Available Online at www.ijcsmc.com

International Journal of Computer Science and Mobile Computing

A Monthly Journal of Computer Science and Information Technology

ISSN 2320–088X

IJCSMC, Vol. 3, Issue. 11, November 2014, pg.170 – 174

RESEARCH ARTICLE

Effective Test Case Prioritization Technique in Web Application for Regression Test Suite

Hari Sankar Chaini[1], Dr. Sateesh Kumar Pradhan[2]

[1]Research Scholar, Dept. of computer Sc. And Application Utkal University, Vani Vihar, INDIA

[2]HOD, Dept. of computer Sc. And Application Utkal University, Vani Vihar, INDIA

[1] harisankar.chaini@gmail.com; [2] sateesh1960@gmail.com

Abstract— Regression testing is retesting of a software system that has been modified to ensure that any bugs have been fixed and that no other previously working functions have failed as a result of the fixes and that newly added features have not created problems with previous versions of the software. Test case prioritization techniques, which are used to improve the cost-effectiveness of regression testing, order test cases in such a way that those cases that are expected to outperform others in detecting software fault are run earlier in the testing phase. In this paper we are describing the test suite prioritization through fault exposed. As a result, this will help us to prioritize the test suite for execution and coverage.

Keywords— Regression Testing, APFD, Regression Test Selection, Test Case prioritization

I. INTRODUCTION

Regression testing objective is to identify newly introduced defects in an existing application or functionality after system changes. The changes could be a major code changes, bug fixes, enhancements or configuration modifications. Software systems are tightly integrated together so that they are meet the intended results and satisfy business objectives. So there is a high probability of existing functionality not working because of the integration of the code changes, bug fixes, etc. with the existing code. This adverse change will result in either a newly introduced defect or a re-emergence of an existing broken code.

Regression testing is a frequently applied but expensive maintenance process that aims to (re)verify modified software. Many approaches for improving the regression testing processes have been investigated. The test case prioritization is important in regression testing. It schedules the test cases in a regression test suite with a view to maximizing certain objectives which help reduce the time and cost required to maintain service-oriented business applications. Test case prioritization seeks to find an efficient ordering of test case execution for regression testing. The most ideal ordering of test case execution is one that reveals faults earliest. Since the nature and location of actual faults are generally not known in advance, test case prioritization techniques have to rely on available surrogates for prioritization criteria [1]. Test suite prioritization is a regression testing technique where test cases are ordered such that faults can be detected early in the test execution cycle. This is useful because tests accumulate over multiple revisions and versions of the system and it is not feasible to execute all the tests in a limited amount of time [2].

© 2014, IJCSMC All Rights Reserved

II. REGRESSION TESTING TECHNIQUES

Let P be a program [3], let P′ be a modified version of P, and let T be a test suite for P. Regression testing consists of reusing T on P′, and determining where the new test cases are needed to effectively test code or functionality added to or changed in producing P′.

There is various regression testing techniques. Figure 1 shows various regression testing techniques.

- Retest all
- Regression Test Selection
- Prioritization of Test cases

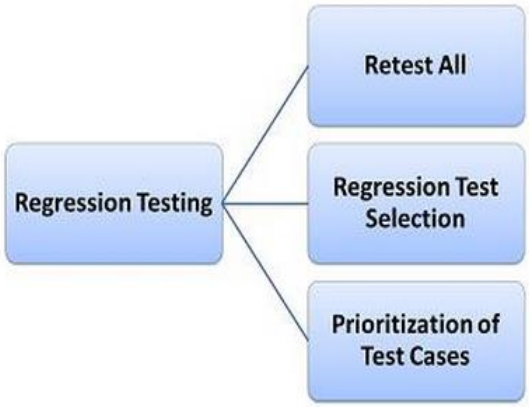

Figure1. Regression Testing Techniques

Retest All: - Retest all method is one of the conventional methods for regression testing in which all the tests in the existing test suite is reran. So the retest all technique is very expensive as compared to other techniques [3]. This method is costly to execute in full as it require more time and budget.

Regression Test Selection (RTS): - We first-rate a part of test suite to replay if the cost of ranking a part of test suite is less than the cost of running the tests that Regression Test selection allows us to ignore. RTS splits the existing test suite into below test cases.

Reusable test cases: - The test cases should be design in such a manner with fact and figure so that whenever same functionalities appear fully or partially, they can be used without any modification. Reusable test cases need not be run because they will give the same result as previous tests.

Re-testable test cases:- This includes both types of test cases which should be repeated because the program constructs being tested are modified, although the specification for the program constructs are not modified, observe that although these test cases specify the correct input/output relations, they may not be testing the same program constructs as before the modification.

