ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 6, Issue 7, July 2019

A Perspective Study on Test Case Suite Reduction Tools of Regression Testing

[1] Bhawna Jyoti, [2] Dr Aman Kumar Sharma
[1] Computer Science Department, Himachal Pradesh University Shimla, India
[2] Professor, Computer Science Department, Himachal Pradesh University Shimla, India

Abstract: **In Modern era, thousands of software are released during every month. Testing and validation of software is an important activity which enhances the quality of software under test. Software testing is very expensive activity in the software development life cycle and is used to evaluate the quality of the software. Regression testing is an activity which ensures that changes to the existing software has not affected the normal functioning of software. This paper presents a review on regression testing tools used in test case suite reduction techniques and describes their strengths and weaknesses. We analyses and categorizes current reduction tools used by research community and identifying future research opportunities in this field.**

Keywords - Test case suit reduction, Tools based on Random testing, Compiler testing, Integer Programming Testing and Fault-Localization based testing.

I. INTRODUCTION

Software testing is an important activity which plays a significant role to detect faults, errors and defects to see whether the system produces correct output and improves the quality of software[1][2]. It is the most expensive practice because resources of developing and maintaining software are related to the testing process for ensuring quality of software [7][10][29]. In modern era, great attention is paid on the test suite maintenance of large scale software systems to overcome the issues of cost, size and fault detection effectiveness of software system under test[7][10]. As a software system spread its dimensions, its test suites need to be updated as well as maintained to verify new or modified functionality of the software[1][2][3][10].
Regression testing is a critical test activity that is used to validate software changes which provides confidence that newly introduced changes during software evolution does not affect the normal functioning of system under test[2][[7][10][28]. Due to time, size and resource constraints, retesting of modified software system becomes difficult when a new version is released [10][22]. It is really important to search for techniques that attempts to find a minimal subset of test cases i.e. reduced test suite that will satisfy all testing requirements matrix similar to the original test suite without affecting the fault revealing capabilities of test suite [1][2][7][25][29].
Regression testing is needed whenever new requirement arises by customer, performance and bug fixing issues, modifications in software are introduced[2][3]. Different techniques have been proposed by researchers to efficiently address cost as well as fault detection rate in the minimized test suite[1][4][5][7][8].

All the regression test suite reduction approaches[3][5][8][11] pays great attention on finding a minimal test suite by permanently eliminating the redundant test cases from the original test suite and as a affect, cost of executing, validating test suites[7][10] over future releases of software is significantly decreased. The test case suite reduction technique can be considered as the minimal hitting set problem[9][28][29] which is based on finding a reduced subset of test cases having minimum cardinality that satisfies all the requirements of original test suite with retaining the powers of fault detection effectiveness[7][10].
Regression testing tools are needed when software change takes place due to change in code, requirements and technology [6][12]. To deal with problems of exhaustive testing in organizational domains, optimal testing tools are very much beneficial to find a reduced test suite by eliminating redundant test cases in original test suite[2][12].
The rest of the paper is organized as follows: Sec 2 presents parameters and taxonomy of Test case suite reduction tools. In Sec 3, analysis of tools is done by comparing their strengths and weakness and Sec 4 gives future research issues followed by conclusion.

II. TEST SUITE REDUCTION TOOLS

A. Parameters

The test case suite reduction tools[12]are based on various parameters which are described as under:
• **Type of approach:** It presents type of methodology used by reduction framework. The methodology may be

Coverage based, Search based, Integer Linear Programming Based and Data mining based [28].

• **Paradigm used for testing:** It tells about the supported language like C, Java etc. in which tool is implemented in test reduction framework [12][28].

• **Type of Optimization used:** There are basically two types of optimization used in tools- single objective and multi-objective. The single-objective based optimization type considers single criteria may be loss in fault-detection capability or effectiveness in terms of size of the reduced subset. In multi-objective both reduced test suite and fault detection effectiveness are calculated as comparison to single-objective optimization[10][12][18][26].

• **Server platform for execution of test cases:** Platform for execution of test cases may be single server and multiple server. In case of multiple servers, divide-and-conquer strategy is used by diving the problem into many sub-problems and each sub-problem is executed on a single server to reduce testing time of the reduction process[12][28].

