Factor Determination in Prioritizing Test Cases for Event Sequences: A Systematic Literature Review

Johanna Ahmad and Salmi Baharom
Software Engineering and Information System Department, FSKTM, UPM, Malaysia.
gs43485@student.upm.edu.my

***Abstract*—The generation of test cases is a challenging phase in software testing. The process of test case generation becomes more expensive and time-consuming when the test suites become larger. Many researchers have proposed the test case prioritization (TCP) technique to schedule test cases, so that those with the highest priority are executed first before lower priority test cases. One of the performance goals of TCP is the rate of fault detection, which is a measure of how quickly faults are detected within the testing phase. However, the existing TCP technique has some limitations. This paper presents the results of a systematic literature review (SLR) of relevant primary studies as evidence of the existence of TCP in the area of event sequences. Consequently, five major techniques and 10 factors were identified and analysed. This study aims to review and identify techniques and factors that influence the process of assigning weight values in TCP processes. The proposed factors need to be evaluated in terms of their contribution to the performance of the TCP technique. Some researchers believe that a combination of factors might be required to produce unique weights during the TCP processes. Nevertheless, most studies applied the random method or did not provide any information regarding the same weight value issues.**

***Index Terms*—Unique Weight; Test Case Prioritization; Systematic Review.**

I. Introduction

In the software development phase, testing software for large systems is often expensive and time-consuming. Hence, the importance of testing grows as the size and complexity of the system increases. Whenever the time for testing increases, the costs will rapidly increase. In recent years, numerous TCP techniques have been proposed and applied. This study is part of the on-going research towards enhancing the existing TCP technique for event sequences. The flexibility of event sequence application enables countless usage scenarios and a combination of interactions [1]. This characteristic makes the application of event sequences even more complex compared to traditional applications due to the possibility of the former having infinite input domain. Within a defined timeframe, it is not practical, and impossible to test every possible input.

TCP has been proven to be beneficial in testing activities [2]. In recent years, numerous researches have proposed methods that combined multiple factors and applied the assigning-weight value approach in their TCP techniques. One of the challenges in TCP is to prioritize test cases that may have the same priority value during these TCP processes. Based on the literature review, most researchers would apply the random technique to break the ties. The random technique is a fundamental testing method in which the test cases will be selected randomly from the test suites [3]. Although the random technique is popular, its effectiveness has been argued by many since it creates bias issues [3], [4]. Based on that reason, researchers have concluded that there is a need for a unique weight approach to solve the same priority value issues. Therefore, this SLR paper aims to represent the techniques and factors that can influence the process to produce unique weight in TCP. This paper is structured as follows; Section II will present details of the systematic review process. Section III will discuss the extraction of information to answer the research questions. Section IV will present discussions of the results. The conclusion will be expressed in Section V.

II. Review Method

The review processes for this SLR used the guidelines proposed by [5], [6]. According to [5], three main phases are involved in this SLR; planning the review, conducting the review, and reporting the review, as shown in Figure 1.

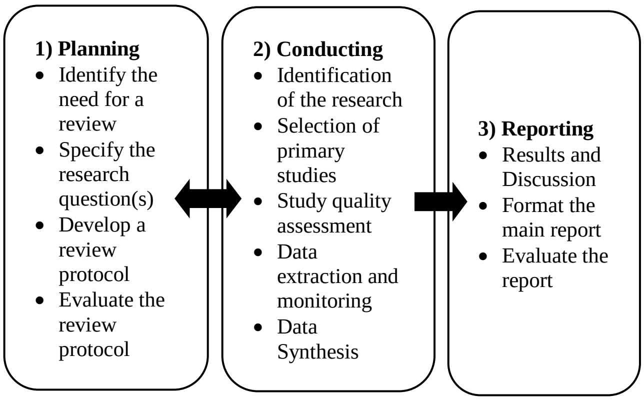

Figure 1: SLR phases and stages in this study

A. Research Questions

Over the years, different methods, approaches, and techniques have been proposed to reduce the effort, time, and cost taken during testing. This SLR seeks to understand and summarise the existing evidence on TCP techniques. Furthermore, this review endeavours to identify the techniques and factors that affect the effectiveness of the existing TCP techniques. According to [6], five components can be used to formulate research questions for the SLR, which are known as the PICOC. Table 1 shows the criteria and scope of such research questions.

To achieve the aim of this study, the two research questions are:

RQ1 : What are the existing techniques used to prioritize test case?

