Journal of Theoretical and Applied Information Technology
31st July 2018. Vol.96. No 14
© 2005 – ongoing JATIT & LLS

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

MODEL-BASED TEST CASE PRIORITIZATION: A SYSTEMATIC LITERATURE REVIEW

[1]MUHAMMAD LUQMAN MOHD SHAFIE, [2]WAN MOHD NASIR WAN KADIR

[1,2]Department of Software Engineering, Faculty of Computing, Universiti Teknologi Malaysia, 81310 Skudai, Johor, Malaysia

E-mail: [1]luqman_1993@yahoo.com.my, [2]wnasir@utm.my

ABSTRACT

Re-testing all test cases during regression testing is costly and time-consuming. These problems motivate researchers to come up with various techniques to overcome them. One of the techniques is Test Case Prioritization that prioritizes test cases in test suite by ordering them according to a desired objective goal. Model-based is one of the approaches which utilizes the system models to make prioritization. The main purpose of this systematic review is to identify and categorize the current state-of-the-art while providing a baseline for future research in model-based Test Case Prioritization. A general search term related to model-based approach was used during the study search in selected digital libraries ranging from 2005 to 2016 to find primary studies that propose model-based approach. A total of 32 primary studies consisting of 21 combinations of conference proceedings, workshop, and symposium and 12 journal articles were finalized after going through a strict selection process. A sum of 48 distinct approaches with their respective characteristics and models used have been identified, and some general constraints of model-based Test Case Prioritization have been highlighted. Future research is recommended to put more focus in detailing the introduced category to benefit the researchers and practitioners.

Keywords: *Model-Based, Test Case Prioritization, Regression Testing, Systematic Literature Review.*

1. INTRODUCTION

A fully developed software system cannot be considered as completely done. Changes in a software system are inevitable and will continuously occur over time because of many factors [1]. As software system changes, parts of the system are modified, added and discarded to satisfy the changes. When changes are implemented in a system, they are re-tested to ensure that no new bugs or defects introduced during the modification. This phase is necessary to ensure that the quality of the system is in top-notch [1] and is particularly known as Regression Testing. The sole purpose of Regression Testing is to make sure that modifications and changes made to the particular software system did not create any negative impact on it [2].

Regression Testing is proved to be one of the most expensive phases in a software development life cycle [3]. Hall et al. [4] stated that almost 80% of the testing budget is spent mostly in Regression Testing. This circumstance occurs mainly because software always undergo modifications and new versions are introduced from time to time to cope with these changes. As a result, the test suite tends to grow in size because new test cases might be added to cover the modified elements for the testing purpose [5]. As the implication, the cost will increase continuously and at a point, re-executing the whole test suite will not be relevant anymore. Apart from that, Regression Testing also consumes much time during the process. These problems are evidenced by a report from an industrial collaborator stating that one of its products with 20,000 lines of code requires seven weeks for the entire test suite to be carried out [6]. This bothersome situation will undoubtedly affect the testing phase significantly in many aspects.

Because of these reasons, researchers had come up with diverse techniques to solve this issue. In their survey, Yoo et al. [2] classified Regression Testing into three main categories which are Test Suite Minimization (TSM), Test Case Selection (TCS) and Test Case Prioritization (TCP). TSM techniques eliminate any obsolete or redundant test cases permanently from the test suite [7] while TCS techniques select test cases from the test suite according to a certain criterion [8]. Last but not least TCP techniques aim to re-order test cases from the original test suite according to a certain goal in a manner that the test cases which serve the goal the most are given the highest priority [9]. All of these techniques possess their strengths and weaknesses in Regression Testing. However, in this paper, we will only retain our focus in TCP, more specifically the model-based approach.

Model-based TCP was firstly introduced by Korel et al. [10] who implemented a different approach in prioritizing test cases. In this particular approach, instead of using the system codes, the system models are utilized to prioritize test cases [10]. Activity diagram, Extended Finite State Machine (EFSM), sequence diagram and use case diagram are some of the models used during prioritization process in the studies we selected [11-14]. One of the advantages of model-based over code-based is cheaper execution cost [10]. Besides, analyzing models would be faster than the source codes, and early feedback can be achieved since models are an abstraction of the actual code which make them simpler to analyze compared to the system itself [3]. Early feedback here means that some bugs can be observed or identified in the system models even before the code is tested by tracing the models back to the requirements to spot for inconsistency. Code-based approaches on the other hand require code knowledge in order to prioritize test cases which means prioritization cannot begin until the source code is available and most of them are language dependent so testing process will become troublesome in cases where the program is written in various programming languages [5]. Catal et al. [3] in their study encouraged researchers to develop more model-based prioritization methods because according to them, the percentage of studies published concerning model-based is growing however at a slower pace. This statement is in tandem with the findings made by Yoo et al. [2] when they stated in their discussion that model-based regression testing approaches are getting more attention. They also predicted that model-based regression testing approaches would be of crucial importance in the forthcoming because of higher level regression testing and scalability. Higher level regression testing means that regression testing can be moved from structural level to functional level because model-based approaches can act as an intermediary between requirements and testing activities. Scalability means that model-based approaches can scale up better than code-based approaches when dealing with industrial scale software system. Despite that, earlier surveys revealed that the number of papers published related to developing new model-based approaches is not very encouraging. According to Catal et al. [3], only four papers proposed or used model-based approach between 2001 and 2009 and six were published during 2009 and 2010 period. However, that survey was pretty much out-of-date. To the best of our knowledge, no latest review has been done that specifically focuses on model-based TCP. Therefore, we feel that there is a need to conduct this review to observe the current progress in model-based TCP and for the sake of future research. Some related works are discussed later in this section.

Driven from the statements above, the motivation or the purpose of this systematic review is to (i) identify and categorize the current state-of-the-art of model-based TCP while providing a baseline or starting point for future researchers in improving the model-based approaches in TCP and to (ii) identify how the existing approaches can handle the common constraints in model-based TCP from their category perspective. The first contribution of this review is that all known model-based TCP approaches starting from 2005 until 2016 were searched exhaustively in well-known digital libraries and organized systematically based on their characteristics in this review. In addition, a quality assessment score is provided for each study to assist researchers in finding quality studies. Therefore, this review can act as a baseline or starting point which can be used for researchers that are curious and eager to learn more about model-based TCP. Secondly, this review contributes by introducing six general categories that clustered all the existing approaches based on their characteristics. This categorization can be useful because it provides a clearer indication of future research of how a better method can be proposed by observing which category performs better and vice versa. Also, a classification of the models used in the existing approaches was also provided. This classification can come in handy for observing the trend and providing assistance on which model to be considered when proposing new approaches. Lastly, this review analyzes how the existing approaches address the common constraints in model-based TCP. It can save a lot of researchers' time by just referring to this review rather than inspecting every study available in the literature and can assist future research to propose a better approach that can ultimately improve prioritization result. There are also several interesting points related to model-based TCP that are not included in this review. They are the evaluation of prioritization effectiveness of each approach, background study for each mentioned model, full comprehensive categorization and thorough analysis of model-based TCP limitations in each approach. This topic is discussed more in Section 5 about discussion and implications for future research.

There exist some related review papers that mentioned model-based TCP. For example, the systematic mapping study [15] conducted by Catal et al. [3] is quite similar with our study when it comes to the discussion about model-based prioritization method in one of their RQs. However, the main difference is that their study is a mapping study that discusses the current trend of TCP while ours is a systematic literature review (SLR). Catal et al. [3] in their explanation to distinguish SLR and mapping study also described that SLR has specific RQ and related to the outcomes of empirical studies while mapping study has general RQ and related to research trends. Furthermore, the SLR conducted by Singh et al. [9] is also quite similar to our study because they conducted a systematic review about TCP and did mention some of the studies which are included in our review but treated them as a modification-based approach because these studies also incorporate code changes in their approaches. Nevertheless, their main intention is to summarize the current state-of-the-art of TCP as a whole. Therefore, the discussion on model-based TCP is pretty much in general, and only some model-based studies are included. In addition, Yoo et al. [2] did a survey on Regression Testing that includes TCP as one of the three major techniques used. Their work also mentioned model-based TCP, but the major distinction is that their survey targets a wider scope which covers major branches in regression testing. Thus, only a small number of model-based studies are included with the less detailed discussion. Apart from that, the studies that are conducted by Joshi et al. [16] and Mohanty et al. [17] apparently focused on model-based TCP based on the title of their study. Nevertheless, both studies only include a few papers from model-based TCP as compared to ours. Therefore, the discussion is not thoroughly made for model-based. Besides, their studies also include discussion on code-based and requirement-based TCP which make their center of attention not entirely on model-based. The key element that makes this particular SLR distinct is its center of attention that specifically focuses on the model-based approach in TCP. Also, the review papers mentioned earlier only included some model-based studies compared to ours might be because of fewer

Table 1: Distinction between Existing Similar Studies.

Reference	Distinction		Model-based TCP studies included (Included in current study)
	Associated existing study	**Current study**	
[3]	Systematic mapping study	Systematic literature review	16 (7)
	General research questions	Specific research questions	
	Reviews all types of TCP approaches generally	Reviews specifically model-based TCP	
[9]	Related approaches are treated as modification-based	Related approaches are treated as model-based	5 (4)
	Focuses on wider scope which is all approaches in TCP	Focuses on narrower scope which is only model-based TCP approaches	
[2]	Survey Regression Testing approaches (TSM, TCS, TCP) in general	Reviews all approaches in model-based TCP specifically	4 (3)
[16]	Survey on code-based, requirement-based and model-based TCP approaches	Reviews only on model-based TCP approaches	3 (2)
	General discussion on model-based TCP and only few studies included	Thorough discussion on model-based TCP and more studies included	
[17]	Survey on code-based, requirement-based and model-based TCP approaches	Reviews only on model-based TCP approaches	2 (1)
	General discussion on model-based TCP and only few studies included	Thorough discussion on model-based TCP and more studies included	

publications that were made during the period. So, we can say that our study is the latest and updated review made for model-based TCP. Table 1 summarizes the distinction between the current study and similar studies in the literature which are explained earlier. The first column shows the references of the associated existing studies. The second column which consists of two sub columns describe the distinction between the existing similar studies and the current study. Lastly, the third column shows the number of model-based TCP studies that are included in the associated studies while the number in the bracket represents the number of those studies that are included in the current study

The systematic review procedure outlined by Keele [18] was referred where necessary to guide the construction of this systematic review. Based on the guideline, we formulated a set of research questions (RQs) that will assist in clarifying the aims of the review stated earlier. From the RQs constructed, we were able to come out with a general search string which is utilized into five selected digital libraries in the effort to find the related studies to answer the corresponding RQs. Then, the search results from all digital libraries were scrutinized and refined in various aspects to ensure that the most relevant and high-quality studies are chosen. Using a particularly designed data extraction form, crucial and related information was excerpted from the 32 finalized studies to be analyzed and evaluated to address the RQs on model-based TCP. The remainder of this paper elaborates our systematic review of the model-based approaches in TCP. Section II presents a more detailed elaboration of model-based TCP as the background information. Section III explains the research method while Section IV demonstrates the execution and results of the research method. Then, Section V discusses the findings and their implications for future research. Section VI presents the threats to the validity of this systematic review, and lastly, Section VII concludes this systematic review.

