International Journal of Engineering & Technology, 7 (4.38) (2018) 1642-1646

International Journal of Engineering & Technology

Website: www.sciencepubco.com/index.php/IJET

Research paper

A Review on Test Case Prioritization Technique for Event Sequence Test Cases

Johanna Ahmad[1*], Salmi Baharom[2], Abd Azim Abd Ghani[3], Hazura Zulzalil[4], Jamilah Din[5]

[1]Faculty of Computer Systems and Software Engineering, Universiti Malaysia Pahang, 26 300 Gambang, Pahang, Malaysia
[2]Faculty of Computer Science and Information Technology, Universiti Putra Malaysia, 43400 Serdang, Selangor, Malaysia
**Corresponding author E-mail: johanna@ump.edu.my*

Abstract

Software testing is a process to verify and validate the correctness of a software product before it is delivered to the customer. Any modifications on the requirements or codes can cause the testing process to be redone all over again. Such occurrences could cause additional time, resources, and cost during testing. Hence, the test case prioritization (TCP) technique has been proposed, with the objective of prioritizing test case sequences and finding faults as early as possible to increase the effectiveness and efficiency of the testing process. Various TCP techniques are available, with the combination of different factors, research area, methodologies, and evaluation metric. This paper summarizes and discusses how the TCP technique can be applied for event sequence test cases. Analysis results from this preliminary study would help the researchers plan for future work, which is to propose an enhancement of the TCP technique for application with event sequence test cases.

Keywords: *Test case prioritization; Event Sequences; Review paper*

1. Introduction

Software testing is one of the stages in the software development life cycle that is conducted to verify and validate the software before it is delivered to the top management. However, a number of possible tests are needed to perform software testing, which would require a lot of time, resources, and cost. Longer duration of software testing will increase the cost, resources, and time. Furthermore, frequent changes may occur during the software development phase and this may increase the expected time to deliver the system. All the modification codes and change requirements must be tested again. This can cause the test suite to grow enormously, and rerunning the entire test suite is time consuming and would delay the project's completion [1]. Another reason for performing a software test is to avoid retesting the unwanted test cases, which would be redundant or obsolete [2]. Hence, numerous techniques have been proposed, such as test minimization, test selection, and test case prioritization (TCP).

These techniques have been proven by an empirical study as being able to reduce time, cost, and resources, apart from avoiding exhaustive testing [3]. Test selection is a technique that focuses on giving the best test cases for the execution during testing. The selection is based on the concept of how relevant the test case is to the system [4]. Hence, the minimization technique will generate a smaller test suite compared to the original test suite [5]. The effectiveness of the smaller test suite is measured based on its performance, which should at least be equal to the performance of the original test suite. The TCP technique was proposed to produce a new ordering of test cases for the execution during testing. [6] has defined the TCP technique as follows:

Current TCP technique

Given: T is a test suite, while PT is a set of permutations of T, and f is a function of PT to the real numbers.

Problem: To find $T' \in PT$, whereby $(\forall T'')\ (T'' \in PT)\ (T'' \neq T')\ [f(T') \geq f(T'')]$

The PT is presented as a set of possible orderings of set T, while f is a function that can be applied with any ordering, yields, and award values that were ordered. The f is represented as a quantification that is used to measure the success of the prioritization technique. The selection of the function f refers to the selected criteria to prioritize the test case, T. The previous definition also states that priority will be given to the higher award values [6]. The main objective of applying the TCP technique is to increase the performance of the test, with the new ordering test case, which is capable of detecting faults faster [7]. This paper presents a summary and discussion related to TCP for the event sequence test cases. TCP for a single event test case is different from the TCP for event sequence test cases. The reason for this will be explained in next section. This paper is organized as follows; related work on TCP technique, discusses how TCP technique can be implemented for event sequence test cases, and how to measure the effectiveness and efficiency of the TCP technique. Lastly, concludes this paper and future work.

