ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)

Vol. 2, Special Issue 10, March 2016

A STUDY PAPER ON TEST CASE PRIORITIZATION TECHNIQUES

Smitha.P.S
Assistant Professor
Velammal Engg College
Smithaps.ap@gmail.com

T.Subashini
Assistant Professor
Velammal Engg College
subhatanish@gmail.com

Preethi.R
Student
Velammal Engg College
preethirani25@gmail.com

Abstract:-
Software Testing plays a crucial role in analyzing the propogation of defects. Regression testing is an important activity in the software life cycle, but it can also be very expensive and can account more cost. Test cases are prioritized to reduce the overall cost associated with testing. There are number of techniques available for test case prioritization to implement them very early in the verification and validation process. Any reduction in the cost of regression testing would help to reduce the software maintenance cost. Test case prioritization techniques schedule test cases for execution in an order that attempts to increase their effectiveness in meeting their performance goal. Various existing test prioritization techniques and the effectiveness of these techniques were studied. However, the collection of dynamic code coverage information on test cases has several associated drawbacks including cost increases and reduction in prioritization precision. After studying various prioritization techniques a comparison has been made which shows the issues and challenges in reducing the cost of testing. Coverage information like statement coverage and branch coverage helps in evaluating the effectiveness of testing by providing data on different coverage items. The goal of test case prioritization technique is to increase a test suite's rate of fault detection and to reduce the cost of regression testing.

Keywords—Software testing, regression testing, test case prioritization.

1. INTRODUCTION

Software testing is a process that is performed to support quality assurance or it is a means of assessing the software to determine its quality. Testing activities is considered as one half of the cost of software maintenance. Regression testing is defined as "the process of retesting the modified parts of the software and ensuring that no new errors have been introduced into previously tested code". It is an expensive activity and consumes significant amount of effort and cost. Testers want to order their test cases so that those test cases with the highest priority run first. Test case prioritization may not be cost effective to schedule test cases in any order, when the time required to execute all test cases in a test suite is short.

Numerous techniques have been proposed to deal with the regression testing costs. Regression test selection techniques select a subset of existing test case set for execution, depending on the changes to the software. Test case prioritization techniques retain the complete test suite, but change the order of test cases before execution but aims to find the defects earlier during the testing. The testing teams need to run regression test case set on many intermediate builds, to ensure that the bug fixes or the software do not adversely affect any portions of software. Test case prioritization techniques schedule test cases for regression testing in an order that attempts to maximize some objective function. When the time required executing all test cases in a test suite is short, test case prioritization may not be cost effective. When the time required running all test cases in the test suite is sufficiently long, however, test case prioritization may be beneficial.

Test case prioritization techniques, which are used to improve the cost effectiveness of regression testing order test case in such a way that those cases that are expected to outperform others in detecting software faults are run earlier in the testing phase. There exists a large variety of prioritization technique with coverage based prioritization technique dominating the field. Regression testing is an integral part of the extreme programming software development method. In this method, design documents are replaced by extensive, repeatable, and automated testing of the entire software package throughout each stage of the software development cycle.

Regression testing has traditionally been performed by a software quality assurance team after the development team has completed work. However, defect found at this stage are the mostly costly to fix. This problem is being addressed by the rise of unit testing. Furthermore, the prioritizing and scheduling test case are one of the most critical tasks during the software testing process. In this situation, test engineers may want to prioritize and schedule those test case in order that those test case with higher priority are executed first. Additionally 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 development cycle, these test cases have generally been either functional test. The selective retesting of a software system 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 reparations and that newly added features have not created problems with previous versions of the software. Regression testing is initiated after a programmer has attempted to fix a recognized problem or has added source code to a program that may have inadvertently introduced errors. It is a Quality control measure to ensure that the newly modified code still complies with its specified requirements and that unmodified code has not been affected by the maintenance activity.

Test case prioritization techniques do not discard test case, they avoid the drawbacks that can occur when regression testing selection and test suite minimization discard test case. Moreover, in cases where the discarding of test case is acceptable, test case prioritization can be used in conjunction with regression testing selection or test suite minimization technique to prioritize the test case in the selected or minimized test suite. In either case, the use of test case prioritization can increase the likelihood that, if testing activities are unexpectedly terminated, the time that was spent on testing will have been spent more beneficially than it might.

Test suite development is an expensive process and additionally even conscientiously maintained test suite can grow quite large.

