International Journal of Advances in Engineering Research http://www.ijaer.com/

(IJAER) 2012, Vol. No. 3, Issue No.IV,April **ISSN: 2231-5152**

ASSESSMENT OF SOFTWARE DESIGN USING REGRESSION TESTING

Prof. Sheo Kumar

ABSTRACT

The purpose of regression testing is to ensure that changes made to software, such as adding new features or modifying existing features, have not adversely affected features of the software that should not change. Regression testing is usually performed by running some, or all, of the test cases created to test modifications in previous versions of the software. Many techniques have been reported on how to select regression tests so that the number of test cases does not grow too large as the software evolves. Our proposed hybrid technique combines modification, minimization and prioritization-based selection using a list of source code changes and the execution traces from test cases run on previous versions. This technique seeks to identify a representative subset of all test cases that may result in different output behavior on the new softwareversion.

KEYWORDS: Regression Testing, Modification-Based Test, Selection, Test Set Minimization, Test Set Prioritization

INTRODUCTION

No matter how well conceived and tested before being released, software will eventually have to be modified in order to fix bugs or respond to changes in user specifications. Regression testing must be conducted to confirm that recent program changes have not adversely affected existing features and new tests must be conducted to test new features. Testers might rerun all test cases generated at earlier stages to ensure that the program behaves as expected. However, as a program evolves the regression test set grows larger, old tests are rarely discarded, and the expense of regression testing grows. Repeating all previous test cases in regression testing after each minor software revision or patch is often impossible due to the pressure of time and budget constraints. On the other hand, for software revalidation, arbitrarily omitting test cases used in regression testing is risky. In this paper, we investigate methods to select small subsets of effective fault-revealing regression test cases for assessmentof softwaredesign.

REGRESSION TESTING IN SOFTWARE PRODUCT

“Regression testing is not an isolated one-off activity, but rather an activity of varying scope and preconditions, strongly dependent on the context in which it is applied”. There exist techniques for regression test selection for single applications that can beeasily adapted on any context of testing process. These approaches have been implemented in Software Product Testing to reduce the testing effort considering user acceptance in the end. By focusing the testing on changes to the system and re-executes past tests to ensure that recent changes haven't broken other parts that the code was not changed, by definition. Regression testing aims at verifying that previously working software still works after a change. Since many variants derived from the common platform and testing to each product with respect to variants is not an easytask.

In order to test the complete product , all generic components have to be tested for all the products. A major challenge in Software Product testing is that a large number of test cases will be stored for both product and product variants. Such testing throughout the product is infeasible and a test selection strategy is required. The variants that are derived from the Software Product are closely related and large amount of testing can be saved and removed by identifying the redundant test cases. Recent research studies have considered the challenge to evaluate possible approaches aiming to minimize the amount of redundant testing in Software Product. The regression test selection approach can be adapted to minimize those redundant test cases from large pool of testcases.

TYPES OF REGRESSION TESTING

There are two types of regression testing that are proposed here, even though it is not being practiced or popular.

A "final regression testing" is done to validate the gold master builds, and "Regression testing" is done to validate the product and failed test cases between system test cycles.

The final regression test cycle is conducted on an "unchanged build for a period of x days" or for a period, which was agreed as the "cook-time" for release. The product is continuously exercised for the complete duration of this cook-time. Some of the test cases are even repeated to find out whether there are failures in the final product that will reach the customer. All the bug fixes for the release should have been completed for the build used for the final regression test cycle. The final regression test cycle is more critical than any other type or phase of testing, as this is the only testing which ensures "the same build of the product that was tested reaches thecustomer".

A normal regression testing can use the builds for a time period that is needed for the test cases to be executed. However unchanged build is highly recommended for each cycle of regression testing.

SELECTING TEST CASES FOR REGRESSION TESTING

It was found from industry data that good number of the defects reported by customers were due to last minute bug fixes creating side effects and hence selecting the test case for regression testing is an art and not that easy.

The selection of test cases for regression testing:

a. Requires knowledge on the bug fixes and how it affect the system.
b. Includes the area of frequentdefects
c. Includes the area which has undergone many/recent code changes
d. Includes the area which is highly visible to theusers

e. Includes the core features of the product which are mandatory requirements of thecustomer

Selection of test cases for regression testing depends more on the criticality of bug fixes than the criticality of the defect itself. A minor defect can result in major side effect and a bug fix for an Extreme defect can have no or a just a minor side effect. So the test engineer needs to balance these aspects for selecting the test cases for regression testing

When selecting the test cases we should not select only those test cases which are bound to fail and those test cases which has no or less relevance to the bug fixes. You need to select more positive test cases than negative test cases for final regression test cycle as this may create some confusion and unexpected heat. It is also recommended that the regular test cycles before regression testing should have right mix of both positive and negative test cases. Negative test cases are those test cases which are introduced newly with intent to break the system.