2. Related Work

TCP technique is a method that is used to increase the effectiveness and efficiency of the test by providing new orderings for the test case execution. The execution starts with a test case that has higher priority, followed by the lower priority. Rothermal et al. [8] have distinguished two types of TCP: general and version-specific. In a general TCP for a given program P, and test suite T, the new ordering will be based on the prioritized technique, which will be useful for the subsequent modified versions of P. The general TCP is suitable for initial testing, where no information from the previous testing is available.

Meanwhile, the version-specific TCP would have information available from the previous testing, which is useful for improving the effectiveness and efficiency of the testing process. For a given program *P*, test suite *T* is prioritized based on the new ordering after a set of changes have been made to *P* and prior to the regression testing *P'*. The version-specific TCP can only be performed after *P'* is available [8]. The TCP technique has been proposed by numerous researches using various approaches since 1997. However, the main goal of applying TCP technique is to reduce time, cost and resources during the testing. This technique can be applied in various ways, for example, to increase the rate of fault detection during testing. It can be applied by prioritizing the test case by executing modules that had failed in the previous testing [9].

The Proportion-Oriented Randomized Algorithm (PORA) prioritizes a test case by optimizing the distance between the test suite and the hierarchy distributions of the test input [10]. It will compare the proposed technique with other existing techniques, such as total greedy, automated random technique (ART), and additional greedy technique. The empirical results gained from the experiments have shown that the PORA is more effective and stable compared to the other three techniques. A model of Software as a Service (Saas) has been widely used where customers can select and pay via web. However, numerous challenges must be faced to maintain the quality of the services. Thus, Hema et al. [11] proposed a model for the regression testing in SaaS environment. One industrial case study was selected to prove the effectiveness and efficiency of the proposed model towards SaaS applications. Towards the end of the experiment, the proposed model, with the prioritized technique, had managed to improve the effectiveness and efficiency of testing compared with the previous version. Furthermore, the information of failure history gained from the previous testing can help to improve the effectiveness of the new version.

Combinatorial interaction testing is a well-known technique that can improve the efficiency of testing. Fixed-strength and variable-strength interactions are some of the models available in combinatorial testing. However, due to the limitations of the existing combinatorial testing, which only supports fixed-strength interaction, [12] proposed two heuristic methods for the implementation of the variable-strength interaction. The experimental results for the proposed method have shown that it is more effective compared with test case generation order, random technique, and fixed-strength interaction.

Realizing the limitations of the test suite reduction, [4] proposed a method with the concept of increasing the rate of fault detection using a new ordering test suite. With this method, even though the test execution is stopped early, the best test cases that can detect faults would already have been executed. The empirical study was conducted using test cases that were based on user-session, with three subject applications. Furthermore, [4] proposed a new measurement to measure the effectiveness of the proposed method, known as *Mod_APFD_C*. *Mod_APFD_C* is the modification of the Average Percentage Fault Detected (APFD) that allows a comparison of test suites with different sizes, and incorporates them with test generation time and cost. There are five criteria that could influence the effectiveness of the proposed method, which are count-based, frequency-based, combinatorial-based, random, and logged order.

As previously mentioned, combinatorial interaction is widely used, as proven by empirical results obtained by numerous researchers. However, when testing resources are limited during the testing, the execution order of the interaction's test suite may be critical. Thus, Huang et al. [13] proposed a new "aggregate-strength prioritization" to improve the effectiveness of the testing. The combination of interaction coverage at different strengths managed to perform better than the test case generation, reverse test-case-generation, and random technique. To find the best test case for execution, all possible permutation test suites need to be investigated. This is to ensure that the best test case is not missing out during the testing. Genetic algorithm has been widely used to solve the TCP, test case selection, and test suite minimization. With the genetic algorithm concept, [14] defines the Epistatic Test Case Segment (ETS) by applying two associated crossover operators, namely, the Epistasis-based Order Crossover (E-Ord) and Partially-Mapped Crossover (PMX). The empirical results of these applications showed that the proposed methods are effective and efficient. Meanwhile, the Average Percentage Statement Coverage (APSC) is often used to evaluate the test case execution sequence. Code coverage is one of the known criteria used in TCP technique. [15] proposed several novel similarity-based TCPs using edit distances or ordered sequences. With five open source programmes, the experimental results have shown that the proposed technique manages to increase the rate of fault detection, while being the most cost-benefit compared with the existing techniques. Several novel similarity-based TCP techniques applied the farthest-first ordered sequence (FOS) algorithm and greed-aided-clustering ordered sequences (GOS) algorithm. The ordered sequences have been proven effective and been used in various fault location [16].

