Test Case Prioritization Using Fault Severity

Dr. Varun Kumar[1], Sujata[2], Mohit Kumar[3]
[1,2,3] Department of Computer Science and Engineering, ITM University, Gurgaon, India.
E-mail : varunkumar333@rediffmail.com, sjt.gpt@gmail.com, tomohitkumar@gmail.com

Abstract : *Regression testing is the process of executing all or some of the tests that already have been conducted to ensure that no new errors have been introduced in the previously tested code. It is very expensive technique. To reduce the cost of regression technique and to increase the effectiveness of testing process we can prioritize the test cases. To date there are number of techniques have been proposed for test case prioritization to achieve the testing objective early in the testing process. One such objective is the increase rate of fault detection early in the testing process. All these techniques are based on code coverage and focus on finding the maximum no of faults rather than severity of faults. In this paper we have proposed a new approach which considers the severity of faults based on requirement prioritization. Aim is to find the severe faults early in the testing process and hence to improve the quality of the software according to customer point of view. Experiment results indicate that our prioritization approach frequently yields faults with high severity.*

Keywords : *test case prioritization, regression testing, requirement prioritization, fault severity, rate of fault detection*

I. Introduction

Regression testing is the re-execution of some subset of test that has already been conducted. [3,5] In regression testing as integration testing proceeds, number of regression tests increases and it is impractical and inefficient to re execute every test for every program function if once change occurs. It is an expensive testing process used to detect regression faults. Research has shown that at least 50% of the total software cost is comprised of testing activities. Companies are often faced with lack of time and resources, which limits their ability to effectively complete testing efforts [5]. Prioritization of test cases in the order of execution in a test suite can be beneficial. Test case prioritization (TCP) techniques organize the test cases in a test suite, allowing for increase in the effectiveness of testing.

We extend the code coverage and function coverage TCP techniques and apply TCP at faults severities. Aim is to early detection of severe faults in the regression testing process and to improve the software quality according to consumer point of view and business value.

We present a new approach for prioritizing the execution of existing test cases by relating them with the requirement severity. The priority is considered to be a function of how desirable it is to include a specific feature. Every proposed feature is rated and assigned a weighting factor.

The main idea of this approach is to group all reused test cases and link with them the requirement which are highly important from business point of view. In order to measure their abilities in detecting regression faults, we also perform an experiment.

II. Related Work

The test case prioritization problem 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 ′ εPT such that (for allT ″)(T ″ ε PT) [f(T ′) ≥ f(T ″)].

In this definition, PT is the set of possible prioritizations (orders) of T, and f is an objective function that applied to any such order, yields an award value for that order.

There are many possible goals for prioritization. For example, testers may wish to increase the coverage of code in the system under test at faster rate, increase their confidence in the reliability of the system at faster rate, or increase the rate at which test suits detect faults in that system during regression testing [6].

A. Code-Based Test Prioritization

The source code of the system is used to prioritize the test cases in code-based test prioritization. These techniques are dependent on information relating the tests of the test suite to various elements of a system's code of the original system (before modification) [4]. For example, a particular code-based technique can utilize information about the number of statements executed, or the number of branches of the code executed, by a test. The system code then executes the test suite and information about executed code elements is collected for each test. Test cases can be prioritized by analyzing the collected information and applying different types of test prioritization heuristics [3]. Achieving early fault detection during regression testing of a modified system code is the main purpose of code-based test prioritization [6]. Experimental studies investigated the effectiveness of various code-based techniques with respect to early fault detection.

A code-based prioritization technique includes total statement coverage, total function coverage, additional statement coverage, additional function coverage and others.

Total statement coverage technique [1,4]: test coverage is measured in terms of program statements. Tests can then be prioritized in terms of the total number of statements they cover by counting the number of statements covered by each test case and then sorting the test cases in descending order of that number. The test cases are prioritized randomly if more than one test covers the same number of statements. For this only the original code is used and not the modified version.

