7-12-2020

How Time-Fault Ratio helps in Test Case Prioritization for Regression Testing

Prem Parashar
Sheridan College, prem.parashar@sheridancollege.ca

Arvind Kalia
Himachal Pradesh University

Rajesh Bhatia
Deen Bandhu Chotu Ram University

How Time-Fault Ratio helps in Test Case Prioritization for Regression Testing

Prem Parashar(1), Arvind Kalia(2), Rajesh Bhatia(3)

(1) Computer Science Department,Himachal Pradesh University (India)
E-mail: prem.parashar@gmail.com
(2) Computer Science Department,Himachal Pradesh University (India)
E-mail: arvkalia@gmail.com
(3) Computer Science Department, Deen Bandhu Chotu Ram University(India)
E-mail: rbhatiapatiala@gmail.com

ABSTRACT

Regression testing analyzes whether the maintenance of the software has adversely affected its normal functioning. Regression testing is generally performed under the strict time constraints. Due to limited time budget, it is not possible to test the software with all available test cases. Thus, the reordering of the test cases, on the basis of their effectiveness, is always needed. A test prioritization technique, which prioritizes the test cases on the basis of their Time -Fault Ratio (TFR), has been proposed in this paper. The technique tends to maximize the fault detection as the faults are exposed in the ascending order of their detection times. The proposed technique may be used at any stage of software development.

Keywords: Fault detection, maintenance, prioritization, test suite.

1- INTRODUCTION

Software maintenance is one of the most expensive phases of software development [1, 8, 11, 12, 13, 15, 18]. After making the required modifications in the software, regression testing is performed to check that (i) the changes made in the software do not lead the system to any undesirable behavior [12], and (ii) the changes made in the software meet with the current requirements of the system. In order to assure (i), and (ii), software should be tested with all existing test cases along with the test cases generated for the changed part of it [4]. Due to limited time and other cost constraints, exhaustive testing is not feasible in maintenance phase. Therefore, most promising test cases are selected from test suite for execution. Different researchers have suggested different test case prioritization techniques [10, 15, 18, 20,22]. The common techniques proposed are, Average Percentage of Fault Detection (APFD) based prioritization (a test case with highest value of Fault-Exposing Potential (FEP) is executed first), random test case prioritization (a test case is selected randomly for execution), optimal test case prioritization (A test case which determines maximum new faults, is executed first), coverage-based prioritization etc.

Most of the prioritization techniques mentioned in different studies are based on FEP of test cases. A test case may detect a number of faults and a fault may be detected by many test cases. While prioritizing test cases for regression testing, test cases are arranged in the test suite according to descending values of FEP and are executed in the same order. At the time of prioritization, FEP plays significant role in deciding the priority value of a test case in test suite. Other prioritization techniques are based on the coverage potential of test case. A test case that covers maximum functions, classes or code fragments of software program within minimum time has highest priority in the prioritize test suite. If two or more test cases cover same amount of source program, a tie is broken and that test case is assigned highest priority which covers the critical sections of software [6]. The critical nature of any part of the software is defined by its usability and importance.

The behavior of a fault plays significant role in deciding the order in which it should be retrieved. A fault may be detected by many test cases. In order to reveal a fault, that test case should be executed which detects it in minimum possible time. In proposed technique, the faults are revealed in the ascending order of their total detection time. In current study, It has been considered that time constraints are strict if time budget allowed for regression testing is below half of the time require to execute the test suite designed for it.

2- REVIEW OF LITERATURE

Rothermel [15, 16] et al. compared the results of different test case prioritization techniques through some experiments. Average Percentage of Fault Detection (APFD) and total Fault- Exposing Potential (FEP) are the metrics used for measuring the effectiveness of each technique. From the experimental studies, it has been observed that most of the times total FEP coverage based test case prioritization performs better than rest of the techniques mentioned in their studies, though results could not be generalized due to varying efficacy of techniques from one program to another. Ma, J. and Zhao, J. [6] prioritized test cases on the basis of the structure of program. They proposed TIM (Testing Importance of Module) metrics in the paper. While prioritizing the test cases of a test suite, both TIM and fault proneness of the test case were taken into consideration. The main advantage of the approach proposed in their paper is that it can be applied to perform testing of new software or for regression testing. Since it deals with the importance of a module, the algorithm proposed by them is more realistic.

Park [10] et al. proposed historical value-based cost-aware test case prioritization approach. The main metrics considered in this case is Average Percentage of Fault detected per cost (APFDc).Their approach mainly depends on the historical value of the test case and the fault severity of the same. The experiment set up considered in the paper is based on open-source Java software *ant*. The cost effectiveness of a test case at any time is estimated on the basis of the previous value of the test case and the fault severity. The main contribution of this paper is that it considered the criticality of software.