2. BACKGROUND

TCP is a technique under regression testing in which test cases are re-ordered from the original test suite according to a particular purpose in a manner that the test cases serving the purpose the most are given the highest priority [9]. We took the definition of TCP problem proposed by Elbaum et al. [19] into consideration for this systematic review which is stated below:

Given: *T*, a test suite; *PT*, the set of permutations of *T*; *f*, a function from *PT* to the real number.
Problem: Find *T'* ∈ *PT* such that

$$(\forall T'')(T'' \in PT)(T'' \neq T')[f(T') \geq f(T'')] \quad (1)$$

In this definition, PT serves as the set of all possible sequences f *T*, while *f* is the function when implemented to ny of the sequences, yields an *award value* for that particular sequence. In short, the definition expects that the higher award values are preferable than the lower ones. The *f* function is the most crucial part that represents the approaches used to prioritize test cases. There are some possible goals when referring to prioritization in this context. Elbaum et al. [19] also stated some of the goals of their study which are:

- To increase the rate of early faults detection when executing test suite.
- To increase the code coverage under test at a faster pace when executing test suite.
- To increase their confidence in the system's reliability at a faster rate
- To increase the possibility of revealing faults associated with particular code changes earlier in the testing process.

Over time, researchers have proposed numerous approaches for TCP. All of these approaches can be divided into two main categories which are code-based and model-based. In code-based TCP, test cases are prioritized by utilizing the source code information of the software system. A survey conducted by Mahdian et al. [5] stated that the vast majority of test selection strategies were code-based. A study carried out by Catal et al. [3] also proved that the most investigated prioritization method was coverage-based that conquered 40 percent of all the various techniques they had gathered. Coverage-based is a kind of code-based prioritization where the more coverage achieved by a test suite, the more chances faults can be revealed earlier during the testing process. Coverage in this context means the code coverage of the software system for example statement, function or code block. The downside of code-based is that code knowledge is needed to prioritize test cases [5] which means prioritization cannot begin until the source code is available. Another drawback of code-based is that most of them are language dependent [5] so the testing process will become troublesome in cases where the program is written in various programming languages.

On the other hand, model-based prioritization manipulates the models of the software system to

Journal of Theoretical and Applied Information Technology
31st July 2018. Vol.96. No 14
© 2005 – ongoing JATIT & LLS
ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

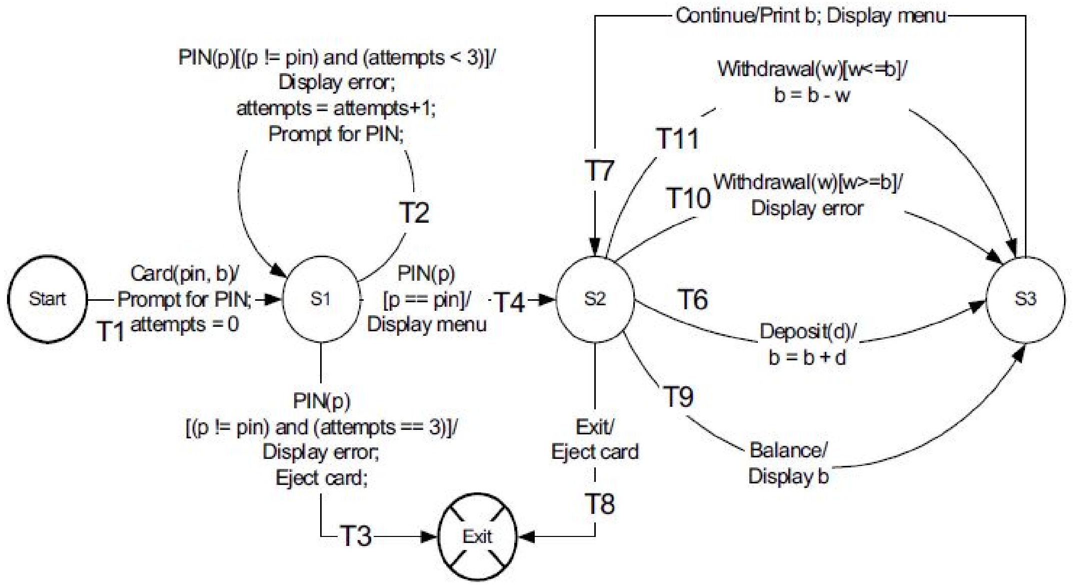

Figure 1: Example of EFSM.

perform prioritization [20]. Any kind of TCP approaches that use the system models in it can be categorized as a model-based approach. Some examples of system models are use case diagram, sequence diagram, state machine diagram and activity diagram. Figure 1 illustrated an EFSM model retrieved from Korel et al. [10] which is used in their proposed work to perform prioritization. The primary advantage of model-based prioritization is that execution of the system models is rather faster than the execution of the system codes itself during testing [10]. The reason is that system models are at a higher level of abstraction thus capturing system's behaviors and structures are less complex compared when using the source codes [21]. Therefore, model-based prioritization is considerably inexpensive compared to code-based prioritization which is both resource-wise and time-wise [10]. Nevertheless, model-based prioritization also possesses their own weaknesses. One of the major flaws is its dependence on the correctness and completeness of the system models [14]. This topic regarding model-based prioritization will be discussed more thoroughly in the next section when we address our research questions.

3. RESEARCH METHOD

This systematic review was produced in tandem with the guidelines proposed by Keele [18]. The methodology consists of five crucial steps which are research questions, search strategy, study selection process, study quality assessment and data extraction. In the first subsection, the research questions that addressed the aims of this whole review were defined. The search strategy will explain the design of searching the studies that are possibly relevant to the defined research questions. Next, the study selection process illustrated how the primary studies were scrutinized and filtered to include the ones that are related to this systematic review. The fourth subsection described how the refined studies were evaluated by implementing the formulated quality assessment criteria. The last sub-topic clarified how the information was extracted from the selected studies.

3.1 Research Questions

The most crucial part in any systematic review is to specify the research questions because they steer the whole systematic review methodology process [18]. To accomplish the motivations of this systematic review, we had formulated the following research questions (RQs) to help to answer the aims of this review.

RQ1. How to categorize the existing approaches in model-based TCP based on their characteristic?
RQ2. How the existing approaches address the inherent constraints in model-based TCP from their category perspective?

To answer RQ1, we first identified all the existing model-based TCP approaches from related studies in the literature from a certain time period. From the selected studies, we extract the important information that will be used to answer RQ1 and

RQ2. From the collected information, we introduced several general categories that group these existing approaches based on their characteristics. To address RQ2, we first recognized the common problems faced by researchers when proposing new model-based TCP approach. From the drawbacks identified, we reflect on how the existing approaches overcome those drawbacks from the perspective of the introduced categorization.

3.2 Search Strategy

As a starting point, all primary studies related to the research questions will be searched. To design the search terms, we constructed a search string to be used for paper searching in digital libraries. The general search string is "model-based test case prioritization". Additionally, possible synonyms and alternative spelling were also considered when searching to prevent the risk of overlooking or missing relevant studies. For example, the synonyms for the keyword "case" would be "suite" and their alternative spelling would be "cases" and "suites". The synonyms and alternative spelling for all keywords in the search string were connected using the Boolean operators AND and OR appropriately. The final search string is as follow: *(((((((((model-based) AND test) OR tests) OR testing) AND case) OR cases) OR suite) OR suites) AND prioritization) OR prioritizing).*

The formulated search string was then used in the search query. The selected digital libraries are shown below with their respective address.

- ACM (dl.acm.org)
- IEEE Xplore (ieeexplore.ieee.org)
- Science Direct (www.sciencedirect.com)
- SpringerLink (www.springerlink.com)
- Web of Science (apps.webofknowledge.com)

All the electronic sources mentioned earlier were selected because they have been mentioned and proven to be relevant in software engineering studies [3, 9, 18]. It should be noted that each digital library has different requirements in their search query. Therefore, necessary adjustments were made to the search string when applied to each digital library to get a correct and precise search result. Furthermore, the period of the published studies was limited to 2005 until 2016. The reason 2005 was chosen is that our early investigation indicated that the first formal approach in model-based was introduced by Korel et al. [10] during the year 2005 based on the citation number.

3.3 Study Selection Process

After all studies that are possibly related to the research questions were identified, different level of inspections was made by the reviewers. These steps are performed to ensure duplicate and irrelevant studies are excluded. Preliminarily, studies were retrieved from the selected digital libraries listed previously by querying the constructed search string suitably for each digital library. Next, in stage 2 namely duplicates removal, the results were filtered to discard any duplicate studies. This duplicate phenomenon occurs because some studies were stored in more than one digital library. The title-based exclusion concentrated on reviewing the titles according to whether the title of a particular study was related to the research questions or vice versa. In this case, only titles that might be related to model-based TCP and use English as the communication language were included. Then, the remaining studies were refined further by undergoing abstract-based exclusion. The studies' abstracts were reviewed to verify that only studies which proposed a model-based approach in TCP were taken into account.

3.4 Study Quality Assessment

Besides the general inclusion/exclusion criteria done in the previous section, a quality assessment is also necessarily important. Some of the rationales why quality assessment is important is to provide a more detailed inclusion/exclusion criterion, to filter out mediocre works and to weight the significance of individual studies when results are being synthesized as stated by Keele [18].

In this particular assessment, we constructed a set of quality assessment questions to evaluate the validity of selected studies. Each question has three possible answers of "Yes", "Partly" and "No" with their score of 1, 0.5 and 0 respectively. Each study received their corresponding quality score by calculating the total score they got in answering each of the assessment questions. We decided that only studies that obtain the quality score higher than half of the maximum score will be included in the review. The maximum score is 5 so only studies that obtained score higher that 2.5 were selected. The following questions influenced by the questions presented by Keele [18] were utilized and altered appropriately for assessing the quality of the selected studies:

QA1. Are the purposes of the study precisely described?

QA2. Is the proposed approach clearly described?

QA3.Are the results and findings clearly described and associated with the aims of research?
QA4.Is the effectiveness of the proposed approach assessed accordingly?
QA5.Does the paper includes conclusions that are related to the stated objectives of study?

3.5 Data Extraction

This last section of the research method focused on extracting necessary information from the finalized primary studies to be recorded into a form designed for data extraction. This process was done so that only crucial information regarding the research questions were extracted and to consistently arrange them while addressing the research questions clearly. The designed data extraction form is shown in Table 2.