Additional statement coverage prioritization [1, 4] iteratively selects a test case that yields the greatest statement coverage, and then adjusts the coverage information on all remaining test cases to indicate their coverage of statements not yet covered and repeats this process until all statements covered by at least one test case. If more than one test has the same maximum number of statements covered then a test is selected randomly from that set of tests. The remaining tests have the coverage

information adjusted to reflect the coverage of statements not yet covered by the selected test.
Function coverage [1,4] is similar to total statement coverage except that only function level is used in the system code. This technique is less expensive and less intrusive as compared to statement coverage. Additional Function coverage is the same as additional statement coverage using the functions in the system code.
Other code-based prioritization techniques include information from the modified system code version. Total Fault Index prioritization, each function is assigned a fault index representing the fault proneness based on functional complexity and complexity of changes introduced in the function. Fault indexes of the original system code and for the modified code are generated and compared. Heuristics similar to total function coverage or additional function coverage can be applied to the fault indexes information collected for each test in the test suite. The specific focus of this paper and that of is to find an ordering of test cases T ′ of a suite T with the goal of increasing the likelihood of revealing severe faults earlier in the testing process.

Test case prioritization using fault severity based on requirement weights.

Following factors can be considered to assign the weights to the requirements:

A. Business Value Measure (BVM)
BVM is a measure in which the requirements are given most importance which is critical to customer's business. The customer assigns a value to each requirement which can range from 1 to 10. Most critical requirements are assigned the highest number (10) and least important requirement is assigned the lowest number (1). Use cases can also be used in analyzing the requirement [2]. Approximately 45% of the software functions are never used, 19% are rarely used, and only 36% of the software functions are sometimes or always used. In case if project running out of time the test efforts should be prioritized based on the rating assigned to requirements. By analyzing and testing the fraction of the requirement with highest importance should be tested first, this is helpful in increasing the customer confidence level as it increases the business value.

B. Project Change Volatility (PCV)
PCV is based on the how many times consumer is modifying the project requirements during the software development cycle. PCV is one of the criteria which help to assess the requirement changes after the start of the implementation.
PCV increase the test efforts significantly and make it difficult to complete the project on time. And due to pressure and lack of time sometimes it leaves the project with high defect of density. Severe defects that escape into the field can cost 100 times more to fix after delivering the project. The most significant factor to cause these project failures were attributed to changing requirements. The biggest causes for the project failures happen to be the lack of user inputs, and changing or incomplete requirements. Roughly 25% of the requirement for an average project change before project completion.

C. Development Complexity (DC)
Each requirement analyzed based on the implementation complexity. Development efforts, technology, environmental constraints and requirement feasibility matrix are considered while measuring implementation complexity. It is becoming difficult to ensure the expected performance of the software. While functioning in an environment, the external behavior of any software system has to be observed. The behavior of its components and the interaction among them is reflected through this external behavior. When the interaction between different components is governed by simple, well-defined and deterministic rules, the overall eternal behavior of the system is predictable to a high degree of accuracy. This predictability provides the basis for differentiating simple or complex systems. Also the non functional requirements are very important for any software.

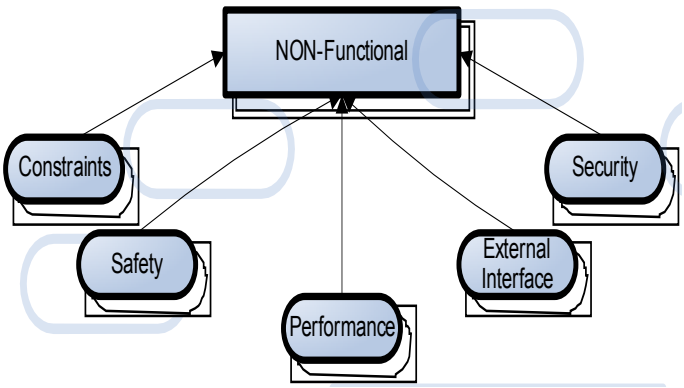

Fig-1

Requirements with higher complexity tend to have a higher number of faults. Security and performance related requirements should have higher rating because of its criticality.