Mark Sherriff [20, 21] et al. proposed a change impact analysis approach to prioritize the test cases for regression testing. They used Singular Value Decomposition (SVD) technique to find the structure of the file association clusters and the amount of variation done by this cluster in the original system after a change. The U and V matrices provide the information about the file association clusters and the values from S represent the variation. A large value of variation signifies that the cluster is problematic and it requires rigorous testing. This technique has been found quite satisfactory if the level of granularity is file and provides encouraging results. But nowadays, the granularity level has been grained to code fragment level [14] for more precise output.

Jiang [3] et al. conducted different empirical studies to find how a subset of test suite with high priority value helps in fault localization. They considered continuous integration of software for locating fault at early stage of software development. By conducting various empirical studies on different types of software it has been observed that coverage-based strategies for prioritizing test cases of a test suite outperform the other strategies in continuous integration testing. For testing software in realistic environment, the field data is leveraged from the users and software is test with the collected field data [9]. This method is very effective but due to non-availability of all types of users, data collected is not sufficient to generalize the results. For business-oriented applications Mei [7] et al. proposed a technique for prioritization of test cases. This study takes human behavior of developer into consideration while the software is maintained. Human behavior is generally underestimated by test case designer. Under such circumstances, for the successful software maintenance, the implementation of coverage based test case prioritization techniques for regression testing is required. The results of an experimental study conducted have shown that by taking significance of an artifact into consideration, efficacy of regression testing can be improved.

Engstrom [2] et al. conducted a systematic review of almost all regression test selection techniques proposed by different researchers from calendar year 1969 to 2006. The review reported 38 studies with 32 techniques. Some techniques evaluated in the review were found software specific. This study gives the insight view of regression testing selection technique. After studying each technique mentioned in their review, it has been observed that some of the techniques are more frequently used where as other are not. It has been further observed that there is no such test selection technique which fulfills all the requirements of regression testing.

Sebastian [18] et al. conducted a set of empirical studies that aim to find, (i) the effectiveness of prioritization techniques to specific modified version, (ii) to find the trade-off between fine granularity prioritization techniques and coarse granularity prioritization techniques. From the empirical studies based on various open source utilities, it has been observed that the fault proneness measure of a test case plays significant role in prioritizing a test case. The analysis results indicate that version –specific prioritization can improve the rate of fault detection significantly.

3- RESEARCH METHODOLOGY

Time budget allowed is usually lesser than the time required for execution of all test cases of test suite designed for regression testing [22]. Therefore, an efficient technique for selection of test cases from the test suite is required that detects maximum distinct faults within given time limit. In most of the studies conducted in this direction, the subset of prioritized test cases contains those test cases that have high values of FEP [15,16,18,20]. The test cases are arranged in the descending order of their FEP values and they are executed in the same order. In this study, FTCP algorithm has been proposed to re-order the test cases in the test suite such that the fault which has been allotted minimum time for its detection in the test suite is detected first. Let F_i be a fault and it is detected by test cases T_a,T_b, or T_c. TFR for this fault in the test suite has been computed as:

$$TFR(F_i) = \frac{t_a}{TFT_a} + \frac{T_b}{TFT_b} + \frac{t_c}{TFT_c} \qquad (1)$$

In equation (1), TFT_a represents total faults detected by test case T_a, and t_a represents the time of execution of test case T_a. The other factors of equation (1) can be interpreted in the same manner. TFR (Fi) represents total time allocated to detect fault F_i in test suite. It has been assumed that if a test case detects *m* faults in *n* seconds, then one fault will be detected in *n/m* seconds. Further, in equation (1), that test case will be executed to detect fault F*i*, which contributes minimum to TFR (F_i). A tie is broken arbitrarily. For example, consider a test suite, Tabulated in Table 1, that contains five test cases which detects five faults. The faults are tabulated in rows and test cases are presented in columns.

Table 1 Fault and test suite representation for regression testing

	T_1	T_2	T_3	T_4	T_5
F_1	1	0	1	0	1
F_2	0	1	0	0	0
F_3	1	1	0	1	0
F_4	0	0	1	0	1
F_5	1	1	0	0	1
Time	**4**	**5**	**3**	**2**	**4**

In Table 1, for a particular row, value 1 present in a column indicates that corresponding fault is detected by the column test case. A fault which has the least value of TFR is detected first. The test case corresponding to this fault (which contributes least to TFR) is at the utmost priority of prioritized set of test cases. The remaining test cases are added to the prioritized set after ignoring those faults which have already been covered with the identified test case and repeating the same steps that we have used for the identification of first test case. FTCP algorithm proposed for prioritization has been shown in Figure 1.