RQ2 : What are the factors that can affect the effectiveness of TCP technique?

Table 1
Criteria and Scope of Research Questions

Criteria	Scope
Population	Sequence Based, Event Based, Search Based, State Based
Intervention	Test case prioritization technique
Comparison	NA
Outcomes	Techniques and factors of TCP technique applied in Sequence-Based, Event Based, Search Based, State-Based
Context	Review(s) of any empirical studies of the test case prioritization

B. *Data Sources*

Ten electronic databases were used to primarily extract data, namely, the ACM Digital Library, Emerald Insight, Elsevier, Google Scholar, IEEE Xplore Digital Library, ScienceDirect, Scopus, SpringerLink, Taylor & Francis Group, and Wiley. These selections were based on the online databases subscribed by the University Putra Malaysia's Library under the Computer Science subject category.

C. *Search Strategy*

The initial search strings were software, test case prioritization, sequence based, search based, event based, and state based. Trial searches with a combination of terms were derived from the research questions. The proceeding search string was then constructed using the Boolean "and", and Boolean "or" operators for alternatives synonyms, and world-class variants of each keyword. The following search keywords were used to find relevant studies based on the title, abstract, and metadata:

("Software" AND "Test") OR ("Test Case Prioritization") OR ("Test Case Prioritization" AND "Sequence-Based") OR ("Test Case Prioritization" AND "Search Based") OR ("Test Case Prioritization" AND "Event Based") OR ("Test Case Prioritization" AND “State-Based")

D. *Study Selection*

Study selection is evidence of the research question. During the first search stage, 2,314 prospective studies were selected. The next stage was the process of eliminating duplicates, and irrelevant studies. After screening the titles and abstracts, only 135 were potentially relevant [5]. These were then subjected to the inclusion and exclusion criteria. Once the 135 primary studies have been selected, the quality of the selected primary studies were evaluated using quality assessment questions, which were proposed by [5]. The quality assessment questions are shown in Table 2.

E. *Inclusion and Exclusion Criteria*

The inclusion and exclusion criteria for this SLR were based on the research questions [5]. The inclusion criteria for this SLR are as follows:

- All papers must be published in English
- All papers must be published from 1 January 2005 to 18 December 2016
- All papers must focus on test case generation and test case prioritization

Next, each paper was filtered using the exclusion criteria before being accepted for the next stage. The exclusion criteria are as follows:

- Papers that are not published in English
- Duplicated study areas
- Papers that only contain opinion pieces, viewpoints, progress research or incomplete results
- Exclude thesis
- Exclude papers with less than three pages
- Papers that do not report any empirical study

F. *Data Extraction and Quality Assessments*

Quality assessment (QA) for this study was achieved by weighting or scoring to obtain relevant studies that would be capable of addressing each research question. Most researchers agree that the quality assessment study checklist can be used to ensure that the data extraction process meets the quality criteria [7]. Some researchers stated that quality assessment can be used to evaluate the completeness and relevance of the selected studies. Table 2 lists the general questions to measure the quality of selected studies. Three scales are coded for the quality assessment checklist, and given scores; Yes =1; Partially = 0.5; No = 0. Based on the item checklist, each article was measured from 0 (very poor) to 4 (very good).

Table 2
Quality Assessment Checklist

No	Item	Answer
SQ1	Were the aim and objective clearly stated?	Yes/No
SQ2	Was the research design clearly specified?	Yes/No/Partially
SQ3	Did the researcher(s) carry out the process of data collection well?	Yes/No/Partially
SQ4	Do the researcher(s) discuss the work limitations clearly?	Yes/No/Partially
SQ5	Did the researcher(s) state enough data to support their proposed factors?	Yes/No/Partially

III. FINDING

After the titles and abstracts have been screened, only 135 papers were potentially relevant. At this stage, irrelevant studies and duplicate studies were eliminated. Then, each full paper was read whenever the title and abstract were insufficient to categorize whether the paper was relevant or not. Finally, 70 primary studies were selected for providing answers to the formulated research questions. Figure 2 depicts the results of the paper search and selection process.

