Prioritization based Test Case Generation using Regression Testing

L.UmaRani
Student,
Department of Computer Applications,
Sri Ramakrishna Engineering College,
Vattamalai palayam, coimbatore-641022.

R.Pradeepa
Assistant Professor,
Department of Computer Applications,
Sri Ramakrishna Engineering College,
Vattamalai palayam, coimbatore-641022.

ABSTRACT

Test case prioritization involves scheduling test cases in an order that increases the effectiveness in achieving some performance goals. One of the most important performance goals is the rate of fault detection. Test cases should run in an order that increases the possibility of fault detection and also that detects the most severe faults at the earliest in its testing life cycle. A test case prioritization, assigns each test case a selection probability according to its potential ability to achieve some certain testing goal. It selects prioritized test cases to run to get higher testing Performance. Test case prioritization techniques could be of great benefit to increasing the effectiveness of test suites in practice. Test case prioritization is a technique. It helps to increase the rate of fault detection or code. Whereas Regression test prioritization is completed in a time constrained execution environment in which testing occurs for only a fixed time period. Based on test case prioritization using regression testing calculating the efficiency for each test cases. This can support to make a better software product.

Keywords: Test Case, Regression Testing, Test Case Prioritization, Error, Testing Efficiency.

1. INTRODUCTION

Test cases are manually generated for every software. As a result it consumes more time for testing and increases the complexity. Regression testing is retesting changed segments of application system. It is performed frequently to ensure the validity of the altered software. In most of the cases, time and cost constraint is prominent; hence the whole test suite cannot be run. Thus, prioritization of the test cases becomes essential. The priority criteria can be set accordingly e.g. to increase rate of fault detection, to achieve maximum code coverage.

Regression testing is a necessary but expensive process in the software lifecycle. One of the regression testing approach, test case prioritization , aims at sorting and executing test case in the order of potential abilities to achieve certain testing objective.

1.1 Test Case Prioritization

Test case prioritization techniques schedule test cases in an execution order according to some criterion. These criteria can test case cost basis or time basis. The purpose of prioritization is to detect faults in the system in minimum time or cost. It also improves the functionality of a quality product. Test case prioritization can address a wide variety of objectives to increase the rate of fault detection that revealing faults earlier in a run of regression tests. It also increases the rate of detection of high-risk faults, locating those faults earlier in the testing process. It may be wish by a testers to increase the likelihood of revealing regression errors related to specific code changes earlier in the regression testing process and to increase their coverage of coverable code in the system under test at a faster rate. With help of this testers may wish to increase their confidence in the reliability of the system under test at a faster rate.

1.2 Regression test selection.

Retest all technique takes time and effort as all test cases are used to test the program again, so may be quite expensive. This technique much better as it uses information about program, modified program, test cases to select subset of test cases for testing.

2. AVERAGE PERCENTAGE FAULT DETECTION(APFD)

Average Percentage of Faults Detected (APFD) metric [1] was used to determine the effectiveness of the new test case orderings, but it considered faults and test cases cost to be uniform.

2.1 Test Case Prioritization Problem

Given: T is a test suite; PT is the set of permutations of T; (all possible prioritizations of T) f is a function from PT to a real number (award value) then Find T' Є PT such that (for all T") T" Є PT, (T" != T') [f (T') >= f (T")

Here, PT represents the set of all possible prioritizations (orderings) of T and f is a function that, applied to any such ordering, yields an award value for that ordering.

Empirical Study: APFD Measures [1] Let T be a test suite containing n test case,and let F be a set of m faults revealed by T .Let T Fi be the first test case in ordering T, of T which

reveals fault I . The APFD for test suite T' is given by the equation:

APFD = 1- [(TF1+TF2+TF3+….+TFm) / nm] + [1/(2*n)]

Table-1a

Fault >> test Cases	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
T1	x				x					
T2					x	x	x			
T3	x	x	x	x	x	x	x			
T4					x					
T5								x	x	x

APFD =1-[(1+3+3+3+1+2+2+5+5+5)/ (5*10)]+(1/(2*5)