Obsolete test cases: - This test includes both types of test cases that can no longer be used.

There are three ways that a test case may become obsolete

1. If a test case specifies an incorrect input-output relation due to a modification to the problem specification, then it can no longer be used.

2. If the targeted program component has been modified, some test cases may correctly specify the input-output relation but may not be testing the same construct

3. Due to the change in architecture, a structural case may no longer contribute to the structural coverage of the program, because all structural test are designed to increase the structural coverage of the program, any structural test which does not increase the coverage measure can be deleted during the testing phase [7].

© 2014, IJCSMC All Rights Reserved

III. TEST CASE PRIORITIZATION

This technique of regression testing 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. This is of two types:

- General prioritization which attempts to select an order of the test case that will be effective on average subsequent versions of software [6].
- Version Specific prioritization which is concerned with particular version of the software.

Test Case Prioritization can be classified further as given below:

- Comparator techniques: These involve of random ordering and optimal ordering.
- Statement level techniques: These methods are also known as Fine Granularity. They involve of total statement coverage prioritization, additional statement coverage prioritization, total fault-exposing-potential (FEP) prioritization an additional FEP prioritization.
- Function level techniques: These techniques are also known as Coarse Granularity. They consist of total function coverage prioritization, additional function coverage, total FEP prioritization, and additional FEP prioritization, and total fault index (FI) prioritization, additional Fault Index (FI) prioritization, total FI with FEP coverage prioritization and additional FI with FEP coverage prioritization.

IV. TEST SUITE IDENTIFICATION THROUGH AVERAGE OF THE PERCENTAGE OF FAULT DETECTED (APFD)

As described in [9], a typical regression testing procedure can be elaborated as:

Step 1:- Involves the regression test selection i.e. selecting a subset T' of T in order to test P'
Step 2:- Involves test suite execution i.e. test P' with T' means executing the test suites and checking test results to measure the correctness of P'
Step 3:- Involves the identification of code coverage i.e. to determine whether P' has new functionality which requires to create new test cases for P'
Step 4:- Involves executing new test cases to test P', establishing correctness
Step 5:- Involves the test suite maintenance i.e. updating and storing test information means maintaining test execution profile for P'

When the user is selecting test suite for the regression testing, it has to be prioritized. We can validate through the APFD. APFD, which measures the weighted average of the percentage of faults detected over the life of the suite. APFD values range from 0 to 100; higher numbers imply faster (better) fault detection rates [3].

Let T be a test suite containing n test cases,
Let F be a set of m faults revealed by T.
Let TFi be the 1st test case in ordering T' of T which reveals fault i.
The APFD for test suite T0 is given by the equation:

$$APFD = \frac{1 - (TF1 + TF2 + TF3 + TF4\ldots\ldots\ldots\ldots + TFn)}{nm} + \frac{1}{2n}$$

For example, consider a web application having four modules and ten different scenarios as per below mentioned table. If user has to find out the regression suit with fault tolerance, then APFD should be calculated. User can get 24 different permutations with respect to four modules "A", "B", "C", "D".

© 2014, IJCSMC All Rights Reserved

TABLE I
MODULES AND SCENARIOS MATRIX

	1	2	3	4	5	6	7	8	9	10
A	X	X	X							
B	X	X	X	X	X	X	X			
C					X	X	X			
D								X	X	X

Considering the above mentioned table (Table-I), we can get the below APFD (Table-II) result. Which show the test suite "BDCA" and "BDAC" have APFD "80%", has capability of fastest fault detecting capabilities than other test suite.

TABLE III
PERMUTATION OF MODULES WITH APFD CALCULATED VALUES

Test Suite	APFD	Test Suite	APFD	Test Suite	APFD	Test Suite	APFD
ABCD	55	BADC	72.5	CABD	52.5	DACB	57.5
ABDC	62.5	BACD	65	CADB	57.5	DABC	60
ACBD	52.5	BCDA	72.5	CBAD	55	DBCA	70
ACDB	57.5	BCAD	65	CBDA	62.5	DBAC	70
ADBC	60	BDCA	80	CDAB	60	DCBA	60
ADCB	57.5	BDAC	80	CDBA	60	DCAB	57.5

From analysis of the above result, we reach to the below APFD Graph (Figure-2), where the user will able to know the coverage of the test cases for a particular test suit and choose the suit as and when it is suitable for the different build or releases.