The TCP can be applied either as code-based or model-based [17]. For code-based TCP, the prioritization technique uses a programme from the system as one of the input besides the test cases. On the other hand, the model-based TCP is based on information retrieved from the developers or testers, whereby these information are useful for developing the model-based technique. Sometimes, modification of an existing model can be done to fill the gap left by previous researchers.

3. Test Case Prioritization Technique for Event Sequence Test Cases

3.1 TCP Technique Limitations

According to the SLR analysis by [18], four major TCP limitations have been identified, namely, failure to prioritize multiple suites, failure to handle same-priority values, ignoring the practical weight factor, and most test cases are small in size. The SLR was conducted using 50 primary studies and the publications were from 2005 to 2015. Most of the previous researches failed to handle the issue of same priority values. Some papers stated that they would pick randomly, while other papers did not mention anything about the same priority value. Furthermore, many researchers believe that the combination of factors may help to break ties [19]. Break ties here refer to the case where more than one test case share the same priority value [20]. The priority value is used to rank the execution of the test case, which uses the concept that the highest priority will be executed first compared with the lower priority.

Table 1 presents some of the techniques and approaches applied by numerous researchers. Columns for whether the probability of same priority value exists, did the paper handle the issue of same priority value, and how they handle this issue were added to the table to summarize how each of these techniques handled same priority value. As seen in Table 1, most of these techniques had randomly applied more than one test case that shared the same priority value after the prioritization processes ([19], [20], [21], [22]).

3.2 Factor Determination

Various combinations of factors have been applied by previous researchers to meet the objective of the TCP technique. To the best of our knowledge, the combination of factors depends on the objective and research area. Additionally, the type of test cases may need to be investigated since the properties of the test case may influence the performance of the TCP technique. A simple analysis was conducted to get an overview of how the combination of factors can be applied in TCP technique. Figure. 1 depicts

10 factors that were identified as potential factors, which can improve the effectiveness and efficiency of the TCP technique. Based on observations, fault has become the most popular factor to be applied in a TCP technique. Out of 70 papers, 42 of them had applied fault in their TCP techniques. Some of these techniques combined fault with other factors, such as execution time [23]. Meanwhile, Zhang et al. [24] combined fault matrix with distance to propose a prioritization technique using the Adaptive Random Sequence (ARS) in order to get higher fault detection. Category-partition-based was applied to assess the diversity of the test cases. Finally, the experimental results showed that the proposed technique had higher fault detection compared to other techniques. As depicted in Figure.1, besides fault, redundancy, complexity, frequency, and requirements had ranked at the top as well. Thus, it can be concluded that there is a need to combine more than one factor to improve the effectiveness and efficiency of the TCP technique.

