Indonesian Journal of Electrical Engineering and Computer Science
Vol. 16, No. 1, October 2019, pp. 429~434
ISSN: 2502-4752, DOI: 10.11591/ijeecs.v16.i1.pp429-434

A review of object-oriented approach for test case prioritization

Umar Farooq, Hannani Aman, Aida Mustapha, Zainuri Saringat
Faculty of Computer Science and Information Technology, Universiti Tun Hussein Onn Malaysia, Malaysia

Article Info

Article history:

Received Nov 1, 2018
Revised Feb 6, 2019
Accepted Mar 15, 2019

Keywords:

Object-Oriented
Regression Testing
Software Testing
Test Case Prioritization

ABSTRACT

The most essential phase in regression testing is Test Case Prioritization (TCP), with its primary objective to increase the fault detection rate at different stages during testing. Prior to achieving the objective, existing evidence of techniques in TCP must be synthesized and analyzed. At present, fault detection for TCP based on object-oriented features only consider statement, module, and class level. The important features of object-oriented (OO) programming like inheritance and polymorphism have not been fully explored for fault detection in TCP. Such OO concepts are important for test case selection and in turn for ranking the test cases (prioritization). This paper reviews various test case prioritization techniques specific to OO systems. This review is hoped to highlight the importance and usage of TCP in relation to object-oriented software development lifecycle.

Corresponding Author:

Hannani Aman,
Faculty of Computer Science and Information Technology,
Universiti Tun Hussein Onn Malaysia,
86400 Parit Raja, Batu Pahat, Johor, Malaysia
Email: hanani@uthm.edu.my

1. INTRODUCTION

Regression testing is a software engineering activity that continues well after software delivery. It is part of maintenance activity in a software development lifecycle that help to ensure any modification resulting from debugging or improvement activities do not affect the existing functionalities and the initial requirement of the design [1]. In practice, regression testing takes almost 80% of the overall testing budget and up to 50% of the cost of software maintenance. Software maintenance activity itself is an expensive phase which account for nearly 60% of the total cost of the software production [2]. Figure 1 shows the types of regression testing.

Regression test selection is an activity within regression testing whereby the testers select test cases from an existing test suite that need to be rerun. This is to ensure that all the modified parts still behave according to the original intention and no modification introduce any sudden faults [3]. The techniques rely on the available surrogates for prioritization criteria, because the position and nature of the faults are not generally known in advance [4]. Prioritization of test cases to be used in the testing will result in reduction of total cost associated with the testing activities. In prioritizing the test cases, the test cases are ordered with the aim of detecting faults early in the test execution cycle. Nonetheless, due to multiple modifications and versioning of the system, the pool of test cases become larger in size and cannot be all executed within limited time [5].

Prioritization of test cases is the main challenge in regression testing because it requires a careful balance between maximizing the cover- age of fault detection while at the same time reducing the execution time. In the literature, this is carried out using various techniques such as slicing-based, inheritance-based, and machine learning-based i.e. Genetic Algorithms and Ant Colony. Nonetheless, beyond test case identification and selection, prioritization requires ranking of the test cases as well as dynamic mechanism to

balance between the testing coverage and time. The objective of TCP techniques is to order test cases based on the rate of code coverage or error discovery. Most of the TCP techniques previously proposed were established in the environment of technical plans. In addition, existing TCP practices do not have any effort for object-oriented programs, hence do not reflect dependencies that arise due to object-relations [6]. The major concern in regression testing is how to prioritize or rank all the test cases.