=1- (30/50) +(1/10)
= 1-0.6+0.1
= 0.4+0.1
= 0.50
= 50%

Table-1b

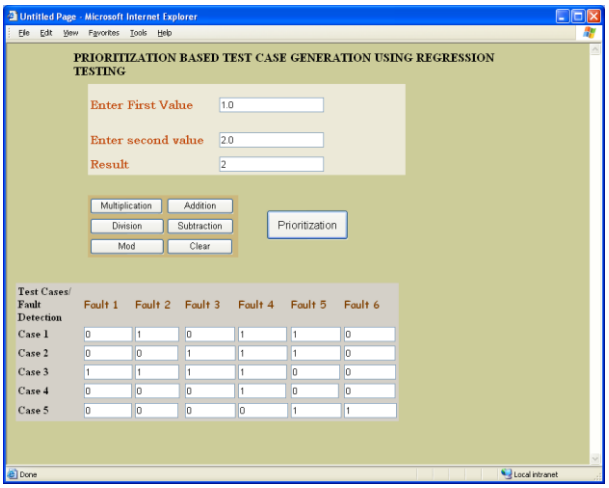
Fault >> test Cases	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
T3	x	x	x	x	x	x	x			
T5								x	x	x
T2					x	x	x			
T1	x				x					
T4					x					

APFD = 1- [(1+1+1+1+1+1+1+2+2+2)/ (5*10)]+(1/(2*5)

=1- (13/50) +(1/10)
= 1-0.26+0.1
= 0.74+0.1
= 0.84
= 84%

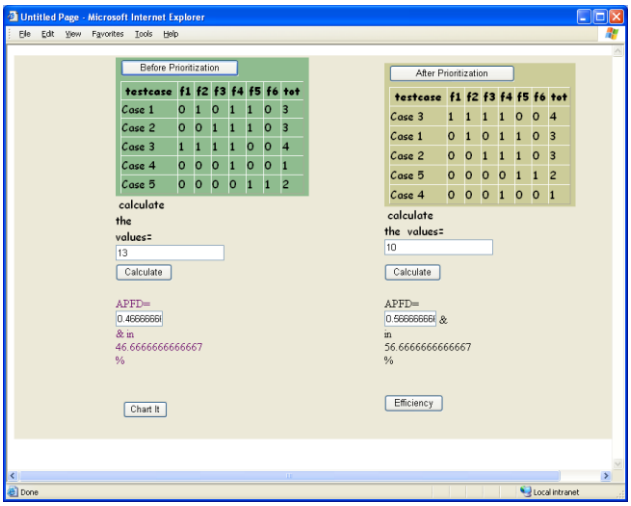
In similar way we can find out APFD for all permutation of test suit .Thus in regression testing ,out of all APFD value the maximum percentage value is most priority test suit for best result in minimum time execution. In other words fault are found as early as possible in regression testing help of test case prioritization.

Figure-1 Prioritization



In Figure. 1, the test case 3 identifies most number of faults. The test case 3 is top in the prioritized order and it is executed first. Thus the proposed prioritization technique will identify most number of severe faults at an early stage.

Figure-2 APFD calculation



In Figure. 2, APFD percentage of test cases executed against percentage of faults detected. The the proposed prioritization technique will identify most number of severe faults at an early stage.

All APFD can be calculated with the following C Language functional code

```
for(i=0;i<tn;i++)
{
for(j=0;j<=fn;j++)
 printf("%3d",t[i][j]);
 printf("\n");
```

```
}
printf("total test case in test suit>>");
 scanf("%d",&tsn);
 printf("enter test case order in test suite>>>"); //i.e.3 5 2 4 1
for(i=0;i<tsn;i++) s
canf("%d",&b[i]);
 printf("\n");
for(i=0;i<tsn;i++)
printf(" b=[%d]%d",i,b[i]);
 m=0; for(i=0;i<tsn;i++)
{
for(j=0;j<tn;j++)
 {
if(b[i] == t[j][0])
 { for(k=0;k<=fn;k++)
c[m][k]=t[j][k];
} }
m++;
}
 printf("\ntcno f1 f2 f3 f4 \n");
for(i=0;i<tsn;i++)
 {
for(j=0;j<=fn;j++)
 printf("%3d",c[i][j]);
printf("\n");
}
for(j=1;j<=fn;j++)
 {
count=0;
for(i=0;i<tsn;i++)
 {
 if(c[i][j] == 1)
{
count=count+1;
}
 if(count>1)
 {
 c[i][j]=0;
count=1;
} } }
for(i=0;i<tsn;i++)
 {
for(j=0;j<=fn;j++)
printf("%3d",c[i][j]);
printf("\n");
 }
s=0;
sum=0;
for(i=0;i<tsn;i++)
 for(j=1;j<=fn;j++)
{
if(c[i][j] ==1) f[++s]=i+1;
}
for(i=0;i<s;i++)
 printf(" %d \t ",f[i]);
for(i=1;i<=s;i++)
sum=sum+f[i];
 if(s<fn)
{
sum =sum+fn;
}
```

