Transactions of the Nigerian Association of Mathematical Physics
Volume 14, (January - March, 2021 Issue), **pp155 –158**

FIREWALL-BASED REGRESSION TESTS SELECTION APPROACHES: A REVIEW

[1]Samaila Musa and [2]Jibril Lawal

[1]Department of Computer Science, Federal University Gusau
[2]Department of Mathematical Sciences, Federal University Gusau

Abstract

Regression testing is about running the entire test and is very important activity in software testing. Regression Test Case Selection (RTCS) aims to reduce regression testing cost by only selecting and running the tests that may be affected by code changes. Various RTCS techniques based on firewall analysing at either statement, methods of class levels have been proposed. This paper presents the result of the study conducted on different Regression Test Case Selection approaches based on firewall showing mostly stressed areas by researchers and the areas where there is a future scope. Papers related to firewall-based Regression Test Case Selection techniques in regression testing are reported and documented to find the current trends and future scope for the researchers to work upon.

***Keyword*:** Regression testing, test case selection, software testing, firewall.

Introduction

One of the most important activities in software maintenance is regression testing [1] that is performed in order to ensure that modifications due to debugging or improvement do not affect the existing functionalities and the initial requirement of the design. According to [2], regression testing comprises execution of an enormous amount of tests and is tedious, but it might not be feasible to repetitively re-execute all the test suite in order to tests the software during regression testing due to resource restraints. In this way, it is reasonable to select test case by selecting part of the test suite so that software is tested for early identification of faults. There are numerous regression tests selection techniques, for example, retest-all or randomly selects test cases, however using some pre-defined criteria for selection of test cases might result in more fault identification. There were recently proposed techniques in regression testing [3-12] in order to improve regression testing performance so as to make it a cost-effective process. These techniques are categorized as test suite minimization, test case se-lection, and test case prioritization [13].

Regression test case selection helps in selecting part of tests from the test suite. The most effortless approach is the tester essentially executes all the current tests to guarantee that the new changes are innocuous and is referred as a retest-all technique [14]. It is the most secure method; however, it is conceivable just if the test suite is small in size. The tests can be chosen arbitrarily to diminish the extent of the test suite. In any case, the greater part of the tests chose arbitrarily can bring about checking little portions of the altered programming, or may not have any connection to the adjusted program. Formally, regression test selection techniques will be an options to the retest-all and random approaches. Regression tests selection techniques for object-oriented programs are grouped into 3 categories [15], namely: Firewall-based regression tests selection, Program model-based regression tests selection and Design-based regression tests selection.

In regression testing, variables are measures from the experiment by measuring the first five values directly and the last four values (Equations 1, 2, 3 and 4indirect measured as used in our previous approaches [38].

1. Number of killed or detected mutants (#killed mutants)
2. Number of test cases killed mutants (selected test cases) #N
3. Execution time is taken to detect mutants by an approach measured in seconds
4. Position of test: Tf_i= position of the first test in T that exposes the faults i
5. A number of executed test cases: Is the number of tests used to killed mutants (# test case executed).
6. Number of undetected mutants (i.e. #equivalent mutants)
   $$\#Equivalent\ mutant\ = Total\ mutants - \#killed\ mutants \quad (1)$$
7. The effectiveness of fault detection (mutation score) measured as:

Correspondence Author: Samaila M., Email: samailaagp@gmail.com, Tel: +2348065670727

$$Eff_{MSP} = \frac{\#killed\ mutants}{Total\ mutants} * 100 \quad (2)$$

9. The efficiency of test effort measured as:

$$RTE = \frac{\#mutants\ killed\ by\ the\ test\ case\ executed}{execution\ time\ to\ detect\ the\ mutants} \quad (3)$$

10. Average percentage of the rate of fault detection:

$$APFD = 1 - \frac{Tf1+Tf2+\cdots+Tfm}{\#N * \#killed\ mutants} + \frac{1}{2*\#N} \quad (4)$$

Literature Survey

There were a lot of studies performed by researchers and a lot selection techniques were introduced, with the aim of enhance rate of fault detection. These techniques were categorized into several domains according to the nature of the study conducted by the researchers. A review was conducted in which various regression test case selection techniques were examined to enhance the significance of the selection of test cases from test suite in regression testing [16]. The study conducted by these researchers is elaborated correspondingly into procedural programs. Also another review on Test Case Selection was presented which was conducted in conferences and journals. But most of the commonly reported techniques are genetic algorithm, adaptive random testing and greedy algorithm.

