A RELATIVE EXPLORATION ON TEST CASE REDUCTION TECHNIQUES

J. Srividhya[1], Dr. R. Gunasundari[2]

ABSTRACT

Cost effective test strategies are necessary to provide reliable, secure, and usable applications. During software maintenance, as test suites grow larger during software evolution the test cases should exactly reflect the usage to uncover faults that are easily encountered the users. It turns into difficult to execute all the test cases in convinced time period. In order to reduce the size of test suites, test suite reduction techniques with respect to some coverage criteria are used. Test suite reduction techniques are critical to the cost effectiveness because a main concern is the cost of collecting, analyzing, and re executing the huge number of test cases created from user data. We performed a logical study on the test case reduction techniques and provided a comparative analysis. Our study investigates both the benefits and the costs of test-suite reduction.

Keywords : Software testing , Test case reduction techniques, Particle Swarm Optimization, Artificial Bee colony optimization, Classification Tree method.

[1]Research Scholar, Department of Computer Science Karpagam University, Coimbatore, Tamilnadu, India. jsrividhya.ku2011@gmail.com

[2]Associate Professor & Head, Department of Information Technology, Karpagam University, Coimbatore, Tamilnadu, India. gunasoundar04@gmail.com

1. INTRODUCTION

In SDLC, testing is essential to produce extremely reliable applications related to space, avionics, etc. Test suite reduction is a vital activity for test maintenance that tries to remove redundancy and reduce execution time and thus decrease the cost of testing and it turns the software cost. It cuts the number of test cases with respect to some coverage measures defined as a set of rules which helps to define whether a test suite has been effectively tested the software or not. Most minimization techniques proposed to date have two main limits: they perform minimization based on a single criterion and produce estimated suboptimal solution. The good minimization technique should allow us to easily encoding a wide spectrum of test-suite minimization problems, handling problems that involve any number of criteria, and computing optimal solutions to minimization problems by the help of modern integer linear programming solvers. Test suite prioritization modifies the order of test cases within a test suite with the goal of increasing the rate of fault detection. These test selection methods are mainly concentrating on giving the tester the best test cases for execution early on, so that under limited time and cost constraints, increased effectiveness in testing may be achieved.

The greater part of the current methodologies think about test suite which hold, experiments to test the practicality, limit values, anxiety, and execution of the software. Any reduction in this test suite size will decrease the testing time, exertion, and cost. A considerable lot of the experiments in this test suite fit in with the usefulness and limit qualities of the software. To make the vision be more extensive, this paper attempts to analyze the techniques currently available for reduction of test suites and analyze their result and performance.

The paper is organized as follows. Section II befits to related works and research in this specific area. Section III discusses about some of the existing related test case reduction techniques. Section IV summarizes the conclusion and Section V discuss about Future work.

II. Related Work

Numerous *Test-suite reduction* [1], [2], [3], [4]) aims to find a representative subset of the original test suite which can satisfy the same *test requirements* as the original test suite. .M.Harder, J.Mellen and M.D Ernst [5] applied an operational difference technique to generate and reduce the number of test cases. Their technique dynamically generates operational abstractions from test suite executions by adding test cases until the operational constructs do not change. Finding the minimal illustrative subset for a given test suite is equivalent to the problem of set covering and has been shown to be NP-complete. Traditional test-suite minimization techniques include greedy techniques, heuristic-based techniques and techniques based on integer linear programming (ILP) [6]. Lingming Zhang, Darko Marinov, Lu Zhang, and Sarfraz Khurshid in [7] says that, several experimental studies were conducted with the proposed test-suite reduction techniques to compare the sizes of reduced test suites. When some researches says that the test-suite reduction does not substantially lower the fault-detection capability of test suites, G. Rothermel, M. Harrold, J. Von Ronne, and C. Hong [8] found that test-suite reduction can severely lower the fault-detection capability. Also when compared the sizes of reduced test suites via a simulation study, H. Zhong, L. Zhang, and H. Mei, [9] studied both the sizes of reduced test suites and the time taken by various test-suite reduction techniques on a set of shell programs. There are also empirical studies on the effect of test-suite reduction on lowering the fault-detection capability. Extended dependence analysis of Yanping Chen, Robert L.Probert, and Hasan Ural [10], discusses about regression test suite reduction. Because of the computational difficulty of multi-criteria minimization, however, most existing techniques mark a much simpler version of the problem: generating a test suite that achieves the exact coverage as the initial test suite with the minimal number of test cases in [11]. In one of our

