DOI: http://dx.doi.org/10.26483/ijarcs.v9i1.5424

ISSN No. 0976-5697

Volume 9, No. 1, January-February 2018

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

CHALLENGES OF REGRESSION TESTING: A PRAGMATIC PERSPECTIVE

Sandeep Dalal
Assistant Professor,
Maharshi Dayanand University,
Rohtak, India

Sudhir
Research Scholar,
Maharshi Dayanand University,
Rohtak, India

Kamna Solanki
Assistant Professor,
Maharshi Dayanand University,
Rohtak, India

Abstract: The prevailing scenario of ever increasing dependency of human beings on software applications has built pressure on software organizations to produce quality software. The quality of software is determined by many factors. One of the vital factors in deciding the software quality is optimized usage of testing tools and techniques employed during regression testing. Extremely high complexity of regression testing makes it necessary to utilize optimized way of running a selected and minimized test suite. Still, various challenges like redundancy, repetition ratio, Recurrence ratio or missing functionalities during regression testing are common. This paper discusses about the problems and challenges encountered during regression testing.

Keywords: Regression Testing; Test Case Prioritization; Software testing

I. INTRODUCTION

The significance of software can be easily perceived as it emerged as a powerful tool which finds its application in every field ranging from mobile phones to medical treatment covering almost all electronic gadgets on earth up to space i.e. satellite communication. Thus, the software has made a way of changing and facilitating the life of humans. If it has such an involvement, it must be developed in a well-defined and sequential way; otherwise, its quality will be compromised leading to an unreliable software. The software is thus developed after undergoing various phases of SDLC.

To ensure the quality of a software, testing is performed. The various testing techniques find numerous faults/errors, which are thus removed by various debugging approaches. One of the approaches to testing is exhaustive testing. It has a limitation that it can be performed only on very small programs. Running exhaustive testing on large programs keeping in mind time and cost constraint is not a feasible idea. Regression testing is about running the entire test ensemble again to ensure that amendments do not negatively affect the system.

Verification and validation of a software product is an essential step towards building quality software. Verification and validation process is also known as software testing. Software testing is performed both statically (verification) and dynamically (validation) to ensure that the software meets the customers' demands and expectations. [1][2][3]. "Software testing is a process of executing a program with the intent of finding the software bugs".

Test cases are executed during software testing to find the failures (bugs). A test case is a collection of set of input, behavior and output conditions. A test case with same expected and observed output is considered as "pass". Any mismatch or discrepancy between expected and observed output makes a test case "fail"[4][5][6]. To manage abundant test cases in software, related test cases are often clubbed together as a

single test suite. A test suite is a collection of related test cases as shown in figure1.

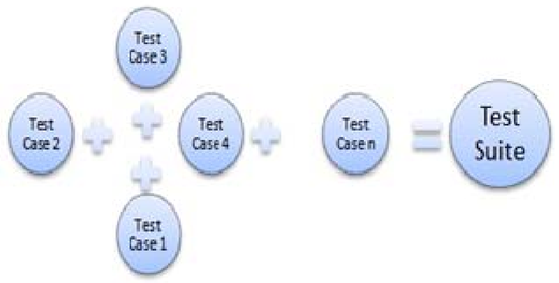


Fig 1: Test Cases and Test Suite

Software testing phase consumes the highest resources in terms of time, resources and efforts. It is the most critical and time consuming process during software development. The testing of software is necessary evil for verifying the software quality[7][8]. Software testing typically requires 40 - 50% of development efforts. It is the most significant phase of the software development life cycle [9][10]. Software testing still can show the presence of bugs or software failures, but it can never confirm regarding the absence of bugs because it is nearly impossible to completely test a software exhaustively due to resource and time constraints.

A. *Regression Testing*

Regression Testing is necessary during software development as the developers alter the code to rectify the bugs that have been reported during software testing; and regression testing ensures to capture and mitigate the undesired side-effects of code amendments.

