A Systematic Literature Review on Regression Test Case Prioritization

Ani Rahmani[1], Sabrina Ahmad[2]*
Intan Ermahani A. Jalil[3]
Fakulti Teknologi Maklumat dan Komunikasi
Universiti Teknikal Malaysia Melaka
Melaka, Malaysia

Adhitia Putra Herawan[4]
Tokopedia Indonesia
Jakarta, Indonesia

***Abstract*—Test case prioritization (TCP) is deemed valid to improve testing efficiency, especially in regression testing, as retest all is costly. The TCP schedule the test case execution order to detect bugs faster. For such benefit, test case prioritization has been intensively studied. This paper reviews the development of TCP for regression testing with 48 papers from 2017 to 2020. In this paper, we present four critical surveys. First is the development of approaches and techniques in regression TCP studies, second is the identification of software under test (SUT) variations used in TCP studies, third is the trend of metrics used to measure the TCP studies effectiveness, and fourth is the state-of-the-art of requirements-based TCP. Furthermore, we discuss development opportunities and potential future directions on regression TCP. Our review provides evidence that TCP has increasing interests. We also discovered that requirement-based utilization would help to prepare test cases earlier to improve TCP effectiveness.**

***Keywords*—Software testing; test case prioritization; regression testing; requirements-based test case prioritization; software engineering**

I. Introduction

Software testing is a significant stage to confirm the quality of the software before it is released. Particularly in the software maintenance process, the study [1] demonstrated that the cost of testing implementation could reach 80% of the total maintenance costs. Therefore, further efforts are needed to reduce execution time in the testing process.

In the iterative-incremental process and the era of agile software development, new functions are increased by a short cycle [2]. Thereby, software development is also a process that is carried out continuously because of adding user needs. When there are changes in the software, new errors might appear. This situation will disrupt the previous stable system [3], [4]. For this reason, regression testing (RT) is needed, because it will verify the software to find the impact of changes to ensure its continued quality.

One of the popular techniques in RT is test case prioritization (TCP). This technique will order test cases in the test suite so that the testing execution will process the test cases with the most potential to find errors. The advantage of TCP implementation is that even if the testing process must be stopped for certain reason, the most significant errors have already been found. According to [5], there are two essential aspects of building TCP: determining the TCP approach and the technique to optimizing the TCP implementation.

In the past years, TCP studies gained significant attention and achievements to improve regression testing effectiveness. The study [6] emphasized that the researchers focus on five aspects: coverage criteria, algorithms, practical concerns involved, measurement techniques, and scenario to implement the technique. On the other side, studies [7], [8] explained that most of research efforts used source code as input resources to obtain the maximum number of faults within a certain period. Utilization of the code information is best applied to unit-level or block-level tests. Therefore, these efforts have limitations when applied to large systems since statements and block levels in source code will be challenging to manage [9], [10]. Utilizing code information will be expensive to implement because the tester must read and understand the source code, and this will take a long time.

Besides code-based, other TCP approaches have also been developed. According to a study [11], since a system is built from many requirements, the use of information from the requirements can increase error discovery. For this reason, some researchers argue it is essential to develop requirements-based TCP, while the studies in this area are still limited.

Therefore, the paper's main objective is to investigate TCP research's state of the art, emphasizing requirements-based TCP. The expected contributions of this study are:

1) To provide an overview of TCP developments in the years range from 2017 to 2020. We intend to highlight requirements-based TCP as one of the TCP approaches worth considering, and as far as we are concern, this is the first review on requirements-based TCP.

2) To present the variations of the TCP approaches and techniques explored so far, the diversity of software under test (SUT) used as an object for empirical evaluation, and the variation of metrics utilization to measure the TCP effectiveness. The results will be helpful to form a basis for future requirements-based TCP research.

Although there have been many studies in the form of TCP surveys, literature review, or mapping, each research has a different emphasis and perspective. In this regard, we have reviewed 48 credible papers from reputable journals and proceedings. Section II explains this in more detail.

*Corresponding Author

This paper is presented in stages, starting by looking at RT in general, followed by a study of TCP, and finally exploring the requirement-based TCP. Following the introduction, Section II presents the motivation and related work in RT and TCP. Section III describes the SLR method, including threats to validity and Section IV presents results and discussion. Subsequently, Section V describes the research findings, and Section VI offers future work and conclusions.

II. Motivation and Related Work

In this section, we explain the motivation and related work of the study conducted.

A. The Motivation

The ideal implementation of RT is to "retest all" or execute all test cases. However, in practice, not all test cases will be retested, especially those implementing RT manually. Several RT practice personal intuition based on experience, and even randomly [12]. Complete testing is complicated, and even worst, in several cases testing needs to be stopped. This condition causes other RT implementation problems, such as an error in the execution of the test case sequencing. On top of that, the RT process may be prolonged, or it may also run out of time. Studies [12], [13] stated that these approaches are inefficient and require high costs.