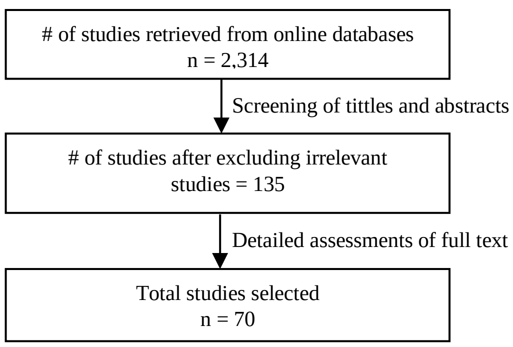

Figure 2: Paper search and selection stage for this SLR

A. Quality of Factors

Table 3 indicates the quality assessment scores for the final identified papers. Six studies (9%) were rated fair, nine studies (13%) were good, and 55 studies (78%) were of very good quality. None of these papers were rated as being of poor quality. As such, all selected papers were included in the next phase for further analysis.

Table 3
Quality Assessment Scores

Quality Scale	Very Poor (>=1)	Poor (>=2)	Fair (>=3)	Good (>=4)	Very Good (=5)	Answer
Number of studies	0	0	6	9	55	70
Percentage (%)	0	0	9	13	78	100

IV. DISCUSSION

This section presents and discusses the results related to the research questions. A detailed description of the findings will be presented with the aim of investigating the major utilised techniques and factors that can affect the performance of the TCP techniques.

A. What Are the Existing Techniques Used to Prioritize Test Case?

Based on this SLR, numerous techniques have been adapted and applied in prioritising test cases. 26% of the selected papers have combinations of more than one technique, as proposed by [8], [9], and [10]. They believe that adopting multiple criteria can maximise the number of discovered faults, thus, improving the effectiveness and efficiency of the proposed technique [11]. In fact, some researchers agree that the multiple criteria could break ties if they are present during the TCP processes. As previously mentioned, a majority of the papers reported the application of the random technique to solve the same priority value issues. Some researchers believe that if one criterion is not performing as expected, the remaining criteria can make up for it to provide the expected result. Table 4 represents the identified techniques used to prioritize a test case.

Table 4
Identified Techniques Used to Prioritize Test Case

No.	Techniques	Authors
1	Code Coverage	[8], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], [33]
2	Requirement Coverage	[8], [17], [24], [26], [34], [35], [36], [37], [38]
3	Execution Time	[9], [25], [26], [27], [28], [39], [40]
4	Fault Coverage	[24], [25], [28], [31], [41], [42]
5	Historical Data	[36], [42], [43], [44], [45]

Code coverage is the most utilised technique to prioritize test cases at 40% of these papers. The second is the requirement coverage at 17%. This is followed by execution time at 13%, fault coverage at 10%, and historical data at 8%. The remaining 2% is for other techniques, such as event coverage, interaction coverage, and state-based behaviour. The following researchers applied for code coverage:[8], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31], [32], and [33]. Higher code coverage can be a good indicator of fault detection capability [46]. There are a number of coverage criteria for code coverage, such as function coverage, statement coverage, branch coverage, and condition coverage. Meanwhile, a combination of branch coverage and function coverage is called the decision coverage. Normally, the decision coverage is applied for safety critical applications, whereby each condition in the program could affect the decision outcome independently. The code coverage is also widely used in the industry. Code coverage becomes one of the requirements in the automotive safety standard, ISO 26262, Road Vehicles-Functional Safety [75].

In terms of the requirement coverage, researchers applied it to maximize user satisfaction [8]. Test cases are mapped with the given requirements, and the requirement coverage will ignore the actual behaviour and the structure of the application. According to [36], requirements complexity and requirements volatility are some of the weight factors proposed by previous researchers to prioritize test cases based on the requirement coverage technique. A recent research had shown that the implementation of requirements complexity and requirements volatility can significantly affect the rate of fault detection in test suites [36].

B. What Are the Factors That Can Affect the Effectiveness of TCP Technique?

As shown in Table 5, 10 factors were identified based on the data extracted from 70 primary studies. All identified factors were found to have affected the effectiveness and efficiency of the TCP technique. Three factors were the most addressed by the primary studies, which include fault matrix in 46 papers, redundancy in 20 papers, and complexity in 18 papers. 57% of the papers applied more than three factors in their TCP technique to achieve more than one competing objective. This shows the interrelation between the identified factors. It also shows the importance of using more than one factor to increase the performance of the TCP technique. In addition, 14% of these papers addressed only one factor. Most of these papers also applied execution time as a factor to prioritize test cases. It was stated that this technique is expected to cover all the statements with a minimal execution time [27].