3-1 FTCP ALGORITHM

Input: A test suite T, set of fault yet to detect (TF), time budget (TB), total number of faults detected by a test case T_i (tft_i), Faults detected by Test case T_i (FT_i) and time taken to execute a test case T_i (t_i). Initially TF contains all faults.
Output: Prioritized test suite (P), set of Fault Detected (FD) (initially FD is empty.)

```
while (( TF≠Φ) and ((t≤TB)
{
  for(i=1;i≤n; i++)
  {
    i) q=∞;     (ii)  r[0]= ∞;
     for (j=1;j≤n;j++)
     {
       if(f[i,j] ≠ 0)
        {
         r[j]= tft_j / t_j ;
         if(r[j]<q)
         {
           q=r[j]; (ii)  F[i]=T_j;
         }
        TFR[i]=TFR[i]+r[j];
       }//endif
     }//end for
  }end for
Min=∞;
for(i=1;i≤n,i++)
{
    if((TFR[i]> 0 and TFR[i]≤Min))
    if(t+ti ≤ TB)
     {
          (i) Min=r[i];   (ii)  p=i;
     }
} //end for loop
t=t+ t_p;
FD=FD U FT_p;
TF=TF-FT_p;
P=P U T_p;
Make TFR[i]=0; and  f[i,j]=0 for all faults detected by T_j and set f[i,j]=0  for test case T_j that is
currently executed }//end while loop
```

Figure 1 FTCP Algorithm

In the proposed algorithm, general mathematical set operations (AUB, A∩B,A-B) have been used for making it simple and understandable.

4- OBJECTIVES

The broad objective of comparative study conducted in this section is to find the relative effectiveness of proposed technique and two parallel available prioritization techniques, APFD based [15, 16, 17, 18, 19], and OTCP [15, 16, 17, 18, 19, 22]. The specific objectives of the study are:
1. To analyze that which of prioritization techniques is most fault-prone.
2. To analyze which technique performs better under strict time constraints.

5- ANALYSIS

For the comparative study, a test suite, tabulated in Table 2 has been taken without any loss of generality.The test suite consists of six test cases, and when executed, it reveals ten possible seeded faults (inserted randomly). The test suite is similar to those taken by other researchers [10, 15, 18, 20]. For the comparision, APFD based and OTCP techniques have been considered as they also intend to maximize faults detection with the minimum execution of test cases. Further, TFR ratio and the time Effective Test Case (ETC) for each fault has been determined with the help of FTCP algorithm and shown in Table 2.

Table 2 Test suits with TFR and ETC

	T_1	**T_2**	**T_3**	**T_4**	**T_5**	**T_6**	**TFR**	**ETC**
F_1	1	0	1	1	1	0	3.516667	T1
F_2	0	1	1	0	1	1	3.8	T3
F_3	1	0	0	1	1	0	2.916667	T1
F_4	0	0	0	0	0	1	1.2	T6
F_5	0	1	1	0	0	1	2.8	T3
F_6	1	0	0	1	0	1	3.116667	T1
F_7	0	0	1	0	1	0	1.933333	T3
F_8	0	1	0	0	0	1	2.2	T2
F_9	0	0	1	1	0	0	1.85	T3
F_{10}	0	1	0	0	0	0	1	T2
Time	**2**	**4**	**3**	**5**	**4**	**6**		
Faults	**3**	**4**	**5**	**4**	**4**	**5**		

TFR value for fault F_{10} is less as compared to others. Hence it will be detected first, and test case executed will be T_2. The execution of this test case also reveals faults F_2, F_5, and F_8. Therefore, according to FTCP algorithm (Figure 1, step 28), all non-zero value of these faults, and FTR values will be set to 0. Also, all non-zero values corresponding to test case T_2 for any fault will be set to 0. The resultant is represented with the help of Table 3. The gray shells represent the changed shells. From Table 3, the next values of TFR values are calculated as calculated for Table 2. The iterative tables are generated until either all faults are detected or total execution time of test cases exceeds time budget for regression testing.

