Journal of Computer Science 9 (5): 654-659, 2013
ISSN 1549-3636

doi:10.3844/jcssp.2013.654.659 Published Online 9 (5) 2013 (http://www.thescipub.com/jcs.toc)

Effective Test Case Selection and Prioritization in Regression Testing

[1]Albert Pravin and [2]Subramaniam Srinivasan

[1]Department of Computer Science and Engineering, Sathyabama University, Chennai, India
[2]Department of Computer Science and Engineering, Anna University, Regional Centre, Madurai, India

Received 2013-04-15, Revised 2013-05-24; Accepted 2013-05-28

ABSTRACT

Regression testing is used to ensure the validity of the changed software. Due to time budget and entire test suite could not be executed. Hence it becomes an essential to minimize the test suite and choose a subset of test cases from test suite which will be executed in least time and has the capability to cover all the faults. Hence reordering the test case on the basis of time fault, test case prioritization technique prioritizes the test cases using fault detection algorithm, which is proposed in this study. After finding the faults in the code, the source code will be processed in the open source system like Webkit.

Keywords: Regression Testing, Webkit, Test Case Prioritization

1. INTRODUCTION

Regression testing is often more essential in order to provide a high quality software. Regression testing plays a vital role while changes occurred in the existing software. The change is to be checked before it is to be implemented. The test case is created to have an coverage over the change that is performed in the application. The main principle in the regression testing is to provide assurance that the recently established changes will not disturb the behavior of the existing, unaffected part of the software. This is checked by executing the test cases to find out the defects. Instead of executing all the test cases the test cases which cover the modification is to be selected and executed. For instance, software development method based on components mostly uses the black-box components, often accept from third party. The internal of third party components are unknown to the users, when any changes are created in third party components might get in the time way with other software system. So far it is very hard to complete the regression testing. Test case selection and test case prioritization are the two major part of the regression testing. Test case selection solves the problem of selecting the test case which will be useful to test the modified part of the software. Finally test case prioritization concerns ordering the test cases which maximize the attractive properties of recent fault detection.

1.1. Related Work

The conceptual of regression test selection method has been used to select the set of test cases Pravin and Srinivasan (2013). In addition to one of the primary assessment features of our techniques has defined the evaluation principle called inclusiveness. It is the part of the failing test cases contained in the choice relation to the whole numeral of fault test cases as soon as performing the entire test suite. In this research use most important principle is selection size. It is calculated as the percentage of numeral of preferred test cases in excess of the total number of without selection. One more review on prioritization and regression test selection has been explained by Yoo and Harman (2012).

The genetic algorithm is analyzed and the effectiveness for test case selection depending on fault detection property of a program modification explained by Singh and Parkash (2012). It will assign the priority for the test cases depending up on the fault detected . But this method doesn't consist of any brief program statement. Many researchers are viewed an inclusiveness. Pravin and Srinivasan (2012) proposed an Evolutionary Algorithm for the purpose of developing the input sets . The test case is arranged in the test suite in the ordered form So the fault is detected earlier in the process proposed by Badhera *et al*. (2012).

Corresponding Author: Albert Pravin, Department of Computer Science and Engineering, Sathyabama University, Chennai, India

The case studies of all subject programs were small programs among less than 500 line of program and arranged by fault seeding. In this case Qt port of Web Kit software tool as specified and its regression test suite were educated. The system basis includes 1.9 million fault were not seed but they were actual having an effect on the software. One more important characteristic of test case selection method is the coverage, which create use of them. Pravin and Srinivasan (2012) uses the knowledge based concept for selection of test cases. Rothermel *et al*. (2002) used all-edges coverage. In this studies showed to the various granularities are utilized in test case selection created various decrement during the test suite size. In this study the purpose of coverage-based test suites will give test selection results better to those offered by means of test suite, which are not coverage-based. Procedure level coverage used in this study, but didn't try to reduce the test suite in provisions of removing "redundant" test cases. Additional studies detailed on various problems of regression test case collection. Kumar *et al*. (2012) describes the testing process which will improve the Effeciency regarding the quality ,cost and effort.