Table 5
Factors That Affect the Effectiveness of TCP Technique

No.	Factors	Authors
1	Fault	[8], [9], [14], [15], [17], [18], [19], [20], [21], [23], [24], [25], [26], [28], [30], [33], [35], [37], [38], [40], [41], [43], [44], [45], [47], [48], [49], [50], [51], [52], [53], [54], [55], [56], [57], [58], [59], [60], [61], [62], [63]
2	Redundancy	[9], [11], [15], [21], [22], [23], [24], [28], [33], [35], [36], [37], [44], [50], [53], [54], [55], [57], [62], [64]
3	Complexity	[14], [18], [26], [28], [30], [33], [35], [37], [43], [47], [49], [50], [51], [52], [53], [54], [62]
4	Frequency	[14], [16], [21], [28], [34], [43], [45], [47], [50], [51], [54], [56], [64], [65], [66], [67]
5	Requirements	[8], [24], [26], [33], [34], [35], [37], [39], [41], [55], [59], [65], [68], [69]
6	Time	[9], [12], [15], [18], [25], [26], [34], [39], [48], [49], [62], [65]
7	Distance	[11], [23], [32], [42], [45], [49], [53], [54], [68], [70]
8	Cost	[8], [17], [34], [36], [44], [45], [60], [65], [71]

No.	Factors	Authors
9	Permutation	[11], [42], [44], [49], [51]
10	Others	[1], [13], [16], [19], [20], [22], [23], [24], [27], [28], [29], [31], [32], [33], [35], [38], [40], [41], [43], [44], [45], [48], [50], [55], [59], [63], [64], [65], [66], [67], [69], [72], [73]

Most of these papers emphasized that fault matrix plays an important in selecting potential factors for the TCP technique. Fault matrix represents the minimal set that covers all faults [9]. The weight of a test case is given based on the ratio of the fault coverage value. Furthermore, the execution time will be reduced with early fault detection, which can affect the effectiveness and efficiency of the TCP technique. Based on the literature, redundancy becomes the second popular factor because of the high possibility for a large test suite to have redundancies. Minimization is one of the techniques to remove redundancy in a test suite. Previous experiments have shown that the implementation of redundancy in TCP technique can save resources and time [14].

The complexity of a system can be considered as a subjective measure. Based on Table 5, 17 papers addressed complexity as one of the factors that can influence the performance of their TCP technique. Some researchers stated in their respective papers that by reducing the test suite, and the program size, the value of complexity for the system can be decreased [14]. High complexity value shows that the system is more complex. Furthermore, the complexity can also be a measure for the case of requirement changes. The complexity is calculated based on the number of times the requirement changes. Numerous complexity metrics are available for measuring complexity, such as McCabe, Lines of Codes, and unique complexity metric. According to [35], requirements with complex functionality can introduce a higher number of faults. Thus, it was concluded that the complexity factor can influence the TCP processes.

The data presented in Table 5 shows that only five papers have considered permutation as one of the factors that can influence the weight of the priority value in TCP. However, based on the literature, previous researchers believed that permutation is actually one of the important factors that help to generate an optimum number of test cases. Furthermore, permutation can also remove redundancies. Due to resource and time constraint, it would be impractical to execute all test cases as some of these test suites can grow very large, especially in event sequences applications [74]. Thus, permutation is needed to select a subset of possible combinations of events.

There were 19% of the selected papers that combined fault matrix, redundancy, frequency, and complexity factors. Based on Table 5, all four factors are in the top rank. 6% papers combined fault matrix and time factors. A similar condition was found with the combination of fault matrix and redundancy factors. On the other hand, 13% of these papers combined fault matrix with other factors, such as dependence structure, relationships among test cases, and the execution of information of the modified program. Thus, it was concluded that there is a need to combine all four factors that belong in the top rank to obtain a high performance TCP technique. However, some limitations can still be found with the existing TCP technique, which may still require enhanced effectiveness and efficiency. Therefore, changes in the existing combinations of factors may be needed to fill the gap in that research area.