It is noticed that several companies have "constant test cases set" for regression testing and they are executed irrespective of the number and type of bug fixes. Sometimes this approach may not find all side effects in the system and in some cases it may be observed that the effort spend on executing test cases for regression testing can be minimized if some analysis is done to find out what test cases are relevant and what arenot.

It is a good approach to plan and act for regression testing from the beginning of project before the test cycles. One of the ideas is to classify the test cases into various Priorities based on importance and customer usage. Here it is suggested the test cases be classified into three categories;

1. Priority-0 – Sanity test cases which checks basic functionality and are run for pre-system acceptance and when product goes thru major change. These test cases deliver a very high project value to both engineering dept and tocustomers.

2. Priority-1 – Uses the basic and normal setup and these test cases deliver high project value to both engineering and tocustomers.

3. Priority-2 – These test cases deliver moderate project value. Executed part of ST cycle and selected for regression testing on needbasis.

There could be several right approaches to regression testing which needs to be decided on "case to case" basis;

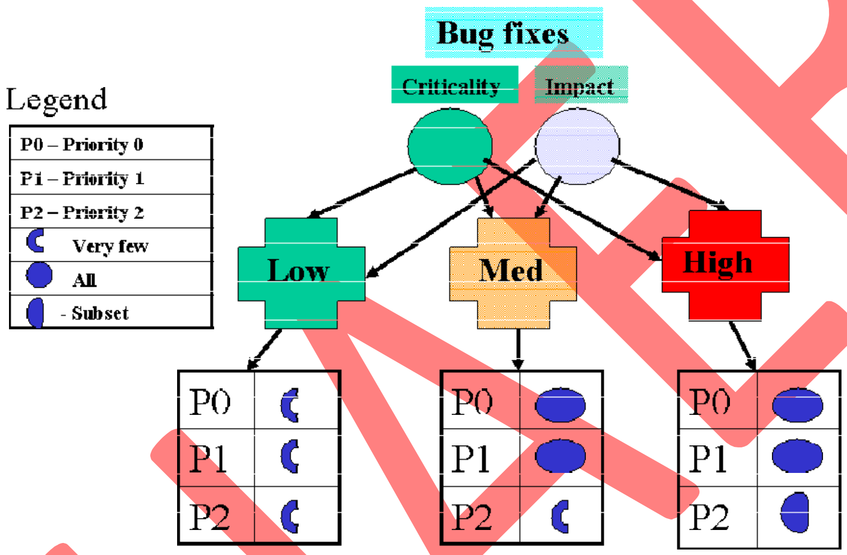
- Case 1: If the criticality and impact of the bug fixes are LOW, then it is enough a test engineer selects a few test cases from TCDB and executes them. These test cases can fall under any Priority (0, 1 or2).

- Case 2: If the criticality and the impact of the bug fixes are Medium, then we need to execute all Priority-0 and Priority-1 test cases. If bug fixes need additional test cases from Priority-2, then those test cases can also selected and used forregression testing. Selecting Priority-2 test cases in this case is desirable but not a must.

- Case 3: If the criticality and impact of the bug fixes are High, then we need to execute all Priority-0, Priority-1 and carefully selected Priority-2 testcases.

- Case 4: One can also go thru the complete log of changes happened (can be obtained from Configuration management engineer) because of bug fixes and select the test cases to conduct regression testing. This is an elaborate process but can give very good results.

This is illustrated in the picture below;



RESETTING THE TEST CASES FOR REGRESSION TESTING

In a big product release involving several rounds of testing, it is very important to note down what test cases were executed with what build and related information. This is called test case result history. In many organizations not all types of testing and all test cases were repeated for each cycle. In such cases resetting the test cases become very critical for the success of regressiontesting.
RESET of test case, are not expected to be done often. Resetting of the test cases needs to be done with following considerations;

a. When there is a major change in theproduct
b. When there is a change in the build procedure which affect the product c Large release cycle where some test cases were not executed for a long time
d. You are in the final regression test cycle with a few selected testcases
e. Where there is a situation the expected results of the test cases could be quite different from previouscycles

When the above guidelines are not met, you may want to RERUN the test cases rather than resetting the results of the test cases. There are only few differences between RERUN and RESET states in test cases, either way the test cases are executed but in case of RESET one has to think “zero base” and expect different result than what was obtained in earlier cycles and therefore those test cases affect the completion rate of testing. In case of RERUN the management need not worry about completion rate as those test cases can be considered complete except for a formality check and are expected to give same results.

To give you an example, if there is a change in Installation of a product, which does not affect the product functionality, then the change can be tested independently by rerunning some test cases and we don't have to RESET the test cases.