• **Mode of computation:** It is the type of computation processing supported by a framework and may be online and offline mode[12].

• **License:** It is the license of tool which may be commercial, academic research and free open source tools.

• **Customizability:** It is defined as an ability of the tool to support alterations in basic functions and it includes full, partial and no customizability.

B. Categories of Existing frameworks of test suit reduction tools:

1. Tools based on Random selection of elements in unit testing:

These tools are based on randomized unit testing and are used for generating method sequences calls based on distinct test inputs easily and quickly from common data structures and is very useful in exposing defects of system under test[6][9][11][12].

ATAC: This tool is based on coverage flow based approach[17].

Rostra: This tool is used to evaluate quality of test suite by finding similar unit test cases of equivalent objects[20]. Source code, time and space taken to find redundant test cases are used[12][20].

GenRed: A feedback-directed tool is used to generate feedback by executing many randomly generated method sequences and reduce object oriented test cases[15]. This tool follows a sequence based test case suite reduction technique and eliminate redundant test cases without their execution[12][15].

RUTE-J : It uses delta debugging technique to isolate the failure-inducing inputs of the program under test[6].

Randoop: This tool takes input as set of classes and contract checkers and produces output as contract violating test and regression test. It does not require pre-existing test suite and removes the test inputs which throw exceptions or belongs to similar object [18].

TOBIAS: This tool is based on stochastic approach and captures the tester knowledge to write test patterns with manual effort. Then test patterns are unfolded to find and eliminate redundant test cases in a test suite [12][19].

2. Tools used in Web Application Testing:

USbRed: This tool reduces a set of recorded user session data and focuses on coverage and fault detection capabilities of reduced test suite in terms user recorded session data[27].

CPUT: This tool is designed to reduce the user-session based test cases to identify faults by using execution traces. A generalized logger is used with Apache server. This tool uses a small sized web application[29].

3. Tools Based On Compiler testing:

This framework focus on testing the back-end of a retargeted compiler for reducing the test suite, since the back-end relies on the targeted processor.

RTL: Testing is done on back-end of a retargeted compiler for intermediate testing of code and based on high level abstraction framework for compilers. In it test generation is based on grammar based approach[12][21].

PLOOSE: The minimal representation of intermediate inputs by generating a number of test cases based on the given C grammar-coverage criteria and converts these into RTL code and optimal test suite is obtained[12][25].

4. Tools based on Integer Linear Programming:

MINTS: This tool is ILP-based, object oriented, Multi-Objective problem domains and more effective than single objective in terms of cost and effectiveness, defined encoding mechanism, easily plug-in different ILP solvers[16].

EDTSO: This tool give code coverage and minimum energy consumption of test suites and suitable for anroid applications. Use multiple ILP solvers in parallel fashion[4].

5. Tools based on Fault Localization: These tools are very much beneficial as fault localization[7][10][24] is most important and expensive activity to access the quality of reduced test suite.

SrTC: This tool works on the uneven distribution of test cases and a framework retained a small number of redundant test cases to improve fault-localization effectiveness. Reduction Processor, w Processor, and Evaluation is used which produces five types of data[1] and fault detection effectiveness is improved[1].

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 6, Issue 7, July 2019

JINSI: This tool works on passing and failing execution of test cases and combine delta debugging techniques with event slicing[14].
GZoltar: This tool is used as plug-in for Eclipse IDE environment. It is based on spectrum, cardinality and find representative suite by determining their test execution covering time. Visual representations of data analyzed under tool[13] are given.

III. ANALYSIS:

Tools based on randomized unit testing[6][12][15][20] are inefficient when there is high dimensional data of large software systems. Tools which are mainly focused on coverage-based approaches[3][8] exploit the coverage of a system under test to determine the reduced suite. Existing TSR tools mainly targeted to solve the single-objective TSR optimization problems which is impractical for a testing scenario containing multiple objectives and constraints. A comparative study is done on these techniques which is presented in tabular form i.e. Table 1.

