Fault Detection Time: An Important Parameter of Test Case Sequencing for Regression Testing

Prem Parashar*
Arvind Kalia**
Rajesh Bhatia***

Abstract

Regression testing is one of the imperative activities of software maintenance. It requires lot of efforts to choose those test cases from test suite which are most promising and the execution of which traces maximum number of faults present in software. Due to limited time budget available for regression testing some test case selection techniques are always needed to meet this requirement. Various test case selection techniques have been proposed by different researchers. A Fault-Detection Time selection technique has been proposed in this paper that selects a test case which detects that fault which has been allotted minimum time in the test suite. Through a comparative study, it has been observed that proposed technique performs equally well as other two parallel test case prioritization techniques; Average Percentage of Fault Detection (APFD) based prioritization, and Optimal Test Case Prioritization (OTCP) with minimum number of random selections of test cases.

Keywords: Fault Detection Time, Test Suite, Test Case Prioritization, Maintenance, Fault Detection

1. Introduction

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 and vice versa. Software maintenance is one of the most expensive phases of software development [1, 8, 11, 12, 14, 17]. 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 [11] 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].

Most of prioritization techniques mentioned in current studies are based on fault - exposing potential (FEP) of test cases. A test case may detect different 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

*Department of Computer Science, Himachal Pradesh University, Shimla, India
**Department of Computer Science, Himachal Pradesh University, Shimla, India
***Department of Computer Science, Deen Bandhu Chotu Ram University, Murthal, India

of software program within minimum time have 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]. Whether a part is critical to software application or not is defined by its usability and importance.

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. Some common techniques proposed are Average Percentage of Fault Detection (APFD) based prioritization, random test case prioritization, optimal test case prioritization and coverage-based prioritization.

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. The results of the comparative study conducted are encouraging as the proposed technique performs equally well to the parallel techniques APFD, and OTCP with minimum number of random test case selection.

2. Review of Literature

In some comparative studies [14,15] the metrics, (i) Average Percentage of Fault Detection (APFD) and (ii) total Fault-Exposing Potential (FEP) are used for measuring the effectiveness test case prioritization. From the experimental studies, it has been observed that most of the times total FEP coverage based test case prioritization techniques perform 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. The metrics TIM (Testing Importance of Module) also plays significant role in prioritization of test cases [6]. In [6], 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 proposed approach is that it can be applied to perform testing of new software or software after maintenance. Since it deals with the importance of a module, the algorithm proposed is more realistic.

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 experimental study conducted have shown that by taking significance of an artifact into consideration, efficacy of regression testing can be improved.

Park [10] et al. proposed historical value-based cost-aware test case prioritization approach. The main metrics considered in this study 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 its fault severity. The main contribution of this paper is that it considered the critical sections of software.

Mark Sherriff [19, 20] 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. 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 [13] for more precise output.

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 the review, it has been observed that some of the techniques are more frequently used as compared to others. It has been further observed that there is no such test selection technique which fulfills all the requirements of regression testing.

Sebastian [17] 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 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

The major challenge to perform regression testing is the time budget allotted [21]. An efficient test case selection technique is always desirable that detects maximum faults in the given time budget. 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. Test cases are arranged in descending order of their FEP values and they are executed in the same order. In

the current study, a test case selection technique has been proposed that re-order test cases in the test suite such that a fault which has been allotted minimum time for its detection in the test suite is detected first.

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. If a fault is detected by more than one test case, the total time allotted to detect that fault will be the sum of the times allotted to all test cases and that test case will be considered for final execution which detects the fault in minimum time. A tie is broken arbitrarily. The simple algorithm SIM proposed for this purpose has been presented in Figure 1.

3.1 SIM Algorithm:

Input: A test suite T, set of fault yet to detect (TF), time budget (TB)

Output: Prioritized test suite (P)

1. Repeat (step 2- step 6) until TF is empty and t >TB
2. Select fault F_i from TF which takes least time for execution
3. If F_i is detected by more than one test case say T_i and T_j, select the one which detects it with minimum time. Let the test case selected be T_i
4. Include test case T_i to P
5. Minus fault F_i and other faults detected by test case T_i from TF. Add time t_i to t
6. Minus T_i from T

Fig. 1: SIM Algorithm

4. Objectives

The main objective of this comparative study is to analyze the effectiveness of SIM algorithm for prioritization of test cases.

1. To analyze how the effectiveness of the techniques varies with time budget.
2. To analyze which technique performs better under strict time constraints.