In above C-Language code we can calculate APFD for different test suite and compare the results for test case prioritazation .tn is number of test cases ,fn is number of faults ,tsn-number of test case in test suit.

fault found by a test case represent show by 1 value in corresponding row and column oterwise it represent 0 value .All 0 and 1 value stored in two dimensional array.

3. TIME BASED TEST CASE PRIORITIZATION

There are many existing approaches to regression test prioritization that focus on the coverage of or modifications to the structural entities within the program under test [3], [4], [5]. None of these prioritization schemes explicitly consider the testing time budget like the time-aware technique presented in paper [6] Elbaum et al. and Rothermel et al. focus on general regression test prioritization and the identification of a single test case reordering that will increase the effectiveness of regression testing over many subsequent changes to the program. They had described a time-aware test suite prioritization technique. Experimental analysis demonstrates that their approach can create time-aware prioritizations that significantly outperform other prioritization techniques.

3.1 Measures of Time-Aware Test Suite Prioritization

Kristen R. Walcott et al [6] described other factors of test case prioritization i.e. Time-Aware Test Suite Prioritization .Problem description as below time aware test suit prioritization Given: (i) A test suite, T, (ii) the collection of all permutations of elements of the power set of permutations of T, perms (2T) , (iii) the time budget, tmax, and (iv) two functions from perms(2T) to the real numbers, time and fit.

Problem: Find the test tuple Smax belongs to perm (2T) such that time (Smax)<=tmax and for all S' belongs to perms(2T) where Smax is not equal to S' and time (S') <=tmax ,fit(smax)>fit(s').

For example, suppose that regression test suite T contains six test cases with the initial ordering for T that contains (T1; T2; T3; T4; T5; T6), as described in Table 2. For the purposes of motivation, this example assumes a priori knowledge of the faults detected by T in the program P. As shown in Figure 1(a), test case T1 can find seven faults, (F1,F2,F3,F4,F5,F6,F7), in nine minutes, T2 finds one fault, (F1), in one minute, and T3 isolates two faults, (F1,F5), in three minutes.

Test cases T4; T5; and T6 each find three faults in four minutes,(F2,F3,F7) ,(F4,F6,F8) and ,(F2,F4,F6), Respectively. Supposing that the time budget for regression testing is twelve minutes. Without a time budget, the test tuple (T1; T4; T5; T6; T3; T2) would execute. Out of this, only the test tuple S1= T1 would have time to run when under a twelve minute time constraint and would only a total of seven faults, as noted in Table 2(b). Since time is a principal concern, it may also seem intuitive to order the test cases with regard to their execution time. In the time constrained environment, a time-based prioritization S2= (T2T3; T4; T5) could be executed and find eight defects, as described in Table-2(b). Another option would be to consider the time budget and fault information together.Therefore, the \intelligent" prioritization, S4, is favored over S2 because it is able to detect more faults earlier in the execution of the tests.

Table-2a

Test Case	#Fault	Time cost (min)	Avg. Fault per min.
T1	7	9	0.778
T2	1	1	1.0
T3	2	3	0.667
T4	3	4	0.75
T5	3	4	0.75
T6	3	4	0.75

	F1	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**
T1	**X**	**X**		**X**	**X**	**X**	**X**	**X**
T2	**X**							
T3	**X**				**X**			
T4		**X**	**X**				**X**	
T5				**X**		**X**		**X**
T6		**X**		**X**		**X**		

Table-2b

	Time Limit: 12minutes			
	Fault S1	Time S2	APFD S3	Intelligent S4
	T1	T2 T3 T4 T5	T2 T1	T5 T4 T3
Total Faults Total Time	7 9	8 12	7 10	8 11

Table-2c

3.2 Prioritization for Multiple Processing Queues

Test case prioritization is an effective technique to detect the faults with minimum time or cost. It helps to increase the rate of fault detection or code coverage in regression testing.