V. CONCLUSION

This paper has presented and discussed the results obtained from 70 primary studies. This study is a part of the research to propose a unique weight approach in TCP technique for event sequences. Therefore, the aim of this SLR paper was to investigate and identify the factors used to develop an effective TCP technique for event sequences. Collecting and identifying the most utilized factors in TCP techniques were useful for the potential improvement of the overall research. The SLR results have 10 factors that should be considered to enhance the existing TCP technique. Moreover, code coverage is widely used in TCP technique, to detect faults as early as possible. Thus, code coverage should be taken under considerations for future researches. However, in order to maximize the number of detected faults, there is a possibility of combining code coverage with other techniques, such as requirement coverage, which was done by [8]. It measured the amount of requirements that can be covered by the test case. Overall, the results confirmed that the 10 identified factors played significant roles in the performance of the TCP technique.

ACKNOWLEDGMENT

The authors would like to acknowledge the Ministry of Higher Education Malaysia (MOHE) for the financial support under the Fundamental Research Grant Scheme (FRGS);Project code-08-01-15-1723FR.

REFERENCES

[1] C. Klammer, R. Ramler, and H. Stummer, “Harnessing automated test case generators for GUI testing in industry,” *2016 42th Euromicro Conf. Softw. Eng. Adv. Appl.*, pp. 227–234, 2016.

[2] R. Krishnamoorthi and S. a. Sahaaya Arul Mary, “Factor oriented requirement coverage based system test case prioritization of new and regression test cases,” *Inf. Softw. Technol.*, vol. 51, no. 4, pp. 799–808, 2009.

[3] T. Y. Chen, F.-C. Kuo, H. Liu, and W. E. Wong, “Code coverage of adaptive random testing,” *IEEE Trans. Reliab.*, vol. 62, no. 1, pp. 226–237, 2013.

[4] T. Menzies and B. Cukic, “Focus,” *Ieee Softw.*, pp. 107–112, 2000.

[5] B. Kitchenham and S. Charters, “Guidelines for performing systematic literature reviews in software engineering,” *Engineering*, vol. 2, p. 1051, 2007.

[6] M. Petticrew and H. Roberts, *Systematic Reviews in the Social Sciences: A Practical Guide*. 2006.

[7] N. H. Hassan, Z. Ismail, and N. Maarop, “Information security culture : a systematic,” no. 205, pp. 456–463, 2015.

[8] M. M. Islam, A. Marchetto, A. Susi, and G. Scanniello, “A multi-objective technique to prioritize test cases based on latent semantic indexing,” *2012 16th Eur. Conf. Softw. Maint. Reengineering*, no. 3, pp. 21–30, 2012.

[9] M. Tyagi and S. Malhotra, “Test case prioritization using multi objective particle swarm optimizer,” *2014 Int. Conf. Signal Propag. Comput. Technol. (ICSPCT 2014)*, pp. 390–395, 2014.

[10] H. Srikanth, M. Cashman, and M. B. Cohen, “Test case prioritization of build acceptance tests for an enterprise cloud application: An industrial case study,” *J. Syst. Softw.*, vol. 119, pp. 122–135, 2016.

[11] X. Zhang, X. Xie, and T. Y. Chen, “Test case prioritization using adaptive random sequence with category-partition-based distance,” *2016 IEEE Int. Conf. Softw. Qual. Reliab. Secur.*, pp. 374–385, 2016.

[12] H. Srikanth and M. B. Cohen, “Regression testing in software as a service: An industrial case study,” *2011 27th IEEE Int. Conf. Softw. Maint.*, pp. 372–381, 2011.

[13] R. Blanco, J. García-Fanjul, and J. Tuya, “A first approach to test case generation for BPEL compositions of web services using scatter search,” *2009 Int. Conf. Softw. Testing, Verif. Valid. Work.*, pp. 131–140, 2009.

[14] Z. Li, M. Harman, and R. M. Hierons, “Search algorithms for regression test case prioritization,” *IEEE Trans. Softw. Eng.*, vol. 33,

