PERFORMANCE ANALYSIS OF REGRESSION TESTING

K. AYAZ AHMED SHARIFF, **Dr. M. Deepamalar**

Research Scholar, SSSUTMS, SEHORE, MP

Research Guide, SSSUTMS, SEHORE, MP

ABSTRACT

The regression testing one of the testing techniques is the testing in which testing is performed on the modified application using the same previously defined sets of Test cases. When an application is developed and it is tested for the first time a set of test cases means test suite is designed to verify and validate its functionality. The tester keeps this test suite with them for further use. When a modification is done in the application then these previously designed test suites are used by testers to ensure that no new errors have been introduced in the previously tested code. But it is not worth checking all test cases for a small change. It is impractical and inefficient to run each test for each program function when a change occurs. In addition, it will be a very expensive technique that also performs the full test for a small change. To reduce the cost of regression technology and make it more efficient, researchers have introduced the concept of prioritizing test cases. So here we are providing brief discussion about the regression testing.

Keywords: Test Case prioritization, TCS algorithm, Test set up

INTRODUCTION

Testing and verifying software systems is becoming increasingly important with the growth in the size and complexity of modern software systems. Due to which, many test cases which were produced during development are reused in regression testing increasing the overall cost of testing. Test Case prioritization (TCP) provides an efficient solution to decrease the costs of regression testing. An efficient test case prioritization technique plays a vital role in increasing efficiency of testing while keeping the cost of testing in check. There are various examples of test case prioritization techniques such as Random Ordering prioritization, Optimal Ordering prioritization, Total statement Coverage Prioritization, Additional Statement Coverage Prioritization, Total Function Coverage Prioritization.

In the test case prioritization the test cases are prioritized and scheduled in order that attempts to maximize some objective function. To decide the priority of the test cases the various factors depending upon the need are decided then the priority is assigned to the test cases. Test case prioritization provides a way to schedule and run test cases, which have the highest priority in order to provide earlier detect faults. Furthermore, in [5] it is mentioned that Gregg Rothermel has proven that prioritization and scheduling test cases are one of the most critical task during the software testing process as he has given an example of industrial collaborators reports, which shows that there are approx 20,000 lines of code, running the entire test cases require seven weeks. In this situation prioritization of test cases plays a vital role to save the time.

Various test case prioritization techniques depending on different parameters such as code coverage, fault detection, function coverage are suggested in this survey. The performance of the numerous prioritization techniques is also analyzed. As the name reveals in the first phase requirement from the point of view of the users as well as other stakeholder are collected, in the second phase designing is done, in the third phase implementation is done and the fourth phase is Testing and Maintenance, the testing include the execution of the program or an application with an intent of finding software errors and faults in the application executed. The testing is also known as the process of validating and verifying that software meets the business and technical requirements that guided its design and developments [3], so that it works according to the need of the stakeholders.

METHODS OF REGRESSION TESTING

Test cases are referred as the collection of test specification, test procedures and test programs, that is developed by software engineers before designing and writing a piece of code. Here prioritization of the test case is important for regression testing. Because of prioritization selective test cases gets executed. Prem et al presents the regression testing is one of the ways, in order to make sure that the modification made on the program lines does not affect the other parts of the software. During the regression testing all the test cases are again tested. The following figure shows that Regression testing has four methods that are,

(1) Rest all
(2) Regression test selection
(3) Test suite reduction
(4) Test case prioritization.

