PRIORITIZATION BASED TEST CASE GENERATION IN REGRESSION TESTING

K.P. Jayant, Research Scholar & Sr. Associate Professor, CSE, ABESEC, Ghaziabad
Prof. (Dr.) Ajay Rana, Director, Amity University

ABSTRACT

This paper is a study on test case prioritization based on cost, time and process aspects. Prioritization concept increases the rate of fault detection or code in time and cost constraints. Test case, test suite prioritization's approaches are used in regression testing. It is because testing plays an important role in software development life cycle. In this paper we have made a survey of an important role of prioritization and their techniques. There are several test suite and test case prioritization techniquesexist.

Keywords" *Test, Regression, Error, Probability, Prioritization, Software, Techniques,Cycle.*

INRODUCTION

Testing is a process of executing a program with the intent of finding an error. Regression testing is compulsory for proper functioning of software product as per current market trend. It is an expensive, time consuming endless process in the software development 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 [1].

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.

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. Gregg Rothermel [2] faults can detect on the basis of different prioritization techniques as -No prioritization, Random prioritization, Optimal prioritization, Total branch coverage prioritization, additional branch coverage prioritization, Total fault-exposing-potential (FEP) prioritization and Additional fault-exposing-potential (FEP)prioritization.

STUDY OF CONCEPT GENERATION

Rothermel et al. [5] define the test case prioritization problem and describe several issues relevant to its solution. The test case prioritization problem is defined (by Rothermel et al.) as follows:

The 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 case	**F1**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**
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 case	**F1**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**	**F**
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.

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++)
     scanf("%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  f1f2f3       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 dimensionalarray.

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.

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 (2^T) , (iii) the time budget, t_{max}, and (iv) two functions from perms(2^T) to the real numbers, time and fit.

Problem: Find the test tuple S_{max} belongs to perm (2^T) such that time (S_{max})<=t_{max} and for all S' belongs to perms(2^T) where S_{max} is not equal to S' and time (S') <=t_{max} ,fit(s_{max})>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. Because we want to find as many faults as possible early on, it would seem intuitive to order the test cases by only considering the number of faults that they can detect. 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= (T_2T3; 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 informationtogether.

In order to do this, we could order the test cases according to the average percent of faults that they can detect per minute. Under the time constraint, the tuple S3=(T2; T1)would be executed and find a total of seven faults. If the time budget and the fault information are both considered intelligently, that is, in a way that accounts for overlapping fault detection, the test cases could be better prioritized and thus increase the overall number of faults found in the desired time period. In this example, the test cases would be intelligently reordered so that the tuple S4 =(T5; T4; T3) would run, revealing eight errors in less time than S2. Also, it is clear that S4 can reveal more defects than S1 and S3 in the specified testing time. Finally, it is important to note that the first two test cases of S2, T2 and T3, find a total of two faultsinfourminuteswhereasthefirsttestcaseinS4T5,detectsthreedefectsinthesametimeperiod.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								
	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			
Test Case	#faults	TimeCost (Mins)	Avg. Faults 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

Table-2c				
	Time limit : 12 minutes			
	Fault S1	Time S 2	APF D S 3	Intelligent S4
	T1	T2 T3 T4 T5	T2 T1	T5 T4 T3
Total faults Total Time	7 9	8 12	7 10	8 11

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 regressiontesting

The Test Case Prioritization Problem in ParallelScenario

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. Therefore, each processing queue should be assigned a prioritized subset of a test suite according to its ability. To extend test case prioritization method to parallel scenario, we have

defined the prioritization problem in such scenario and proposed corresponding algorithms based on task scheduling methods.

The definition of the test case prioritization problem suggests that prioritization techniques aim at finding a best prioritized test suite. However, when there are many processing queues, the definition would no longer be applicable. Based on the characteristics of parallel scenario definition changed [7]

Comparison of two different definition of prioritization in parallel scenario involves one more step, dividing test suiteto 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.

CONCLUSION

Regression testing is the verification that previously functioning software remains after a change. Regression testing is time consuming and expensive process. 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. This can support to make a better software product.

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, May2001.

[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,andG.Rothermel.Testcaseprioritization:Afamilyofempiricalstudies.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, October2001.

[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]Sebastian Elbaum, Alexey Malishevsky, Gregg Rothermel,"Incorporating Varying Test Costs and Fault Severities into Test Case Prioritization" IEEE,Pg:329-338,2001

[11] 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.

Authors would like to extend their heartfelt thanks to the academic and infrastructural support received from their respective Dept./University.