no. 4, pp. 225–237, 2007.
[15] P. K. Mishra, “Analysis of test case prioritization in regression testing using genetic algorithm,” vol. 75, no. 8, pp. 1–10, 2013.
[16] P. R. Srivastava, A. Vijay, B. Bariikha, P. S. Senear, and R. Sharma, “An optimized technique for test case generation and prioritization using ‘tabu’ search and data clustering,” *Proc. 4th Indian Int. Conf. Artif. Intell. IICAI 2009*, pp. 30–46, 2009.
[17] D. D. N. B, A. Panichella, A. Zaidman, and A. De Lucia, “Search-Based Software Engineering,” vol. 9275, pp. 157–172, 2015.
[18] M. N. Nawar and M. M. Ragheb, “Multi-heuristic based algorithm for test case prioritization,” pp. 449–460, 2014.
[19] V. Panthi and D. P. Mohapatra, “Generating and evaluating effectiveness of test sequences using state machine,” *Int. J. Syst. Assur. Eng. Manag.*, no. Jorgensen 2008, 2016.
[20] C. R. Panigrahi and R. Mall, “Regression test size reduction using improved precision slices,” *Innov. Syst. Softw. Eng.*, vol. 12, no. 2, pp. 153–159, 2015.
[21] B. Miranda and A. Bertolino, “Scope-aided test prioritization, selection and minimization for software reuse,” *J. Syst. Softw.*, vol. 0, pp. 1–22, 2016.
[22] S. U. R. Khan, S. P. Lee, R. W. Ahmad, A. Akhunzada, and V. Chang, “A survey on Test Suite Reduction frameworks and tools,” *Int. J. Inf. Manage.*, vol. 36, no. 6, pp. 963–975, 2016.
[23] B. S. Ahmed, “Test case minimization approach using fault detection and combinatorial optimization techniques for configuration-aware structural testing,” *Eng. Sci. Technol. an Int. J.*, vol. 19, no. 2, pp. 737–753, 2015.
[24] H. Srikanth, C. Hettiarachchi, and H. Do, “Requirements based test prioritization using risk factors: An industrial study,” *Inf. Softw. Technol.*, vol. 69, pp. 71–83, 2016.
[25] P. Parashar, A. Kalia, and R. Bhatia, “Pair-wise time-aware test case prioritization,” pp. 176–186, 2012.
[26] N. Prakash and K. Gomathi, “Improving Test Efficiency through Multiple Criteria Coverage Based Test Case Prioritization,” *Int. J. Sci. Eng. Res.*, vol. 5, no. 4, pp. 430–435, 2014.
[27] Y. B. B, S. Kirbas, M. Harman, Y. Jia, and Z. Li, “Search-based software engineering,” vol. 9275, pp. 221–227, 2015.
[28] S. Chaudhury, A. Singhal, and O. P. Sangwan, “Event- driven software testing – an overview,” vol. 5, no. 4, 2016.
[29] Z. Li, Y. Bian, R. Zhao, and J. Cheng, “A fine-grained parallel multi-objective test case prioritization on GPU,” *Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics)*, vol. 8084 LNCS, pp. 111–125, 2013.
[30] F. Yuan, Y. Bian, Z. Li, and R. Zhao, “Search-based software engineering,” vol. 9275, pp. 109–124, 2015.
[31] M. G. Epitropakis, S. Yoo, M. Harman, and E. K. Burke, “Empirical evaluation of pareto efficient multi-objective regression test case prioritisation,” *Proc. 2015 Int. Symp. Softw. Test. Anal. - ISSTA 2015*, pp. 234–245, 2015.
[32] J. F. Silva Ouriques, E. G. Cartaxo, and P. D. Lima Machado, “Revealing influence of model structure and test case profile on the prioritization of test cases in the context of model-based testing,” *J. Softw. Eng. Res. Dev.*, vol. 3, no. 1, p. 1, 2015.
[33] C.-Y. Huang, C.-S. Chen, and C.-E. Lai, “Evaluation and analysis of incorporating Fuzzy Expert System approach into test suite reduction,” *Inf. Softw. Technol.*, vol. 79, pp. 79–105, 2016.
[34] L. S. De Souza, P. B. C. De Miranda, R. B. C. Prudencio, and F. D. a. Barros, “A multi-objective particle swarm optimization for test case selection based on functional requirements coverage and execution effort,” *2011 IEEE 23rd Int. Conf. Tools with Artif. Intell.*, pp. 245–252, 2011.
[35] C. Hettiarachchi, H. Do, and B. Choi, “Risk-based test case prioritization using a fuzzy expert system,” *Inf. Softw. Technol.*, vol. 69, pp. 1–15, 2016.
[36] S. Roongruangsuwan and J. Daengdej, “Test case prioritization techniques,” *J. Theor. Appl. Inf. Technol.*, pp. 45–60, 2010.
[37] R. Krishnamoorthi and S. a. Sahaaya Arul Mary, “Factor oriented requirement coverage based system test case prioritization of new and regression test cases,” *Inf. Softw. Technol.*, vol. 51, no. 4, pp. 799–808, 2009.
[38] D. Hao, X. Zhao, and L. Zhang, “Adaptive test-case prioritization guided by output inspection,” *2013 IEEE 37th Annu. Comput. Softw. Appl. Conf.*, pp. 169–179, 2013.
[39] A. Ansari, A. Khan, A. Khan, and K. Mukadam, “Optimized regression test using test case prioritization,” *Procedia Comput. Sci.*, vol. 79, pp. 152–160, 2016.
[40] S. S. and A. Singh, “Model based test case prioritization using greedy approach,” *Int. J. Emerg. Trends Eng. Dev.*, vol. 6, no. 5, pp. 80–88, 2016.
[41] Y. T. Yu and M. F. Lau, “Fault-based test suite prioritization for specification-based testing,” *Inf. Softw. Technol.*, vol. 54, no. 2, pp. 179–202, Feb. 2012.