D. Fault Proneness of Requirements
Developer can identify the requirements which are error prone based on historical data [5]. This also includes the requirement failures reported by the customer. As different versions of the system released the data collected from the previous versions can be analysed. FP is based on the number of filed failures and in-house system test failures found in the code that implements a requirement.

Research has shown that fault prone modules are more likely to cause field failures than the modules, which are not fault prone.
Example table (Table-1) for assigning the priority which can be prepared by customer and development team.

Table 1: Sample Prioritization

Factors	R^1	R^2	R^3	R^4	Weights
Business Value Measure	7	4	10	8	0.35
Project Change Volatility	10	5	4	9	0.35
Development Complexity	10	8	5	5	0.15
Fault Proneness of Requirements	4	6	5	7	0.15
WP	8.05	5.25	6.4	7.75	1.00

Weights to each factor are assigned by the development team according to the project.
Assigned total weight (1.0) is divided amongst the PFs. For every requirement, Equation 1 is used to calculate a weighted prioritization (WP) factor that measures the importance of testing a requirement earlier.

$$WP = \sum_{PF=1}^{n} (Pf\ value * Pf\ weight) \quad (1)$$

WP represents the weighted prioritization for the four requirements computed using Equation 1. The results of the table show the prioritization of test cases for the four requirements as follows: R1, R4, R3, and R2. The WP will vary with a change in factor-weights and factor-values.

IV. Test Case prioritization algorithm

Each fault is assigned severity measure (SM) on a 10 point scale as shown below:

- Complex (Severity 1): SM value of 9-10
- Moderate (Severity 2): SM of 6
- Low (Severity 3) :SM of 4
- Very Low (Severity 4): SM of 2

To calculate the Total percentage of fault detected (TSFD), we use severity measure(SM) of each fault. Once the fault has been detected then we assign some severity measure to each fault according to requirement weights, to which it is mapped.

Total Severity of Faults Detected (TSFD) is the summation of severity measures of all faults identified for a product.

$$TSFD = \sum_{i=1}^{n} SM\ (severity\ measure)$$

This equation shows TSFD for a product where n represents total number of faults identified for the product.

Input: Test suit, T and total factor value of all requirements.
Output: Prioritized test suit T'.
Algorithm:
Build the requirement prioritization matrix based on the priority (*Table-1*).
Select *T'* from *T based on the* requirement coverage
Draw a test case from *T'*. Run it and find fault severity using table-1.
Reorder rest test cases using fault severity information
Repeat step 3 and 4 until testing
resource is exhausted.

First step is to build the prioritization matrix based on the factor value of the requirement. Then map the test cases against each requirement. Execute the test cases based on the assigned priority and analyze the results based on fault severity.

A. Evaluation of new approach

For the evaluation of this approach consider the example program with number of fault detected for particular test cases and severity value against those faults.

Table-2: Faults exposed and assigned severity values to those faults

	Assign Severity value								
Fault detected for some test case.	1	2	3	4	5	6	7	8	9
T1 - 3	x	x	x						
T2 - 4	x		x	x	x				
T3 - 2		x	x						
T4 - 2								x	x
T5 - 4			x	x			x		

Execute the test case first which has highest severity values. The total severity becomes for the test cases as given below: T1=6, T2=13, T3=5, T4=17, T5=14.

These severity values assign to these detected faults are shown in table. According to our approach test cases will be executed in the order: T4-T5-T2-T1-T3
Comparison of this proposed approach with the random and fault base prioritization is explained in the following table.

Table 3: Fault-Severity Based prioritization

T1	F1	F2	F3						
T2	F1		F3	F4	F5				
T3		F2	F3						
T4								F8	F9
T5			F3	F4			F7		

T4	T5	T2	T1	T3
2 faults	3faults	4 faults	3faults	2 faults
17 sev.	14 sev.	13 sev.	6 sev.	5sev.