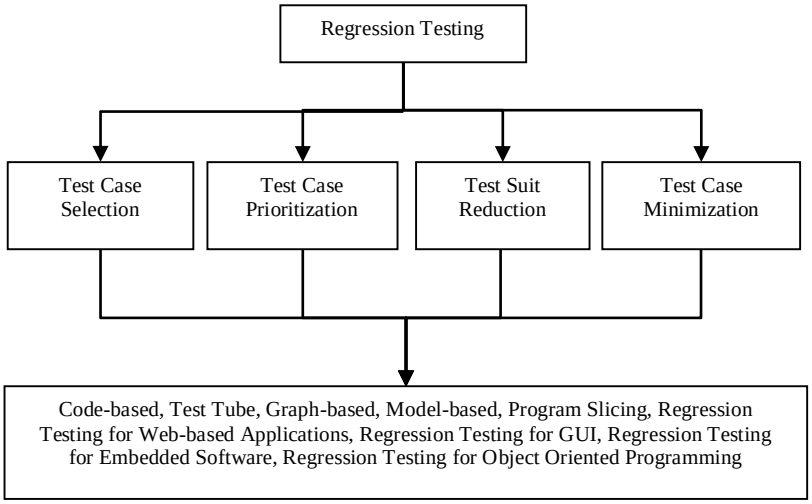

Figure 1. Types of regression testing

This paper reviews the object-oriented approaches, where it is hoped that best prioritization of regression test cases could be selected. The prioritization is important to reduce the execution time during software testing and to maximize the coverage in detecting faults. The remainder of this paper proceeds as follows. Section 2 reviews the works related to test case prioritization in regression testing. Section 3 discusses the analysis of literature and finally Section 4 concludes with some indication for future work.

2. RESEARCH METHOD

Test Case Prioritization (TCP) is one of the most essential phase in regression testing. This review is important to analyze the potential correlation between TCP within the framework of Object-Oriented Programming (OOP). Figure 2 shows some techniques of TCP.

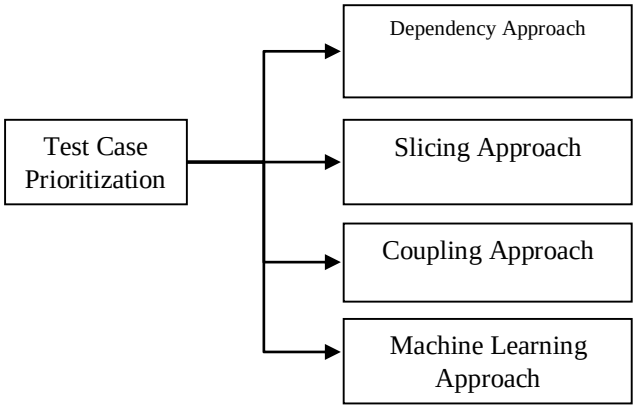

Figure 2. Techniques for TCP

2.1. Dependency approach

In data dependency approach, [7] proposed a novel prioritization algorithm for regression test suite that maximize the percentage of faults dependency detection. As regression test is dependent or exe-cuted for only dependent modules or code, the APFDD metric will measure how fast a prioritized test case can find faults and depen- dency. The work also introduced a new metric in order to measure the effectiveness of prioritization algorithm called the Average of the Percentage Fault Dependency Detected (APFDD). The presented test suite has detection ratio of 83.3% APFDD when compared against the non-prioritized test suite, which was 43.3%. This fault detec- tion metric has been proven to exhibit the performance of a TCP technique when related to random test case prioritization method. Nonetheless, other parameters like function dependency, variable dependency, and inheritance have not been considered.

C. R. Panigrahi and R. Mall, [8] proposed an improved dependency model called the Extended System Dependence Graph (ESDG) that is designed to prioritize the test cases. ESDG is especially useful in object-oriented program so as to de- termine which nodes are being affected. The main assumption in the ESDG approach is that all the test cases are consuming the same costs and all the errors share same rigorousness. Once the test cases have been represented by the ESDG, the modified nodes which are being affected will be removed [9] ranked the test cases based on the degree of error discovery and influence of error. The proposed algorithm classified the former error during the testing procedure, measured the efficiency of ranked test case, and assessed the non-prioritized test cases based on APFD. APFD for prioritized test case was 0.70 while for non-prioritized test case was 0.63. This showed that the average proportion of error discovered has been improved by the set of prioritized test cases related to arbitrary collection.