previous study [12], we analyze the Genetic approaches available for test suite reduction. To our knowledge, the only other work that sponsors assembling a reduced suite is by Antonia Bertolino, Emanuela Cartaxo, Patrycia Machado, Eda Marchetti, and Joao Felipe Ouriques [13] where they propose measuring the rate of fault detection with APFD.A evaluation of test suite reduction is dealt by few [14]. McMaster, S., and Memon, A [15] have presented a new coverage criterion for test suite minimization based on the set of unique call stacks. Regression test suite minimization using dynamic interaction patterns with increased fault detection efficiency has been proposed in [16].

III. Testcase Reduction Techniques

A. Combined Classification Method :

In [17] P. Prema, B.Ramadoss, and S. R. Balasundaram, discussed about the proportion of test suite lessening for various test case generation methods such as Equivalence Class Partition (ECPM) Method, In-house and External Classification Tree Methods, and Combined Classification Tree Method (CCTM), also CCTM is anticipated for reducing redundancy test case generation.

The Classification Tree Method (CTM) is a method for test design. CTM is a black-box grounded testing technique which ropes any type of system under test and provides a systematic approach to generate test cases from the functional requirement specification. The classification tree method contains of two major steps. Identification of test related features and their parallel values as well as grouping of different classes from all classifications into test cases. A classification-tree is a determined set of one or more groupings such that: i) there is a especially designed classification A called the root, ii) the remaining groupings, which is straight below A are called subclasses A1, A2 ... An, where n e" 0 and each of these sets is a tree. $A_1, A_2 \ldots A_{n,}$ are called the sub-trees. Each sub-tree can be sectioned into Subclasses. A terminal sub-tree comprises a classification with terminal class and their classification. In CTM, test cases are produced by joining data values of all different terminal classes. Test choice strategy involves choosing test input from all the terminal classes.

The following algorithm is used to generate classification tree :

1. Build the sub-trees associated with predecessor relations
2. Reorder the related sub-trees which is designed in step one
3. Create sub-trees for individual classifications

4. Last classification tree is developed by integrating steps 2&3.

Prema in [17] stating that, overlapping the new test method [18] for Commercial Off-The Shelf (COTS) software based on CTM to identify test cases by initially finding groupings based on the requirement specifications (in-house) and groupings based on the off-the shelf specifications (external) methods a combined classification tree was developed. To prove the combined classification Prema took the Home Security system and related the test case generation and reduction results from ECPM, CTM and CCTM. Home Security System (HSS) is an automated system that pedals the module of an alarm system. It assists mainly to guard a property against invaders by signaling an alarm and / or warning a central observing station if the sensor device detects action when the system is fortified. It also provides a treasured line of protection that can safeguard a home in the event of an tried break-in. Figure [1] & [2] shows the combined classification Tree for Home Security System and relative analysis on test cases vs test coverage criteria perception.