Most of the times running an entire suite is not possible as it takes significant amount of time to run all tests in a test suite.
So researchers have given various techniques for minimizing test suite. Test suite minimization technique lower costs by reducing a test suite to a minimal subset that maintains equivalent coverage of original set with respect to particular test adequacy criterion. Test suite minimization technique, however, can have 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 unminimized original, other empirical evidence shows that the fault detection of test suites can be severely compromised by minimization.

The paper is organized as follows: Section 2 focus on test case prioritization and section 3 shows the various techniques of prioritization and Section 4 gives the issues and challenges in test case prioritization. Section 5 concludes the discussion.

2. TEST CASE PRIORITIZATION

Regression testing is the re execution of some subset of test that has been conducted [1].Test case prioritization schedules test case in order to increase their ability to meet some performance goals. Test case prioritization techniques [3] schedule test cases for regression testing in an order that attempts to maximize some objective function. Test case prioritization techniques [1] organize the test cases in a test suite, allowing for increase in the effectiveness of testing. The main aim is to detect faults in regression testing and improves software quality. Test case prioritization mainly focuses on the test scheduling. During regression testing test case prioritization goal is to maximize the criterion that follows

1) Increased rate of fault detection
2) Increased rate of coverage
3) Increased rate of fault likelihood exposure

It is a measurement criterion and the percentage of faults detected over the life of test suite can be calculated.

The test prioritization problem [1] can be defined as follows Given:

a test suite T, PT is the set of permutations of T;

a function f from PT to the set of real numbers,

Problem: Find T'∈PT such that (for all T'') (T''∈PT) [f(T')≥ f(T'')].
In this definition, PT is the set of possible prioritizations of T, and f is an objective function that applied to any such order, yields value for that order

Test case prioritization [2] addresses a wide variety of objectives which are as follows:-

1. Testers may increase the rate of fault detection which reveals faults in regression testing
2. Testers locate faults earlier in testing process.
3. Testers reveal regression errors related to specific code changes.
4. Testers may wish to increase their confidence in the reliability of the system.

Test case prioritization can be used to locate defects quickly to begin debugging and deliver in increments ASAP and to locate the majority errors quickly. It can prioritize according to

1) module or functional dependencies
2) module size
3) module complexity

module suspected to be the most error prone. Test case prioritization techniques have been shown to beneficial for improving regression testing activities. With prioritization, the rate of fault detection is improved thus allowing testers to detect faults earlier in the system testing phase. Most of the prioritization techniques to date have been code coverage based. These techniques may treat all faults equally. Test case prioritization techniques schedule test cases foe execution so that those with higher priority, according to some criterion are executed earlier than those with lower priority to meet some performance goals. In cluster based test case prioritization techniques, by clustering test cases, based on their dynamic run time behaviour we can reduce the required number of pair wise comparisons significantly. There is a value driven approach to system level test case prioritization techniques called the prior of require for test. The results show that the prioritization approach at the system level improves the rate of fault detection of severe faults.

3. TEST CASE PRIORITIZATION TECHNIQUES

Numerous test case prioritization have been presented by Gregg Rothermel ,Roland H.Untch [3] and Gurinder Singh and Dinesh Gupta [2] Developing a test suite is not as easy it costs much. Test cases can be prioritized on certain criteria is the test assistance way. In these prioritizing techniques, testers order their test cases, so that test cases with higher priority are executed much faster than lower priority test cases [2]. We can avoid the disadvantage of test case minimization techniques by not discarding test cases in the test case prioritization techniques. [2] shows that for a project to be successful one, its cost to be minimized quality to be maximized and should be less delivery time. Our objective is to improve the test suite rate of fault detection and provide faster feedback on the system under test, or early evidence that quality goals have not met. It also let debuggers to begin their work [2]. We prioritize the test case in terms of failure rates of the modules. Alternatively, we may prioritize test cases in terms of increasing cost per coverage of code components or in terms of their increasing cost per coverage of features listed in requirement specifications. The prioritization criteria is to increase the likelihood that the prioritize test suite can meet the objective better than adhoc or random ordering of test cases [3]. Our intent is to reveal the faults earlier in the testing process.

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)

Vol. 2, Special Issue 10, March 2016

There are twelve different prioritize test cases techniques. They are as follows from papers [2, 3].

T1: No Prioritization

This prioritization technique refers to the application of no technique. Let us consider the untreated test suites and it depends upon its construction.