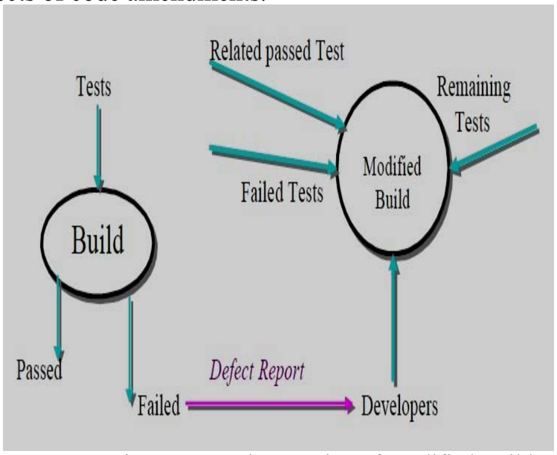


Fig 2: Regression Testing of Modified Build

Figure 2 describes the importance of regression testing and depicts that the testing team needs to re-run the test suite after code amendments to make sure that the previously failed test cases are now passed after bug fixation (or code change) and status of previously passed test cases is still "pass" so as to capture the undesired side effect of software code change [5][9]. Regression Testing makes sure that "fixing one bug do not introduce several new bug" due to code change as shown in fig. 3 [11][12][13][14][15][16].

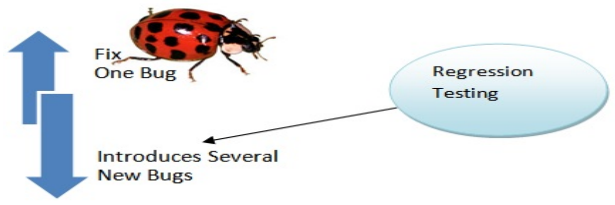


Fig. 3. Need of Regression Testing

"Regression testing is selective retesting of a system or component to verify that modifications have not caused undesirable effects and that the system or component still complies with its specified requirements" [4] [5].

Test case prioritization approach increases test feasibility in the testing of software [3]. Software evolves with time, so the size of software test suite also increases which often makes it costly to execute. Numerous researchers have shown that regression testing is a costly process and therefore it requires most of the collective expenditure of the software.

A.1 Test suite minimization

This approach intends to eliminate the test cases, which becomes redundant with regard to coverage of some set of program requirement and therefore decreasing the number of test cases in a regression test suite. It clearly states that only a subset of the test suite is actually economical to use. Minimization is at times called "test suite reduction" denoting that the exclusion of test cases are everlasting.

A.2. Test case selection

This approach targets to pick the test cases from the original test suite, that focuses on testing the customized part of the software program. It does not remove test cases, rather it filters the test cases which are related to the customized portion of source code.

A.3. Test case prioritization

Prioritization approach is followed because execution of all the test cases is not feasible due to resource constraints. In this, the test cases are ordered such that those with higher priorities are run earlier than those with lower priorities based on some criterion [10]. Prioritization techniques are usually preferred because prioritization deals with the original test suite and no test cases are eliminated from the initial test suite.

B. Complexity of Regression Testing

The complexity of regression testing can be understood from the fact that it is impossible to completely test a software using a test suite collection even once due to resource constraints. Regression testing requires to re-run the entire ensemble of the test suite after every new build of software code is launched due to code change for fixing software bugs. Hence, one can easily judge that running the entire test suite again for every build is not feasible.[17][18][19][20]

Hence, optimization of test suite is greatly required for regression test suite execution. There are three ways of optimizing a test suite for regression testing: selection, minimization and prioritization. Many researchers have developed techniques for regression test optimization

II. CHALLENGES IN REGRESSION TESTING

Regression testing is often performed to capture the "Return of a bug". A number of challenges are associated with regression testing. Some of them are listed below

- Fairly large size of test suite after every successive regression run is a big challenge which needs to be optimized using different tools and techniques.
- Deciding the frequency of the regression test suite run is a major challenge. A regression test suite can run after a group of bug fixes, or after every new build, or after every modification.
- Minimization or optimization of the original test suite for regression can decrease the fault detection capacity or can negatively affect the code coverage achieved and many other factors. So, optimization is still an open issue.
- Determining the "Stoppage Criteria"/"Entry and Exit Criteria" for a regression test suite is still an open challenge. As almost all the software's are released with known bugs due to market pressure, deciding the stopping criteria plays an important role[21][22][23][24].
- Maintaining a balance between the ever-growing test suite size and limited constraints it-self is the biggest challenge in regression testing.
- Choosing a right automation tool based upon the nature of software application and availability of resources is one of the common challenges [25][26][27][28].