In this study testing shows that as the number of variation between the program versions increased and the number of test cases chosen also grew quickly as well as the efficiency of test selection increased. Have to apply test case selection to all version of important changes are present and study how frequently a complete retest is needed to realize consistent results. Test case prioritization methods also included in this study Singh and Parkash (2012). In this proposed hybrid techniques has minimization, combining modification and prioritization based selection to recognize a delegate division of all test cases that will result in various output performance on the new software version Ahmed and Hermadi (2008). The case studies on regression test case prioritization have been included. Test data generation technique based on adequacy based testing criteria was proposed by Malhotra and Garg (2011). Shahid *et al*. (2011) This study provides a study of the current testcoverage researches conducted by other researchers for test coverage in testing process and the existing approaches; gaps and uncovered measurement of Test Coverage can be explored further Panigrahi and Mall (2012) developed an Regression Test Selection technique for object oriented programs and the UML state machines for the affected classes.The contorl and the data dependancy also captured.

1.2. Test Case Selection Based on Time Fault

1.2.1. Test Case Prioritization

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. Test case Prioritization is of selecting test cases in an order of higher priority with an earlier execution and has components that specifies input, operation and an expected outcome, that determine if a properties of the particular application is working correctly.

1.3. Prioritization Methods

- Initial ordering
- Reverse ordering
- Random ordering
- Based on fault detection ability

1.4. Test Suite Reduction

In software application development, a test suite, less commonly known as a suite for checking the validity of the software, is a collection of test cases that are intended to be used to test a software program to show that it has some specified set of functionality. A test suite will contain a detailed set of instructions or goals for each collection of test cases and information on the system configuration to be used during testing process. A set of test cases may also contain prerequisite states or steps and descriptions of the following tests.

1.5. Fault Detection Algorithm

Time budget to test the software is very less when compared to the time needed to execute the all test cases of test suite. Hence, an effective technique for choosing the test cases from the test suite is essential, which identify maximum distinct fault within the given period of time. The majority of the study conduct in this method, prioritized test case consists of test case which has high value of fault values. These test cases are arranged in the descending order of their fault values and which are examined in the same order.

In this study, proposed a Fault detection algorithm, which is mainly used for re-arrange the test cases in the test suite such that the fault of the Fault detection algorithm has been selected least time for its detection in the test suite is detected first. Let Mi be a fault and it is detected by test cases L_a, L_b, or L_c. TF for this time fault in the test suite has been calculated as:

$$TF(M_i) = 1_a / LML_a + 1_b / LML_b + 1_c / LML_c \quad (1)$$

In Equation (1), LMLa characterize total faults detected by test case La and la represents the time of execution of test case La. The other factors of Equation (1) can be understood in the same way. TF (Mi) represents total time allocated to detect fault Mi in test suite and it has been assumption of a test case detects p faults in n seconds, at that time fault will be detected in n/p seconds. Additionally, in Equation (1), to facilitate test case may be performed to detect fault Mi, test case add minimum to TF (Mi). A bond is busted randomly.

For an example, consider a test suite, Tabulated in **Table 1**, in **Table 1** includes five test cases and it detects five faults. These faults are tabulated in rows and test cases are specified in columns.

In **Table 1**, for a specific row value 1 represented in a column point out with the matching fault is detected by the column test case. Least TF value is detected first in the fault detection.

In the prioritized test cases, a test case whose TF value minimum will be considered as highest priority fault. The residual test cases are added to the prioritized set. After eliminate those faults that have been covered before with that of the recognized test case. Also replication the similar steps have used with the purpose of recognition of first test case. In order to prioritize the test cases fault detection algorithm is proposed it is shown in following algorithm.

1.6. Input

A test suite S, set of fault so far to detect (tf), time budget (tb), total number of faults detected by a test case Ti (tgsi), Faults detected by Test case Si (MSi) and time taken to execute a test case Si (si). At first TF consist of all faults.

1.7. Output

Prioritized test suite (p), set of Fault Detected (fd) (initially fd is empty)

The general mathematical sets of operations (BUA, B∩A, BA) are in the proposed algorithm is mainly used because it is of:

- Simple
- Understandable

```
While ((tf‡ø) and ((s<=tb))
{
for (a=1;a<=n; a++)
{
i) q=oe; ii) r[0]= oe;
for (b=1;b<=n; b++)
{
if(m[a,b] ‡0)
{
i)q=r[b]; ii)M[a]=Sb;
}
TF[a]=TF[a]+r[b];
}//end if
}//end for
} //end for
Min= oe;
for (a=1;a<=n;a++)
{
if((TF[a]>0 and TF[a]<=Min))
if(s+sa<=CY)
{
i)Min=r[a]; ii)pn=a;
}
}//end for loop
s=s+spn;
fd=fd U MS pn;
tf=tf- MS pn;
p=p U S pn;
Make TF[a]=0; and m[a,b] =0 for all faults
detected by Sb and set m[a,b] =0 for test case Sb
that is currently executed
}//end while loop
```

Open source software system is a very complex, huge and lively developing software system. This is preserved by hundreds of developers in the region of world.