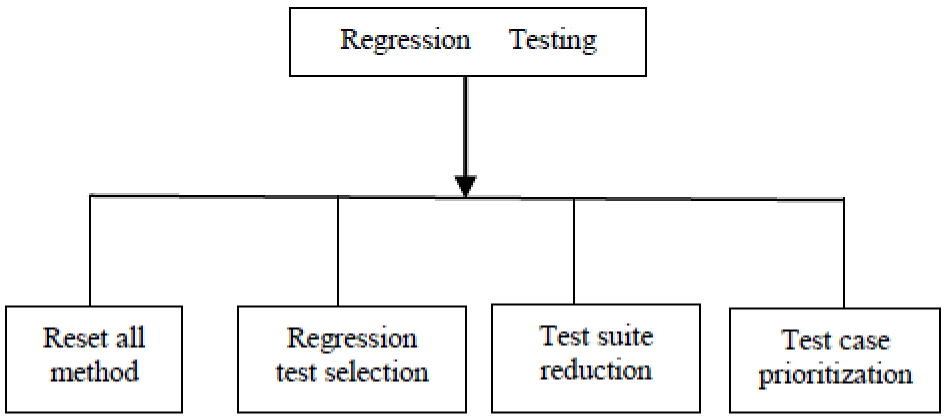


Figure 1: Flow of Regression Testing

Reset all means the test cases that no longer apply to modified version of the programs are discarded and remaining set of test cases are used to reset the modified program. Regression test selection technique used to select the test cases for testing based on the information about modified program. Test suite reduction is different from regression test selection. This technique has the information about the program and removes the test cases. It is not permanently remove the test cases. But select which test cases are required. Hence it will be a time consuming process and expensive. To overcome this problem the developers incorporate prioritization of test cases for selective testing. Prioritization means scheduling (or) changing the order of test case execution, based on the factors (or) methods. Prioritization is used to identify the defects early and improve the effectiveness of regression testing. In this paper we reviewed the number of factors (or) methods used in test case prioritization in detail. Sujatha et al presents a technique, genetic algorithm is proposed to prioritize the test cases. The source code (or) binary code is used to prioritize the test cases, which is not available. The paper prioritizes the test cases completely based on requirements of the system and technique considers the maximum requirement coverage and potential of fault detection during the testing process. Severe fault test cases are executed first which are covering highly prioritized requirements. This approach is used to find the more severe faults, early in the testing process than the other random as well as fault based approach.

Test case selection and prioritization [1] provides an effective selection and prioritization of test cases based on the code coverage. This approach is used to reduce the cost and time for the regression testing. This approach has three techniques (1) Test case minimization (2) test case selection (3) Test case prioritization TCS algorithm for Test case selection and TCP algorithm for test case prioritization. In TCS algorithm test cases are grouped into 3 clusters (1) Out dated (2) Required (3) Surplus. Software engineers often save the test suites they develop for their software so that they can reuse those test suites later as the software evolves. Such test suite reuse, in the form of regression testing, is pervasive in the software industry [24] and, together with other regression testing activities, has been estimated to account for as much as one-half of the cost of software maintenance [4, 2]. Running all of the test cases in a test suite, however, can require a large amount of effort.

For example, one of our industrial collaborators reports that for one of its products of about 20,000 lines of code, the entire test suite requires seven weeks to run. One For this reason, researchers have considered various techniques for reducing the cost of regression testing, including regression test selection and test suite minimization techniques. Regression test selection techniques (e.g. [1, 2]) reduce the cost of regression testing by selecting an appropriate subset of the existing test suite, based on information about the program, modified version, and test suite.

Test suite minimization techniques are lower costs by reducing a test suite to a minimal subset that maintains equivalent coverage of the original test suite with respect to a particular test adequacy criterion. Regression test selection and test suite minimization techniques, however, can have drawbacks. For example, although some empirical evidence indicates that, in certain cases, there is little or no loss in the ability of a minimized test suite to reveal faults in comparison to its minimized original, other empirical evidence shows that the fault detection capabilities of test suites can be severely compromised by minimization [3]. Similarly, although there are safe regression test selection techniques (e.g. [2, 4]) that can ensure that the selected subset of a test suite has the same fault detection capabilities as the original test suite, the conditions under which safety can be achieved do not always hold [8, 9]. Test case prioritization techniques [3, 6] provide another method for assisting with regression testing. These techniques let testers order their test cases so that those test cases with the highest priority, according to some criterion, are executed earlier in the regression testing process than lower priority test cases. For example, testers might wish to schedule test cases in an order that achieves code coverage at the fastest rate possible, exercises features in order of expected frequency of use, or exercises subsystems in an order that reflects their historically demonstrated propensity to fail. When the time required to reexecute an entire test suite is short, test case prioritization may not be cost-effective it may be sufficient simply to schedule test cases in any order.

When the time required to execute an entire test suite is sufficiently long, however, test case prioritization may be beneficial, because in this case, meeting testing goals earlier can yield meaningful benefits. Because test case prioritization techniques do not themselves discard test cases, they can avoid the drawbacks that can occur when regression test selection and test suite minimization discard test cases. Alternatively, in cases where the discarding of test cases is acceptable, test case prioritization can be used in conjunction with regression test selection or test suite minimization techniques to prioritize the test cases in the selected or minimized test suite. Further, test case prioritization can increase the likelihood that, if regression testing activities are unexpectedly terminated, testing time will have been spent more beneficially than if test cases were not prioritized.

Software testing is a comprehensive set of activities conducted with the intent of finding errors in software. It is one activity in the software development process aimed at evaluating a software item, such as system, subsystem and features (e.g. functionality, performance and security) against a given set of system requirements. Also, software testing is the process of validating and verifying that a program functions properly. Many researchers have proven that software testing is one of the most critically important phases of the software development life cycle, and consumes significant resources in terms of effort, time and cost. Arden [33] said that “The impact of software errors is enormous because virtually every business in the United States now depends on software for the development, production, distribution, and aftersales support of products and services.