TABLE 1: A Comparative Study on Different Test suite reduction Techniques

Category	Tool Name	Strengths	Weaknesses
Tools based on Random selection	ATAC	Elimination of test cases is done that are redundant. Use of data flow coverage metric and execution slices. Coverage based approach which follows structured testing and focus on cost in terms of size of representative set[17]. License: Free, full customizability	Dependent very much on human interactions and needs more testing time to create test data and their evaluation. Whenever there is increase in program size, memory storage of collected coverage information and management cost is very much increased. Fault detection capability is less as it supports only selective testing[17].
	Rostra	Formal Object- oriented unit testing framework based on Java language. Coverage based and single objective which finds similar unit test cases based on equivalent objects[20].License: Research	Sometimes there is possibility not to find minimal solution. Test oracle problem. When executing source code special attention have to pay manually because changes in the existing redundant code can make a test case non-redundant in the new equivalent classes. No customizability[20]
	GenRed	Object- oriented, Coverage based selection of methods and reduction technique is based on sequence of method calls of source code. single objective. License: Research[15]. This tool uses combination of code coverage based reduction and method sequence based reductions and provides significant improvement in test suite minimization[12][15].	It needs good experience to handle tool for entering some technical input of system under test[15].No customizability[12][15].
	Randoop	Feedback directed mechanism based. Object- oriented, Coverage based useful to create similar objects or throw exceptions ,Single objective. License: Free, full customizability[18].	Requires high human interactions to set time limits for finding representative set and it finds small number of redundant test cases[18]. It is based on generated feedback and methods can be tested only if there are relevant feedbacks[18].
	TOBIAS	Semi-automated based on combinatorial testing, Object- oriented, coverage based and single	Less effective in terms of cost and effectiveness as compared to Multi-objective criteria[19]

ISSN (Online) 2394-6849

International Journal of Engineering Research in Electronics and Communication Engineering (IJERECE)
Vol 6, Issue 7, July 2019

		objective[19]License: Research[19]	
	Open-SourceRed	Two open source components: Proteja and Modificare Search and coverage based, object oriented and single objective License: Free, full customizability	Need high user experience regarding tool[26]
	TEMSA	Multi-objective using feature coverage and search based .It generates minimized test suite[22].License: Research	It may be very time consuming in case of evaluating fitness function followed by different possible candidate solutions[22]. No customizability.
	Raspect	Requires less manual code inspection, Aspect oriented, Coverage based, Single objective[9],License: Research	Performance degraded as it is extension of Rostra in case of non availability of algebraic expressions of program[9]
	RUTE-J	Coverage based, single objective and object oriented[12]License: Free, full customizability	Requires high human interactions to enter some technical inputs[12].
Tools used in Web Application testing	USbRed	Reduce user session data, Concept analysis based, Object oriented and Single objective. License: Research[27].	Not adequate for capturing of execution traces, limited code coverage because it needs particular for given program paths. User session data will become invalid if there is small modifications in web applications. No customizability[27].
	CPUT	A general tool to test small web applications, use black box testing method so cost effective in comparison to coverage based techniques[29].	Limited code coverage by tool as particular type of input is required for exercising certain paths of a program[29].
Tools based on Compiler Testing	RTL	Testing is done on back-end of a retargeted compiler for intermediate testing of code. In it test generation is based on grammar based approach[21].	This tool is not suitable for large scale applications[21].
	PLOOSE	Extension of RTL tool. In it , test generations are based on C grammar coverage and converts tests into RTL code by using translator and then eliminate redundant test[25].	It needs to be enhanced to support new test suite reduction techniques[25].
Tools based on Integer Linear Programming	MINTS	ILP-based, Object oriented, Multi-Objective problem domains and more effective than single objective in terms of cost and effectiveness, defined encoding mechanism, easily plug-in different ILP solvers[16]. Reduced computational time[16].License: Free	Very effective but fully dependent on expertise in testing field. Non-availability of historical data can lead to difficulty in finding optimal solution[16].
	EDTSO	This tool give code coverage and minimum energy consumption of test suites and suitable for anroid applications. Use multiple ILP solvers in parallel fashion[4].	For execution of test cases, high time and memory complexity is needed[4].
Tools Based on Fault	SrTC	This tool gives concept of relative redundancy and fault detection effectiveness is significantly improved[1].	Consider coverage information instead of concrete path and may remove test cases which are relevant and thus provide less fault-detection effectiveness[1].