Firewall-based techniques for the Object-Oriented Programs depended on the idea of a firewall characterized for the procedural program [18]. The techniques depended on examination of data and control conditions among modules in the procedural system. Firewall-based methodology for OOP identifies the changed classes for the altered rendition of the program as a firewall. The tests to be chosen are those that at any rate practiced one class from the inside of the firewall. Firewall-based approach for object-oriented program aims to detect the changed classes for the amended form of the program. They present a selective retest technique directed specifically at inter procedural regression testing that handles both code and specification changes. Their technique determines where to place a *firewall* around modified code modules. Where test selection from T is concerned, the technique selects unit tests for modified modules that lie within the firewall, and integration tests for groups of interacting modules that lie within the firewall.

A lot of firewall-based approaches for object-oriented programs have been proposed [19, 27]. The notion of class firewall analysis by taking into account the features of OO languages, such as inheritance was proposed [28]. A state-of-the-art static file level RTCS technique based on class firewall analysis called STARTS was proposed {24}. STARTS computes the set of classes that might be affected by the changes and builds a "firewall" around those classes; then, any test classes within the class firewall can potentially be affected by code changes and are selected as affected test cases**.** STARTS performs class firewall analysis based on the Intertype Relation Graph (IRG).

A technique that implemented the traditional class firewall analysis as a practical static RTS tool named STARTS, and performed an extensive study of STARTS on a number of real-world GitHub projects was presented [24]. The experimental results verified that STARTS can significantly better than static method-level RTCS. White and Abdullah (1997) proposed a technique [26] that constructs a firewall to enclose the set of classes affected by the changes; such classes are the only ones that need to be retested. Another technique by Hsia and colleagues [25] is also based on the concept of class firewall, but leverages a different representation of the program. But the techniques are limited because they do not handle certain object-oriented features (e.g., exception handling) and perform analysis only at the class level, which can be imprecise. Furthermore, they are not implemented and, thus, there is no empirical evidence of their effectiveness or efficiency.

Techniques based on the idea of a class firewall [18, 27] were proposed that enclose the types that need to be retested because they may be impacted by a code change. The concept of firewall to cover all the affected modules that were modified at the module integration level was presented [22]. The firewall was based on the call graph defined and the effort can be decreased by retesting only those modules and links that were present in the firewall of the modified or affected modules instead of retesting the whole of the system. Later on the authors also introduced data flow based firewalls that were based on the data flow diagram of the affected modules due to coupling effect.

The work was extended to handle interactions involving global variables **[**28**]**. A firewall analysis for regression testing with integration test cases in the presence of small changes in functionally-designed software was developed [18, 22, 28]. Firewall analysis restricts regression testing to potentially-affected system elements directly dependent upon changed system elements [22, 29]. Affected system elements include modified functions and data structures, and their calling functions. The firewall concept has been utilized on object-oriented systems [30, 31] and for regression testing of graphical user interfaces [32].

Firewall methods can only be guaranteed to select all modification-revealing tests and to be safe if all unit and integration tests initially used to test system components are reliable. *Modification-revealing test cases* are those test cases, when executed before and after the modification, for which the program will generate different output [33]. A *safe* RTS process guarantees that the subset of tests selected contains all test cases in the original test suite that can reveal faults based upon the modified program [33, 34]. Tests are *reliable* if the correctness of modules exercised by those tests for the tested inputs

implies correctness of those modules for all inputs [33]. Since test suites are typically not reliable in practice [29], the firewall technique may omit modification-revealing tests and/or may admit some non-modification-traversing tests. However, via empirical studies of industrial real-time systems, firewall was shown to be effective despite these theoretical limitations [29].

A class firewall regression test selection technique was reported on the empirical evaluation in combination with scenario testing, on a large scale industrial software system using the Java byte code in the analysis [35]. It was conducted on a large complex distributed software system in one of Sweden's largest banks. The results show that not all test cases were selected by the class firewall selection technique. A hypothesis was formulated stating that the empirical observations of class firewall are not incidental, but an inherent property of the Class firewall technique [36]. It was proved that the hypothesis holds for Java in a stable testing environment, and concluded that the effectiveness of the Class firewall regression testing technique can be improved without sacrificing the defect detection capability of the technique, by removing the class firewall. A technique that investigates situations when data-flow paths are longer and the testing of modules and components only one level away from the changed elements may not detect certain regression faults; an *extended firewall* considers these longer data paths. The results show that empirically the degree to which an extended firewall detected more faults, and how much more testing was required to achieve this increased detection [37].