REGRESSION TEST PROCESS

A regression test process is exhibited in Figure 2. The process assumes that P’ is available for regression testing. There is usually a long series of tasks that lead to P’ from P. These tasks, not shown in, include creation of one or more modification requests and the actual modification of the design and the code. A modification request might lead to a simple error fix, or to a complex redesign and coding of a component of P. In any case, regression testing is recommended after P has been modified and any newly added functionality tested and found correct. The tasks in are shown as if they occur in the given sequence. This is not necessarily true and other sequencings are possible.

Several of the tasks shown can be completed while P is being modified to P'. It is important to note that except in some cases, for test selection, all tasks shown in the figure are performed in almost all phases of testing and are not specific to regression testing.

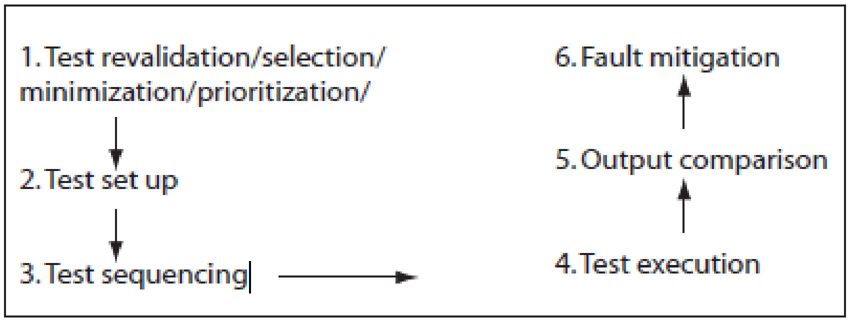


Figure 2: Test process

1. Test Revalidation/Selection/Minimization/Prioritization

While it would be ideal to test P' against all tests developed for P, this might not be possible for several reasons. For example, there might not be sufficient time available to run all tests. Also, some tests for P might become invalid for P' due to one or more reasons such as a change in the input data and its format for one or more features. In yet another scenario, the inputs specified in some tests might remain valid for P' but the expected output might not. These are some reasons that necessitate. Test revalidation refers to the task of checking which tests for P remain valid for P'. Revalidation is necessary to ensure that only tests that are applicable to P' are used during regression testing.

Test selection can be interpreted in several ways. Validated tests might be redundant in that they do not traverse any of the modified portions in P'. The identification of tests that traverse modified portions of P' is often referred to as test selection and sometimes as the *regression test selection* (RTS) problem. However, note that both test minimization and prioritization described next are also techniques for test selection. Test minimization discards tests seemingly redundant with respect to some criteria. For example, if $t1$ and $t2$ test function f in P then one might decide to discard $t2$ in favor of $t1$. The purpose of minimization is to reduce the number of tests to execute for regression testing. Test prioritization refers to the task of prioritizing tests based on some criteria. A set of prioritized tests becomes useful when only a subset of tests can be executed due to resource constraints. Test selection can be achieved by selecting a few tests from a prioritized list. However, several other methods for test selection are available as discussed later in this chapter. Revalidation, followed by selection, minimization, and prioritization is one possible sequence to execute these tasks.

2. Test Setup

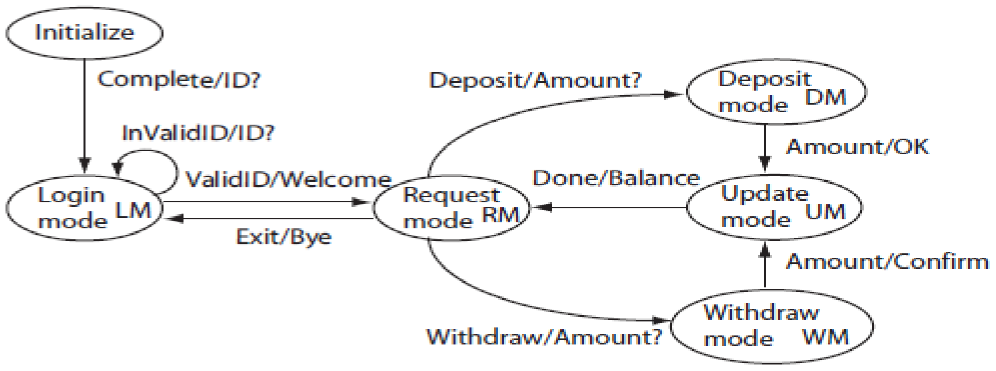
Test set up refers to the process by which the application under test is placed in its intended, or simulated, environment ready to receive data and able to transfer any desired output information. This process could be as simple as double clicking on the application icon to launch it for testing and as complex as setting up the entire special purpose hardware and monitoring equipment and initializing the environment before the test could begin. Test set up becomes even more challenging when testing embedded software such as that found in printers, cell phones, Automated Teller Machines, medical devices, and automobile engine controllers. Note that test set up is not special to regression testing, it is also necessary during other stages of testing such as during integration or system testing. Often test set up requires the use of simulators that allow the replacement of a "real device" to be controlled by the software with its simulated version. For example, a heart simulator is used while testing a commonly used heart control device known as the pacemaker.