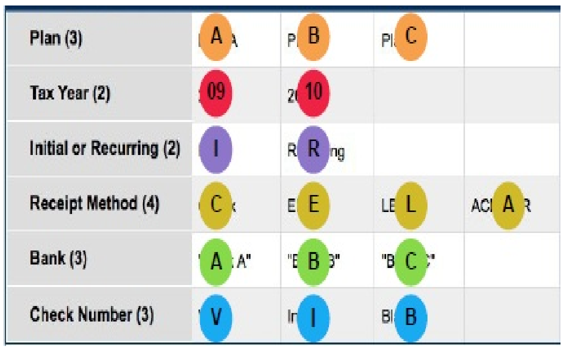
A. Minimizing Recurrence Ratio

This metric is used to measure the quality of the regression testing procedure. Value of this metric depicts the percentage of those bugs (failures) whose fixation introduced some "newer" bugs in the software system that were not present earlier.

It measures the level of quality of the regression testing as how well the bugs have been fixed in response to the bug reports. Fixing a bug must be performed by altering software code in such a way that it introduces least or no negative impact on rest of the "passed" test cases. Recurrence ratio determines the degree to which previously correct functionality is being negatively affected by the altered software code for bug fixation [29][30][31][32][33].

B. *Minimizing Repetition Ratio*

Most of the test cases that constitute a test suite are actually testing some kind of functionality. The basic challenge during regression testing is that there are many test cases that are testing the same functionality or path again and again. This leads to drastic wastage in the time and resources. Hence, utmost care must be taken to ensure that redundancy in terms of repetition of functionality test must be avoided by removing un-necessary test cases [34][35][36].



Plan (3)	A	B	C	
Tax Year (2)	09	10		
Initial or Recurring (2)	I	R		
Receipt Method (4)	C	E	L	A
Bank (3)	A	B	C	
Check Number (3)	V	I	B	

Fig. 4. Example1 for Repetition Ratio

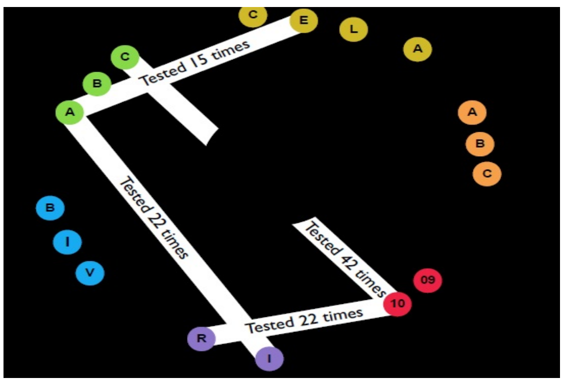


Fig. 5. Repetition Ratio

C. *Maximizing Functionality Coverage*

Test cases are executed again during regression testing after every new build is launched. Optimization of the test suite to be executed can cause loss of functionality coverage or code coverage or can minimize the defect detection capability of a test suite [37][38][39][1][2]. So, it must be made sure that a regression test suite that has been optimized does not compromise with functionality coverage of the original test suite as shown in fig. 6.

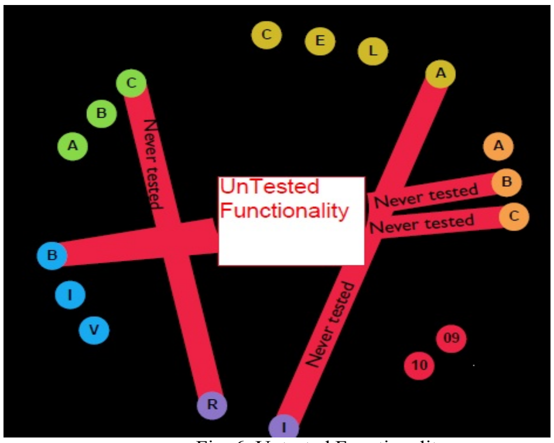


Fig. 6. Untested Functionality

D. *Focusing Defect Cluster Based Testing*

This is one of the most important factor which can play a vital role in regression testing. As per the Pareto Rule of software testing: "Not all the software modules are equally buggy. There are nearby 20 percent of software modules which contributes towards 80 percent of the software bugs." Therefore, regression test suite optimization must ensure to focus on the modules where defect clusters are highest[2][3][4][5].

E. *Avoid Pesticide Paradox*

New test cases must be added to the existing original test suite to avoid the "pesticide paradox" rule for software testing which states that if the same type of test cases is executed repeatedly, then they can no longer find new bugs or defects. So, test case optimization during regression testing must ensure to enhance the fault detection capacity by exercising focus on different parts of software system.