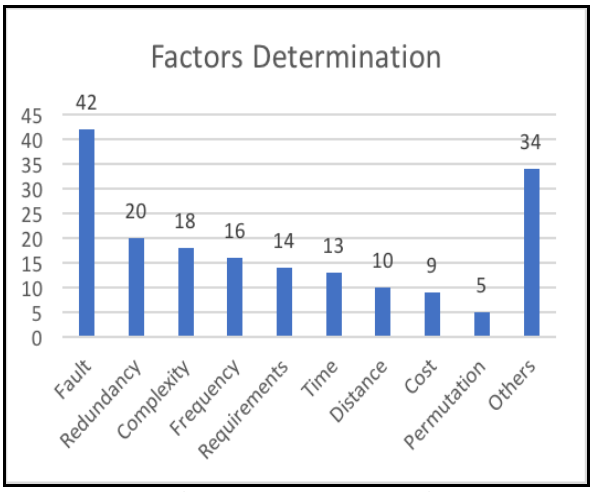

Figure 1: Number of Papers that Applied Each of the Listed Factors

Table 1: Test case prioritization techniques

Technique	Probability of Same Priority Value Exists?	Same Priority Value Issue is Handled or Not	How do They Handle Same Priority Value Issue
Proportion-Oriented Randomized Algorithm	Yes	No	Pick Randomly
Model for Regression Testing in SaaS	Yes	No	Not Available
Weighted dependence propagation model	Yes	No	Pick Randomly
Two Heuristic Methods in Order to Prioritize VCA	Yes	No	Pick Randomly
Prioritizing Test Cases Using Business Criticality Test Value	Yes	No	Not Available
Modified Cost-Cognizant Test Case Prioritization (MCCTCP)	Yes	No	Not Available
Aggregate-strength prioritization	Yes	No	Pick Randomly
Cluster-based test case prioritization technique	Yes	No	Not Available
Epistatic Test Case Segment (ETS)	Yes	No	Not Available
Model structure and test case profile	Yes	No	Pick Randomly
Multi-objective genetic algorithm method	Yes	No	Not Available
Novel similarity-based test case prioritization techniques	Yes	No	Pick Randomly
Test Case Prioritization Based on Genetic Algorithm	Yes	No	Not Available

3.3 Existing TCP Technique

Various techniques and approaches for the TCP technique have been proposed since 1997. From the simple analysis conducted for this paper, it was found that code coverage and requirement coverage are the most utilized techniques since the TCP technique. Meanwhile, fault coverage, interaction coverage, historical data, statement coverage, and execution time were applied by four previous papers, followed by input information with three papers, and lastly, programme changes with two papers. Some of the previous researches had combined more than one technique to increase the number of faults detected [25]. Some researchers agreed that a combination of more than one technique could optimize the testing process by detecting faults earlier and increasing the number of faults detected [26]. Code coverage has become the most utilized technique because of its ability to enhance failure-detection and the confidence on software reliability [9], [27]. However, various perspectives have been reported regarding the effectiveness of the code coverage to be applied in TCP technique. Figure. 2 presents a summary of the analysis done regarding the number of papers for each of the technique that exists in TCP.

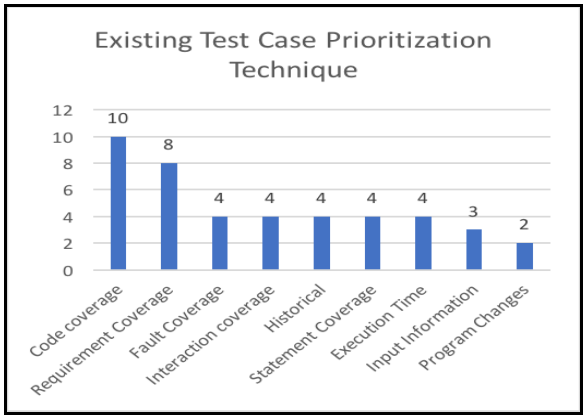

Figure 2: Number of Papers that Applied Each of the Listed Factors

3.4 TCP Technique for Event Sequence Test Cases

TCP technique can be applied either for event sequence test cases, or for a single event test case. Based on the SLR analysis that was conducted in 2016, out of 50 primary studies, only 36 per cent had applied the TCP technique for event sequence test cases [18]. Previous studies have pointed out that the event sequence test case is more complex compared with the single event test case due to several reasons, such as the huge amount of test cases, with considerable degree of redundancy [28]. The large input sequence may lead to the possibility of the test case to have a combination of events [29]. Furthermore, [30] has addressed that the complexity of the event sequence testing was due to the large test space, different positions of the events, and also because of the curious permutations of inputs.