Table 2: Data Extraction Form.

Category	Data Extraction Category	Description	Addresses
General data	Identification number	Unique identifier for each primary study	
	Extraction date	Data extraction date	
Study particulars	Title	Title of primary study	
	Author	Name of the primary study author(s)	
	Publication year	Year of publication	
	Type of paper	Journal, conf. paper, book chapter, etc	
	Publication medium	Name of publisher	
Study content	Approach used	The approach used in model-based TCP	RQ1, RQ2
	Model/input used	Activity diagram, state machine diagram, etc	RQ1, RQ2
	Process involved	What are the processes during execution?	RQ1, RQ2
	Effectiveness Measurement	How the effectiveness of approach measured?	RQ2
	Weaknesses of Approach	What are the identified weaknesses?	RQ2
	Advantages of approach	How the inherent constraints in model-based TCP is handled	RQ2

4. EXECUTION AND RESULTS

4.1 Primary Studies

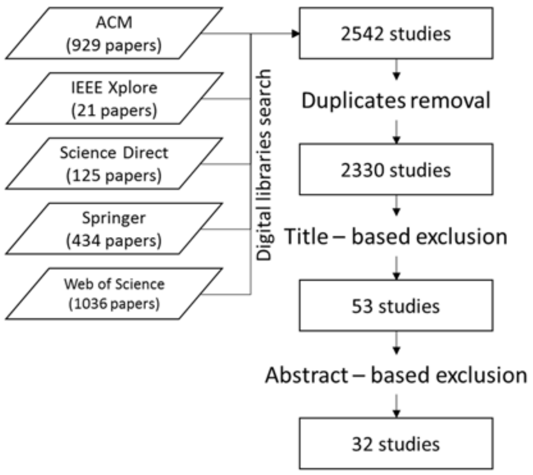

Figure 2: Study selection process.

Figure 2 illustrates the stages from the searching of studies to the selection process. A total of 2542 studies were found from the digital libraries search described in the previous section. An overwhelming set of articles were obtained in the initial step because it seems that the search from digital libraries also returned results of the individual term from the search string thus including other unrelated domains. Regardless of how many articles were obtained, due to the strict methodology that must be followed, all of them must be recorded. On the other hand, the utilized search string also influenced the result and the search string used in this review is sufficient but

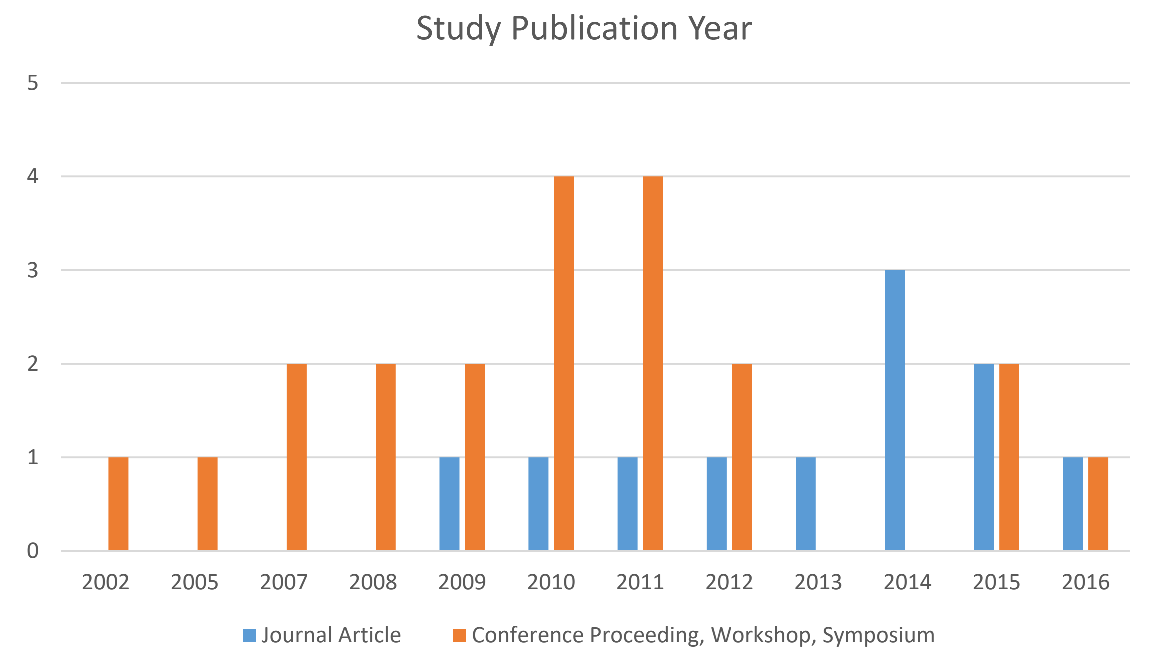

Figure 3: Study's publication year.

might not be the best combination available. Then, all of the titles from the result were listed in an Excel sheet. The duplicate removal step was done by using the remove duplicates feature available in the Excel. The process eliminated 212 studies, and a total of 2330 studies remained. The next step focused on title-based exclusion where we reviewed each individual title to select the ones related to model-based TCP. This step disposed a large number of 2277 studies and left 53 remaining studies. The main reason why many studies was inapplicable is that as mentioned earlier, the individual keyword in the search string was also used in other unrelated domains from this review and they were also included in the result list. Then, a more specific filtering was done by investigating the abstract of each remaining study to verify that only studies which proposed a model-based approach in TCP were taken into account and a total of 32 finalized studies were obtained. This number includes the one that we inspected from the references in the finalized studies to find more approaches and found one study which is published in 2002, earlier from our lowest period of 2005. The reason why we insisted in including this study although we claimed that the first model-based TCP was introduced in 2005 is that this study is still considered as model-based but not as formal as the one introduced by Korel et al. [10] in 2005. Figure 3 depicts the publication year of each finalized study which consists of 21 combinations of conference proceedings, workshop, and symposium and 11 journal articles where the x-axis represents publication year while the y-axis represents number of papers published. Figure 4 illustrates the citation visualization of each study where incoming arrow means a study is referring to that particular study the arrow is pointing to, and bigger circle means that a study is cited more. Discussion for Figure 3 and

Figure 4 is presented in Section 5. The overview for each study which includes type of paper, publication year and publication medium is shown in Table A2 in Appendix A.

As mentioned earlier, a study quality assessment is crucial in providing a more thorough inclusion/exclusion criterion, filtering out average works and weighing the significance of individual study when results are being synthesized. After inspecting through all of the selected studies, we recorded the quality assessment score for each particular study into a table for future reference. All of the chosen studies were rewarded a quality assessment score higher than 2.5 which is the limit that we agreed. This finding does not mean that our quality assessment is ineffective because, from our investigation, all studies included were undoubted of quality works. Therefore, none should be removed. The quality score for all chosen studies is shown in Table 3. From our observation, most studies that received high quality score are from journal publication. This is not surprising because unlike journal publication, paper publication from conferences are bounded to a certain number of pages thus less information can be fitted in. This circumstance is the reason why studies from conferences did not received high quality score. It also shows that the quality assessment is correct and useful. Last but not least, we recorded all the necessary information from all selected studies in the data extraction form illustrated earlier for the consistent arrangement of information retrieved while addressing the research questions explicitly.

4.2 Model-based TCP Approaches and Their Categories (RQ1)

After the data extraction process was done, we identified that many types of distinctive approaches had been proposed over the period. A total number of 48 different approaches were revealed from the selected studies. It is noteworthy that some papers proposed more than one approach. Also, there were some extended version papers from the same authors with more detailed contents of their previous papers which proposed the same approaches. This circumstance occurs when a conference paper is selected to be published in the journal article. To be specific, the study published by Korel et al. [10] is mostly similar with the one published by Korel et al. [22] and Korel et al. [20] with only a few tweaking done while studies from Tahat et al. [23] and Tahat et al. [21] are the extended versions of those approaches. For this particular situation, we decided to only include approaches from Korel et al. [10] and Korel et al. [22] because they were the earliest

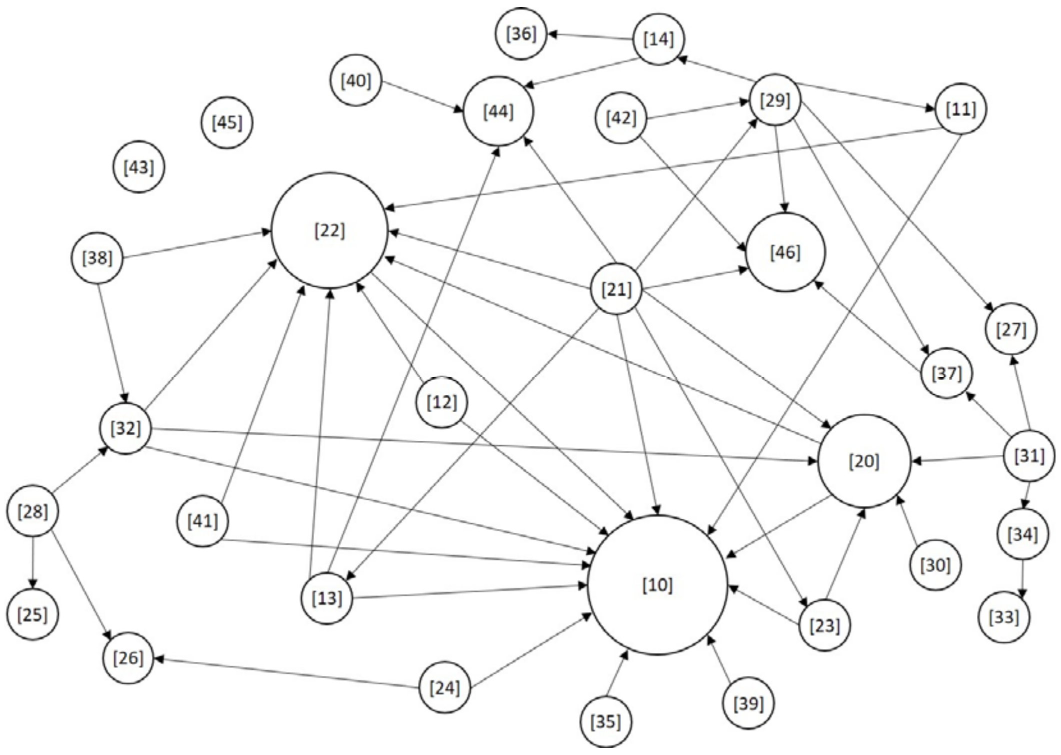

Figure 4: Study's citation.

studies published among the ones stated above and shown different proposed approaches. Therefore, the number of approaches discovered will not be the same as the number of studies included. After going through all the identified approaches, six general categories were introduced where all the approaches with same characteristics were clustered together in their appropriate category. They are model modification based (MMB), weight / complexity / priority / risk assignment based (WAB), genetic algorithm based (GAB), fuzzy logic based (FLB), probability-based (PB) and graphical user interface (GUI) based (GUIB). All these categories were introduced so that the approaches identified can be grouped to show the similarity in their approaches' execution while highlighting for possible weakness. This grouping process is also crucial to assist in getting the general idea of how a particular approach operates. Each of the general category and their brief description is shown in the following subsection.