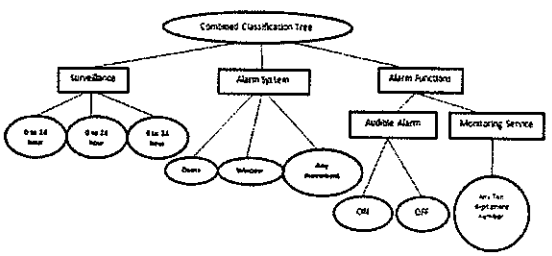

Figure-1 : Combined classification tree for HSS

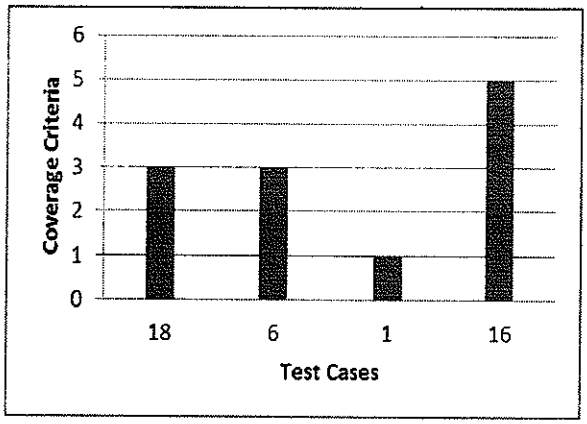

Figure-2: Test cases vs Test coverage criteria

A. A Hybrid Technique based on BCO and GA :

Bharti Suri, Isha Mangal and Varun Srivastava in [19], discussed about an effort to develop an algorithm for lessening test cases from a enormous test suites using GA and bee colony optimization to decrease both time and effort and generating optimal results.

A genetic algorithm (GA) is an optimization technique which can be useful to numerous problems, including those that are NP-hard. It uses a "survival of the fittest" technique, where the finest solutions survive and are diverse until we get a good creation. To apply GA, it is necessary to satisfy two main requirements. (a) An encrypting is used to denote a solution of the solution space, and (b) An detached function i.e a fitness function which measures the greatness of a solution. Bharti Suri states that Encryption is done for the firmness to the problem. Using fitness-based occupation like roulette wheel selection and event selection, initial

populace is chosen. The second group population of solutions is produced from first generation using genetic operators like crossover and mutation. Fresh population will be chosen and further takes part in generating the next generation. The course is monotonous until a decision state is reached (i.e. the end result has been found or, determined number of generations attained).

Artificial Bee Colony (ABC) algorithm is a swarm based meta-heuristic algorithm encouraged by the intelligent hunting behavior of honey bees and construct that scavenging action of honey bees. The definitive goal of the bees' is to ascertain the location of the food source spots with high nectar quantity. By [20], Employed bees search their food source and return to hive and execute a dance on this area. The employee bees who find unrestrained food source becomes a spy and find a new food source again. Observers decided their food source depending upon the dances of employed bees. A fluid source is chosen by each bee by subsequent a nest buddy whose food source has already exposed. The bees dance on the hive, to inform that they discovered of fluid sources and encourage their nest mates to follow them. To get nectar, other bees follow the dancing bees to one of the nectar areas. On collecting the nectar they come back to their hive, bestowal the nectar to a food store bee.

In the proposed approach by Bharti, crossover operation is implemented at the second step of GA life cycle. This is the method of assimilating the information components of two individuals that will yield two more new children .Here splitting of the two strings at the user crossover edge and exchange the two. The will result new population. The method chooses the set of test case from the available test suite that will shield all the errors detected earlier, in minimum execution time. Here bees are used as representatives who explore the minimum set of test cases. Fifty percent of the bees will start scavenging with randomly selected test cases. Now bees will append new test cases on their covered path if adding of a test case increases its fault detection capability. After adding one more test case, the bees coming back to their hive, and swap the information based on crossover operation of GA. The newly created set of test case formed after crossover is used by other bees to forage. This repetitive process continues till any of the bees has exposed a set of test cases that shields all faults detection and starts a joggle dance. The requirement for this process is an initial test suite 'T' consist of 'n' test cases. The outcome is a subset 'S', of m test cases where (m<=n), such that the test cases are carefully chosen based on maximum fault handling capacity in minimum execution time. Initially the 'n/2' that is fifty percent of the bees starts foraging. Each bee that has started forage will take the test case unsystematically.