This approach finds 7 unique and severe faults early in the testing process. (up to execution of T2 test case.) If the severity of the two test cases is same then we can prioritized them in any random order. For example if the severity of T1 and T2 is 3 then either we can execute T1 first or we can execute T2 first.

Table 4: Fault-Based Prioritization

T1	x	x	x						
T2	x		x	x	x				
T3		x	x						
T4								x	x
T5			x	x			x		

T2	1	T5	T3	T4
4 faults	3faults	3 faults	2faults	2 faults
13 sev.	6 sev.	14 sev.	5 sev.	17sev.

This approach finds 6 unique faults early in the testing process (up to the execution of T5) but they may or may not be severe.

In fault based prioritization technique test cases are arranged in descending order of finding maximum number of faults. So the order of prioritization is:

T2-T1-T5-T3-T4
If the two test cases are finding same number of faults then they can be prioritized according to some random order.

Table 5: Random Prioritization:

T1	x	x	x						
T2	x		x	x	x				
T3		x	x						
T4								x	x
T5			x	x			x		

T2	T3	T1	T5	T4
4 faults	2 faults	3 faults	3faults	2 faults
13 sev.	5 sev.	6 sev.	14 sev.	17sev.

This approach is finding 5 (up to the execution of T1 test case.) unique faults early in the testing process, again they may or may not be severe ones.
The comparison is drawn for the prioritized and non prioritized test suit and the results shows that prioritized test suit is more effective than the non-prioritized one.
Test suite fraction (TSF) is taken on x-axis and total severity of fault detected (TSFD) is taken on y-axis.

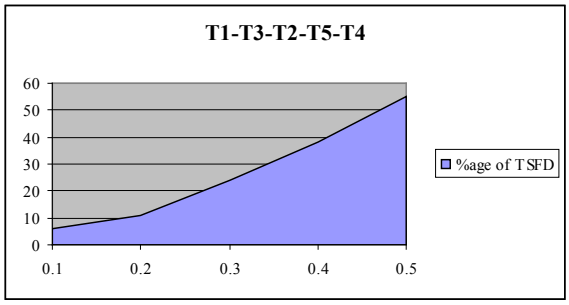

Fig-2

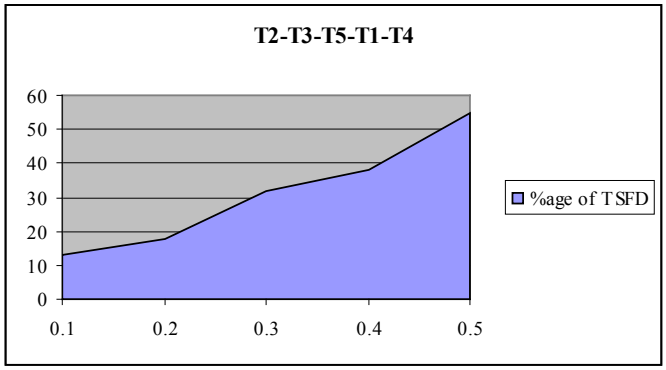

Fig-3

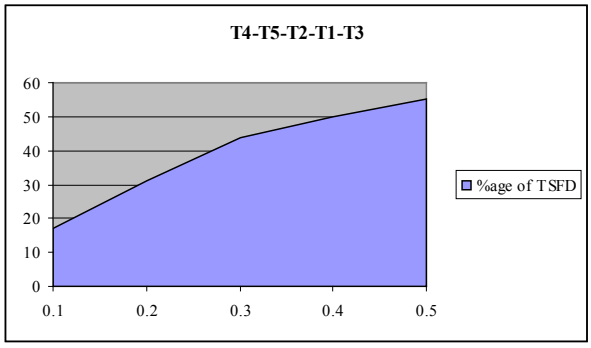

Fig-4

Fig-2 shows when the test cases are executed in some random order which isT1-T3-T2-T5-T4. The area covered is approximately 44%.