In a more recent work, [10] used the dependency graph model along with the Genetic Algorithm to identify the failed test cases based on the dependency graph. The methodology selected a set of test cases from the test suite and used the selected test cases in testing the modified program. The test cases are then prioritized in the order of their detecting faults. This method identified many essential issues like test selection, test suite maintenance, change identification and test execution that regression testing approaches need to address. Subsequent work by [3] extended the dependence graph approach to include identifying the changes at the statement level within the source codes. The required changes were stored in a separate file, and the information on coverage for each test case from the source code was generated with the help of Genetic Algorithm. Evaluation using the APFD metric produced efficiency of only 20% from the test cases able to detect all the faults. This method claimed to reduce the cost of regression testing by efficient selection and prioritization of test cases.

S. Musa, et al. [11] suggested a structure for regression testing for object-oriented software constructed on prolonged system necessity graph model of the program that is being affected. Semantic examination of the code provides a base for this method. The main target was to find the changes in the body method due to control dependence, data depen- dence and dependency due to object relations. The work highlighted significant matters that regression testing approaches required to cater; which are transformation identification, test collection, test implementation, and change identification all of which are still at method level within class only.

2.2. Slicing approach

Program slicing is a technique for simplifying programs by focusing on selected aspects of the program semantics. In TCP, [8] proposed a an improved slice-based approach called the Slice-based Regression Test Prioritization (S-RTP), which ordered the test cases based on the number of model elements that are affected. The influenced portions of the program were those associated with objects and data control dependencies. This method is promising because it highlighted the possibility to explore other object-oriented programming structures like threads, exclusions, boundaries and class libraries. Nonetheless, the research claimed to only achieve an average outcome, which is 25.70% in- crease in the APFD value when compared against other techniques. [12] also focused code slices at the class level only, when reducing the prioritization problem to an ordinary quer-based analysis.

2.3. Coupling approach

H. Kumar and N. Chauhan [13] proposed a novel method for TCP called the component cou-pling approach. This approach depends on linked information among the components of the program and is capable of finding the critical module with the possibility to have serious bugs when compared against other approaches. The experiments used a sample program which comprises of 10 modules and testing was carried out based on the coupling information [14] also used the coupling approach to enhance the efficiency of TCP in regression testing. The analysis showed that the efficiency and effectiveness of prioritized test cases able to expose the errors timely during the regression test cycle. Pri- oritization of test cases was performed on the

source of fault prone or affected nodes' coverage. In the test suite, the test cases having high weight are given high priority.

Along the line with the coupling approach, [15] proposed the use of inheritance in detecting fault coverage. This means prioritization of the classes is carried out based on the inherited qualities or traits, number of descendants that a class has, and level of class within the pyramid of inheritance. This means classes having higher level of likelihood of fault dissemination (higher degree of error propa- gation) in the inheritance pyramid will be given first priority. In other words, test cases having high rate of fault identification or detection are prioritized at first place. The research showed that the inheritance hierarchy is able to to save assets like cost and time when the experiment achieved 85% APFD as compared to random test case prioritization methods.

2.4. Machine learning approach

For requirements prioritization based on machine learning approach, [16] proposed the use of Genetic Algorithm for enhancing the num- ber of error discovery among severe faults. The suggested technique covered different aspects of prioritizing the test cases such as the data flow, fault proneness, length of test cases as well as code coverage. Other aspect includes severity of faults and perceived code complexity. Prioritization of the system test cases is based on six factors, which include fault impact, implementation complexity, completeness, priority of the clients, variations in requirement, and traceability. The efficiency of the proposed prioritization procedure was assessed using the Average Percentage of Fault Detection (APFD) metric and was found efficient when prioritizing the test cases. Subsequent work by [17] also demonstrated that test cases produced by GA produced sound result with regards to code coverage and time.

S. Musa, et al. [11] suggested the use of Genetic Algorithm to optimize the selected test cases built from a dependency graph model. The proposed approach produced better results in terms of average proportional rate of fault exposure. On the basis of measured performance gained from the outcomes, GA with decreased strictness of error was able to prioritize chosen test cases with more efficiency as compared to the use of GA with same level of severity of fault and non-prioritized test cases. GA with decreased severity of errors also gives higher priority to designated cases more efficiently when associated to the usage of GA with the similar level of severity of error [18]. In the long run, this success is able to decrease the overall regression esting cost.