F. *Context Based Regression Optimization*

Every software application is different from other software application. Testing different types of application requires different type of skills, tools, technology and strategy. So, testing is actually context dependent. Testing a safety critical application requires completely different strategy than testing a commercial website[5][6][8].

G. *Absence of Defects Fallacy*

Software having least number of defects or bugs can still becomes useless for end user as it does not provide the intended functionality intended to user. Therefore, testers must make sure that focus for testing must be utilized constructively for testing only the correct functionalities[5][7][8].

H. *Update Regression Pack Regularly*

Test suite collection which is executed after every software build or update is known as regression pack. Test cases in a regression pack must be updated regularly as per the changing user requirement.

I. *Performance Measurement*

Time to time measurement of the performance of the test procedures or tools employed for regression testing is a big challenge as there are not sufficient metrics available to measure the performance of a regression technique. Most commonly used metrics are APFD ("Average Percentage of Faults Detected"), PTR ("Percentage of Test Suite Executed for Complete Fault Coverage") etc. [24][26].

J. *Balancing Between Large Test Suite Size and Limited Resources*

The major challenge in regression testing is to maintain a balance between the ever-increasing size of the test suite and the limited resources as shown in fig. 7. As the size of the test suite keeps on growing at a faster pace with every new build and the time to achieve the quality testing keeps on decreasing due to hard deadlines [27][22].

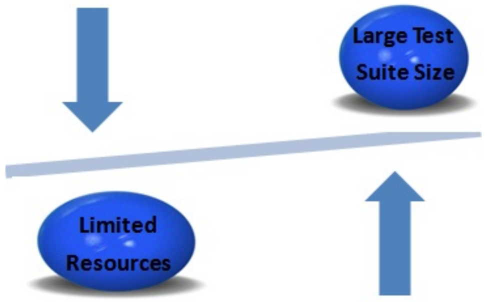


Fig. 7. Balancing Resources and Test Suite

III. CONCLUSION

The paper discussed about the major challenges and issues during regression testing in a pragmatic way. Any regression testing technique or tool does not provide any golden solution to all the challenges associated with regression process. The study of literature related to regression testing revealed some of the critical and vital challenges and issues, which have been described in detail in the paper.

IV. REFERENCES

[1] K. Onoma, W.T. Tsai, M. Poonawala and H. Suganuma. "Regression testing in an Industrial Environment". Communications. Of ACM, Vol.41, No. 5, pp 81–86, 1988.

[2] B. Beizer. "Software Testing Techniques". Van Nostrand Reinhold, New York, NY, 1990.

[3] H. Leung and L. White. "Insights into Regression Testing". Proceedings of IEEE International Conference on Software Maintenance, pp 60–69, 1989[Online].

[4] G. Myers. "The Art of Software Testing", NY,USA: John Wiley, 1979

[5] A. P. Mathur. "Foundations of software testing". China Machine Press, 2008.

[6] C. Kaner, J. Bach, and B. Pettichord,. "Lessons Learned in Software Testing". John Wiley & Sons, 2008. Paul C. Jorgensen." Software Testing:A Craftsman's Approach", CRC Press, 4th Edition..

[7] http://www.softwaretestinggenius.com/

[8] http://www.cs.swan.ac.uk/~csmarkus/CS339/presentations/20061124_Schlingloff_Testing_Introduction.pdf

[9] https://venkatreddyc.wordpress.com/2007/03/24/taking-on-testing-triangles-a-classic-excercise/

[10] W Wong, J. Horgan, S. London and H. Agrawal. "A study of effective regression testing in practice". Proceedings of IEEE Eighth International Symposium on Software Reliability Engineering. pp. 264-274, 1997.

[11] S. Yoo and M. Harman. "Regression Testing Minimisation, Selection and Prioritization : A survey". Journal of software testing , Verification and Reliability, Vol. 22, No. 2, pp. 67-120, 2012.

[12] Z. Li, M. Harman and R. M. Hierons. "Search algorithms for regression test case". IEEE Transactions on Software Engineering, San Francisco, CA, USA, pp. 225-237, 2007.

[13] Y. Singh, A. Kaur and B. Suri. "Regression Test Selection and Prioritization Using Variables: Analysis And Experimentation", New Age International Publishers, New Delhi, pp. 1-15, 2008.