An evolutionary regression testing for object-oriented software based on extended system dependence graph model of the affected program was presented in one of previous work [39]. The approach is based on optimization of selected test case from dependency analysis of the source codes. The goal is to identify changes in a method's body due to data dependence, control dependence and dependent due to object relation such as inheritance and polymorphism, select the test cases based on affected statements. The number of affected statements determined how fit a test case is good for regression testing. A case study is reported to provide evidence of the feasibility of the approach and its benefits in increasing the rate of fault detection and reduction in regression testing effort compared with retest-all. It was shown that our approach needs 30% of the test cases to cover all the faults, while 80% is needed to cover all the faults using retest-all, which is time consuming and costly.

Conclusion

This paper presents a review on regression test selection techniques based on firewall which assesses the research work related to this area. The paper summarises the research papers along with the approaches based on firewall. It can conclude that there are so many approaches that are used for test case selection based on firewall and each approach has its advantages and disadvantages. Based on the requirement a tester/researcher can use any of the firewall-based approach.

REFERENCES

[1] Zhang, L., Hou, S. S., Guo, C., Xie, T., & Mei, H. (2009, July). Time-aware test-case prioritization using integer linear programming. In *Proceedings of the eighteenth international symposium on Software testing and analysis* (pp. 213-224). ACM.

[2] Rothermel, G., Roland, H.U., Chu, C., &Harrold, M.J. (2001). Prioritizing test cases for regression testing. *IEEE Transactions on Software Engineering,* 27(10), 929-948

[3] Ngah A, Munro M, Abdallah M. An Overview of Regression Testing. *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)*. 2017; 9(3-5): 45-49.

[4] Pandey A, Banerjee S. Test suite minimization in regression testing using hybrid approach of ACO and GA. *International Journal of Applied Metaheuristic Computing.* 2018; 9(3): 88-104.

[5] Chen L, Zhang L. *Speeding up mutation testing via regression test selection: An extensive study*. Proceedings of the 11th IEEE International Conference on Software Testing, Verification and Validation (ICST 2018). 2018: 58-69.

[6] Jimenez I, Watkins N, Sevilla M, Lofstead J, Maltzahn C. *Quiho: Automated performance regression testing using inferred resource utilization profiles*. Proceedings of the ACM/SPEC International Conference on Performance Engineering. 2018: 273-284.

[7] Minhas N. M, Petersen K, Ali N, Wnuk K. *Regression testing goals-view of practitioners and researchers*. Proceedings of the 24th Asia-Pacific Software Engineering Conference Workshops (APSECW 2017). 2018: 25-31.

[8] Tulasiraman M, Kalimuthu V. Cost cognizant history based prioritization of test case for regression testing using immune algorithm. *International Journal of Intelligent Engineering and Systems.* 2018; 11(1): 221-228.

[9] L. Zhang. *Hybrid regression test selection*. Proceedings of the IEEE/ACM 40th International Conference on Software Engineering (ICSE 2018). 2018: 199-209.

[10] A. Panichella and R. Oliveto, M. Di Penta, and A. De Lucia. Improving Multi- Objective Test Case Selection by Injecting Diversity in Genetic Algorithms. *IEEE Transactions on Software Engineering*. 2015; 41(4): 358-383.

[11] S. Dahiya, R. K.Bhatia, and D. Rattan. Regression Test Selection using Class, Sequence, and Activity Diagrams. IET Software. 2016; 10(3): 72-80.

[12] S. S. Emam and J. Miller. Test Case Prioritization using Extended Digraphs. ACM Transactions on Software Engineering and Methodology (TOSEM). 2015; 25(1): 6:1-6:41.

[13] Yoo, S., & Harman, M. (2012). Regression testing minimization, se-lection and prioritization: a survey. *Software Testing, Verification and Reliability*, *22*(2), 67-120. https://doi.org/10.1002/stv.430.

[14] M. Rani and A. Singh (2014). Review of Regression Test Case Selection Techniques, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181.IJERTV3IS051262 www.ijert.org. Vol. 3 Issue 5, May – 2014.

[15] S. Narula (2016) Review Paper on Test Case Selection. Saakshi Narula| IJCSET(www.ijcset.net) |April 2016 | Vol 6, Issue 4, 126-128

[16] M. Rani and A. Singh (2014). Review of Regression Test Case Selection Techniques, International Journal of Engineering Research & Technology (IJERT), ISSN: 2278-0181.IJERTV3IS051262 www.ijert.org. Vol. 3 Issue 5, May - 2014

[17] S. Narula (2016) Review Paper on Test Case Selection. Saakshi Narula| IJCSET (www.ijcset.net) |April 2016 | Vol 6, Issue 4, 126-128

[18] Leung, H. and White, L., "A Study of Integration Testing and Software Regression at the Integration Level," International Conference on Software Maintenance, San Diego, 1990, pp. 290-301.