Generally, there are seven types of transition criteria for the event sequence test case, namely, single event single outcome, many events and single outcome, event to event, event to component, component to event, and component to component [30]. Event sequence test cases mostly consist of a combination of methods in one class, thus the link between methods should be taken into consideration to avoid exhaustive testing. [31] proposed the idea that since the event sequence test case has an enormous number of states, thus, every state should be tested. Hence, the change of internal data state from one state to another needs to be under consideration since it involves the interaction between the events. The properties of the event sequence test case, as previously introduced, are useful for proposing a TCP technique for the event sequence test cases, with the goal of offering effective prioritized test suite compared to the original test suite.

4. Evaluation to Determine the Effectiveness of Test Case Prioritization Technique

Numerous evaluation metrics have been proposed to measure the effectiveness and efficiency of TCP technique. Based on the literature, existing evaluation metrics include the Average Percentage Fault Detected (APFD), Average Percentage Statement Coverage (APSC), Average Percentage of Faults Detected per Cost (APFDc), and Normalized Percentage of Faults Detected (NAPFD). Most researches would use the APFD. The SLR analysis by [18] had also shown that out of 50 primary studies, 58 per cent of the existing TCP techniques had applied APFD as their evaluation metric, to measure the effectiveness and efficiency of the proposed technique.

The APFD is often used to measure how quickly faults can be detected within the testing process. [9] proposed the APFD in 2001, with the objective of quantifying the rate of fault detection for the prioritized test suite. The APFD value ranges between 0 and 100. The higher APFD value shows that the technique is effective compared to other techniques. The expectation is that the prioritized test suite should obtain higher APFD value compared to the original test suite. However, there are two limitations that need to be satisfied before the APFD can be applied, as listed below:

- All faults must have equal fault severities.
- All test cases must cost the same.

If these assumptions are not fulfilled, unsatisfactory APFD values would be produced [32]. According to [9], the APFD can be calculated using the following Equation (1):

$$APFD = 100\,x\left(1 - \frac{TF_1 + TF_2 + \cdots + TF_n}{nm} + \frac{1}{2n}\right) \qquad (1)$$

where n denotes the number of test cases, while m is the number of faults revealed, and TF_i is the position of the first test case, and T reveals the fault.

5. Conclusion

The main goal of the TCP technique is to improve the effectiveness and efficiency of the testing process. Detection of faults earlier in the TCP technique can reduce time, cost, and resources of testing. In the TCP technique, a test case that has high priority value will be executed first compared to the lower priority. This paper has summarized several researches on TCP technique, which included different techniques, evaluation metrics, approaches, and methodologies. Each technique had applied different factors, different areas, and has its own advantages and disadvantages. The essence of this review paper will be used to identify areas of improvements in TCP technique for event sequence test cases.

Acknowledgement

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] G. Rothermel, R. H. Untch, C. C. C. Chu, and M. J. Harrold, "Test case prioritization: an empirical study," *Proc. IEEE Int. Conf. Softw. Maint. - 1999 (ICSM'99). 'Software Maint. Bus. Chang. (Cat. No.99CB36360)*, 1999.

[2] A. A. Haider, A. Nadeem, and S. Rafiq, "On the Fly Test Suite Optimization with FuzzyOptimizer," *2013 11th Int. Conf. Front. Inf. Technol.*, pp. 101–106, 2013.

[3] S. Nayak, C. Kumar, and S. Tripathi, "Effectiveness of prioritization of test cases based on Faults," *2016 3rd Int. Conf. Recent Adv. Inf. Technol. RAIT 2016*, pp. 657–662, 2016.

[4] 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.

[5] M. A. Sapaat and S. Baharom, "A Preliminary Investigation Towards Test Suite Optimization Approach for Enhanced State-Sensitivity Partitioning," no. November, pp. 40–45, 2011.

