Department of Electronics and Communication Engineering

[1]B.H.Gardi college of Engineering, Rajkot – 360 001, Gujarat, India

Abstract— In these days we required Electronic design automation(EDA or ECAD) software tools for designing electronic systems such as printed circuit boards and integratedcircuits. These tools work together in a design flow that chipdesigners use to design and analyze entire semiconductorchips. Silicon Package Board(SPB) Front End Allegro Design Workbench (ADW) and Library Workbench (LW) are well known EDA tools, which is useful to design printed circuit boards and to maintain an electronics library component. Software testing is automated to increase the test coverage, improve efficiency in finding defects and enhance the effectiveness of the development process. In large, complex and time critical projects where code needs to be tested frequently and repeatedly in the same areas of application, test automation increases efficiency and reduces test cycles. We will discuss the various aspects of how to achieve regression testing automation and techniques to run testcases. And run testcases from different locations.

Key words: Regression Testing, Regression Test Selection, Software Maintenance, Test Prioritization

I. INTRODUCTION

The process of using special automation software to control the execution of software tests to compare the outcome with the expected results is called Automation Testing. Test automation is normally used for projects that require frequent testing of the same sections of code when the requirements do not change frequently, when the applications have to be tested for load and performance with several users and finally when the time available for testing is short. Software Testing should not be a distinct phase in System development but should be applicable throughout the design development and maintenance phases. Software testing is often used in association with terms verification & validation [8, 9].

During the maintenance of a software system or as the software evolves, the regression testing is the expensive but definitely necessary task. There have been a lot of researches on regression testing [1, 3, 4, 5]. Since regression testing is quite expensive, the researches mainly focus on how to reduce such cost. The topics include test selection, minimization, prioritization, etc. In [2], Onoma emphasized such problem, and gave one example that in practice some companies will rerun all test cases in regression testing while the research communities are still working on minimizing the test cases.

The rest of the paper is organized as following: Section 2 is a briefly review of the current regression testing research literatures. In Section 3, the current popular commercial tools will first be introduced, and then some case studies are presented as they are some practices to apply the regression testing technology. Some interesting observations have been found according to Section 2 and Section 3, and they are presented in Section 4. Conclusions are given In Section 5.

II. REGRESSION TEST SELECTIONS

Regression testing is acknowledged to be an expensive activity. It consumes large amounts of time as well as effort, and often accounts for almost half of the software maintenance costs [6, 7]. The extents to which time and effort are being spent on regression testing are exemplified by a study [10] that reports that it took 1000 machine-hours to execute approximately 30,000 functional test cases for a software product. It is also important to note that hundreds of man-hours are spent by test engineers to oversee the regression testing process; that is to set up test runs, monitor test execution, analyze results, and maintain testing resources, etc. [10]. Minimization of regression test effort is, therefore, an issue of considerable practical importance, and has the potential to substantially reduce software maintenance costs.

Regression test selection (RTS) techniques select a subset of valid test cases from an initial test suite (T) to test that the affected but unmodified parts of a program continue to work correctly. Use of an effective regression test selection technique can help to reduce the testing costs in environments in which a program undergoes frequent modifications

So, for that few techniques for regression test selection:

A. Prioritization Techniques:

Regression testing is expensive and in order to reduce such cost, researchers have done many works other than test selection technologies. [5, 15, 16] address the problem of the test case prioritization technology. They order (prioritize) the test cases by certain measures. Then in the regression testing cycling, the test cases will be used to test the modified program P' according to the order, so that the "better" test cases can run first. The goal of the prioritization is to increase the rate of fault detection (how quickly the test suite can detect the faults during the test process), or, increase the rate of code coverage (how quickly the test suite can increase the coverage of the program). For example, let t1, t2, t3 be the three test cases. Also assume that t1 has the coverage of 75%, t2 has the coverage 25% and t3 has the coverage of 50%. According to the second goal, the result of apply such technology is to run the test cases in the order of t1, t3, t2. And similarly, according to the first goal, the order of the three test cases will depend on their ability to expose the fault.

Computation of proposed practical prioritization factors such as (1) customer allotted priority, (2) developer observed code execution complexity, (3) changes in requirements, (4) fault impact (5) completeness and (6) traceability , is essential for prioritizing the test cases because they are used in the prioritization algorithm.