[19] Jang, Y., Munro, M.,& Kwon, Y. (2001). An improved method of selecting regression tests for C++ programs. *Journal of Software Maintenance: Research and Practice*, 13(5), 331–350.

[20] Rothermel, G., & Harrold, M. J. (1997). A safe, efficient regression test selection technique. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, *6*(2), 173-210.

[21] Kung, D., Gao, J., Hsia, P., Wen, F., Toyoshima, Y. and Chen, C. (1996). On regression testing of object oriented programs. *Journal of Systems and Software*, 32(1), 21–40.

[22] Leung, H.,& White, L. (1992, November). A firewall concept for both control-flow and data-flow in regression integration testing. *In Proceedings of the Conference on Software Maintenance*,(pp. 262 – 271). IEEE.

[23] D. C. Kung, J. Gao, P. Hsia, J. Lin, and Y. Toyoshima, "Class firewall, test order, and regression testing of object-oriented programs," JOOP, vol. 8, no. 2, pp. 51–65, 1995

[24] O. Legunsen, F. Hariri, A. Shi, Y. Lu, L. Zhang, and D. Marinov, "An extensive study of static regression test selection in modern software evolution," in FSE, pp. 583–594, *November 13-19, 2016, Seattle, WA, USA.* 2016 ACM. ISBN 978-1-4503-4218-6/16/11. DOI: http://dx.doi.org/10.1145/295029

[25] P. Hsia, X. Li, D. Kung, C.-T. Hsu, L. Li, Y. Toyoshima, and C. Chen. A technique for the selective revalidation of OO software. Software Maintenance: Research and Practice, 9:217–233, 1997.

[26] L. J. White and K. Abdullah. A firewall approach for regression testing of object-oriented software. In Proceedings of 10th Annual Software Quality Week, May 1997.

[27] A. Orso, N. Shi, and M. J. Harrold. Scaling regression testing to large software systems. In Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, pages 241–251, 2004.

[28] Leung, H. and White, L., "Insights into Testing and Regression Testing Global Variables," *Journal of Software Maintenance*, 2(4), 1991, pp. 209-222.

[29] White, L. and Robinson, B., "Industrial Real-Time Regression Testing and Analysis Using Firewall," International Conference on Software Maintenance, Chicago, 2004, pp. 18-27.

[30] Kung, D., Gao, J., Hsia, P., Wen, F., Toyoshima, Y., and Chen, C., "Change Impact Identification in Object-Oriented Software Maintenance," International Conference on Software Maintenance, Victoria, B.C., Canada, 1994, pp. 202-211.

[31] White, L. and Abdullah, K., "A Firewall Approach for the Regression Testing of Object-Oriented Software," in *Software Quality Week*, San Francisco, 1997

[32] White, L., Almezen, H., and Sastry, S., "Firewall Regression Testing of GUI Sequences and Their Interactions," International Conference on Software Maintenance, Amsterdam, The Netherlands, 2003, pp. 398-409.

[33] Rothermel, G. and Harrold, M., "Analyzing regression test selection techniques," *IEEE Trans. on Software Engineering*, 22(8), 1996, pp.529-551.

[34] Bible, J., Rothermel, G., and Rosenblum, D., "A Comparative Study of Course- and Fine-Grained Safe Regression Test-Selection Techniques," *ACM Transactions on Software Engineering and Methodology*, 10(2), 2001, pp. 149-183.

[35] M. Skoglund and P. Runeson. A Case Study of The Class Firewall Regression Test Selection Technique on a Large Scale Distributed Software System

[36] M. Skoglund and P. Runeson Improving Class Firewall Regression Test Selection By Removing The Class Firewall. International Journal of Software Engineering and Knowledge Engineering Vol. 17, No. 03, pp. 359-378 (2007)

[37] L. White, K. Jaber, Brian Robinson and Václav Rajlich (2008) firewall for regression testing: an experience report Journal of Software Maintenance and Evolution: Research and Practice November 2008

[38] Musa, S., Sultan, A. B. M., Ghani, A. B., & Baharom, S. (2015). Software regression test case prioritization for object-oriented programs using genetic algorithm with reduced-fitness severity. *Indian Journal of Science and Technology*, *8*(30), 1-9.

[39] Musa, S., Sultan, A. B. M., Abdul Ghani, A. A. B., & Baharom, S. (2014). Regression Test Case Selection & Prioritization Using Dependence Graph and Genetic Algorithm. *IOSR Journal of Computer Engineering*, *4*(3), 2278-661.