The simulator allows the pacemaker software to be tested without having to install it inside a human body. The test set up process and the set up itself, are highly dependent on the application under test and its hardware and software environment. For example, the test set up process and the set up for an automobile engine control software is quite different from that of a cell phone. In the former one needs an engine simulator, or the actual automobile engine to be controlled, while in the latter one needs a test driver that can simulate the constantly changing environment.

3. Test Sequencing

The sequence in which tests are input to an application may or may not be of concern. Test sequencing often becomes important for an application with an internal state and that is continuously running. Banking software, web service, engine controller, are examples of such applications. Sequencing requires grouping and sequencing tests to be run together. The following example illustrates the importance of test sequencing.

For example consider a simplified banking application referred to as SATM. Application SATM maintains account balances and offers users the following functionality: login, deposit, withdraw, and exit. Data for each account is maintained in a secure database.



4. Test Execution

Once the testing infrastructure has been set up, tests selected, revalidated, and sequenced, it is time to execute them. This task is often automated using a generic or a special purpose tool. General purpose tools are available to run regression tests for applications such as web service. However, most embedded systems, due to their unique hardware requirements, often require special purpose tools that input a test suite and automatically run the application against it. The importance of a tool for test execution cannot be overemphasized. Commercial applications tend to be large and the size of the regression test suite usually increases as new versions arrive. Manual execution of regression tests might become impractical and error prone.

5. Output Comparison

Each test needs verification. This is also done automatically with the help of the test execution tool that impares the generated output with the expected output. However, this might not be a simple process, especially in embedded systems. In such systems often it is the internal state of the application, or the state of the hardware controlled by the software application, that must be checked. This is one reason why generic tools that offer an oracle might not be appropriate for test verification. One of the goals for test execution is to measure an application's performance. For example, one might want to know how many requests per second can be processed by a web service. In this case performance, and not functional correctness, is of interest. The test execution tool must have special features to allow such measurements.

LITERATURE REVIEW

Innovations in fields ranging from robotic manufacturing to nanotechnology and human genetics research have been enabled by low-cost computational and control capabilities supplied by computers and software." Also, a study conducted by NIST in 2002 reports that software bugs cost the U.S. economy $59.5 billion annually. More than a third of this cost could be avoided if better software testing was performed. Boris [5] claimed that software testing should take around 40-70% of the time and cost of the software development process. Many approaches have been proposed to reduce time and cost during software testing process, including test case prioritization techniques and test case reduction techniques. For example, [20], [13], [14], [15], [17], [4], [7] and [8]. Also, many empirical studies for prioritizing test cases have been conducted, like [4], [5], [3], [4], [14] and [15]. Furthermore, Gregg Rothermel [15] has proven that prioritizing and scheduling test cases are one of the most critical tasks during the software testing process. He referred to the industrial collaborators reports, which shows that there are approximately 20,000 lines of code, running the entire test cases requires seven weeks.

In this situation, test engineers may want to prioritize and schedule those test cases in order that those test cases with higher priority are executed first. Additionally, he [13], [16] stated that test case prioritization methods and process are required, because: (a) the regression testing phase consumes a lot of time and cost to run, and (b) there is not enough time or resources to run the entire test suite (c) there is a need to decide which test cases to run first. Test case prioritization techniques prioritize and schedule test cases in an order that attempts to maximize some objective function. For example, software test engineers might wish to schedule test cases in an order that achieves code coverage at the fastest rate possible, exercises features in order of expected frequency of use, or exercises subsystems in an order that reflects their historical propensity to fail. When the time required to execute all test cases in a test suite is short, test case prioritization may not be cost effective - it may be most expedient simply to schedule test cases in any order [13], [16]. When the time required to run all test cases in the test suite is sufficiently long, the benefits offered by test case prioritization methods become more significant. Although test case prioritization methods have great benefits for software test engineers, there are still outstanding major research issues that should be addressed. The examples of major research issues are: (a) existing test case prioritization methods ignore the practical weight factors in their ranking algorithm (b) existing techniques have an inefficient weight algorithm and (c) those techniques are lack of the automation during the prioritization process.