All rights reserved by www.ijsrd.com

Weights are assigned to each test case in the software according to these factors. Then, test cases are prioritized based on the weights assigned. [11]

1) Customer Allotted Priority (CP):

It is a measure of the implication of a requisite to the customer. The values of each need are assigned by the customers. The values vary from 1 to 20, where 20are used to identify the highest customer priority. So, improving customer's fulfillment imposes the initial testing of the highest priority needs of the customers. . It has been proved that customer Allotted value and satisfaction can be improved by fixing on customer needs for development.

2) Fault Impact of Requirements (FI):

It allows the development team to distinguish the requirement that had customer reported failures. Developers can recognize requirements that are expected to be error free by using the prior data collected from older versions as a system evolves to several versions. The number of in-house failures and field failures determine the fault impact of requirements.

3) Developer observed Code Implementation Complexity (IC):

It is an individual measure of the complexity expected by the development team in implementing the necessity. First every necessity is evaluated. The developer assigns a value from 1 to20 on the basis of its implementation complexity and a higher complexity is implied by a larger value. Large number of faults that could be occurs in a requirement that has high implementation complexity.

4) Changes in Requirements (RC):

It is a degree assigned by the developer in the range of 1 to 20 for indicating the number of times arequirement is changed in the development cycle with respect to its origin date. The volatility values for all the needs are expressed on a 20-point scale is the need is altered more than 20 times. The number of changes for any requirement I divided to the highest number of changes for any requirement among all the project requirements yields the change in requirement RI of that requirement i. If the ith requirement is changed M times and N is the maximum number of requirements then the requirement change of i, RI can be calculated in Eqn (1) as

$$RI = (M/N) \times 10 \quad (1)$$

The errors introduced in the requirement level are approximated to 50% of all faults detected in the project. The change in requirements is the major factor attributable to the failure of the project

5) Completeness (CT):

This part indicates what is needed as per the requirement for a function to be executed, the rate of success, the litations to be followed for the function is to be executed and any limitation which manipulate the expected solution for example the boundary constraints. The consumer assigns value from 1 to 20.When the condition is selected for reuse after scrutinizing the completeness of each requirement into consideration, customer satisfaction like stronghold of the software response to the user request can be enhanced.

6) Traceability (TR):

Relation between requirement and assessment can be calibrated by means of Traceability. Defining whether a requirement is properly tested is cumbersome for evaluators. If the test cases are not concerned to individual requirement, the common problem reported is scarcity of traceability, hence poor traceability leads to failure and going beyond the desired limit of the project. The evaluator allots value in the range from 1 to 20, after assessing individual requirement for the concerned traceability and the standard of software can be improved by opting the traceability of the requirement into consideration is chosen for subsequent usage.

B. Coverage Techniques:

Test coverage measures the amount of testing performed by a set of test. Wherever we can count things and can tell whether or not each of those things has been tested by some test, then we can measure coverage and is known as test coverage.

The basic coverage measure is where the 'coverage item' is whatever we have been able to count and see whether a test has exercised or used this item. [17]

$$\text{Coverage} = \frac{\text{Number of coverage items exercised}}{\text{Total number of coverage items}} \times 100\%$$

There is danger in using a coverage measure. But, 100% coverage does *not* mean 100% tested. Coverage techniques measure only one dimension of a multi-dimensional concept. Two different test cases may achieve exactly the same coverage but the input data of one may find an error that the input data of the other doesn't.

C. Minimization Techniques

Minimization techniques are similar to Coverage techniques except that they select minimum set of testcases. Minimization techniques aim to identify redundant test cases and to remove them from the test suite in order to reduce the size of the test suite

III. TESTLIST TECHNIQUE

In this we will discuss the method to run testcases from different locations. We have some testcases which are failed, and these testcases are failed at different locations. So, we have to check reasons behind that failed testcases. For that we have to run only those failed testcases.