[42] B. Jiang, W. K. Chan, and T. H. Tse, “PORA: proportion-oriented randomized algorithm for test case prioritization,” *2015 IEEE Int. Conf. Softw. Qual. Reliab. Secur.*, no. 61202077, pp. 131–140, 2015.
[43] J. A. Parejo, A. B. Sánchez, S. Segura, A. Ruiz-Cortés, R. E. Lopez-Herrejon, and A. Egyed, “Multi-objective test case prioritization in highly configurable systems: a case study,” *J. Syst. Softw.*, vol. 122, pp. 287–310, 2016.
[44] Y.-C. Huang, K.-L. Peng, and C.-Y. Huang, “A history-based cost-cognizant test case prioritization technique in regression testing,” *J. Syst. Softw.*, vol. 85, no. 3, pp. 626–637, 2012.
[45] A. K. Pandey and V. Shrivastava, “Early fault detection model using integrated and cost-effective test case prioritization,” *Int. J. Syst. Assur. Eng. Manag.*, vol. 2, no. 1, pp. 41–47, 2011.
[46] X. Cai and M. R. Lyu, “The effect of code coverage on fault detection under different testing profiles categories and subject descriptors,” pp. 1–7.
[47] K. K. Aggarwal, Y. Singh, and A. Kaur, “A multiple parameter test case prioritization model,” *J. Stat. Manag. Syst.*, vol. 8, no. 2, pp. 369–386, 2005.
[48] A. P. Conrad, R. S. Roos, and G. M. Kapfhammer, “Empirically studying the role of selection operators duringsearch-based test suite prioritization,” *Proc. 12th Annu. Conf. Genet. Evol. Comput. - GECCO ’10*, p. 1373, 2010.
[49] R. Huang, J. Chen, T. Zhang, R. Wang, and Y. Lu, “Prioritizing variable-strength covering array,” *2013 IEEE 37th Annu. Comput. Softw. Appl. Conf.*, pp. 8–11, 2013.
[50] F. Belli, M. Eminov, and N. Gokce, “A Comparative Soft-Computing Approach and Case Studies,” *Comput. Surv.*, pp. 425–432.
[51] S. Khandai, A. A. Acharya, and D. P. Mohapatra, “Prioritizing test cases using business criticality test value,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 3, no. 5, pp. 103–110, 2012.
[52] C. Y. Huang, J. R. Chang, and Y. H. Chang, “Design and analysis of GUI test-case prioritization using weight-based methods,” *J. Syst. Softw.*, vol. 83, no. 4, pp. 646–659, 2010.
[53] A. Khajeh-Hosseini, D. Greenwood, J. Smith, and I. Sommerville, “The cloud adoption toolkit: supporting cloud adoption decisions in the enterprise,” *Softw. - Pract. Exp.*, vol. 43, no. 4, pp. 447–465, 2012.
[54] A. E. V. B. Coutinho, E. G. Cartaxo, and P. D. de L. Machado, *Analysis of distance functions for similarity-based test suite reduction in the context of model-based testing*, vol. 24, no. 2. 2016.
[55] A. Khalilian, M. Abdollahi Azgomi, and Y. Fazlalizadeh, “An improved method for test case prioritization by incorporating historical test case data,” *Sci. Comput. Program.*, vol. 78, no. 1, pp. 93–116, 2012.
[56] S. Sampath and R. C. Bryce, “Improving the effectiveness of test suite reduction for user-session-based testing of web applications,” *Inf. Softw. Technol.*, vol. 54, no. 7, pp. 724–738, Jul. 2012.
[57] C. P. Indumathi and K. Selvamani, “Test cases prioritization using open dependency structure algorithm,” *Procedia Comput. Sci.*, vol. 48, no. Iccc, pp. 250–255, 2015.
[58] C. Dubois, Y. Fazlalizadeh, A. Khalilian, M. Abdollahi Azgomi, and S. Parsa, “Incorporating historical test case performance data and resource constraints into test case prioritization,” *Lect. Notes Comput. Sci.*, vol. 5668, pp. 43–57, 2009.
[59] C. Fang, Z. Chen, K. Wu, and Z. Zhao, “Similarity-based test case prioritization using ordered sequences of program entities,” *Softw. Qual. J.*, pp. 1–27, 2013.
[60] R. C. Bryce, S. Sampath, J. B. Pedersen, and S. Manchester, “Test suite prioritization by cost-based combinatorial interaction coverage,” *Int. J. Syst. Assur. Eng. Manag.*, vol. 2, no. 2, pp. 126–134, 2011.
[61] L. Zhang, J. Zhou, D. Hao, L. Zhang, and H. Mei, “Jtop: Managing JUnit test cases in absence of coverage information,” *ASE2009 - 24th IEEE/ACM Int. Conf. Autom. Softw. Eng.*, pp. 677–679, 2009.
[62] R. Huang, J. Chen, D. Towey, A. T. S. Chan, and Y. Lu, “Aggregate-strength interaction test suite prioritization,” *J. Syst. Softw.*, vol. 99, pp. 36–51, Jan. 2015.
[63] S. Yoo, M. Harman, P. Tonella, and A. Susi, “Clustering test cases to achieve effective and scalable prioritisation incorporating expert knowledge,” *Proc. ISSTA*, pp. 201–212, 2009.
[64] M. Zanoni, F. Perin, F. A. Fontana, and G. Viscusi, “Pattern detection for conceptual schema recovery in data-intensive systems,” *J. Softw. Evol. Process*, vol. 26, no. 12, pp. 1172–1192, 2014.
[65] D. Qiu, B. Li, S. Ji, and H. Leung, “Regression testing of web service: a systematic mapping study,” *ACM Comput. Surv.*, vol. 47, no. 2, pp. 1–46, 2014.
[66] H. Srikanth and S. Banerjee, “Improving test efficiency through system test prioritization,” *J. Syst. Softw.*, vol. 85, no. 5, pp. 1176–1187, 2012.
[67] L. Tahat, B. Korel, G. Koutsogiannakis, and N. Almasri, “State-based