CONCLUSION

In this paper, we describe several techniques for prioritizing test cases for regression testing. We then describe several empirical studies we performed with these techniques to evaluate their ability to improve rate of fault detection a measure of how quickly faults are detected within the testing process. An improved rate of fault detection during regression testing provides earlier feedback on a system under test and lets debugging activities begin earlier than might otherwise be possible. Our results indicate that test case prioritization can significantly improve the rate of fault detection of test suites.

REFERENCES

1. Acharya, M, Xie, T, Pei, J & Xu, J 2007, ‘Mining API patterns as partial orders from source code: From usage scenarios to specifications’, Proceedings of ACM SIGSOFT Symposium on the Foundations of Software Engineering, pp. 25–34.

2. Aggrawal, KK, Yogesh Singh, Kaur, A 2004, Code coverage based technique for prioritizing test cases for regression testing, ACM SIGSOFT Software Engineering Notes, vol. 29, no. 5, pp. 1-4.

3. Ahmed, AA, Shaheen, M, Kosba, E 2012, Software testing suite prioritization using multi-criteria fitness function, Proceedings of 22nd International Conference on Computer Theory and Applications (ICCTA), pp. 160 – 166.

4. Anwar, Z & Ashan, A 2014, ‘Exploration and analysis of regression test suite optimization’, ACM SIGSOFT Software Engineering Notes, vol. 39, no. 1, pp. 1-5.

5. Belli, F & Budnik, Ch. J 2005, 'Towards minimization of test sets for coverage testing of interactive systems', 18th international conference on Innovations in Applied Artificial Intelligence, pp. 79-90.

6. Beszedes, A, Gergely, T, Schrettner, L, Jasz, J, Lango, L & Gyimothy, T 2012, 'Code coverage-based regression test selection and prioritization in WebKit', Proceedings of 28th IEEE International Conference on Software Maintenance, pp. 46–55.

7. Bharti Suri & Shweta Singhal, 2014, "Understanding the effect of time-constraint bounded novel technique for regression test selection and prioritization", International Journal of System Assurance Engineering and Management, vol. 6, no. 1, pp 71-77.

8. Bo Jiang, Zhenyu Zhang, Chan, WK & Tse, TH 2009, 'Adaptive random test case prioritization', IEEE/ ACM International Conference on Automated Software Engineering, pp. 233–244.

9. Boehm, B & Huang, L 2003, 'Value-based software engineering: a case study', IEEE Computer, pp. 33–41.

10. Bogdan Korel, George Koutsogiannakis & Luay Tahat, H 2008, 'Application of System Models in Regression Test Suite Prioritization', IEEE International Conference on Software Maintenance, pp. 247-256.

11. Bryce, RC, Sampath, S & Memon, A 2011, 'Developing a single model and test prioritization strategies for event-driven software', IEEE Transactions on Software Engineering, vol. 37, no. 1, pp. 48–64.

12. Bryce, RC, Sampath, S, Pedersen, JB & Manchester, S 2011, 'Test suite prioritization by cost-based combinatorial interaction coverage', Int. Journal Systems Assurance Eng. and Management, pp. 126-134.

13. Cagatay Catal & Deepti Mishra, 2012, Test case prioritization: a systematic mapping study, Software Quality Journal, vol. 21, no. 3, pp. 445-478

14. Chhabi Rani Panigrahi & Rajib Mall 2013, 'An approach to prioritize the regression test cases of object-oriented programs' CSI Transactions on ICT, vol. 1, no. 2, pp. 159-173

15. David Leon & Andy Podgurski 2003, 'A comparison of coverage based and distribution based techniques for filtering and prioritizing test cases', Proceedings of the 14th international symposium on Software Reliability Engineering, pp. 442 – 453.

16. Do, H, Mirarab, S, Tahvildari, L & Rothermel, G 2008, 'An empirical study of the effect of time constraints on the cost-bene□ts of regression testing', Proceedings of the SIGSOFT 2008/FSE16. ACM Press, New York, pp. 71–82.

17. Do, H, Rothermel, G & Kinneer, A 2006, 'Prioritizing JUnit test cases: An empirical assessment and cost-benefits analysis', Journal of Empirical Software Engineering, vol. 11, no. 1, pp. 33–70.

18. Elbaum, S, Alexey G Malishevsky & Gregg Rothermel 2002, 'Test Case Prioritization: A Family of Empirical Studies', IEEE Transactions on Software Engineering, vol. 28, no. 2, pp. 159–182.