5. Major Findings

The results of the comparative study are based on an example of test suite tabulated in Table 1. The test suite example has been framed without any loss of generality. The comparative results show that SIM performs equally well as APFD based prioritization and OTCP but the number of random selection of test cases is usually lesser than other techniques. Different time budget values have been considered for getting better idea about the behavior of proposed algorithm in comparison to APFD and OTCP technique.

5.1 Analysis

For the comparative study, a test suite tabulated in Table 1 has been taken. The test suite consists of five test cases and when executed, it reveals eight possible faults. Further, Fault-Time Ratio (FTR) and Promising Test case (PTC) corresponding to each fault have been determined with the help of SIM algorithm and tabulated in Table 1.

Table 1: Test suits with FTR and PTC
'1' means that row fault is detected and '0' means that row fault is not detected by column test case.

	T1	T2	T3	T4	T5	FTR	PTC
F1	1	0	0	1	0	1.50	T1,T4
F2	0	1	0	0	1	1.67	T5
F3	0	1	0	1	0	1.75	T4
F4	1	0	1	0	1	3.08	T5
F5	0	0	0	0	1	0.67	T5
F6	1	0	1	1	0	3.17	T1,T4
F7	1	1	1	1	0	4.17	T1,T4
F8	0	1	0	0	0	1.00	T2
Time	**3**	**4**	**5**	**3**	**2**		

After the execution of proposed SIM algorithm described in Figure 1, the prioritized test suit P will contain test cases T_5, T_2, T_1/T_4 to reveal all faults.

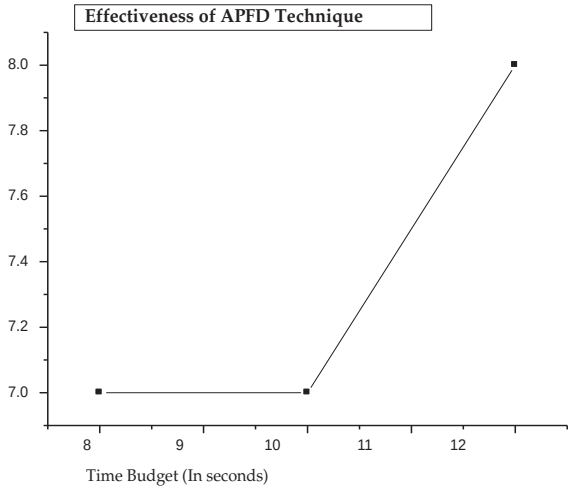

Fig. 2: Effectiveness of APFD Technique at different Time Budgets

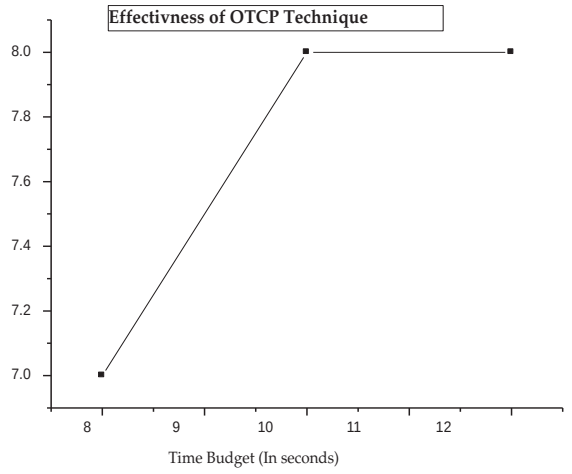

Fig. 3: Effectiveness of OTCP Technique at different Time Budgets

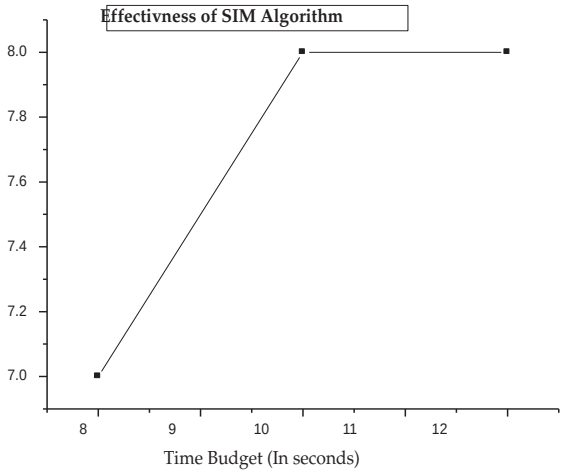

Fig. 4: Effectiveness of SIM algorithm at different Time Budgets

The results of SIM algorithm has been compared with other parallel techniques APFD based prioritization and OTCP technique at three time budgets i.e. 8 seconds, 10 seconds and 12 seconds. The behaviors of APFD, OTCP and SIM algorithm at three times have been represented in Fig 2, Fig 3 and Fig 4 respectively.