4.2.1 MMB

In this category, an approach prioritizes test cases based on information collected from the modification identified regarding the original model and the modified model. The prioritization goal of approaches in this category is to increase the possibility of revealing faults associated to particular code changes earlier in testing process because they rely on code changes between the original model and modified model to prioritize tests. EFSM is normally used as the input model for this category of prioritization based on the approaches in this category. An EFSM model contains two types of element which are the set of states and the transitions connecting the states [23]. This model mainly captures the behaviors of the system because it can visualize all the possible states or behaviors the system can be. In a particular state, a transition is triggered that execute a sequence of actions (shift to another state) when a specific event occurs, and the defined condition is satisfied. A simple example for this category is the Selective Test Prioritization approach proposed by Korel et al. [10]. In this particular approach, when the original source code of the software system is modified, the original system models will change too to reflect the modifications. According to the authors, to identify a transition related to a particular source code modification is pretty easy because usually actions in the model are implemented as functions in the source code [20]. After these modified transitions are spotted, the test cases that cover them will be given high priority than the others.

4.2.2 WAB

In this category, an approach implements the assignment of weight / complexity / priority / risk to the nodes or edges in the system model based on certain criteria to prioritize test cases. The prioritization goal of approaches in this category is to increase the rate of early faults detection because according to Kaur et al. [24], the tests that have the highest complexity which means having the most fault occurrence probability are prioritized first. It is observed that most studies in this category utilized activity diagram for prioritization purpose [14, 24-28]. An activity diagram is a behavioral model because it captures the dynamic behaviors of a system. According to Swain et al. [27], activity diagram illustrates the sequential or parallel control flow between the activities in a system. To clarify this category in further detail, the path complexity approach proposed by Kaur et al. [24] is used for explanation. In their approach, the activity diagram is converted to control flow graph. In brief, for each basis test path generated from the control flow graph, their complexity is calculated by summing the number of nodes, weight of path, number of predicate nodes traversed, and number of logical conditions traversed by them. Lastly, the path with the highest complexity will be assigned the highest priority.

Table 3: Quality Score Result of Selected Studies.

Paper reference	**Score**					**Total**
	QA 1	**QA 2**	**QA 3**	**QA 4**	**QA 5**	
[29]	1	1	0.5	0	1	3.5
[11]	1	0.5	0.5	0	1	3
[30]	1	0.5	1	0	1	3.5
[31]	1	1	1	1	1	5
[25]	1	0.5	0.5	0.5	1	3.5
[32]	1	1	1	1	1	5
[12]	1	1	0.5	0.5	1	4
[24]	1	1	1	0	1	4
[10]	1	1	0.5	1	1	4.5
[22]	1	1	0.5	1	1	4.5
[20]	1	1	0.5	1	1	4.5
[13]	1	1	1	0.5	1	4.5
[33]	0.5	1	0	0.5	1	3
[34]	0.5	1	1	0.5	0.5	3.5
[35]	1	0.5	0.5	0.5	1	3.5
[36]	1	1	1	1	1	5
[37]	1	1	1	1	1	5
[38]	1	1	1	0.5	1	4.5
[28]	1	1	0.5	0	1	3.5
[14]	1	0.5	0.5	0.5	1	3.5
[39]	1	1	1	0	1	4
[40]	0.5	1	0.5	0.5	0.5	3

[41]	1	0.5	0.5	0	1	3
[26]	1	0.5	0.5	1	1	4
[27]	1	1	1	0.5	1	4.5
[23]	1	1	1	1	1	5
[21]	1	1	1	1	1	5
[42]	1	1	0.5	0.5	1	4
[43]	0.5	0.5	0.5	0.5	1	3
[44]	1	0.5	0.5	0	1	3
[45]	1	1	1	1	1	5
[46]	1	1	1	0	1	4

4.2.3 GAB

In this category, a subset of soft computing, which is known as genetic algorithm (GA), is utilized by adapting their process into test cases prioritization using model-based approach. GA is one of the approaches in soft computing, a search algorithm that imitate the way nature evolves species using a natural selection of the fittest individuals as its main inspiration [46]. Based on the study conducted by Sabharwal et al. [46], the test that have the highest information flow (IF) score which is calculated using GA are prioritized first. This IF metric is utilized to compute the complexity of a node which also determines the probability of fault occurrence of a node. Therefore, the prioritization goal of approaches in this category is to increase the rate of early faults detection. This prioritization goal is also stated in the study from Wang et al. [42]. All the identified approaches applied GA in their implementation with added metaheuristic algorithm in several studies. For example, Nejad et al. [34] designed four memetic algorithm based on GA. Each of them is different in term of its local search algorithm which are stochastic local search, hill climbing, random iterative improvement and simulated annealing. These local search algorithms are incorporated to improve prioritization result. Meanwhile, Wang et al. [42] introduced a hybrid GA which combined GA and particle swarm optimization (PSO) to keep the best information of local individual during iteration process. Most of them utilized activity diagram in their approaches, a behavioral model as the input model [34, 42, 46]. An example for this category is the Combination of Basic IF metric & GA approach proposed by Sabharwal et al. [46]. In their proposed approach, the activity diagram is converted into control flow graph, and a set of test paths that cover all branches are generated. Then, weight is assigned to all nodes in the control flow graph using Basic IF model and complexity for each path is calculated by summing the weighted nodes a particular path traversed. Decision nodes of the control flow graph will form the chromosomes to be utilized in the GA part. The number of bits in the chromosomes is determined by the number of decision nodes in the control flow graph. The chromosomes' fitness value is calculated by applying the complexity obtained using the Basic IF model for the path that satisfies the decision nodes direction. Lastly, the chromosome will undergo crossover and mutation to find the chromosome with the highest fitness value. The chromosome value with the highest fitness value will represent the decision nodes an are referred to find the highest priority path. Then the other tests are prioritized based on the fitness value.

4.2.4 FLB

In this category, the fuzzy logic concept is applied to a particular approach to prioritize test cases. There are only three from all of the selected studies that were considered as fuzzy logic based. Two of them utilized the Event Sequence Graph (ESG) [30, 32] while the other one used symbolic execution tree (SET) [38]. The studies that utilized ESG described that their approaches are coverage-based while the study that used SET created fuzzy input sets that aim to give high priority to tests with larger coverage. Therefore, it can be deduced that the prioritization goal of the approaches in this category is to increase the code coverage under test at a faster pace when executing test suite. According to Belli et al. [30], ESG portrays a system's behavior interacting with user's actions. SET is a model generated from the symbolic execution of a system which depicts all achievable execution paths of the model, based on symbolic inputs, as well as any constraints on those inputs [47]. The approach from Rapos et al. [38] is used for the explanation because it illustrates the whole fuzzy logic concept in TCP clearly. For starter, the fundamental steps in a common fuzzy control system are fuzzification, inference, composition, and defuzzification. In this approach, first, they determine the input and output set which map to the fuzzification and defuzzification in the fuzzy control system. The input set consists of test suite size, SET size, relative test case size and output significance with each of them having three fuzzy sets of small, medium and large. In this case, the information from the SET model is used to determine their values. The output set consists of test case priority that has four fuzzy sets of low, medium, high and very high. Next, a set of rules is designed which maps to the inference step in the fuzzy control system. An example rule taken from that study is small test suite size, small SET size, small relative test case size with low or medium output significance yields a high priority test case. The steps mentioned earlier explained generally how fuzzy logic based prioritize test cases in an easy way to understand. The implementation of the other two approaches are more complex to be explained here, but the concept of fuzzy logic is still applied in them.

4.2.5 GUIB

In this category, an approach utilizes the GUI components of the system to prioritize test cases in model-based testing. Majority of the approaches in this category used a unified model from a study conducted by Bryce et al. [31] because most of them are actually from the same study. Basically, each of the related approaches is used as criteria to be fitted into a generic function "OrderSuite" that yield prioritized test suite based on the chosen criteria. We treated each approach differently because their individual concept can still be applied as a standalone approach. This unified model was designed by them, so it can test both the GUI and web application during test prioritization. This model is considered as a behavioral model because it captures the different states the GUI or web application are currently in based on the event triggered by users such as opening menus, checking checkboxes, selecting radio-buttons, and clicking button. We take some approaches from the study conducted by Bryce et al. [31] for a brief explanation. First is the 1-way Parameter-value Interaction Coverage-based which prioritize test cases by giving highest priority to those that cover the maximum number of parameter values that do not appear in the previously selected tests. In this context, the term "parameter" is the widgets while "value" is the setting for the widgets. For instance, the "vehicle" dropdown is a parameter with the selected value "car" or the "male" checkbox parameter with value "true". Another example approach is Unique Window Coverage Count-based which prioritize test cases that cover unique windows which had not been covered in the previously selected test. The term "window" here means a GUI window for GUI application and a page for a web application. These terms that pair the elements between GUI and web application are what that made the unified model. Based on the approaches, the prioritization goal is considered to be increasing the code coverage under test at a faster pace when executing test suite. To prove this, the 1-way Parameter-value Interaction Coverage-based tries to cover all the parameter values available as fast as possible during testing while the Unique Window Coverage Count-based tries to cover all the unique windows as early as possible.