3.3 The Test Case Prioritization Problem in Parallel Scenario

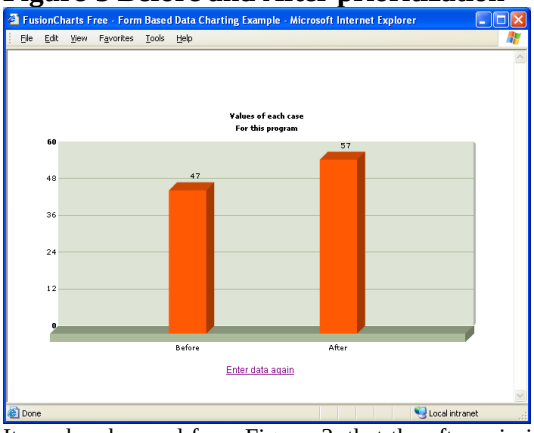
As per Bo Qu Changhai Nie et al [7] all existing methods can only prioritize test cases to a single queue. Once there are two or more machines that participate in testing, all exiting techniques are not applicable any more. To extend the prioritization methods to parallel scenario, this paper defines the prioritization problem in such scenario and applies the task scheduling method to prioritization algorithms to help partitioning a test suite into multiple prioritized subsets. Basically in Existing test case prioritization techniques prioritize test cases in a single set, potentially assuming that there is only one processing queue that selects test cases to run[8,9]. However, there would often be two or more testers/machines that participate in testing in practice.

Comparison of two different definition of prioritization in parallel scenario involves one more step, dividing test suite to several subsets. So at a high level, test case prioritization in parallel scenario works as follow: (1) apply an RTS technique to test suite *T*, yielding *T'*, (2) divide *T'* to several subsets, (3) assign a selection probability to each test case in every subsets, (4) for each processing queue, select a test case from corresponding subsets using the probabilities assigned in step 3, and run it, (5) collect feedbacks and adjust the probabilities if necessary, and (6) repeat step 4 until there is no more test case or resources are exhausted. But problem is how to divide the test suite and set their selection probabilities.

3. BEFORE PRIORITIZATION AND AFTER PRIORITIZATION

The comparison between before prioritization and after prioritization test case, which shows that number of test cases needed to find out all faults are less in the case of prioritized test case compared to non prioritized test case. It can be observed from Figure. 3, that the after prioritization technique needs 57% of test cases to find out all the faults. But 47% of test cases are needed to find out all the faults in the case of before prioritization, if test cases are executed in before prioritized order.

Figure-3 Before and After prioritization



It can be observed from Figure. 3, that the after prioritization technique needs 57% of test cases to find out all the faults. But 47% of test cases are needed to find out all the faults in the case of before prioritization, if test cases are executed in before prioritized order. APFD percentage of test cases executed against percentage of faults detected.

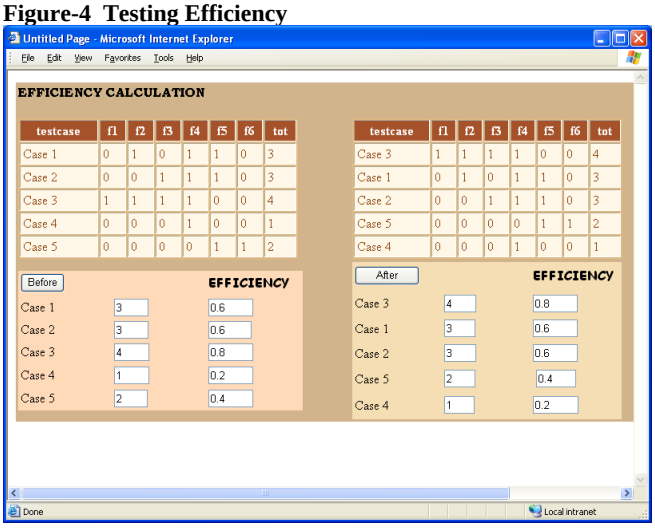
5. TESTING EFFICIENCY

Test efficiency is not only about test execution alone, but all or most of test activities, like test planning, comprehension, test cases creation, review, execution, defect tracking and closure.