From the graphs generated, it is clear that proposed algorithm detects equal or more faults than APFD based and OTCP technique at any given time. Further, in SIM, the possibility of random selection of test cases rarely arises as the value of FTR calculated for each fault is usually distinct. Though the final values of OTCP and SIM are same but still SIM performs better and it always gives testers an alternate sequence of test cases that generate the same results as other parallel techniques. Also this algorithm takes into account the award value of a test case which usually changes with the change in test suite T.

5.2 Results and Discussion

It is evident from the graphs represented in Fig 2, Fig 3, and Fig 4, that SIM technique helps 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, this algorithm rarely assigns same value to two faults, which reduce the random selection of test cases. It also comments silently on the issue that the priority value of a test case is highly dynamic in nature and changes as the size of test suite starts changing.

6. References

1. *Beizer, B. Software Testing Techniques, New Delhi: Dreamtech Press, 2008.*
2. *Engstrom, E., Runeson, P., Empirical evaluation of regression test selection Techniques: Asystematic Review. ESEM'08, October 9-10, 2008.*
3. *Jiang, B., Zhang, Z., Tse, T.H., Chen, T.Y. How well do test case prioritization techniques support statistical fault localization ICSAC'09, 2009.*
4. *Kagdi, H.Z., Maletic J.I. Software-Change Prediction: Estimated + Actual. In Proc. IEEE Workshop on Software Evolvability, page 38-43, 2006.*
5. *Korel, B., Ali, M., and Al, Y.: Automated Regression Test Generation. ISSTA'98 ,1998.*
6. *Ma, Z., Zhao, J. Test case prioritization based on analysis of program structure. APSEC, 2008.*
7. *Mei, L., Zhang, Z., Chan, W.K., and Tse, T. H. Test case prioritization for regression testing of service-oriented business applications. WWW'09, April 20-24, 2009.*
8. *Meyer, B., Ciupa, I., Leitner, A., Liu, L. Automatic testing of object-oriented Software. SOFSEM '07. 20-26, January 2007.*
9. *Orso, A., Apiwattanapong, T.,Harrold, M.J. Leveraging field data for impact Analysis and regression testing. ESEC/FSE'03. September, 2003.*
10. *Park, H., Ryu, H., Baik, J. Historical value-based approach for cost-cognizant test case prioritization to improve the effectiveness of regression testing. SSIRI, 2008.*
11. *Pressman, R. S., Software Engineering : A Practitioner's approach, New Delhi: Mc-Graw Hill Higher Education, 2008.*
12. *Rajina, E., Janzen, D. Effects of dependency injection on maintainability. In Proc. IASTED, 2007.*
13. *Rajlich, V., Patrenko, M. Variable granularity for improving precision of impact Analysis. ICPC-2009.*
14. *Rothermel, G., Untch, R.H., Chu, C., Harrold, M.J. Test case prioritization: An empirical Study ICSM'99. September, 1999.*
15. *Rothermel G., Untch R.H., Chu C., Harrold, M.J. Prioritizing test cases for regression testing. IEEE Transaction on Software Engineering. Jan 3, 2001.*
16. *Rothermel, G. and Harrold, M.J. A safe, efficient regression test selection technique. ACM Transaction on Software Engineering and Methodology, 6(2), pages 173-210, 1997.*
17. *Sebastian E., Malishevsky, A.G., and Rothermel, G. Test case prioritization: A family of empirical studies. IEEE Transactions on Software Engineering, Vol. 28, No. 2, February 2002.*
18. *Sebastian E., Malishevsky, A.G., and Rothermel, G. Incorporating varying test costs and fault severities into test case prioritization. In Proc. ICSE, page 329-338, 2001.*
19. *Sherriff, M., Lake, M., Williams, L. Prioritization of regression tests using singular value decomposition with empirical change records. International Symposium on Software Reliability Engineering, November 2007.*
20. *Sherriff, M., Lake, M., Williams, L. Empirical software change impact analysis using singular value decomposition. ICST'08, 2008.*
21. *Zhang, L., Hou, S., Guo, C., Xie, T., and Mei,H.: Time-Aware Test-Case Prioritization using Integer Linear Programming. ISSTA'09. July19-23, 2009.*