[14] P.R. Srivastava, A. Vijay, B. Barukha, P. S. Sengar, and R. Sharma. "An Optimized technique for Test Case Generation and Prioritization Using Tabu Search and Data Clustering". Source available on DBLP and SCOUPS.

[15] D. Jayamala and V. Mohan. "Quality Improvement and Optimization of Test cases- A hybrid genetic algorithm based approach". ACM SIGSOFT Software Engg. Notes, Vol. 35, No. 3, pp. 1-14, 2010

[16] S. Elbaum, A. Malishevsky and G.Rothermel. "Test case prioritization: A family of empirical studies". IEEE Transactions on Software Engineering, Vol. 28, No. 2, pp 159-182, 2002

[17] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: a family of empirical studies," IEEE Transactions on Software Engineering, vol. 28, no. 2, pp. 159–182, 2002.

[18] S. Raju, G.V. Uma,"Factors oriented test case prioritization technique in regression testing using genetic algorithm", (2012)

[19] Gaurav Duggal and Bharti Suri, "Understanding regression testing techniques ", (2008)

[20] Hareton and Lee White, "Insights into regression testing", (1989)

[21] Hyuncheol Park, Hoyeon Ryu, Jongmoon Baik "Historical value-based approach for cost cognizant test case prioritization to improve effectiveness of regression testing", (2008)

[22] Alexey G. Malishevsky, Gregg Rothermel, and Sebastian Elbaum, "Modelling cost-benefits tradeoffs for regression testing techniques", (2002)

[23] A. Askarunisa, L.Shanmugapriya, N. Ramaraj, " Cost and Coverage Metrics for Measuring the Effectiveness of Test Case Prioritization Techniques", (2009)

[24] K. Solanki, Y. Singh, S. Dalal."Test Case Prioritization: An approach based on modified ant colony optimization". Proceedings of IEEE International Conference on Computer, Communication and Control. 2015 Sept; Indore: India .Available at IEEE-xplore Digital Library and SCOPUS.

[25] K. Solanki, Y. Singh, S. Dalal."Experimental Analysis of m-ACO Technique for Regression Testing". Indian Journal of Science and Technology, vol. 9, no. 30, DOI: 10.17485/ijst/2016/v9i30/86588.

[26] Dalal S, Chhillar, RS. Empirical study of root cause analysis of software failure. ACM SIGSOFT Software Engineering Notes. 2013 Jul, 38 (4), pp. 1-7.

[27] Dalal S, Chhillar RS. Software Testing-Three P Paradigm and Limitations. International Journal of Computer Applications. 2012 Sep, 54 (12), pp. 49-54.

[28] Thangavel Prem Jacob and Thavasi Anandam Ravi, "A novel approach for test suite prioritization",(2013)

[29] J. Albert Mayan and T. Ravi,"Structural software testing: Hybrid algorithm for optimal test sequence selection during regression testing", (2015)

[30] Y. Singh, "Systematic Literature Review on Regression Test Prioritization Techniques Difference between Literature Review and Systematic Literature", (2012)

[31] Catal and D. Mishra, "Test case prioritization: a systematic mapping study," Software Quality Journal, vol. 21, no. 3, pp. 445–478, 2012.

[32] A. Kumar and K. Singh, "A Literature Survey on test case prioritization," Compusoft, 2014.

[33] J. Anderson, S. Salem, H. Do, "Improving the Effectiveness of Test Suite through Mining Historical Data", (2014)

[34] S. Elbaum, G. Rothermel, J. Penix, "Techniques for Improving Regression Testing in Continuous Integration Development Environments", (2014).

[35] P. Kiran and K. Chandraprakash, "A Literature Survey on TCP-Test Case Prioritization using the RT- Regression Techniques," Global Journal of, 2015.

[36] K. Solanki, Y. Singh, S. Dalal, and P. Srivastava, "Test Case Prioritization: An Approach Based on Modified Ant Colony Optimization," Emerging Research in, (2016).

[37] K. Solanki, Y. Singh, S. Dalal, " A Comparative Evaluation of "m-ACO" Technique for Test Suite Prioritization", Indian Journal of Science and Technology, Vol 9(30), (2016).

[38] A. Labuschagne, L. Inozemtseva, R. Holmes. "Measuring the Cost of Regression Testing in Practice- A Study of Java Projects Using Continuous Integration",(2017).

[39] I. Alagoz, T. Herpel, R. German, "A selection method for black box regression testing with a statistically defined quality level", (2017)