models in regression test suite prioritization," *Softw. Qual. J.*, pp. 1–40, 2016.

[68] M. Staats, P. Loyola, and G. Rothermel, "Oracle-centric test case prioritization," *2012 IEEE 23rd Int. Symp. Softw. Reliab. Eng.*, pp. 311–320, 2012.

[69] S. Elbaum, P. Kallakuri, A. Malishevsky, G. Rothermel, and S. Kanduri, "Understanding the effects of changes on the cost-effectiveness of regression testing techniques," *Softw. Testing, Verif. Reliab.*, vol. 13, no. 2, pp. 65–83, 2003.

[70] R. K. Saha, "An information retrieval approach for regression test prioritization based on program changes," *Proceeding 37th Int. Conf. Softw. Eng. (ICSE 2015)*, pp. 268–279, 2015.

[71] L. Briand, Y. Labiche, and K. Chen, "A Multi-objective genetic algorithm to rank state-based test cases," pp. 66–80, 2013.

[72] W. Zhang, B. Wei, and H. Du, "Test case prioritization based on genetic algorithm and test-points coverage evaluation of test case prioritization," pp. 644–654, 2014.

[73] S. Kim and J. Baik, "An effective fault aware test case prioritization by incorporating a fault localization technique," *Proc. 2010 ACM-IEEE Int. Symp. Empir. Softw. Eng. Meas. - ESEM '10*, p. 1, 2010.

[74] C. Fang, Z. Chen, K. Wu, and Z. Zhao, "Similarity-based test case prioritization using ordered sequences of program entities," *Softw. Qual. J.*, vol. 22, no. 2, pp. 335–361, 2014.

[75] "ISO 26262-6:2011 - Road vehicles -- functional safety -- part 6: product development at the software level". *ISO*. N.p., 2017. Web. 3 Feb. 2017.