The bees will choose test case based on the adding test case will rise the fault detection ability which can be verified by 'OR' operation.. The number of 1's existent in the result is the number of faults covered. The scavenged bees will return to their hive and inherently swap information by crossover method. Crossover will be performed between the bees that has minimum total execution time and bees with the next minimum execution time. We adopt that the bees that find out a set of test case in minimum time will create a new set of test case that will be achieved in minimum time and supportive in future to foresee improved and optimum result. If the resulted test case's total performance time is less than the maximum implementation time offered by the subset of test cases, the new bees will search using those subset of test cases as its initial track.

The outcome after crossover does not create new test will not be considered. If both test sets created after crossover will not create new set, no new bee will hunt. Now again the bees will select the test case. Repeating the same steps mentioned above till any of the bee has travel around a set of test cases that can cover all the test faults. As soon as the minimum sets of test case are created, bee started to dance for announcing the success. So the other bees can follow this test case trace. As the number of repetitions increase the system will give the optimum result. To explain, Bharti Suri considered a test suite, covering a total of 10 faults. The regression test suite 'T' as given in Table 1, contains eight test cases {T1, T2, T3, T4, T5, T6, T7, T8} and set of faults 'F' as {F1, F2, F3, F4, F5, F6, F7, F8, F9, F10}, and after positive application of his method, the outcome of the coverage and execution time is presented in the Table 2.

Table 1: Initial Test suite 'T'

Test Case	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10
T1	X		X			X			X	
T2		X						X		
T3		X			X		X			
T4				X		X		X	X	
T5	X					X				X
T6				X	X			X		
T7	X						X	X		
T8			X							X

Table 2: Foraged BEES return hive & their execution time

BEES	Test case selected	Total Execution time	No. of Faults Covered	Binary Form of Fault Covered
B1	T1,T5 T2,T6	19	9	1111110111
B2	T2,T6, T5,T3	17	8	1101111101
B3	T3,T7, T8,T4	15	10	1111111111
B4	T4,T8 T7,T6	16	9	1011111111
B5	T3,T8, T1,T2	17	9	1110111111
B6	T4,T7, T3,T8	15	10	1111111111

By the above results, it clearly shows that Bharti Suri's technique provided a minimized test suite {T3,T4,T7,T8} which cover all the faults or criteria with a minimum execution time.

A. Extended Dependence Analysis :

Yanping Chen, Robert L.Probert and Hasan Ural,,in [10], discussed to reduce the size of given test suite by investigating interface design covered by each test case in the given suit, that is according to a given set of elementary modifications(*EMs*) on an Extended Finite State Machine model (EFSM) of the requests of a system under test (SUT).

Yanping considered 3 types of EMs, i.e., an addition or deletion or a change of a transition. For each EM in the given set of EMs, data and control requirements are used to seizure possible relations between EFSM transitions, and three communication patterns are worked out: three communication patterns are calculated: (1) affecting communication design replicating the effects of the EM on the EFSM model, (2) impacted interaction design imitating the effects of the EFSM model on the EM, (3) side-effect communication design reflecting the side-effects presented by the EM. By Korel, B., Tahat, L.H., and Vaysburg, B, in [36], the Data Dependence (DD) captures the concept that one changeover states a value for a variable and the same or some other changeover may perhaps use this value and Control Dependence (CD) custodies the perception that one changeover may "stimulus" the track way of another transition. Control dependence between evolutions is defined in expressions of the theory of post-dominance. Static Dependence Graph (SDG) vividly characterizes DDs and CDs in an EFSM. In SDG, nodes denote EFSM evolutions and directed edges denotes DDs and CDs. The model centered testing has three steps 1) Testing the paraphernalia of the model on the adaptation, 2) Testing the special effects of the change on the model, 3) Testing the side-effects of the variation on the original parts of the model. Yanping further discussed about the three types of interface patterns associated to each EM: 1) an affecting communication pattern, 2) an affected interface pattern, and 3) a side-effect collaboration design. Within software, "design" states to key phases of a common strategy structure and providing a graphical demonstration as in Figure 3. By thus Yanping presented twelve diverse requirements, namely new dependences per modification (NDPM), rising from added extras, deletions, or changes of evolutions in the EFSM model.