Another machine learning algorithm proposed for prioritizing the test cases is the Ant algorithm [19, 20]. This method identified the bugs during the initial phase of execution therefore capable in terminating early for regression testing. This is necessary for system devel- opment with constrained with resources. The results showed that ordering of the test cases achieved 100% APFD with 11.66 minutes consumption for the rquired stock program.

3. EVALUATION METRICS

The standard evaluation metrics for Test Case Prioritization (TCP) is called the Average Percentage of Faults Detected (APFD) metric developed by [21]. APFD measures the rate of fault detection of test suite execution as shown in Equation 1, where n is the number of test cases, m is the number of faults, $T_{f1} + T_{f2} + \cdots + T_{fm}$ represents the position of first test T that exposes the fault.

$$APFD = 1 - \frac{T_{f1}+T_{f2}+\cdots+T_{fm}}{mn} + \frac{1}{2n} \quad (1)$$

Equation 1 shows that APFD is calculated by taking the weighted average of the percentage of faults detected during the execution of the test suite. The values range from 0 to 100, whereby the higher the APFD value, the better fault detection rate that a system has. Other metrics used for TCP includes [22, 23]:

- Average Percentage Block Coverage (APBC)
- Average Percentage Decision Coverage (APDC)
- Average Percentage Fault Dependency Detected (APFDD)
- Average Percentage Statement Coverage (APSC)
- Average Percentage Loop Coverage (APLC)
- Average Percentage Condition Coverage (APCC)
- Problem Tracking Reports (PTR)

4. CONCLUSIONS

Test Case Prioritization (TCP) in regression testing capitalizes on different features of object-oriented approach such as data dependence, control dependence, and dependency due to object relations. Various techniques have been reviewed and discussed in accordance to the literature. Some notable limitation in existing TCP techniques include lack of fault detection at class level as well as under-utilization of object-oriented features such as relationships from class inheritance. At present, fault detection for TCP based on object-oriented features only consider statement, module, and class level. The important features of object-oriented programming like inheritance and polymorphism have not been fully explored for fault detection in TCP. Such OO concepts are important for test case selection and in turn for ranking the test cases (prioritization). In the future, inheritance-based TCP will be further investigated in effort to reduce execution time and maximize the coverage of fault detection during software testing among other state-to-the-art techniques such as based on error probability and severity of UML models [24] as well as reliability risk analysis [25].

ACKNOWLEDGEMENTS.

This work is supported by Universiti Tun Hussein Onn Malaysia under the Tier 1 Grant Scheme U894.