Localization	JINSI	This tool works on passing and failing execution of test cases and combine delta debugging techniques with event slicing[14]	No visual representations of report[14] as is given by GZoltar.
	GZoltar	Used as plug-in for Eclipse IDE environment. Based on spectrum based, cardinality and find representative suite by determining their test execution covering time. Give visual representations of data analyzed under tool[13]	Only limited customizability[13]

IV. CONCLUSION AND FUTURE WORK

Tools are mainly based on Coverage code methodologies to determine the reduced test case suite, which seldom emphasis on fault-detection effectiveness[1][7][25][27]. Alternatively, by injecting the diversity in test cases, the search based approaches explore significant potential to detect real faults. Most of the existing reduction tools mainly targeted to solve optimization problems having single objectives but impractical platform for testing under Multi-objective scenario. More attention can be paid on Multi-objective based optimization problems for achieving better cost effectiveness and fault detection capability of system under test. The Hybrid solutions are required to be formulated and augmented with existing tools to target Multi-objective optimization problems. Focus is needed to be paid on similarity based reduction techniques to obtain optimal solutions by focusing on automation tool support to efficiently work with Multi-objective and multi -server test suite reduction problems[30][31].

REFERENCES

1. X. Zhang , Gu, Q., Chen, X., J. Qi and D. Chen, “A study of relative redundancy in test-suite reduction while retaining or improving fault-localization effectiveness,” in Proceedings of the ACM Symposium on Applied Computing (SAC'10), Switzerland: ACM, 2229-2236, 2010.
2. Singh, Rajvir and Mamta Santosh, “Test case Minimisation techniques: a review,” International Journal of Engineering Research and Technology, vol 2, no. 12, 2013.
3. S. U. R. Khan and A. Nadeem, “TestFilter : A Statement Coverage based Test case Reduction Technique,” in Proceedings of international Multitopic Conference, IEEE, Dec 2006.
4. D. Li, Y. Jin, C. Sahin, J. Clause,& W.G. Halfond, “Integrated energy-directed test suite optimization,” Proceedings of the ACM International Symposium on Software Testing and Analysis (ISSTA'14), 339-350, 2014.
5. A. Emilia ,V. B. Coutinho, E. G. Cartaxo, D. L. Patrica , Machado, “ Test Suit Reduction Based on similarity of test cases,” Software Quality Journal, vol 24,no. 2, June 2016.
6. J. H. Andrews, S. Haldar, Y. Lei and F. C. H. Li, “Tool support for randomized unit testing,” Proceedings of the First ACM International Workshop on Random Testing, pp. 36-45,2006.
7. A G. Rothermel, M. J. Harrold, J. Ostrin and C. Hong, “An Empirical Study of the Effects of Minimization on the Fault Detection Capabilities of Test Suites,” Proceedings of the International Conference on Software Maintenance, Washington, D.C., November, 1998.
8. D. Hao, L. Zhang, “On demand test suit reduction,” ICSE, 2012.
9. T. Xie, J. Zhao, D. Marinov and D. Notkin, “Detecting Redundant Unit Tests for AspectJ Programs,” Technical Report , UW-CSE-04-10-03,October 2004.
10. A. B. Taha, S.M. Thebaut, S.S. Liu, “An Approach to software fault localization and revalidation based on incremental data flow analysis,” Proceedings of the international Computer Software and Applications Conference, IEEE Computer Society Press,1989,527-584.
11. Y. Lei and J. H. Andrews, “Minimization of Randomized Unit Test Cases,” Proceedings of the 16th IEEE International Symposium on Software Reliability Engineering, 2005.
12. S. U. R .Khan, S. P. Lee, R .W. Ahmad, A. Akhunzada and V. Chang, “A Survey on Test Suite Reduction Frameworks and Tools,” International Journal of Information Management, vol. 36 ,no. 6, pp. 963-975, 2016.
13. J. Campos, A. Riboira, A. Perez and R. Abreu, “GZoltar: An Eclipse Plug-In for Testing and Debugging,” Proceedings of the 27th IEEE/ACM International Conference on Automated Software Engineering,2012.
14. M. Burger and A. Zellar, “Minimizing reproduction of software failures,” in Proceedings of the 2011 ACM International Symposium on Software Testing and Analysis (ISSTA'11), pp. 221-231, 2011.