T2: Random Prioritization

We apply random prioritization, in which we randomly order the test in a test suite.

T3: Optimal Prioritization

For any test suite, which test case expose which faults and thus we can determine an optimal ordering of test case in a test suite for maximizing that suite's rate of fault detection. It can be used to measure the effects of prioritization techniques on the rate of fault detection.

T4: Total Branch coverage prioritization

Prioritization of test cases according to the total number of branches they cover simply by sorting in the order of total branch coverage achieved. This can be done in time O (n log n) for n branches.

T5: Additional branch coverage prioritization

Additional branch coverage iteratively selects a test case that yields the greatest branch coverage, then adjusts the coverage information on subsequent test cases with the indication of coverage of branches which is not yet covered, and then repeats till all branches cover at least one test case that have to be covered. Therefore, its cost is O (N) for programs of n branches.

T6: Total statement coverage prioritization

It is similar to total branch coverage prioritization where, instead of decisions the test coverage is measured in terms of program statements.

T7: Additional statement coverage prioritization

It resembles the same as additional branch coverage prioritization. In this technique test cases measured in terms of program statements rather than decisions.

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)

Vol. 2, Special Issue 10, March 2016

Table1: Prioritization Techniques.

T8: Total fault exposing potential (FEP) prioritization.

The ability of a fault to be exposed by a test case depends not only on whether the test case reaches a faulty statement. It also determines whether the approximate could yield a prioritization technique superior in terms of rate of fault detection The fault get exposed, when the test case reaches a faulty statement. This is much more expensive than code coverage based technique

Table2: Statement coverage

Statement	Testcase1	Testcase2	Testcase3
1	X	X	X
2	X	X	X
3		X	X
4		X	
5	X		
6	X		X
7	X		X

Total statement coverage: **(TC3, TC1, TC2)** Additional statement coverage: **(TC3, TC2, TC1)**

T9: Additional fault exposing potential (FEP) prioritization

An extension of fault exposing potential is the additional fault exposing potential. This shows additional executions of a statement are less valuable than initial executions. In additional FEP prioritization, after selecting a test case't', we lower the award values for all other test cases.

T10: Total function coverage prioritization

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)

Vol. 2, Special Issue 10, March 2016

It is same as total statement coverage prioritization [5] but operating at the level of functions, this technology prioritizes test cases according to the total number of functions they execute.

T11: Additional function coverage prioritization

It is same as total statement coverage prioritization [5] but operating at the level of functions, this technology prioritizes test cases according to the total number of additional functions they cover.

T12: Total Fault Index (FI) prioritization

Faults [5] will not equally found in each function instead certain functions are more likely to faults. Each function is assigned an
unique Fault Index representing the fault proneness for each function based on the complexity of the changes introduced into the function.

Statement	Testcase1	Testcase2	Testcase3
1	0.4	0.5	0.3
2	0.5	0.9	0.4
3		0.01	0.4
4		1.0	
5	0.6		0.4
6	0.5		0.2
	2.0	2.41	1.7

Total FEP: **(TC2, TC1, TC3)**

Table 3: FEP based prioritization

ISSN 2395-695X (Print)
ISSN 2395-695X (Online)
International Journal of Advanced Research in Biology Engineering Science and Technology (IJARBEST)

Vol. 2, Special Issue 10, March 2016

4. ISSUES AND CHALLENGES IN TEST CASE PRIORITIZATION

When there are error prone tasks in testing process, the testers [Everton L. G. Alves] who are the test specialist must reorder the test cases and interpret their results. It is a hard process for the tester to select the suitable prioritization technique when the needed details are not available [6].

While working with different prioritization techniques in the software projects we have to face several challenges [6]. They are as follows

1) Automation: To make the prioritization cost effective automation is required.
2) Technique Selection: Testers will be in confused state while selecting the exact technique. If the technique is not to the extent then it will get wasted.
3) Database: After executing the prioritization it is necessary to store the data in the database. [6] It is a hard to take in charge of these data.

The metric for quality evaluation of a prioritized test suite is the APFD which is the suite rate of fault detection. It compares the ability of different prioritization techniques [2]. Through ordering of test cases, it creates faster detecting test suites. The evolution of the suites fault localization according to each technique and to increase a subset of the test suite's rate of fault detection to quantify the goal can seen through APFD [7]. During the run of the test suite the APFD can be calculated by considering the weighted average of the number of faults detected [7].In [10], Gregg Rothermel define APFD by the following formula

$$\text{APFD} = 1 - \frac{\sum_{i=1}^{num_f} TF_i}{num_t \times num_f} + \frac{1}{2\times \text{num}_t}$$

In the formula [10] num_f denotes the total number of detected faults, num_t denotes the total number of test cases. APFD range is [0, 1].