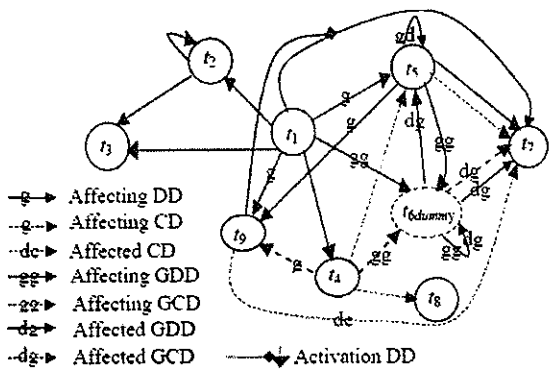

Figure 3 – Modified SDG for Simplified ATM Model

For a given new EFSM model, its static reliance graph, a test suite, and a set (M) of EMs, the anticipated test suite reduction routine first implement a set of EMs to given EFSM model to get the improved EFSM model, and then produces SDG for modified EFSM model using initial EFSM model, modified EFSM model and set of EM . Then for each EM m in set of EM, detects a subgroup of test suite of RTS consisting of test cases in initial test suite encompassing the changeover equivalent to *m* for each EM *m* in set of EM and for each test case *ts* in subclass of give test suite, builds up to 3 interface pattern for ts by using initial SDG and SDG of modified EFSM model and ts is encompassed in the reduced RTS if at least one of its interface designs has not been created for any of the test cases in the reduced Test suite. After lessening a given RTS, the proposed process tests the set of EMs and detects any unverified EMs. For each untried EM, one test case from those detached test cases that pass through the changeover corresponding to the EM has been randomly selected, and add this test case into the condensed RTS. In addition, some NDPMs may not be enclosed by the reduced RTS, and interface designs of an EM may not be fully sheltered. Retain trail of NDPM exposure and report all disclosed NDPMs at the end.

IV. Conclusion

A standard investigation was conducted on few of the popular Test suite Reduction Techniques. The CCTM method by Prema concentrates on reducing redundancy test case generation by classification which in turns provides the reduced test suites, whereas hybrid approach by Bharti Suri focus on finding a subset of test suite which covers maximum coverage criteria with the minimum execution time. And the dependency analysis by Yanping discuss about the effects of adding, deleting and changing a transition and its side effects in an EFSM model and providing reduced test suite as a result from the given large test suite. All of the algorithms studied are derived methods and have some common as well as their own distinctive characteristics. This study provides a relative exploration on test case reduction techniques.

V. Future work

Future work includes analyzing the test suite optimization techniques by using the Non-dominated sorting Genetic Algorithm which implements pareto ranking approach and also investigating a solution for better and more accurate results. And also to identify the external validity, variety of different real world problems has to be exercised in order to generalize the technique.

References

[1] Hennessy, M., and Power, J. F. 2005. *An analysis of rule coverage as a criterion in generating minimal test suites for grammar*

based software. In Proceedings of the 20th IEEE/ACM International Conference on Automated Software Engineering (ASE'05), (Long Beach, CA, USA, Nov. 2005), 104–113.

[2] McMaster, S., and Memon, A. 2005, *"Call stacks coverage for test suite reduction"*, In Proceedings of 21st IEEE International Conference on Software Maintenance.539-548.

[3] HwaYou Hsu and Alessandro Orso, *"MINTS: A General Framework and Tool for Supporting Test suite Minimization"*, Georgia Institute of Technology.

[4] Sreedevi Sampath and Renée C. Bryce. *"Improving the effectiveness of test suite reduction for user-session-based testing of web applications"*, Information and Software Technology 54 (2012) 724–738.

[5] M. Harder, J. Mellen, and M. D. Ernst, *"Improving test suites via operational abstraction"*, In International Conference on Software Engineering, 2003.