[6] G. Rothermel, R. H. Untch, C. Chu, M. J. Harrold, and I. C. Society, "Prioritizing Test Cases For Regression Testing Prioritizing Test Cases For Regression Testing," *IEEE Trans. Softw. Eng.*, vol. 27, no. 10, pp. 929–948, 2001.

[7] H. Do, G. Rothermel, and A. Kinneer, "Prioritizing JUnit Test Cases : An Empirical Assessment and Cost-Benefits Analysis," pp. 33–70, 2006.

[8] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: a family of empirical studies," *IEEE Trans. Softw. Eng.*, vol. 28, no. 2, pp. 159–182, 2002.

[9] S. Elbaum, A. Malishevsky, and G. Rothermel, "Incorporating varying test costs and fault severities into test case prioritization," *Proc. 23rd Int. Conf. Softw. Eng. ICSE 2001*, pp. 329–338, 2001.

[10] 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.

[11] 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.

[12] 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.

[13] 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.

[14] F. Yuan, Y. Bian, Z. Li, and R. Zhao, "Search-Based Software Engineering," vol. 9275, pp. 109–124, 2015.

[15] 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.

[16] M. Renieres and S. P. Reiss, "Fault localization with nearest neighbor queries," *Autom. Softw. Eng. 2003 ...*, pp. 30–39, 2003.

[17] G. Pardha Sagar and P. V. R. D. Prasad, "A Survey on Test Case Prioritization Techniques for Regression Testing," *Indian J. Sci. Technol.*, vol. 10, no. 10, pp. 1–6, 2017.

[18] J. Ahmad and S. Baharom, "A Systematic Literature Review of the Test Case Prioritization Technique for Sequence of Events," *Int. J. Appl. Eng. Res.*, vol. 12, no. 7, pp. 1389–1395, 2017.

[19] S. Sampath, R. Bryce, and A. M. Memon, "A uniform representation of hybrid criteria for regression testing," *IEEE Trans. Softw. Eng.*, vol. 39, no. 10, pp. 1326–1344, 2013.

[20] Z. He and C.-G. Bai, "GUI Test Case Prioritization by State-coverage Criterion," *2015 IEEE/ACM 10th Int. Work. Autom. Softw. Test*, 2015.

[21] A. Ammar, S. Baharom, A. A. A. Ghani, and J. Din, "Enhanced Weighted Method for Test Case Prioritization in Regression Testing Using Unique Priority Value," in *Information Science and Security (ICISS), 2016 International Conference*, 2016.

[22] R. C. Bryce and A. M. Memon, "Test suite prioritization by interaction coverage," *Work. Domain Specif. approaches to Softw.*

test Autom. conjunction with 6th ESEC/FSE Jt. Meet. - DOSTA '07, pp. 1–7, 2007.
[23] 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.
[24] 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.
[25] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege, "A systematic review of the application and empirical investigation of search-based test case generation," *IEEE Trans. Softw. Eng.*, vol. 36, no. 6, pp. 742–762, 2010.
[26] 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.
[27] 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.
[28] R. C. Bryce, S. Sampath, and A. M. Memon, "Developing a single model and test prioritization strategies for event-driven software," *IEEE Trans. Softw. Eng.*, vol. 37, no. 1, pp. 48–64, 2011.
[29] 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.
[30] H. Reza, S. Endapally, and E. Grant, "A Model-Based Approach for Testing GUI Using Hierarchical Predicate Transition Nets Nets (HPrTNs) and Model Based," pp. 1–5, 2007.
[31] O. Kumar, P. K. Bhargavi, and V. Kumar, "A Single Model for Event-Driven Software," *Int. J. Adv. Comput. Theory Eng. Ex.*, vol. 2, pp. 31–36, 2013.
[32] L. Zhang, S.-S. Hou, C. Guo, T. Xie, and H. Mei, "Time-aware test-case prioritization using integer linear programming," *Proc. eighteenth Int. Symp. Softw. Test. Anal. - ISSTA '09*, pp. 401–419, 2009.