Test			F	a	u	L	T			
	1	2	3	4	5	6	7	8	9	10
A	X				X					
B	X				X	X	X			

C	X	X	X	X	X	X	X			
D					X					
E								X	X	X

Table 4: Weighted Average Percentage Faults Detected

We are also interested in the research questions and it is given below

RQ1: Are there any more methods for performing test case prioritization?

Most recent approaches involve no memory of previous test suite execution. This approach uses previous test "runs", not just previous test case, to prioritize the test case. For the first test run, it use some of the old prioritization techniques like random, FEP, statement, branch coverage prioritization. For 2…n test runs, prioritize test cases based on whether they have been run before.

Also give high prioritization to test cases that exercise the same branches (Fault finding).We can also decrease test case prioritization based on whether it had done before (Fault avoiding)

In fact, the random prioritization technique produce very similar results when compared with the coverage based ones. The tester work can be made so sophisticated by giving the generation of execution files and results [6].Other than PriorJ, none of the tools give any help, to the prioritization technique is more suitable for a specific context. All activities can be implemented by PriorJ tool. This tool helps to conduct software development and prioritization researchers.

PriorJ has built in support [6] for various new features

1) Data management
2) Early verification
3) Prioritizing of test results.

PriorJ generates results that give numeric and visual comparison [6] of the prioritization results. PriorJ has the JUnit infrastructure which creates a java class that calls the test case using the java reflection commands.

A code change analysis must be performed if a change based prioritization technique is chosen. PriorJ has the dedicated module that discovers the change [6] methods and statements. PriorJ is an easily extendable tool.

5. CONCLUSION

In this paper, test case prioritizations have been described which shows their abilities to improve quick fault detection by those suites. In the present work we have a detailed study on several test case prioritization techniques. After conducting a deep study on these prioritization techniques, we had certain issues and challenges on the test case prioritization techniques. The results of the study suggest, the prioritization techniques reveal high risk faults earlier**.**

6. REFERENCES

[1] "Test case Prioritization Using Fault Severity," Dr. Varun Kumar, Sujata, Mohit Kumar In Proceeding of the International Journal of Computer Science and Telecommunications Vol 1,Issue 1,Sep 2010

[2] "Test Case Prioritization: An Empirical Study," Gregg Rothermel, Roland H. Untch, Chengyun Chu, Mary Jean Harrold In Proceeding of the International Conference on Software Maintenance, Oxford,UK,Sep, *1999*.

[3] " Regression Based Test Case Selection: An Empirical Study," Gurinder Singh and Dinesh Gupta In Proceeding of the International Journal for Science and Engineering 8-11 (2013)

[4] "Comparative Study of Test Prioritization Testing Technique," Rimmi Saini, Deepa Gupta, Ajay Rana In Proceeding of the International Journal of Engineering Sciences & Research (3): May 2012

[5] "Prioritizing test cases for Regression Testing," Sebastian Elbaum, Alexey G. Malishevsky, Gregg Rothermel presented in International Symposium of Software Testing and Analysis, 102-112, August 2000.

[6] "Test Case Prioritization Using PriorJ,"
Everton L. G. Alves, Samuel T. C. Santos, Patricia D. L. Machado, Tiago Massoni.

[7] "Test Case Prioritization," Praveen Ranjan Srivastava In Proceeding of the Journal of Theoretical and Applied Information Technology

[8] "A Study on Test Coverage in Software Testing," Muhammad Shahid,Suhaimi Ibrahim,Mohd Naz'ri Mahrin In Proceeding of the International Conference on Telecommunication Technology and Applications, vol.5 (2011).

[9] "An Approach to Cost Effective Regression Testing in Black-Box Testing Environment," A. Amanda Rao, Kiran Kumar J In Proceeding of the International Journal of Computer Science Issues,Vol. 8,Issue 3,No. 1,May 2011.

[10] "A Static Approach to Prioritizing JUnit Test Cases," Hong Mei, Dan Hao, Lingming Zhang, Lu Zhang, Ji Zhou and Gregg Rothermel IEEE Transactions on Software Engineering, Vol. 38, No.6, Dec 2012.