4.2.6 PB

In this category, an approach combines the probability calculation and uses the probability score of test cases to prioritize them. In the selected studies, only one approach is considered to be in this category which is the Reinforcement Learning & Hidden Markov Model (RL-based HMM) approach [45]. The study introduced and used a new model called extended digraph in their approach. According to them, this extended digraph is a behavioral model that supports more set of information than the traditional model regular digraph and also offers better performance. The steps of this approach are quite complex and lengthy to be explained thoroughly in this review, so only a brief description is provided. The first step is to determine initial RL-based HMM parameters based on the generated test cases from MBT techniques. Next is to train an RL-based HMM with maximum likelihood using Baum-Welch algorithm. Baul-

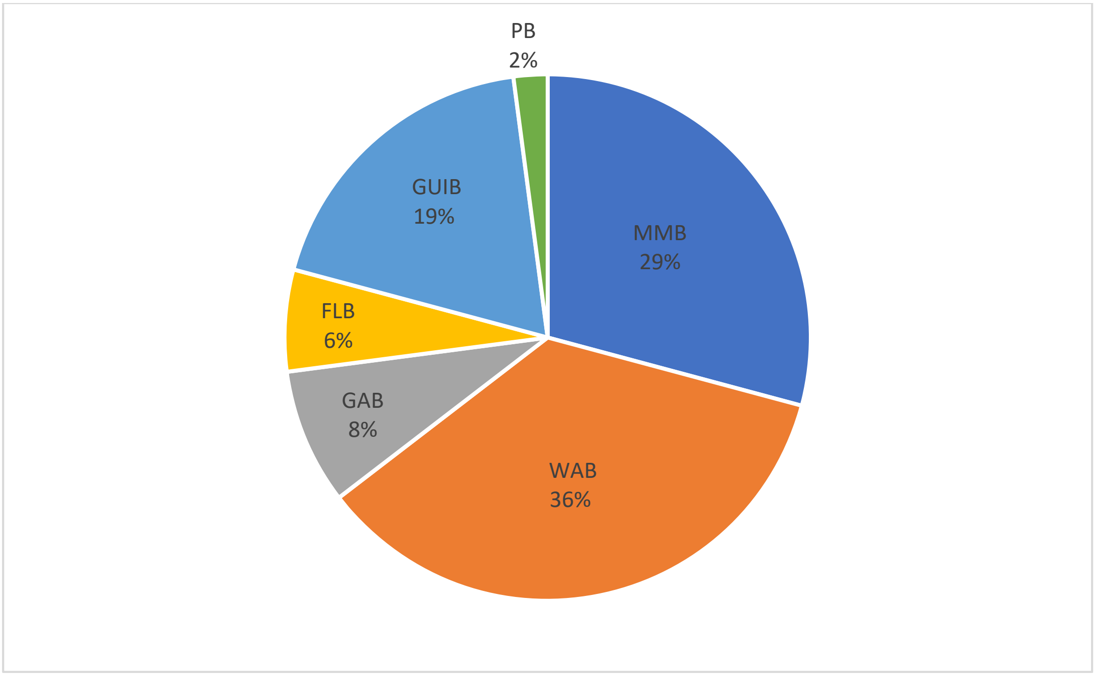

Figure 5: Distribution of existing approaches.

Welch is an algorithm that determines the most optimal model as that with the most likelihood of the estimated parameters [48]. Finally is to compute the test cases forward probabilities using forward algorithm and prioritize test cases. Emam et al. [45] stated that the prioritization of their approach is based on the quantity of changes that can be generated in GUI state by performing each sequence of actions. Higher quantity of changes mean more critical events or larger volumes of computations which give higher possibilities of uncovering faults during testing. They prioritize test cases with higher forward probabilities that have more chances of containing such events. So, it can be said that the prioritization goal of this category is to increase the rate of early faults detection.

4.2.7 Distribution of existing approaches

Figure 5 illustrates the distribution pie chart of the existing approaches in their respective category. Table 4 shows all the introduced categories with their respective studies. Discussion for Figure 5 and Table 4 is presented in Section 5. The full lists of all approaches with their respective descriptions are shown in Table A1 in Appendix A.

Table 4: Model-based TCP Categories with Associated Studies.

No	Category	Associated Study
1.	MMB	[10-12, 22, 35-37, 41]
2.	WAB	[13, 14, 24-29, 33, 40, 43, 44]
3.	GAB	[34, 39, 42, 46]
4.	FLB	[30, 32, 38]
5.	GUIB	[31, 45]
6.	PB	[45]

4.2.8 Distribution of models used

In addition to classification based on general characteristics, we also clustered all the identified approaches according to the models used as the input by a particular approach. The sole reason for this grouping is to observe the trend of what models are currently used in performing model-based TCP and the sense of using them. Figure 6 depicts the distribution pie chart of the models used in all the approaches discovered from the selected studies. Models that are cited in less than two studies were grouped into “Others”. Some individual study proposed more than one approach therefore models included in “Others” does not mean that it is only utilized for a single approach. Table 5 represents all the models used associated with the studies citation that utilized them. Some approaches make use of more than one models in their proposed approach thus some studies will appear more than once in the indexes. Again, in this categorization, it should be noted that models used for studies from Korel et al. [20], Tahat et al. [23] and Tahat et al. [21] were not included for the same reason stated earlier in Section 4.2. We decided to only include approaches from Korel et al. [10] and Korel et al. [22] because they were the earliest studies published and shown different proposed approaches.

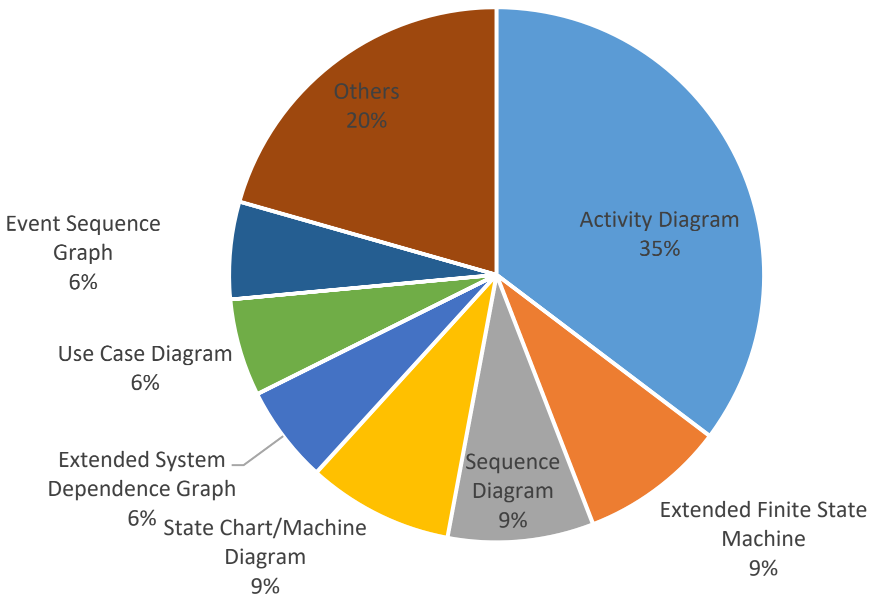

Figure 6: Distribution of Models Used.

4.3 Inherent Constraints in Model-based TCP and How They Are Handled (RQ2)

From the review done to all the selected studies, some critical challenges in model-based TCP had been revealed. One of the main challenges is the absolute dependence to the system models being used. In other words, the correctness and completeness of the system models will ultimately determine whether the prioritization results are accurate or vice versa. It means even the most sophisticated model-based TCP approaches will not be effective if the system models utilized are inadequate or unreliable. Sapna et al. [14] highlighted this issue in their study of prioritizing use cases. They asserted that, it is obvious that if a requirement is not captured in the model, tests that satisfy this requirement will not be generated. If this happens, not only the prioritization result will be

Table 5: Model-based TCP Categories with Associated Studies.

No	Model	Associated Study
1.	Activity Diagram	[11, 14, 24-28, 34, 39, 41, 42, 46]
2.	Extended Finite State Machine	[10, 12, 22]
3.	Sequence Diagram	[13, 41, 44]
4.	State Chart/Machine Diagram	[33, 39, 43]
5.	Event Sequence Graph	[30, 32]
6.	Extended System Dependence Graph	[36, 37]
7.	Use Case Diagram	[14, 44]
8.	Object Relation Diagram	[29]
9.	Combined Model (specific name not stated)	[31]
10.	Extended Object-Oriented System Dependence Graph	[35]
11.	Symbolic Execution Tree	[38]
12.	State Transition Diagram	[40]
13.	Communication Diagram	[27]
14.	Extended Digraph	[45]

inaccurate, even the testing process itself will be invalid. Belli et al. [30] also argued that different models focus on different features of system under test (SUT) thus prioritization results will possibly be inconsistent if disparate models are used for a particular approach. Furthermore, GÖKÇE et al. [32] in their study stated that the proposed prioritization approach can be influenced by modifications in the model which proved the approach’s dependencies to the model. From this challenge, it can be deduced that the effectiveness of an approach can be preliminarily assessed by judging the completeness and correctness of the model used itself.

The model completeness challenge is addressed by approaches in MMB category which utilized EFSM model. EFSM model is considered to be a complete model because of several reasons. Firstly, it is a behavioral model as with many other models discussed in the model-based TCP categories earlier. The behavioral model is the right choice when it comes to testing because based on the details of the system shown in the model, the expected output (test oracle) can be illustrated clearly to be compared with the actual output of the SUT to find for defects. On the other hand, a structural model such as a class diagram or component diagram cannot provide much information for testing because they do not resemble the behavior of the SUT. Secondly, EFSM is a complete model because it provides sufficient details of the system. This characteristic is what that make EFSM superior from other models discussed in this review even though they are all behavioral models. Although it is an abstraction (simpler version) of the system itself, crucial details are not abstracted out which make it executable on its own. For this reason, EFSM can be exploited to generate abstract test cases which can be run on the actual system during testing. This is actually one of the processes which are done in Model-Based Testing. More details can be obtained from the study conducted by Utting et al. [49].

In addition, another constraint in model-based TCP is that an approach might be far too complex to be understood and executed. The obvious effect of this issue is the increase of execution time and resource used. For the sake of clarification, an approach proposed by Korel et al. [10] which they claimed as a complex approach is used as an example. The approach is the Model Dependence-based Test Prioritization from MMB. They elaborated this approach in further details in two of their extended version of studies for modification made both in the software system and models and for modification for which models are not modified (only source code is modified) [21, 23]. In short, this approach makes use of model dependence analysis to determine the patterns of how added and deleted transitions communicate with the modified model and lastly utilizes this information to prioritize test cases. In their empirical study, it is highlighted that this approach needs more analysis and gathers extra information from the model to do prioritization than other approaches they had proposed [23] thus increasing execution time. Also, more resources are required because the whole model execution trace must be stored to compute the interaction patterns [22]. Nevertheless, they reported that the prioritization result using this approach was much better and encouraging than the other approaches proposed by them [23].

The approaches from WAB category were introduced to overcome the complexity issue mentioned earlier. This is because the implementation of approaches in this category is pretty simple and straightforward. They implement the assignment of weight / complexity / priority / risk to the nodes or edges in the system model based on certain criteria in order to prioritize test cases. This type of prioritization is proven to be competent among other categories because it measures the importance of nodes or edges thus giving priorities to the test cases that cover them with a high degree of importance which may contain faults. For instance, the approach Degree Measure Method (DMM) proposed by Al-Herz et al. [29] ranks components based on fan-in degree then prioritizes test cases that cover the highest ranked components. The fan-in degree in this context means the number of components that lead or traverse through this particular component. The rationale behind this approach is that most of the other components will fail to get services if this high fan-in degree component breaks down. As the conclusion, even though the approaches in this category is not too complex, the prioritization result is considered to be promising, but this depends mainly on how the weight assignment is done.