Fig-3 shows when the test cases are executed in some another random order which is: T2-T3-T5-T1-T4.The area covered is approximately 48%.

Fig-4 shows the results when the test case prioritization is used. The order is T4-T5-T2-T1-T3 which is according to faults severity. Area covered is approximately 72%.

We have implemented this algorithm on some project and the results of comparison of random prioritization and Test case prioritization based on faults severity is shown in Fig.-5. We have used the WPFD (weighted percentage of fault detected) metric to measure the effectiveness of this algorithm.

With increase in the % Test suit fraction we found increases in the %TSFD. Hence graph proves the effectiveness of new algorithm as it detects the severe faults early in the testing process.

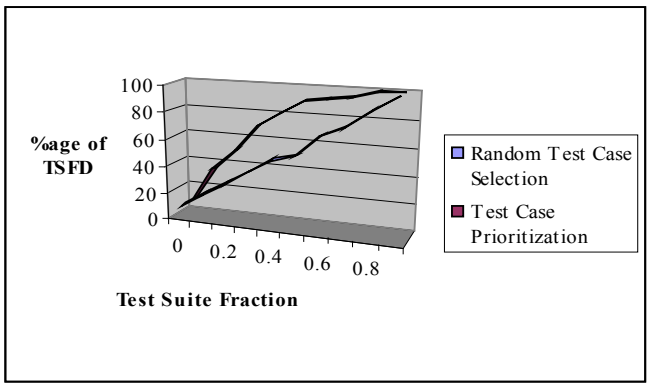

Fig-5

V. Conclusion

This paper proposed an algorithm for test case prioritization to improve the . regression testing . Analysis is done for the prioritized and non prioritized cases using WPFD metrics. Graphs shows that prioritized is more effective than the non prioritizedtest suit.
In future we will try on test case prioritization over requirement analysis using APFD metrics

References

[1] 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, February, 2002.
[2] F. Moisiadis, *"Prioritizing Use Cases and Scenarios,"* 37th International Conference on Technology of OO Languages and Systems, Sydney, NSW, 2000, pp. 108-119.
[3] R. Gupta, M. Harrold, M. Soffa, *"An Approach to Regression Testing Using Slices,"* Proc. IEEE International Conference on Software Maintenance, pp. 299-308, 1992.
[4] 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, pp. 929-948, Oct. 2001.
[5] Roger S. Pressman, Software engineering a practitioner's approach 6/e, 2005
[6] Sebastian Elbaum, Gregg Rothermel, Satya Kanduri,

Alexey G. Malishevsky, Selecting a Cost-Effective Test Case Prioritization Technique, April 20, 2004

Dr.Varun Kumar received his Ph.D in Computer He has more than 12 years of teaching experience in the field of Computer Science and Engineering. Presently, He is working as an HOD, Computer Science and Engineering in ITM University, Gurgaon, Haryana, India. His area of specialization is OOAD, Artificial Intelligence, Data Mining, Web Technologies.

Sujata received her M.Sc. in Computer Science and M.Tech. in Software Engineering. She has more than 4 years of teaching experience in the field of Computer Science. Presently, She is working as an Assistant Professor in ITM University, Gurgaon, Haryana, India. Her area of specialization is Software Testing, Quality and UML.

Mohit Kumar received his M.S. in Software Systems, BITS, Pilani and He has also completed his Masters degree in Computer Application (M.C.A.). He has 11 years of experience in the field of Computer Science and Technology. Presently, he is working as a Project Manager in ICF International India Pvt. Limited. His Domain of expertise is Supply Chain Management System, E-commerce (B2B and B2C), Project Management, Billing and Revenue Management, Billing through Handheld Device (First time implemented in India), Price analysis and forecasting for capacity, transmission, environmental cost for power plants.