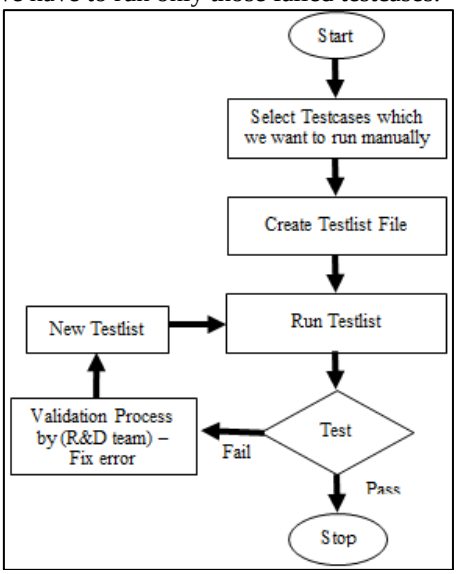

Fig. 1:

All rights reserved by *www.ijsrd.com*

By this technique run testcase from same location. For that we create one *.txt text file, which contain path for each testcase.

IV. CONCLUSION

The objective of this chapter is to develop a new technique to improve the cost-effectiveness of the testing techniques. By this techniques we don't required to run testcases again which are already passed and we ran those testcases at its present location. So, this technique is useful to reduce time space and cost.

REFERENCES

[1] G. Rothermel and M. J. Harrold, "Analyzing regression testselection techniques," IEEE Transactions on Software Engineering, V.22, no. 8, August 1996, pages 529-551.
[2] K. Onoma, W-T. Tsai, M. Poonawala, and H. Suganuma, "Regression testing in an industrial environment," Communications of the ACM, 41(5):81–86, May 1998.
[3] K. Onoma, W-T. Tsai, M. Poonawala, and H. Suganuma, "Regression testing in an industrial environment," Communications of the ACM, 41(5):81–86, May 1998.
[4] G. Rothermel and M. J. Harrold, A Safe, "Efficient regression test set selection technique," ACM Transactions onSoftware Engineering and Methodology, V.6, no. 2, April 1997,pages 173-210.
[5] S. Elbaum, .A. Malishevsky and G. Rothermel, "Test case prioritization: a family of empirical studies", IEEE TransSoftware Engg. , vol. 28, no. 2, pp. 159-182, Feb. 2002.
[6] . H. Leung and L. White. "Insights into regression testing," In Proceedings of the Conference on Software Maintenance, pages 60–69, 1989.
[7] G. Kapfhammer. "The Computer Science Handbook, chapter on Software testing," CRC Press, Boca Raton, FL, 2nd edition, 2004
[8] Rajender Kumar, Maurya V. N. and Maurya Avadhesh Kumar, "A cost- benefit model for evaluating regression testing technique," International Journal of Software Engineering and Computing, Serials Publications, New Delhi, Vol.4, No. 2, pp. 84-89, 2012, ISSN: 2229-7413
[9] Samtinger Johannes, "Software Engineering with Reusable Components," Springer- Verlag, March, 1997
[10] H. Do, S. Mirarab, L. Tahvildari, and G. Rothermel. "The effects of time constraints on test case prioritization: A series of controlled experiments," IEEE Transactions on Software Engineering, 36(5):593–617, September 2010.
[11] Thillaikarasi Muthusamy, Seetharaman.K "A new effective test case prioritization for regression testing based on prioritization algorithm,"International Journal of Applied Information Systems (IJAIS) – ISSN: 2249-0868, Volume 6– No. 7, January 2014
[12] Chandana Bharati, Shradha Verma "Analysis of different regression testing approaches", International Journal of Advanced Research in Computer and Communication Engineering Vol. 2, Issue 5, May 2013
[13] L.UmaRani, R.Pradeepa "Prioritization based test case generation using regression testing", International Conference on Research Trends in Computer Technologies (ICRTCT - 2013)
[14] Seifedine Kadry "A new proposed technique to improve software regression testing cost", International Journal of Security and Its Applications Vol. 5 No. 3, July, 2011.
[15] G. Rothermel, R. Untch, C. Chu, and M.J. Harrold. "Prioritizing test cases for regression testing," IEEE Transactions on Software Engineering, 27(10):929–948, October 2001.
[16] W.E. Wong, J.R. Horgan, S. London, and H. Agrawal. "A study of effective regression testing in practice," In Proceedings of the Eighth International Symposium on Software Reliability Engineering, pages 230–238, November 1997.
[17] http://istqbexamcertification.com/what-is-test-coverage-in-software-testing-its-advantages-and-disadvantages/

All rights reserved by *www.ijsrd.com*