RESET is also decided based on how stable the functionalities are. If you are in Priority- 1 and have reached a stage of comfort level on Priority-0 (say for example more than 95% pass rate) then you don't RESET Priority-0 test cases unless there is a major change. This is true with Priority-1 test cases when you are in Priority-2 test phase.

PRE-SYSTEM TEST CYCLE PHASE

For pre-system acceptance only Priority-0 test cases are used. For each build that is entering the system test, the build number is selected and all test cases in Priority-0 are reset to NOT RUN. The system test cycle starts only if all pre-system test cases (Priority-0) pass.

SYSTEM TEST CYCLE – PRIORITY-1 TESTING PHASE

After pre-system acceptance is over, Priority-1 test cases are executed. Priority-1 testing can use multiple builds. In this phase the test cases are RESET only if the criticality and impact of the bug fixes and feature additions are high. A RESET procedure during this phase may affect all Priority-0 and Priority-1 test cases and these test cases are reset to NOT RUN.

SYSTEM TEST CYCLE – PRIORITY-2 TESTING PHASE

Priority-2 testing starts after all test cases in Priority-1 are executed with an acceptable pass % as defined in test plan. In this phase several builds are used. In this phase the test cases are RESET only if the criticality and impact of the bug fixes and feature additions are very high. A RESET procedure during this phase may affect Priority-0, Priority-1 and Priority-2 testcases.

CONCLUSION

Effective regression testing is a trade-off between the number of regression tests needed and the cost. The greater the numbers of regression tests, the more complete the program revalidation. However, this also requires a larger budget and greater resources which may not be affordable in practice. Running fewer regression tests may be less expensive, but has the potential of not being able to ensure that all the inherited features still behave as expected, except where changes are anticipated. In this paper, we propose a modification-based technique followed by test set minimization or prioritization to determine which regression tests should bererun.

Software Product plays a huge impact in major software industry for creating large number of product variants from a common platform. Creating a product from scratch is a time consuming process. So, analyzing the commonalities and variabilities between different products and their variants helps to reuse the coreassets.

Software Product Testing is a laborious task since large number of product variants can be derived from a product line. Due to the enormous number of possible products, individual product testingbecomes more and more infeasible. A framework has been proposed and it is supported by various testing tools to extract the different test cases through various testing processes. When design changes have been determined, it gives good support to plan and adapt regression testing effort in the Software ProductLine.

In the future, it is important to run additional case studies to assess the drawbacks and advantages of proposed framework by implementing in different systems. To get better results, combinatorial testing can be performed while deriving the Feature Model diagram from specificationrequirements.

REFERENCES

[1] G.Rothermel et al "Test Case Prioritization: An Empirical Study" Proceedings oftheInternationalConferenceonSoftwareMaintenance,Oxford,UK, September, 1999

[2] S. Elbaum, A. Malishevsky,and G. Rothermel.Prioritizing test cases for regression testing.Proc. Int'1 Symp. Softw. Testing and Analysis, Pages 102-112,Aug.2000.

[3] D.Leon and A. Podgurski. A comparison of coveragebased and distribution- based techniques for filtering and prioritizing test cases. In Int'1. Symp. Softw. Rel. Eng., Pages 442-453,Nov.2003.

[4] J.kim and A. Porter. A history-based test prioritization technique for regression testing in resource constrained environments. In Int'1. Conf. Softw. Eng.,Pages 119-129, May 2002.

[5] D.Jeffrey and N. Gupta. Test case prioritization using relevant slices. In Int'1. Comp. Softw. Appl. Conf., pages 411-420,Sept.2006.

[6] Roger S. Pressman, Software engineering a practitioner's approach 6/e, 2005

[7] Alexey G. Malishevsky, Joseph R. Ruthruff, Gregg Rothermel, Sebastian Elbaum, Cost- cognizant Test Case Prioritization,2006

[8] Pavan Kumar Chittimalli& Mary Jean Harrold ISEC'08, February 19-22, 2008, Hyderabad,India.

[9] MaruanKhoury,Cost-Effective Regression Testing 2006

[10] Kristen R. Walcott et al," Time Aware Test Suite Prioritization",ACM, ISSTA'06,, Portland, Maine,USA, July 17-20,2006

[11] B. Qu, C. Nie, B. Xu and X. Zhang. Test Case Prioritization for Black Box Testing. In Proceedings of the International Computer Software and Applications Conference, pages 465-474, July,2007.

[12] Qurat-ul-annFarooq, Muhammad Zohaib Z. Iqbal, Zafar I Malik," An Approach for Selective State Machine based Regression Testing", ACM 978-1-59593-850-3/07/0007,AMOST'07,London,UK.July 9-12,2007

[13]Wang L., J.Yuan,X.Yu, J. Hu, X. Li,and G. Zheng. Generating Test Cases from UML Activity Diagram based on Gray-Box Method. In 11th A s i a -Pacific Software Engineering Conference (APSEC 2004),Pages284-291,2004.