5. DISCUSSION AND IMPLICATIONS FOR FUTURE RESEARCH

From the results obtained, it can be observed that the quantity of study published for model-based TCP throughout the period is more or less constant with no significant rise or downfall (Refer Figure 3). There is a noticeable increase in journal publication which means that some researchers have given serious attention to model-based TCP. There is also an interesting pattern that can be observed in the study's citation in Figure 4 where the study that got cited the most is from Korel et al. [10] published at 2005. This pattern proves that they are among the pioneers that firstly introduced model-based TCP into the literature and the reason why we choose the year 2005 as the limit in the search strategy. Overall, the growth of model-based TCP is still moderate compared to code-based TCP. Nonetheless, more publications have been made in this recent years as compared to few years ago if we compare the number with the study from Catal et al. [3].

On the other hand, WAB category dominates the overall number of existing approaches identified with a percentage of 36. From the review conducted to the studies, an observable reason why most of the approaches proposed utilized this approach is that of its simplicity. For example, in the Tree structure approach proposed by Sapna et al. [28], first an activity diagram is converted into tree structure using depth first search algorithm. After the tree structure is obtained, weights are assigned to the nodes and edges. In activity diagram, the nodes are like action/activity, fork-join that handles concurrency and branch-merge that checks Boolean expression for possible branches to be followed. In their approach, priority of 3, 2 and 1 are given to fork-join, second to branch-merge and action/activity respectively. For edges, weights are assigned based on the product of number of incoming dependencies and number of outgoing dependencies for starting node and ending node of an edge respectively. Next, the weight of each path (scenario) is calculated by summing up the weights of nodes and edges it traverses. Finally, all paths are prioritized according to the value of weight. This approach is pretty much straightforward, but its effectiveness and prioritization result will heavily rely on the criteria of how weights are assigned. From Table 4 that shows model-based TCP categories with associated studies, we can also observe that most of the studies are in WAB category, in tandem with WAB dominance in number of existing approaches. Note that in distribution of existing approaches, GUIB is the third largest category but in categories with associated studies, there are only two studies for GUIB. This circumstance happens because the one study from Bryce et al. [31] proposed eight distinct approaches that can be treated independently. For that reason, we separated them which caused the number of approaches to be more that the number of studies in GUIB category.

Another observation worth discussing is the percentage of models used. From the results depicted in Figure 6, it is fairly obvious that most of the approaches used the activity diagram of the system to perform prioritization with a percentage of 35 over other models used. A simple reason for this circumstance is because, obviously, the most dominant category is WAB where most of its approaches utilized activity diagram. However, there are also some other reasons why this model is mostly used when proposing model-based TCP. Sharma et al. [39] in their proposed approach defined activity diagram as a model that is used to represent the dynamic behavior of the system. Swain et al. [27] stated in their study that activity diagram is a perfect model to portray the realization of the operation in the design phase. An activity diagram is also used to illustrate the scenarios of relating use case and are utilized by system's stakeholders to comprehend their functionality [27]. In addition, Wang et al. [42] also stated in their study that activity diagram is a critical basis for system testing because it has the capability to portray the system's work flow and parallel activities. Not to mention that activity diagram is a behavioral model. In testing, the ultimate goal is to ensure that a system's actual behaviors conform to its desired behaviors and behavioral model appears to be the right candidate for this circumstance. From these statements, it can be concluded that activity diagram is a complete illustration of the system behavior ergo adding to the reasons why most of the approaches used this diagram.

We consider this study as a starting point for researchers to further pursue this topic in the future. However, this review is still far from perfection and possesses several limitations that can still be improved. There are still many factors to be considered so that researchers and practitioners can fully utilize the categorization of the approaches in the future. For that reason, several recommendations for future research are provided. Firstly, it is recommended that further analysis should be done to find the most successful or efficient approach available by comparing the quality of each one of them in term of prioritization. Additionally, further investigation should be made so that more details can be extracted from the categorization. Such details are like the suitable application a category can be implemented in, the required programming language to implement them or the advantages and drawbacks of using them. Moreover, the constraints of model-based TCP mentioned in this review are basically general which is based on the category level. This issue can be further investigated by elucidating weakness in individual approach and how they can affect prioritization result.

6. THREATS TO VALIDITY

One of the major threats to the validity of this systematic review is the search strategy process. When automatic search from digital libraries is being done, the general keyword "model-based test case prioritization" has the possibility of not capturing important studies that use uncommon terms in their contents. We leave no stone unturned and overcome this circumstance by also considering all possible synonyms and alternative spelling related to the general search term and joined them all using the Boolean operators AND and OR appropriately as stated previously. As the implication of using this solution, unrelated studies were also included in the search result. During the study selection process, a manual title-based inclusion was thoroughly conducted by the reviewers to ensure that no essential studies were inadvertently excluded. A systematic data extraction was also utilized to ensure that no important information was missed or left behind during the extraction of data from selected studies. The data extraction form explained earlier also helped in addressing the research questions clearly and transparently.

In addition, the accuracy and transparency of this systematic review are also affected by the publication biases of the selected studies. This situation arises because some scholars tend to deliberately highlight the positive results of their research while obscuring or concealing the negative ones to prove that their proposed approach is a solid improvement. To cope with this phenomenon, we constructed the study quality assessment to provide a minimum requirement for a study to be included in the review. This solution will prevent the inclusion of ambiguous and bias studies that can affect the results of this review. As explained in the previous section about the quality assessment scoring scheme, we had carefully awarded the scores for each study and decided only studies that have the quality score higher than 2.5 will be included in the review.

7. CONCLUSION

The main purposes of this systematic review are to identify and categorize the current state-of-the-art of model-based TCP while providing a baseline or starting point for future researchers in improving the model-based approaches in TCP and to identify how the existing approaches can handle the common constraints in model-based TCP from their category perspective. The essence of this systematic review is to recognize the gaps in model-based approaches thus proposing possible improvements or contributions to fill in the gaps. These objectives were steered by the guidelines proposed by Keele [18] in performing SLR. In the introduction, we interpreted what is Regression Testing, the problem faced by practitioners when implementing it and the techniques that are feasible in solving the problem. A brief explanation of the TCP was elaborated then we went deeper into describing the model-based TCP. Next, we went into the planning of the systematic review, the research method. After the thorough discussion was made, two research questions were formulated that can adequately address the aims of this review. Five different digital libraries were selected in order to find studies that can possibly relate to the research questions. The preliminary search result that contains mixed studies was gone through general inclusion/exclusion criteria in order to establish the studies that are most relevant to the constructed research questions. In addition, a quality assessment was done to the selected studies to provide more detailed inclusion/exclusion criteria while preventing biases and ambiguous results. A systematic data extraction form was used to extract only essential information that can address the research questions. Execution and results established illustrated that from 32 primary studies selected, a total number of 48 distinctive approaches were identified where each of them is clustered into six general categories to distinguish approaches with similar characteristics. Results obtained also proved that activity diagram is the most used model in performing model-based prioritization because it can perfectly portray the realization of the operation and illustrates the scenarios of relating use case. Some of the model-based prioritization weaknesses that worth discussing were also elaborated. Lastly, some recommended future research are discussed will focus on working out the stated problems by proposing an approach in model-based prioritization.

ACKNOWLEDGMENT

The authors would like to express their deepest gratitude to Research Management Center (RMC), Universiti Teknologi Malaysia (UTM) and Ministry of Higher Education Malaysia (MOHE) for their financial support under Fundamental Research Grant Scheme (Vote number R.J130000.7816.4F824) and

Research University Grant (Vote number Q.J130000.2516.19H64).

REFERENCE

[1] M. Rava and W. M. Wan-Kadir, "A Review on Prioritization Techniques in Regression Testing," *International Journal of Software Engineering and Its Applications*, vol. 10, No. 1, 2016, pp. 221-232.

[2] S. Yoo and M. Harman, "Regression testing minimization, selection and prioritization: a survey," *Software Testing, Verification and Reliability*, vol. 22, No. 2, 2012, pp. 67-120.

[3] C. Catal and D. Mishra, "Test case prioritization: a systematic mapping study," *Software Quality Journal*, vol. 21, No. 3, 2013, pp. 445-478.

[4] T. Hall, S. Beecham, D. Bowes, D. Gray, and S. Counsell, "A systematic literature review on fault prediction performance in software engineering," *IEEE Transactions on Software Engineering*, vol. 38, No. 6, 2012, pp. 1276-1304.

[5] A. Mahdian, A. A. Andrews, and O. J. Pilskalns, "Regression testing with UML software designs: a survey," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 21, No. 4, 2009, pp. 253-286.

[6] S. Elbaum, P. Kallakuri, A. Malishevsky, G. Rothermel, and S. Kanduri, "Understanding the effects of changes on the cost-effectiveness of regression testing techniques," *Software testing, verification and reliability*, vol. 13, No. 2, 2003, pp. 65-83.

[7] C.-T. Lin, K.-W. Tang, and G. M. Kapfhammer, "Test suite reduction methods that decrease regression testing costs by identifying irreplaceable tests," *Information and Software Technology*, vol. 56, No. 10, 2014, pp. 1322-1344.

[8] M. Grindal, B. Lindström, J. Offutt, and S. F. Andler, "An evaluation of combination strategies for test case selection," *Empirical Software Engineering*, vol. 11, No. 4, 2006, pp. 583-611.

[9] Y. Singh, A. Kaur, B. Suri, and S. Singhal, "Systematic Literature Review on Regression Test Prioritization Techniques," *Informatica (Slovenia)*, vol. 36, No. 4, 2012, pp. 379-408.

[10] B. Korel, L. H. Tahat, and M. Harman. "Test prioritization using system models", in *21st IEEE International Conference on Software Maintenance (ICSM'05)*, 2005, pp. 559-568.

[11] B. Athira and P. Samuel. "Web services regression test case prioritization", in *International Conference on Computer Information Systems and Industrial Management Applications (CISIM)*, 2010, pp. 438-443.

[12] X. Han, H. Zeng, and H. Gao. "A heuristic model-based test prioritization method for regression testing", in *International Symposium on Computer, Consumer and Control (IS3C)*, 2012, pp. 886-889.

[13] D. Kundu, M. Sarma, D. Samanta, and R. Mall, "System testing for object-oriented systems with test case prioritization," *Software Testing, Verification and Reliability*, vol. 19, No. 4, 2009, pp. 297-333.

[14] P. Sapna and H. Mohanty. "Prioritizing Use Cases to aid ordering of Scenarios", in *Third UKSim European Symposium on Computer Modeling and Simulation*, 2009, pp. 136-141.

[15] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson. "Systematic Mapping Studies in Software Engineering", in *12th International Conference on Evaluation and Assessment in Software Engineering (EASE)*, 2008, 8: pp. 68-77.

[16] S. A. Joshi and B. Tiple, "Literature Review of Model Based Test case Prioritization," *International Journal of Computer Science & Information Technologies*, vol. 5, No. 5, 2014.

[17] S. Mohanty, A. A. Acharya, and D. P. Mohapatra, "A survey on model based test case prioritization," *International Journal of Computer Science and Information Technologies*, vol. 2, No. 3, 2011, pp. 1042-1047.