In that our many department members are included. Recently with the object to develop the inner quality of the system together with the consistency and effectiveness of the regression testing system we have started a long term project. As an initial step, we begin with the estimation of the code coverage and other regression test suite elements.

Table 1. Test suite representation with Fault

	S1	S2	S3	S4	S5
M1	1	0	1	0	1
M2	0	1	0	0	0
M3	1	1	0	1	0
M4	0	0	1	0	1
M5	1	1	0	0	1
Time	4	5	3	2	4

1.8. In Open Source Software System

This is consisting of regarding 2.2 million lines of code. In that typically JavaScript, c++ and python between others. In this study we only intended on c++ components. This features to regarding 86% of the code.

The system has a comparatively big regression test collection that consisting almost 24 thousand test cases. Open source software system has a huge, geographically dispersed development community and its building environment is a usual one for dispersed team, which contains grave configuration administration and strict addition rules. For example a test set must pass before any patch could land any of the mechanism in the version control repository.

There are 113914 revisions present on April 11, 2012 and each day 90 revisions are shaped .The requirement of performing all regression tests could not fulfilled in many cases. Because the big size of the regression test suite and very regular revisions.

The programmed build system always makes regression tests. but due to the insufficient server capacity failure may occur, if occurred it will be difficult to find which revision is responsible. Therefore performing the correction is very difficult. In addition the completeness and consistency of the regression test set has not at all been methodically checked, therefore presently there is no real suggestion regarding regression test suite defect detection capability. There may be faults are detected by several test cases else can Stay undetected. Forced by these problems, required to examine the option to increase the speed of the test case selection and prioritization execution, while keeping the issue detection capability and same level of reliability.

1.9. Research Goals

There is three important phases of our research are presented in the starting of our project in autumn 2011. In the first round we found an open source software system revisions decent number for collect information and get a deeper approach into the code. The changes performed to it, the regression test set, the source code coverage and the test executions. Often in second phase, based on the obtained result we applied an optimized and somewhat different version of test choice in the official open source software built system and started its continuous process in similar to the construct processes. We observed the system performance for a particular period of time and made additional analysis and measurements of the gathered data in order to confirm the initial findings.

Our current research movement is to further improve the effectiveness of the execution by initiating test prioritization in adding to easy selection. We support our approach for test optimization on the long reputation statement. From that it is likely to choose the more applicable tests from the test set depending on the set of changes implemented to the system.

2. MATERIALS AND METHODS

Fault detection algorithm, which is mainly used or re-arrange the test cases in the test suite such that the fault of the Fault detection algorithm has been selected least time for its detection in the test suite is detected first.The input for the algorithm is the test suite,time and the total numer of fault detected by the test cases. TF for this time fault in the test suite has been calculated as:

$$TF(M_i) = 1_a / LML_a + 1_b / LML_b + 1_c / LML_c$$

The general mathematical sets of operations (BUA, B∩A, BA) is used in the proposed algorithm.The number of test case and fault coverage is listed.Initially the code coverage is done

3. RESULTS

Proposed technique performance is known by comparing the existing technique of APFD with the different time budget values. The main advantage of fault detection algorithm is to reduce the random selection of test case by assigning same values to two faults.

Based on assumption, fault detection algorithm takes all test cases in equal time to their detection of faults. Based on assumed test suite with known faults, it is not easy task to predict the performance of test case and producing the test case is a difficult task as shown in **Fig. 1** the performance of proposed technique is measured in terms of number of test cases and efficiency. When the number of test case increase the efficiency fault detection also increases.

From the **Fig. 2** it explains about the comparison of existing and proposed system is shown, the FD value is higher than the APFD. Hence FD compare with APFD the FD is detecting more faults then APFD.