REFERENCES

[1] G. Rothermel and M. J. Harrold, "Selecting regression tests for objectoriented software." in *ICSM*, vol. 94, pp. 14–25. 1994.
[2] R. S. Pressman, Software engineering: a practitioner's approach. Palgrave Macmillan, 2005.
[3] S. Musa, A. B. M. Sultan, A. A. Ghani, and S. Bahrom, "Regression test framework based on extended system dependence graph for object-oriented programs," *International Journal of Advances in Software Engineering & Research Methodology–IJSERM*, vol. 1, no. 2, 2014.
[4] D. Jeffrey and N. Gupta, "Improving fault detection capability by selectively retaining test cases during test suite reduction," *IEEE Transactions on software Engineering*, vol. 33, no. 2, 2007.
[5] J.-M. Kim and A. Porter, "A history-based test prioritization technique for regression testing in resource constrained environments," in *Proceedings of the 24th international conference on software engineering*. ACM, pp. 119–129. 2002.
[6] C. R. Panigrahi and R. Mall, "An approach to prioritize the regression test cases of object-oriented programs," *CSI transactions on ICT*, vol. 1, no. 2, pp. 159–173, 2013.
[7] M. I. Kayes, "Test case prioritization for regression testing based on fault dependency," in *Electronics Computer Technology (ICECT), 2011 3rd International Conference* on, vol. 5. IEEE, 2011, pp. 48–52.
[8] C. R. Panigrahi and R. Mall, "A heuristic-based regression test case prioritization approach for object-oriented programs," *Innovations in Systems and Software Engineering*, vol. 10, no. 3, pp. 155–163, 2014.
[9] H. Raperia and S. Srivastava, "An empirical approach for test case prioritization," *International Journal of Scientific & Engineering Research*, vol. 4, no. 3, pp. 1–5, 2013.
[10] A. B. M. Sultan, A. A. A. Ghani, S. Baharom, and S. Musa, "An evolutionary regression test case prioritization based on dependence graph and genetic algorithm for object-oriented programs," in *2nd International conference on emerging trends in engineering and technology*, pp. 22–26. 2014.
[11] S. Musa, A.-B. M. Sultan, A.-A. B. Abd-Ghani, and S. Baharom, "Software regression test case prioritization for object-oriented programs using genetic algorithm with reduced-fitness severity," *Indian Journal of Science and Technology*, vol. 8, no. 30, 2015.
[12] R. K. Saha, L. Zhang, S. Khurshid, and D. E. Perry, "An information retrieval approach for regression test prioritization based on program changes," in *Proceedings of the 37th International Conference on Software Engineering-Volume 1. IEEE Press*, pp. 268–279. 2015.
[13] H. Kumar and N. Chauhan, "A coupling effect based test case prioritization technique," in *Computing for Sustainable Global Development (INDIACom), 2nd International Conference on. IEEE*, 2015, pp. 1341–1345. 2015.
[14] S. Panda, D. Munjal, and D. P. Mohapatra, "A slice-based change impact analysis for regression test case prioritization of object-oriented programs," Advances in Software Engineering, vol. 2016, p. 1, 2016.
[15] N. Chauhan, H. Kumar et al., "A hierarchical test case prioritization technique for object oriented software," in Contemporary Computing and Informatics (IC3I), *International Conference on. IEEE*, 2014, pp. 249–254. 2014.
[16] S. Raju and G. Uma, "Factors oriented test case prioritization technique in regression testing using genetic algorithm," *European Journal of Scientific Research*, vol. 74, no. 3, pp. 389–402, 2012.
[17] P. K. Mishra et al., "Analysis of test case prioritization in regression testing using genetic algorithm," *International Journal of Computer Applications*, vol. 75, no. 8, 2013.
[18] S. Musa, A. B. M. Sultan, A. A. B. Abd-Ghani, and S. Baharom, "Regression test cases prioritization for object-oriented programs using genetic algorithm with reduced value of fault severity," *International Journal of Soft Computing*, vol. 11, no. 4, pp. 247–254, 2016.

[19] D. Azar and J. Vybihal, "An ant colony optimization algorithm to improve software quality prediction models: Case of class stability," *Information and Software Technology* 53, no. 4 : 388-393. 2011.
[20] M. S. Kumar and P. Srinivas, "An ant colony algorithm to prioritize the regression test cases of object-oriented programs," *Indian Journal of Science and Technology*, vol. 9, no. 19, 2016.
[21] S. Elbaum, A. G. Malishevsky, and G. Rothermel, Prioritizing test cases for regression testing. ACM, vol. 25, no. 5. 2000.
[22] R. Pradeepa and K. Vimaldevi, "Effectiveness of testcase prioritization using apfd metric: survey," in *IJCA Proceedings on international conference on research trends in computer technologies*, pp. 1–4. 2013.
[23] I. Sharma, J. Kaur and M. Sahni, "A test case prioritization approach in regression testing," *International Journal of Computer Science and Mobile Computing* 2, pp. 607-614. 2014.
[24] T. Zhang, X. Wang, D. Wei and J. Fang, "Test Case Prioritization Technique Based on Error Probability and Severity of UML Models," *International Journal of Software Engineering and Knowledge Engineering* 28, no. 06, pp. 831-844. 2018.
[25] Y. Wang, Z. Zhu, B. Yang, F. Guo and H. Yu, "Using reliability risk analysis to prioritize test cases," *Journal of Systems and Software* 139, pp. 14-31. 2018.