[18] S. Keele, *Guidelines for performing systematic literature reviews in software engineering*, in *Technical report, Ver. 2.3 EBSE Technical Report. EBSE*. 2007, sn.

[19] S. Elbaum, A. G. Malishevsky, and G. Rothermel, *Prioritizing test cases for regression testing*. Vol. 25. 2000: ACM.

[20] B. Korel, G. Koutsogiannakis, and L. H. Tahat. "Application of system models in regression test suite prioritization", in *International Conference on Software Maintenance (ICSM)*, 2008, pp. 247-256.

[21] L. Tahat, B. Korel, G. Koutsogiannakis, and N. Almasri, "State-based models in regression test suite prioritization," *Software Quality Journal*, vol., 2016, pp. 1-40.

[22] B. Korel, G. Koutsogiannakis, and L. H. Tahat. "Model-based test prioritization heuristic methods and their evaluation", in *Proceedings*

of the 3rd international workshop on Advances in model-based testing, 2007, pp. 34-43.

[23] L. Tahat, B. Korel, M. Harman, and H. Ural, "Regression test suite prioritization using system models," *Software Testing, Verification and Reliability*, vol. 22, No. 7, 2012, pp. 481-506.

[24] P. Kaur, P. Bansal, and R. Sibal. "Prioritization of test scenarios derived from UML activity diagram using path complexity", in *Proceedings of the CUBE International Information Technology Conference*, 2012, pp. 355-359.

[25] A. Gantait. "Test case Generation and Prioritization from UML Models", in *Second International Conference on Emerging Applications of Information Technology (EAIT)* 2011, pp. 345-350.

[26] H. Stallbaum, A. Metzger, and K. Pohl. "An automated technique for risk-based test case generation and prioritization", in *Proceedings of the 3rd international workshop on Automation of software test*, 2008, pp. 67-70.

[27] R. K. Swain, V. Panthi, D. P. Mohapatra, and P. K. Behera, "Prioritizing test scenarios from UML communication and activity diagrams," *Innovations in Systems and Software Engineering*, vol. 10, No. 3, 2014, pp. 165-180.

[28] P. Sapna and H. Mohanty. "Prioritization of scenarios based on UML Activity Diagrams", in *First International Conference on Computational Intelligence, Communication Systems and Networks (CICSYN)*, 2009, pp. 271-276.

[29] A. Al-Herz and M. Ahmed. "Model-based web components testing: a prioritization approach", in *International Conference on Software Engineering and Computer Systems*, 2011, pp. 25-40.

[30] F. Belli and N. Gökçe. "Test prioritization at different modeling levels", in *International Conference on Advanced Software Engineering and Its Applications*, 2010, pp. 130-140.

[31] R. C. Bryce, S. Sampath, and A. M. Memon, "Developing a single model and test prioritization strategies for event-driven software," *IEEE Transactions on Software Engineering*, vol. 37, No. 1, 2011, pp. 48-64.

[32] N. GÖKÇE, F. Belli, M. EMİNLİ, and B. T. Dincer, "Model-based test case prioritization using cluster analysis: a soft-computing approach," *Turkish Journal of Electrical Engineering & Computer Sciences*, vol. 23, No. 3, 2015, pp. 623-640.

[33] S. Mohanty, A. A. Acharya, and D. P. Mohapatra. "A model based prioritization technique for component based software retesting using uml state chart diagram", in *3rd International Conference on Electronics Computer Technology (ICECT)*, 2011, 2: pp. 364-368.

[34] F. M. Nejad, R. Akbari, and M. M. Dejam. "Using memetic algorithms for test case prioritization in model based software testing", in *1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC)*, 2016, pp. 142-147.

[35] C. R. Panigrahi and R. Mall, "Model-based regression test case prioritization," *ACM SIGSOFT Software Engineering Notes*, vol. 35, No. 6, 2010, pp. 1-7.

[36] 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, 2013, pp. 159-173.

[37] 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, 2014, pp. 155-163.

[38] E. J. Rapos and J. Dingel. "Using Fuzzy Logic and Symbolic Execution to Prioritize UML-RT Test Cases", in *8th International Conference on Software Testing, Verification and Validation (ICST)*, 2015, pp. 1-10.

[39] C. Sharma, S. Sabharwal, and R. Sibal, "Applying genetic algorithm for prioritization of test case scenarios derived from UML diagrams," *International Journal of Computer Science*, vol. 8, No. 3, 2014, pp. 433-444.

[40] S. Sharma and A. Singh, "Model Based Test Case Prioritization Using Greedy Approach," *International Journal of Emerging Trends in Engineering and Development*, vol. 6, No. 5, 2016, pp. 80-88.

[41] R. S. Silva Filho, C. J. Budnik, W. M. Hasling, M. McKenna, and R. Subramanyan. "Supporting concern-based regression testing and prioritization in a model-driven environment", in *34th Annual Computer Software and Applications Conference Workshops (COMPSACW)*, 2010, pp. 323-328.

[42] X. Wang, X. Jiang, and H. Shi. "Prioritization of test scenarios using hybrid genetic algorithm based on UML activity diagram", in *6th International Conference on Software Engineering and Service Science (ICSESS)*, 2015, pp. 854-857.

[43] S. Weißleder. "Towards Impact Analysis of Test Goal Prioritization on the Efficient Execution of Automatically Generated Test Suites Based on State Machines", in *11th International Conference on Quality Software*, 2011, pp. 150-155.

[44] F. Basanieri, A. Bertolino, and E. Marchetti. "The cow_suite approach to planning and deriving test suites in UML projects", in *International Conference on the Unified Modeling Language*, 2002, pp. 383-397.

[45] S. S. Emam and J. Miller, "Test case prioritization using extended digraphs," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 25, No. 1, 2015, pp. 6.

[46] S. Sabharwal, R. Sibal, and C. Sharma. "Prioritization of test case scenarios derived from activity diagram using genetic algorithm", in *International Conference on Computer and Communication Technology (ICCCT)*, 2010, pp. 481-485.

[47] E. J. Rapos, *Understanding the effects of model evolution through incremental test case generation for UML-RT models*. 2012: Queen's University (Canada).

[48] L. E. Baum, T. Petrie, G. Soules, and N. Weiss, "A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains," *The annals of mathematical statistics*, vol. 41, No. 1, 1970, pp. 164-171.

[49] M. Utting, A. Pretschner, and B. Legeard, "A taxonomy of model-based testing approaches," *Software Testing, Verification and Reliability*, vol. 22, No. 5, 2012, pp. 297-312.

APPENDIX A

Table A1: Full List of Approaches for Model-based TCP.

Approach Name	Description
Model Modification Based (MMB)	
Model Dependence-based * [11]	1. Identify differences between original and modified model 2. Identify and mark data and control dependences during modified model execution 3. Identify most promising paths and prioritize test cases that cover those particular paths
Improved Heuristic * [12]	1. Find the test case executing largest number of modified transitions and execute it 2. Update the counts of modified transitions traversed during execution of previous test case 3. Get set of modified transitions executed least number of times 4. Select a random transition in the set then select and execute a test case that covers the transition
Selective Test Prioritization Version 1 [10]	Assign high priority to tests that execute modified transitions (only added transitions)
Selective Test Prioritization Version 2 [10]	Assign high priority to tests that execute modified transitions (added and deleted transitions)
Model Dependence-based Test Prioritization [10]	1. Uses model dependence analysis to identify different ways in which added and deleted transitions interact with the remaining parts of the model and use this information to prioritize high priority tests 2. Weakness – Too complex, computing interaction pattern test distribution
Heuristic #1 * [22]	1. Give higher priority to tests executing higher number of modified transitions 2. H#1 order prioritization based on number of modified transition, different from Selective Test Prioritization that only has high priority set and low priority set 3. Weakness – considering only the number of transition execution may not have significant influence on improving early fault detection
Heuristic #2 * [22]	Modified version of H#1, give more chances to lower priority tests to be selected
Heuristic #3 * [22]	1. Tests with higher frequency of execution of modified transitions given higher priority than tests with lower frequency of execution of modified transitions 2. Weakness – transition frequencies may not be the best type of info to be used for prioritization
Heuristic #4 * [22]	Modified version of H#3, give more chances to lower priority tests to be selected
Heuristic #5 * [22]	1. Each modified transition should have same opportunity to be executed 2. Balance number of executions of modified model 3. Steps a. Keeps counts of transitions executions b. Randomly select first test to be executed, the count of transitions executed on the test are updated c. Determine set of modified transition executed least number of times, randomly select one and execute the test that traverses that selected transition then update the count of transitions executed on that test d. Repeated until transition counts are the same
Model-based Regression Test Case Prioritization technique (M-RTP) [35]	1. Compare Extended Object-oriented System Dependence Graph (EOSDG) models for original and modified model 2. Mark the model with the identified changes 3. Construct forward slicing to identify all model elements affected by modification 4. Find test cases associated with affected elements

	5. Prioritize according to number of affected elements covered by test case
Slice based-Regression Test Prioritization (S-RTP) [36]	1. Steps a. Construct Extended System Dependence Graph (ESDG) model of original program b. Identify changes between original and modified program c. Update ESDG model correspond to modified program d. Construct forward slice of ESDG using modified nodes e. Determine nodes affected to modification f. Find test cases that cover affected node and prioritize based on number of affected node covered 2. Weakness – Less efficient in program with low interdependency
Heuristic based-Regression Test Prioritization (H-RTP) [37]	1. Same with S-RTP 2. Difference – weight is assigned to each modified node and decreased every time the corresponding test case is selected into prioritization 3. Weakness – assume all test cases have equal cost and faults are equally severe
Concern-based * [41]	1. Two options a. Reorganize test procedures based on number of steps originated in modified elements in the model – allow test procedures that cover the highest number of modified elements to be executed first b. User can define individual risk for each activity, prioritize those with higher risks
Weight/Complexity/Priority/Risk Assignment Based (WAB)	
Minimum Independent Dominating Set Method (MIDSM) [29]	1. Steps a. Choose node with the highest degree b. Delete all neighbors c. Choose next highest degree node d. Delete neighbors 2. Weakness a. Which node to select when more than one with the same degree b. Not considering importance of the direction that may impact component importance c. Might delete important neighbor component
Degree Measure Method (DMM) [29]	1. Rank components based on fan-in degree Weakness - Which component to select when more than one with same fan-in degree
Betweenness Measure Method (BMM) [29]	1. Steps a. Find shortest paths between any components pairs b. Go over all individual components and see which paths they exist c. Highest priority to components with most existence 2. Weakness a. Which component to select when more than one with same betweenness measure
Branch Probability * [25]	1. Steps a. Find minimum number of flows covering all edges b. If there is more than one combination of activity flows to cover the activity diagram, take the highest weight of flow to reduce TC. c. Weight of an only one outgoing edge in a node is 1 d. Weight of *n* outgoing edges in a node is equal to 1 e. Weight of a flow is the product of all the weights of the edges in the flow 2. Weakness a. Prioritization based on possibility of usage, may not be feasible at runtime
Path complexity * [24]	1. Convert activity diagram to control flow graph (CFG) 2. Basis path generation 3. Count number of nodes in each path