[6] J. Black, E. Melachrinoudis, and D. Kaeli, *"Bi-criteria models for all uses test suite reduction,"* in Proceedings of International Conference on Software Engineering, IEEE Computer Society, 2004, pp. 106–115.

[7] Lingming Zhang, Darko Marinov, Lu Zhang, Sarfraz Khurshid, *"An Empirical Study of JUnit Test-Suite Reduction"*, 2011 22nd IEEE International Symposium on Software Reliability Engineering, 1071-9458/11 $26.00 © 2011 IEEE DOI 10.1109/ISSRE.2011.26

[8] G.Rothermel, M.Harrold, J.Von Ronne, and C. Hong, *"Empirical studies of test-suite reduction,"* Software Testing, Verification and Reliability, vol. 12, no. 4, pp. 219–249, 2002.

[9] H. Zhong, L. Zhang, and H. Mei, *"An experimental study of four typical test suite reduction techniques"*, Information and Software Technology, vol. 50, no. 6, pp. 534–546, 2008.

[10] Yanping Chen, Robert L.Probert, Hasan Ural, *"Regression Test Suite Reduction Using Extended Dependence Analysis"*, SOQUA'07, September 3-4, 2007, Dubrovnik, Croatia Copyright 2007 ACM ISBN 978-1-59593-724-7/07/09.

[11] S. Tallam and N. Gupta. *A concept analysis inspired greedy algorithm for test suite minimization.* In Workshop on Program Analysis for Software Tools and Engineering, pages 35–42, September 2005.

[12] Srividhya J, K Alagarsamy, *"A Novel Method for Reduction of Regression Testing Cost"*, International Journal of Inventions in Computer Science and Engineering ISSN (Online): 2348 – 3539, ISSN (Print): 2348 – 3431 Volume 1 Issue 3 2014.

[13] Antonia Bertolino, Emanuela Cartaxo, Patry' cia Machado, Eda Marchetti, Joao Felipe Ouriques, *"Test suite reduction in good order: comparing heuristics from a new viewpoint"*, International Conference on Testing Software and Systems: Short Papers, October 2010, pp. 13–18.

[14] Zhong, H., Zhang. L., and Mei, H, *"An experimental comparison of four test suite reduction techniques"*, ICSE 2006, ACM, 636-639.

[15] McMaster, S., and Memon, A, 2005, *"Call stacks coverage for test suite reduction"*, In Proceedings of 21st IEEE International onference on Software Maintenance.539-548.

[16] Selva Kumar, S., and N. Ramarai, *"Regression test suite minimization using dynamic interaction patterns with improved FDE"*, European Journal of Scientific Research, 49, 3 (2011), 332-353.

[17] P. Prema, B.Ramadoss, S. R. Balasundaram, *"Combined Classification Tree Method for Test Suite Reduction"*, 2nd International Conference and workshop on Emerging Trends in Technology (ICWET) 2011 Proceedings published by International Journal of Computer Applications® (IJCA).

[18] Leung, H., and Prema, P, *"Testing COTS with classification tree method"*. In Proceedings of the IASTED International Conference on Software Engineering and Applications (L.A., Nov. 2003). SEA 2003, 270-276.

[19] Bharti Suri, Isha Mangal & Varun Srivastava, "Regression Test Suite Reduction using an Hybrid Technique based on BCO And Genetic Algorithm", Special Issue of International Journal of Computer Science & Informatics (IJCSI), ISSN (PRINT) : 2231–5292, Vol.- II, Issue-1, 2.

[20] Korel, B., Tahat, L.H., and Vaysburg, B, "Model-based regression test reduction using dependence analysis", In Proc. of ICSM'02 (Montréal, Canada, October 3-6, 2002). IEEE Computer Society Press, Washington, DC, 2002, 214-223.

Author's Biography

Dr.R. Gunasundari received the Ph.D. Degree in Computer Science from Karpagam University, Coimbatore in 2014. She is working as an Associate Prof & Head in the Department of Information Technology, Karpagam University, Coimbatore. She has published 20 National and 21 International papers in various journals. Her broad field of research is in Data mining.