Test Efficiency helps to calculate the efficiency of testing i.e. how many defects were leaked to the customer as compared to number of defects reported by the testing team. Generally almost 10-15 % of defects will be leaked and is considered acceptable. In the recent years, Companies have stared spending huge amount of money for developing quality. Due to this defect leakage percentage has come down to less than 10%.

Testing Efficiency(Test Case Based) = Number of Defects Found by test case/Number of test cases

Figure-4 Testing Efficiency



From the Figure.4, it is observed that the prioritized test cases identify the faults at an early stage. The APFD measures of prioritized test cases are higher In Figure.1, Test case 3 can identify more number of faults when compared to others. Thus Test case 3 will be first executed. From Figure.1, we observed that the proposed method identifies the severe fault in the early stage. So it will reduce the computational time.

6. CONCLUSION

Regression testing is the verification that previously functioning software remains after a change. A large number of test case executions are expensive and time consuming during regression testing. Where Test case prioritization (TSP) is an effective and practical technique in regression testing to reduce it. It schedules test cases in order of precedence that increases their ability to meet some performance goals, such as code coverage, rate of fault detection. We have study and conclude that prioritization of test case or test suits have different aspects of fault detection. On the basis of prioritization techniques, functionality of regression testing can improved in minimum time and recourses. Based on test case prioritization using regression testing calculating the efficiency for each test cases.This can support to make a better software product. Test effectiveness and cost can be calculated for future work.

REFERENCES

[1] S. Elbaum, A. Malishevsky and G. Rothermel. Incorporating varying test costs and fault severities into test case prioritization. In Proceedings of the 23rdInternational Conference on Software Engineering, pages 329-338, May 2001.

[2] Gregg Rothermel et al" Test Case Prioritization: An Empirical Study"PICSM , Oxford, UK, September, 1999

[3] H. Do, G. Rothermel, and A. Kinneer. Empirical studies of test case prioritization in a JUnit testing environment. In Proc. Of 15th ISSRE, pages 113{124, 2004.

[4] S. Elbaum, A. G. Malishevsky, and G. Rothermel. Test case prioritization: A family of empirical studies. IEEE Trans. Softw. Eng., 28(2):159{182, 2002.

[5] G. Rothermel, R. J. Untch, and C. Chu. Prioritizing test cases for regression testing. IEEE Trans. on Softw. Eng.,27(10):929{948, 2001.

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

[7] Bo Qu Changhai Nie et al, "Test Case Prioritization for Multiple Processing Queues", ISISE-08,pg. 646-49 IEEE, 2008.

[8] G. Rothermel, R. H. Untch, C. Y. Chu, M. J. Harrold. Prioritizing test cases for regression testing. IEEE Transactions on Software Engineering, 27(10):929-948, October 2001.

[9] 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.

[10] W.E. Wong, J.R. Horgan, S. London, and H. Agrawal, "A Study of Effective Regression Testing in Practice," Proc. Eighth Int'l Symp.Software Reliability Eng., pp. 230-238, Nov. 1997.

[11] Sandeep Kaur and Rajandeep Kaur, "Various Techniques of Regression Testing," An International Journal of Engineering Sciences, Vol. 4, pp.391-397, Sep 2011.

[12] Xiaofang Zhang, Changhai Nie, Baowen Xu and Bo Qu, "Test Case Prioritization based on Varying Testing Requirement Priorities and Test Case Costs", Proceedings of Seventh International Conference on Quality Software (QSIC'07), 2007.

[13] Hema Srikanth and Laurie Williams, "Requirements-Based Test Case Prioritization", North Carolina State University, ACM SIGSOFT Software Engineering, pages 1-3, 2005.

[14] Elbaum S, Rothermel G, Kanduri S, Malishevsky AG (2004) Selecting a cost-effective test case prioritization technique. Soft Qual J 12(3):185–210.