	4. Calculate weight of each path by calculating total weight of all the path's nodes using Information Flow (IF) model 5. Count number of predicate nodes in each path 6. Count number of logical conditions within each predicate node covered by the corresponding path 7. Calculate each path complexity by summing the four parameters values (Number of nodes, weight of path, number of predicate nodes traversed and number of logical conditions traversed) 3. Path with the highest weight will be the highest complexity, therefore, assigned highest priority
System Testing for Object-Oriented systems with test case Prioritization (STOOP) [13]	1. Steps a. Converts a set of sequence diagrams into graph representation, sequence graph (SG) b. Generate test cases from SG c. Prioritize test cases using Sum of Message weight, Average weighted path length, and Code weight d. Take ranked test cases and generate test data 8. Weakness – only consider sequence diagrams for one use case at a time
Component Interaction Graph * [33]	1. Use statechart diagram to model each component and construct CIG 2. Count max state changes and max database access of each test case 3. Calculate the objective function value 2. Prioritize test case according to decreasing value of objective function
Tree Structure * [28]	1. Steps a. Convert activity diagram to tree structure b. Assign weights to nodes and edges c. Calculate path(scenario) weight d. Prioritize scenarios 4. Weakness – Depend solely on structural aspects of the activity diagram
Structural Aspects of Use Case & Activity Diagram * [14]	1. Steps a. Capture data from all use case diagrams b. Obtain actor priority and compute use case priority from use case diagram c. Obtain customer prioritization of use cases d. Calculate UC priority by summing Customer Priority and Technical Priority e. Extract scenarios from activity diagram which is elaborated from use case f. Prioritize scenarios by assigning weights to nodes and edges in activity diagram g. Calculate weight of path (scenario) then finally prioritize by summing the sum of the priorities starting at level 1 of the schema and moving down adding the weights of all the nodes up to the scenario weight 2. Weakness a. Depend on correctness and completeness of UC an AD b. Scenarios will not be generated if requirement not captured in AD
Greedy Approach * [40]	1. Traverse the test suite for each TC 2. Calculate number of intrastate and interstate covered by each TC 3. Define unit time 4. Calculate objective function of each TC 3. Prioritize TC by sorting in descending order of objective function
Risk-based test case Derivation and Prioritization (RiteDAP) [26]	1. Derive unordered test case scenarios (TCS) from test model 2. Calculate sum of risks of all actions covered by TCS 5. Order TCS based on value of risk using Total Risk Score Prioritization (TRSP) or Additional Risk Score Prioritization (ARSP)
Prioritizing Test Scenarios through COMMACT tree (PRITECOMMACT) [27]	1. Convert communication diagram and activity diagram to testing flow tree (ComTree & ActTree) 2. Merge to COMMACT tree and traverse through it 3. Generate test scenarios

	3. Prioritize the scenarios by calculating weights of nodes and edges and based on main or alternate scenario
Far Element First/Last (FEF/FEL) [43]	4. Sorts test goals according to the referenced model element's distance in descending/ascending order
High Branching Factor First/Last (HBFF/HBFL) [43]	5. Sorts all test goals according to the branching factor of the referenced model element in descending/ascending order
Many Atomic Conditions First/Last (MACF/MACL) [43]	6. Sorts all test goals according to the number of atomic conditions in descending/ascending order
High Positive Assignment Ratio First/Last (HPARF/HPARL) [43]	7. Sorts all test goals according to their positive assignment ratio in descending/ascending order
COWtest plus UIT Environment (COW_SUITE) [44]	1. Identify & organize the graphs representing the design model structure 2. Trees derivation 3. Assign weights to the nodes 4. Integration stage selection & weighted tree derivation 8. Cowtest_ing
<u>Genetic Algorithm Based (GAB)</u>	
Memetic Algorithm * [34]	1. Combination of GA algorithm and a local search algorithm (stochastic local search, hill climbing, random iterative improvement or simulated annealing) 2. Steps a. Convert activity diagram to Control Flow Graph (CFG) b. Use fitness function to compute paths value in AD Use memetic algorithm to prioritize test case
Combination of Basic IF metric, Stack & GA * [39]	1. Convert activity diagram to control flow graph (CFG) and statechart diagram to state dependency graph (SDG) 2. Assign weight to nodes in CFG and SDG using stack-based weight and basic IF model 3. Selection – form chromosome using decision nodes of CFG and SDG 4. Crossover - swap genes or sequence of bits in the chromosome 3. Mutation – bring diversity in population to avoid local optima
Hybrid Genetic Algorithm (GA +PSO) [42]	1. Convert AD to CFG 2. Generate all possible independent & non-redundant paths using Depth First Search (DFS) method 5. Find fittest test path using HGA
Combination of Basic IF metric & GA * [46]	1. Convert AD to CFG 2. Assign weight to nodes using FAN-IN & FAN-OUT 3. Selection – Turn decision nodes of CFG into chromosomes 4. Crossover - swap genes or sequence of bits in the chromosome 5. Mutation – bring diversity in population to avoid local optima
<u>Fuzzy Logic Based (FLB)</u>	
Gustafson Kessel Clustering * [30]	1. Steps a. Construct set of events b. Cluster event using GK clustering c. Classify event into *c* fuzzy groups d. Determine importance degrees of groups e. Determine importance index of event groups f. Order Complete Event Sequence (CES) as test cases using preference degree 2. Weakness a. Heavily depends on model used b. A model usually focuses on selected features; result may differ for other features
Cluster Analysis (FL + NN) * [32]	1. Steps a. Construct set of events b. Cluster the events using both Adaptive Competitive Learning (ACL) and Fuzzy C-Means (FCM)

	c. Classify events into c crisp groups (using ACL) & fuzzy qualified groups (using FCM) d. Determine importance degrees of groups e. Determine importance index of event groups f. Order CES as test cases using corresponding preference degree 2. Weakness a. Affected by changes in model concerning the generated test sequences b. Only behavioral sequence-based faults are revealed, logical and/or calculation errors are ignored
Fuzzy Control System * [38]	1. Fuzzification & defuzzification 2. Rule selection 3. Presentation of results 3. Implementation
Graphical User Interface Based (GUIB)	
1-way Parameter-value Interaction Coverage-based [31]	Select next test to maximize the number of parameters values that don't appear in previously selected tests.
2-way Parameter-value Interaction Coverage-based [31]	Select next test to maximize the number of 2-way parameters value interaction between windows
Unique Window Coverage Count-based [31]	Prioritize tests by giving preference to test cases that cover unique windows that previous tests have not covered
Action Count-based [31]	Prioritize tests by number of actions in each test (duplicates included)
Parameter-Value Count-based [31]	Prioritize tests by number of parameters that are set to values in a test case (duplicates included)
Most Frequently Present Sequence (MFPS) of Windows Frequency-based [31]	1. Identify most frequently present sequence of windows in the test suite 2. Order test cases in decreasing order of number of times that particular sequence appear in test cases
All Present Sequence (APS) of Windows Frequency-based [31]	The frequency of occurrence of all sequences is used to order test suite
Weighted Sequence of Windows (Weighted-Freq) Frequency-based [31]	Assign each test case-weighted value based on all of the windows sequences it contains and the importance of the window sequence (window sequence importance is calculated by number of times the sequence appears in suite)
Accumulated Q-value [45]	1. Prioritize test cases based upon the number of computations activated by the corresponding action in each test case (Q-values) 2. Rank every test case in descending order, then label the one with the highest accumulated Q-value as highest priority
Probability-Based (PB)	
Reinforcement Learning & Hidden Markov Model (RL-based HMM) [45]	1. Determine initial RL-based HMM parameters based on the generated test cases from MBT techniques 2. Train an RL-based HMM with maximum likelihood using Baum-Welch algorithm 3. Computes TC's forward probabilities using forward algorithm and prioritize test cases

Table A2: Overview of Selected Studies.

No.	Type of Paper	Publication Year	Publication Medium	Reference
1.	Journal	2016	Software Quality Journal	[21]
2.	Journal	2016	International Journal of Emerging Trends in Engineering and Development	[40]
3.	Journal	2015	Turkish Journal of Electrical Engineering & Computer Sciences	[32]
4.	Journal	2015	ACM Transactions on Software Engineering and Methodology	[45]

5.	Journal	2014	Innovations in Systems and Software Engineering	[27]
6.	Journal	2014	International Journal of Computer Science	[39]
7.	Journal	2014	Innovations in Systems and Software Engineering	[37]
8.	Journal	2013	CSI transactions on ICT	[36]
9.	Journal	2012	Software Testing, Verification and Reliability	[23]
10.	Journal	2011	IEEE Transactions on Software Engineering	[31]
11.	Journal	2010	ACM SIGSOFT Software Engineering Notes	[35]
12.	Journal	2009	Software Testing, Verification and Reliability	[13]
13.	Conference	2016	1st Conference on Swarm Intelligence and Evolutionary Computation	[34]
14.	Conference	2015	6th IEEE International Conference on Software Engineering and Service Science	[42]
15.	Conference	2015	IEEE 8th International Conference on Software Testing, Verification and Validation	[38]
16.	Conference	2012	CUBE International Information Technology Conference	[24]
17.	Conference	2011	11th International Conference on Quality Software	[43]
18.	Conference	2011	3rd International Conference on Electronics Computer Technology	[33]
19.	Conference	2011	Second International Conference on Emerging Applications of Information Technology	[25]
20.	Conference	2011	International Conference on Software Engineering and Computer Systems	[29]
21.	Conference	2010	International Conference on Computer and Communication Technology	[46]
22.	Conference	2010	International Conference on Advanced Software Engineering and Its Applications	[30]
23.	Conference	2010	International Conference on Computer Information Systems and Industrial Management Applications	[11]
24.	Conference	2009	First International Conference on Computational Intelligence, Communication Systems and Networks	[28]
25.	Conference	2008	IEEE International Conference on Software Maintenance	[20]
26.	Conference	2005	21st IEEE International Conference on Software Maintenance	[10]
27.	Conference	2002	International Conference on the Unified Modeling Language	[44]
28.	Symposium	2012	International Symposium on Computer, Consumer and Control	[12]
29.	Symposium	2009	Third UKSim European Symposium on Computer Modeling and Simulation	[14]
30.	Workshop	2010	IEEE 34th Annual Computer Software and Applications Conference Workshops	[41]
31.	Workshop	2008	3rd International Workshop on Automation of Software Test	[26]
32.	Workshop	2007	3rd international workshop on Advances in model-based testing	[22]