Table 3 Test suits after First iteration
Colored shells represent the changed values

	T_1	T_2	T_3	T_4	T_5	T_6	TFR	ETC
F_1	1	0	1	1	1	0	3.516667	T1
F_2	0	0	0	0	0	0	0	
F_3	1	0	0	1	1	0	2.916667	T1
F_4	0	0	0	0	0	1	1.2	T6
F_5	0	0	0	0	0	0	0	
F_6	1	0	0	1	0	1	3.116667	T1
F_7	0	0	1	0	1	0	1.933333	T3
F_8	0	0	0	0	0	0	0	
F_9	0	0	1	1	0	0	1.85	T3
F_{10}	0	0	0	0	0	0	0	
Time	**2**	**4**	**3**	**5**	**4**	**6**		
Faults	**3**	**0**	**3**	**4**	**3**	**2**		

The prioritized test suites generated by APFD, OTCP, and FTCP for four time budgets, i.e. 15 seconds (TB15), 12 seconds (TB12), 9 seconds (TB09), and 6 seconds (TB06) are tabulated in Table 4. Table 5 indicates the total number of faults detected by different techniques for varying time budgets. From the values generated, it is clear that proposed technique detects equal or more faults than APFD based, and OTCP technique. Further, in FTCP, the possibility of random selection of test cases rarely arises as the value of TFR calculated for each fault is usually distinct.

Table 4 Test Suites generated by different Prioritization techniques for different time budgets

Technique	TB15	TB12	TB09	TB06
APFD	T_3, T_1, T_2, T_5	T_3, T_1, T_2	T_3, T_1, T_2	T_3, T_1
OTCP	T_3, T_6, T_2, T_1	T_3, T_6, T_1	T_3, T_6	T_3, T_1
FTCP	T_2, T_3, T_6, T_1	T_2, T_3, T_1	T_2, T_3, T_1	T_2, T_1

Table 5 Total faults detected by different techniques for given time budgets

Time Budget	APFD	OTCP	FTCP
TB15	9	10	10
TB12	9	9	9
TB09	9	8	9
TB06	7	7	7

5-1 FINDINGS

The results of the comparative study are based on an example of test suite tabulated in Table 2. The comparative results indicate that FTCP performs equally well as APFD based prioritization and OTCP but the number of random selection of test cases is usually lesser than other techniques [15, 16, 20, 22]. Different time budget values have been considered for getting better idea about the behavior of proposed technique in comparison to APFD, and OTCP technique.

The results of Table 5 are also shown graphically in Figure 2. It is evident from the graph that FTCP technique is helpful in prioritization of test cases and competes with two other techniques proposed by different researchers. The main advantage of proposed algorithm is that contrary to APFD, and OTCP, FTCP algorithm rarely assigns same value to two faults, which reduce the random selection of test cases.

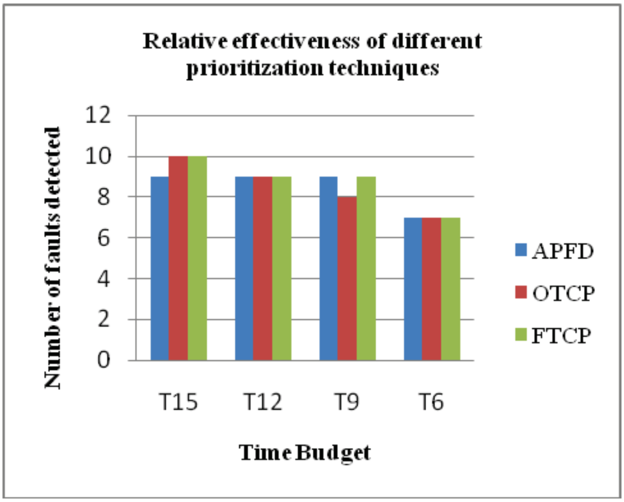

Figure 2 Relative effectiveness of different prioritization techniques

5-2 THREATS TO VALIDITY

Main threat to validity is that FTCP is based on an assumption that all faults of a test case take equal time for their detection. The comparative study is based on an assumed test suite with known faults. In real practice, it is difficult to predict the behavior of a test case and hence, to generate test cases and their oracles is always challenging.The generalization of this technique requires some empirical studies of real-world test suites.

6- CONCLUSION AND FUTURE SCOPES

The selection of test cases for the prioritized test suite is challenging task as their criteria of selection are very complex. In this study, FTCP technique for the prioritization of test case reorders the test cases on the basis of their efficacy to detect those faults which have been allocated minimum time for detection in test suite. From the results of comparative study, it is evident that proposed technique is an effective time-aware test case prioritization technique. The FTCP algorithm reduces the possibility of random selection of test cases while forming prioritized test suite. The current study can be extended further by experimenting with real test suites generated for open source software and by considering the business values of the affected functions in regression testing for the test case prioritization.

REFERENCES

[1] B. Beizer," *Software Testing Techniques".* New Delhi: Dreamtech Press, 2008.

[2] E. Engstrom, M. Skoglund, and P. Runeson, "Empirical evaluation of regression test selection Techniques: A systematic Review". Proc. of the Second ACM-IEEE international symposium on Empirical software engineering and measurement, ACM: New York, NY, USA, pp. 22-31, 2008.