15. H. Jaygarl, Kai-Shin Lu, C. K. Chang, "GenRed: A Tool for Generating and Reducing Object-Oriented Test Cases," in IEEE 34th Annual Computer Software and applications Conference, Dec 2010.
16. H. Hsu, A. Arso, "MINTS: A general framework and tool," IEEE publications, in Proceedings of the 31st IEEE International Conference on Software Engineering (ICSE'09) pp. 419-429, 2019.
17. J. R. Horgan and S. London, "A data flow coverage testing tool for C," Proceedings of the Second IEEE Symposium on Assessment of Quality Software Development Tools, pp. 2-10, 1992.
18. C. Pacheco, and M. D Ernst, "Randoop : feedback-directed random testing for Java," Proceedings of the 22nd ACM SIGPLAN Conference on Object-Oriented Programming Systems and Applications Companion, pp. 815-816, 2007.
19. F. Dadeau, Y. Ledru, L. D. Bousquet, "Directed random reduction of combinatorial test suites," in Proceedings of the 2nd International Workshop on Random Testing: co-located with the 22nd IEEE/ACM International Conference on Automated Software Engineering (ASE'07), ACM, pp.18-25, 2007.
20. T. Xie, D. Marinov, & D. Notkin, "Rostra: A framework for detecting redundant object-oriented unit tests," in Proceedings of the 19th IEEE International Conference on Automated Software Engineering (ASE'04), pp. 196-205, 2004.
21. G. Woo, H. S. Chae and H. Jang, "An intermediate representation approach to reducing test suites for retargeted compilers," In Reliable Software Technologies–Ada Europe, Springer Berlin Heidelberg, 2007.
22. S. Wang, S. Ali and A. Gotlieb, "Cost-effective test suite minimization in product lines using search techniques," Journal of Systems and Software, 103, 370-391, 2015.
23. Y. Huang, L. Lu , "A methodology for test suit reduction in user-session-based testing," in IEEE Fifth International Conference on Bio-Inspired Computing: Theories and Applications, pp. 23-26 Sept. 2010.
24. P. Velmurugan, R. P. Mohapatra, "Effective Test Case Minimization and Fault Detection Capability Using Multiple Coverage Technique," in International Journal of Applied Engineering Research, ISSN 0973-4562, vol 11, no 8, pp 5389-5394, 2016.
25. H. S. Chae, G. Woo, T. Y. Kim, J. H. Bae,& W. Y. KIM, "An automated approach to reducing test suites for testing retargeted C compilers for embedded systems," in Journal of Systems and Software, pp. 2053-2064, 2011.
26. X. Wang, S. Jiang, P. Gao, X. Ju, R. Wang, Y. Zhang, "Distance-based Test-Suite Reduction for Efficient Testing-based Fault Localization", International Conference on Software Analysis, Testing and Evolution,2016.
27. S. Sampath, S. Sprenkle, E. Gibson, L. Pollock and A. S. Greenwald, "Applying concept analysis to user-session-based testing of web applications," IEEE Transactions on Software Engineering, vol 33, pp. 643-658, 2007.
28. S. Yoo and M. Harman, "Regression Testing Minimisation, Selection and Prioritization: A Survey," in Software testing verification and reliability, 2007.
29. S. Sampath , R. C. Bryce , S. Jain and S. Manchester, "A tool for combination-based prioritization and reduction of user-session-based test suites," Proceedings of the 27th IEEE International Conference on Software Maintenance (ICSM'11), pp. 574-577, 2011.
30. S. Yoo and M. Harman, "Pareto efficient multi-objective test case selection," in Proceedings of the International Symposium on Software Testing and Analysis (ISSTA'07), ACM, pp. 140-150,2007.
31. J. M. Kauffman and G. M. Kaphammer, " A framework to support research in and encourage industrial adoption of regression testing techniques," in Proceedings of the Fifth IEEE International Conference on Software Testing, Verification and Validation (ICST'12), pp. 907-908, 2012.