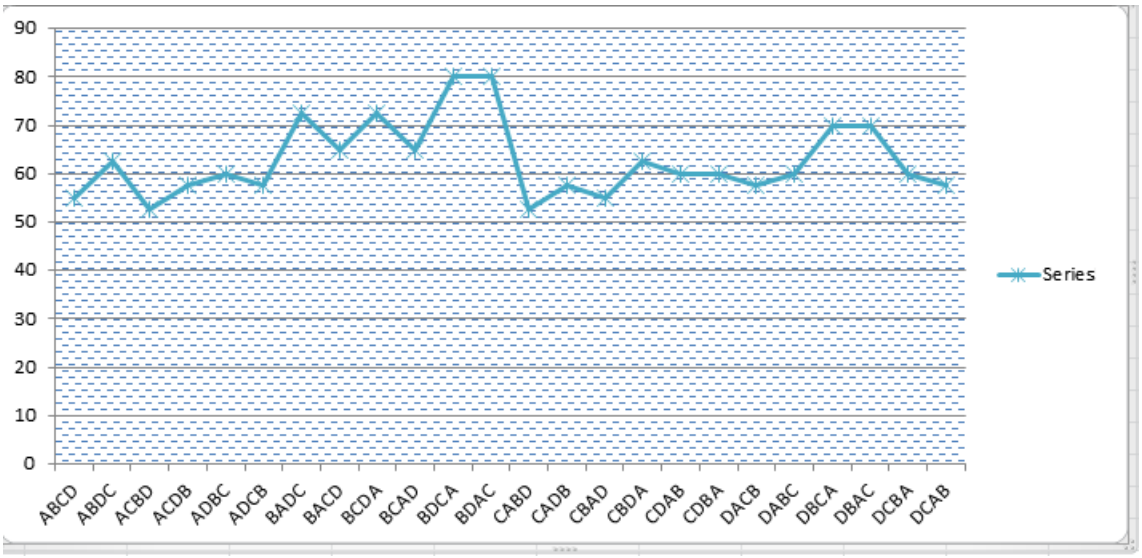

Fig 2 APFD Graph

© 2014, IJCSMC All Rights Reserved

There are several prioritization techniques available for identification of regression suite, but we must to find out prioritization technique which is best applicable for specified application. It contains the analysis of the below mention points.

- How do techniques differ in terms of their ability to reduce regression testing costs?
- How do techniques differ in terms of their ability to detect faults?
- What trade-offs exist between test suite size reduction and fault detection ability?
- When is one technique more cost-effective than another?
- How do factors such as program design, location and type of modifications, and test suite design affect the efficiency and effectiveness of test selection techniques?

V. CONCLUSION

The greatest persistent need for future work, involves additional studies of prioritization applied to a wider variety of applications, modified applications, test suites, and faults. More over data will help us better understand the characteristics inducing cost-effectiveness, and help us further distinguish techniques through additional metrics. We can extend our analysis to multiple types of faults; develop time-series-based models, capturing notions of amortized analysis and non-constant fault densities, rerun these experiments using larger programs with more complex fault distributions to find an efficient and effective regression suite.

REFERENCES

[1] Shin Yoo, Mark Harman, Paolo Tonella and Angelo Susi, "Clustering Test Cases to Achieve Effective & Scalable Prioritisation Incorporating Expert Knowledge," In.Proc.of the eighteenth international symposium on Software testing and analysis, pp. 201 - 211, 2009.

[2] Renee Bryce, Sreedevi Sampath, Jan B. Pedersen and Schuyler Manchester, "Test Suite Prioritization by Cost-based Combinatorial Interaction Coverage," International Journal of Systems Assurance Engineering and Management, Vol.2, No.2, pp.126-134, Apr 2011.

[3] Sebastian Elbaum, Praveen Kallakuri, Alexey G. Malishevsky, Gregg Rothermel, Satya Kanduri, "Understanding the Effects of Changes on the Cost-Effectiveness of Regression Testing Techniques," Journal of Software Testing, Verification, and Reliability, 13(2) pages:65-83, June 2003.

[4] H. Leung and L. White, "Insights into regression testing," In Proceedings of the Conference on Software Maintenance, pages 60-69, Oct. 1989.

[5] K.K.Aggarwal & Yogesh Singh, "Software Engineering Programs Documentation, Operating Procedures," New Age International Publishers, Revised Second Edition – 2005.

[6] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold, "Prioritizing Test Cases for Regression Testing," IEEE Trans. Software Eng., vol. 27, no. 10, pages 929-948, Oct. 2001.m1014047

[7] Allent Kent, James G. Wllllams, "Encyclopedia of Computer science and Technology", page 311

[8] Sachin Kumar, Vipin Kumar, "Introduction of Software Maintenance Testing", International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 5, May 2014.

[9] G. Rothermel and M. J. Harrold, Analyzing Regression Test Selection Techniques, IEEE Transactions on Software Engineering, V.22, no. 8, August 1996, pages 529-551.

© 2014, IJCSMC All Rights Reserved