[3] B. Jiang, Z. Zhang, T.H. Tse, and T.Y. Chen, "How well do test case prioritization techniques support statistical fault localization". Proc. of the 33rd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2009), vol. 1, IEEE Computer Society Press, Los Alamitos, CA, pp. 99-106, 2009.

[4] H. Z. Kagdi and J. I. Maletic, "Software- Change Prediction: Estimated + Actual. Proc. IEEE Workshop on Software Evolvability", pp. 38-43, 2006.

[5] B. Korel, M. Ali, and Y. Al,"Automated Regression Test Generation." Proc. of International Symposium of Software Testing and Analysis, pp. 143-152, 1998.

[6] Z. Ma and J. Zhao, "Test case prioritization based on analysis of program structure". Proc. of Asia-Pecific Software Engineering Conference, pp. 471-478, 2008.

[7] L. Mei, Z. Zhang, W.K. Chan, and T.H. Tse, "Test case prioritization for regression testing of service –oriented business applications." Proc. of International Conference on World Wide Web, pp. 901-910, New York, April 20-24,2009.

[8] B. Meyer, I. Ciupa, A. Leitner, and L. Liu, "Automatic testing of object-oriented Software." Proc. of the 33rd conference on Current Trends in Theory and Practice of Computer Science, pp. 114-129, 20-26 January, 2007.

[9] A. Orso, T. Apiwattanapong, and M.J. Harrold, "Leveraging field data for impact Analysis and regression testing". Proc. of the 9th European software engineering conference held jointly with 11th ACM SIGSOFT inter-

national symposium on Foundations of software engineering, pp. 128 - 137. September, 2003.

[10] H. Park, H. Ryu, and J. Baik, "Historical value –based approach for cost-cognizant test case prioritization to improve the effectiveness of regression testing." Proc. of Second International Conference on Secure System Integration and Reliability Improvement, pp 39-46,USA 2008.

[11] P. Parashar, A. Kalia, and R. Bhatia,"Change impact analysis: A tool for effective regression testing". Proc. of International conference on Information Systems and Technology Management(ICISTM 2011), Communication in Computer and Information Systems Springer-Verlag, vol. 141, pp. 160-169, 2011.

[12] R. S. Pressman, "Software Engineering: A Practitioner's approach" New Delhi: Mc-Graw Hill Higher Education, 2005.

[13] E. Rajina and D. Janzen, "Effects of dependency injection on maintainability." Proc. of International Conference on Software Engineering and Applications (ICSEA'07),pp 7-12, CA, USA, 2007.

[14] V. Rajlich and M. Patrenko,"Variable granularity for improving precision of impact Analysis." Proc. of International Conference on Program Comprehension (ICPC'09), pp. 10-19, Vancouver, British Columbia, Canada, May 17-19, 2009.

[15] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold,"Test case prioritization: An empirical Study". Proc. of International Conference on Software Maintance, pp. 179-188 DC, USA, September, 1999.

[16] G. Rothermel, R.H. Untch, C. Chu, and M.J. Harrold, "Prioritizing test cases for regression testing." Proc. of the ACM SIGSOFT International symposium on Software Testing and Analysis (ISSTA), pp. 102-112, NY, USA, 2000.

[17] G. Rothermel and M.J. Harrold, "A safe, efficient regression test selection technique." ACM Transaction on Software Engineering and Methodology, pp. 173-210, 1997.

[18] E. Sebastia, A.G. Malishevsky, and G. Rothermel,"Test case prioritization: A family of empirical studies". IEEE Transactions on Software Engineering, vol. 28, no. 2, February 2002.

[19] E. Sebastian, A.G. Malishevsky, and G. Rothermel,"Incorporating varying test costs and fault severities into test case prioritization." Proc. of International Conference on Software Engineering, pp. 329-338, 2001.

[20] M. Sherriff, M. Lake, and L. Williams, "Prioritization of regression tests using singular value decomposition with empirical change records". International Symposium on Software Reliability Engineering, DC, USA, pp. 81-90 Nov. 2007.

[21] M. Sherriff, M. Lake, and L. Williams, “Empirical software change impact analysis using singular value decomposition.” Proc. of International Conference on Software Testing, pp. 268-277, 2008.

[22] L. Zhang, S. Hou, C. Guo, T. Xie, and H. Mei,“Time-Aware Test-Case Prioritization using Integer Linear Programming.” Proc. of International symposium on Software testing and analysis (ISSTA’09), NY, USA. pp. 213-224, July19-23, 2009.