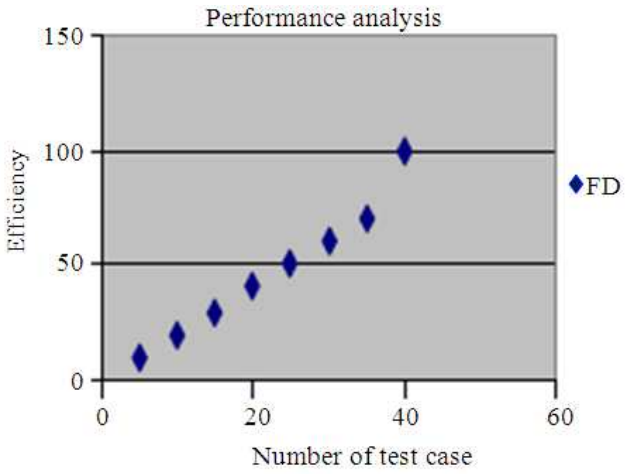

Fig. 1. Number of test cases Vs efficiency

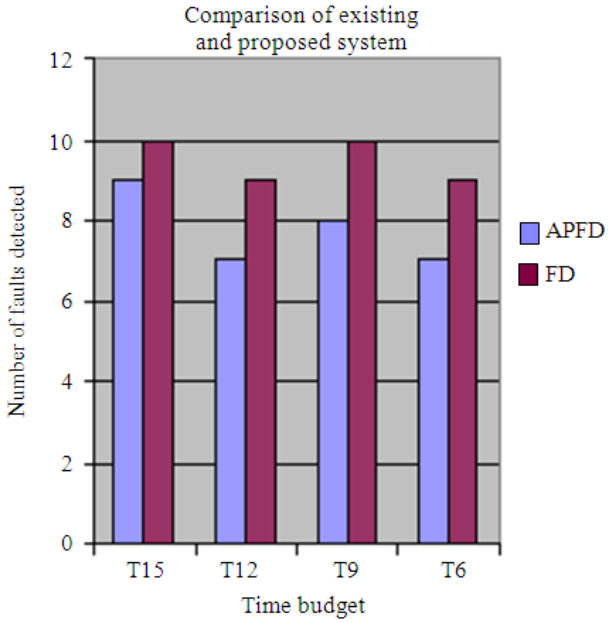

Fig. 2. Comparison of existing and proposed approach

4. DISCUSSION

The performance of proposed technique is measured in terms of number of test cases and efficiency. When the number of test case increase the efficiency fault detection also increases.The perfomance is calculated by comparing the existing and the proposed methods .From the **Fig. 2** we know that FD value is higher than the AFD.

5. CONCLUSION

In the prioritized test suite, selection of test cases is not easy task as the principles of selections are difficult. Fault detection algorithm is proposed in order to reorder the test cases on the basis of their efficiency to find the faults which have least assigned time in test suite. While forming the prioritized test suite, Fault detection algorithm minimizes the opportunity of selection of test cases. After finding the fault, experiments the real test suites with the open source software.

6. REFERENCES

Ahmed, M.A. and I. Hermadi, 2008. GA-based multiple paths test data generator. J. Comput. Operat. Res., 35: 3107-3124. DOI: 10.1016/j.cor.2007.01.012

Badhera, U., G.N. Purohit and D. Biswas, 2012. Test case prioritization algorithm based upon modified code coverage in regression testing. Int. J. Soft. Eng. Applic., 3: 29-37. DOI: 10.5121/ijsea.2012.3603

Kumar, A., S. Gupta, H. Reparia and H. Singh, 2012. An approach for test case prioritization based upon varying requirements. Int. J. Comput. Sci. Eng. Applic., 2: 99-107. DOI: 10.5121/ijcsea.2012.2309

Malhotra, R. and M. Garg, 2011. An adequacy based test data generation technique using genetic algorithms. J. Inform. Proc. Syst., 7: 363-384.

Panigrahi, C.R. and R. Mall, 2012. A hybrid regression test selection technique for object-oriented programs. Int. J. Soft. Eng. Applic., 6: 17-34.

Pravin, A. and S. Srinivasan, 2012. Evolutionary algorithm for knowledge based unit testing. CiiT Int. J. Soft. Eng.

Pravin, A. and S. Srinivasan, 2013. An efficient algorithm for reducing the test cases which is used for performing regression testing. Proceedings of the 2nd International Conference on Computational Techniques and Artifical Intelligence, Mar. 17-18, Dubai, pp: 194-197.

Rothermel, G.G., M.J. Harrold, J.V. Ronne and C. Hong, 2002. Empirical studies of test-suite reduction. Software Test. Verificat. Reliab., 12: 219-249. DOI: 10.1002/stvr.256

Shahid, M., S. Ibrahim and M.N. Mahrin, 2011. A study on test coverage in software testing. Proceedings of the International Conference on Telecommunication Technology and Applications, (CSIT' 11), pp: 5-5.

Singh, A. and K. Parkash, 2012. Fault based analysis to perform test case prioritization in regression testing. Int. J. Adv. Res. Comput. Sci. Soft. Eng., 2: 165-171.

Yoo, S. and M. Harman, 2012. Regression testing minimization, selection and prioritization: A survey. Soft. Test. Verificat. Reliab., 22: 67-120. DOI: 10.1002/stv.430