In many cases, RT is performed in high-pressure situations since testing execution requires a very long time. For example, the testing process conducted in an industry takes up to seven weeks to program with 20,000 lines [14]. In another case, Google has reported that there are more than 20 code changes every minute and that there is a change of 50% of files per month, resulting in a very long execution [15], [16]. The other example is a software development product with up to 30,000 functional test cases that need over 1000 hours. Besides, engineers need hundreds of hours to oversee the implementation of regression testing, supervise tests, monitor test results, and maintain test cases, oracles, and everything needed to support automated testing. Therefore, the study [17] concluded that RT is costly due to thousands of effort hours.

It is then understandably if several researchers emphasize that the common problem in RT is time constraint or insufficient [8], [18]–[20]. Through various surveys, research in the RT field will continue to grow, with the increasingly diverse types of approach or a broader application domain, for more effective methods.

RT techniques are divided into three types [2], [21]: regression test minimizing (RTM), regression test selection (RTS), and regression test prioritization (RTP), or also known as test case prioritization (TCP). A study [22] summarizes the comparison of the three techniques which are presented in Table I.

TABLE I. THE COMPARISON OF REGRESSION TESTING APPROACH [22]

Component	Regression Test Approaches		
	Minimizing (RTM)	Selection (RTS)	Prioritization (TCP)
Strategy	Eliminate test case	Modification aware test case	Test case permutation by ordereing and prioritizing
Strength	Effective in reducing test case	Effective in selecting modification-aware test cases	Usefull when new test case will always be considered in the test case permutation
Limitation	Test case are not modification-aware	New test cases might be missed out in the temporary selection that is modification-aware	Time consuming, larger test-suuite

RTM reduces test cases by removing many test cases for a particular reason, such as redundant ones. Meanwhile, RTS selects test cases that can potentially find errors. The selection process refers to specific criteria. Both RTM and RTS will permanently remove some test cases from the test suite. Unlike RTS and RTM techniques, TCP does not remove test cases but orders the test cases according to the criteria. The test case with the most potential to find an error in the program will have a higher priority and be executed earlier.

B. Related Work

Some surveys or reviews have been conducted on RT and the TCP techniques. This section describes the study, SLR, and mapping obtained from many digital libraries in 2010-2021 range.

Regression testing survey is available in several studies [2], [16], [21]. The study [2] surveyed RT in the scope of the technical side, metrics, strategy, software under test (SUT), and an overview of the optimization technique in the form of automation, or using a traditional approach. Meanwhile, the study [16] described the techniques and advantages of all three types of regression testing. Study [21] on the other hand, reviewed articles with the most extended ranges from 1977 to 2009. This study discussed the approaches and techniques covering test case minimizing effort, test case selection, and TCP in great detail.

The specific survey on TCP was performed in [5], [22], [23] [24], [25], [26], 33], [27], [28], and [29]. Survey [22] and [23] are two very detailed surveys and have been cited by many TCP researchers to date. The study [22] reviewed 80 articles from 1999 to 2016, while [23] reviewed 65 papers from 1997 to 2011. Generally, the aspects explored in the two studies include approaches and techniques on TCP, metrics, and software under test (SUT).

In analysing TCP, study [5] explained that there are two approaches to categorize TCP implementation: input resources (the information sources for the TCP process) and optimization strategies (methods or algorithms for executing the TCP technique). This study classified the TCP approaches and the TCP optimization strategies according to these two categories. This method is a more straightforward step to facilitate TCP classification. In measuring the TCP effectiveness, this study proposes another view of the metric used by many researchers, which is the average percentage error detection (APFD). However, APFD has limitations because it treats all test cases as having the same weight.

A survey [24] has mapped and reviewed 108 articles from 1999 to 2016. The author mapped article content into several aspects: the place of publication, the number of articles on the approach, and the number of metrics. Furthermore, the review includes the use of tools, the TCP effectiveness for each study investigated, the analysis of APFD factors, and a review of APFD in some SUT applications.

The model-based TCP has been studied [25] which reviewed 32 articles from 2005 to 2016. The authors classified the TCP models based on approaches, their characteristics, and how they can overcome obstacles in TCP implementation using model-based as an input resource.

The study conducted by Mukherjee and Patnaik [26] surveyed 90 TCP articles from 2001 to 2018. The purpose of the survey is to investigate several aspects: TCP Metric, the program or SUT, and identify the TCP method commonly used. This study concludes three essential perspectives: 1) the APFD metric is the most extensive to measure the effectiveness of TCP, 2) the program in the SIR repository is the most widely used as SUT, and 3) the coverage-aware, requirements-based, and model-based are the three approaches that are getting more attention, currently.

In 2019, Lio et al. [30] surveyed 191 articles on TCP published in the 1997 to 2016 range. They analyzed TCP trends based on six categories: constraints, algorithms, criteria, measurements, scenarios, and empirical studies. In addition to this, they highlighted several improvements during the development of test cases in 2004–2005, 2008–2009, and 2014–2015. They analyzed the trends of the period from various points of view as a basis. More specifically, the analysis was related to the emergence of technologies that allow online repositories to host software projects.

Meanwhile, a study [27] have reviewed TCP trends from 2017-2019. An essential aspect of this study is to answer whether the taxonomy proposed in the previous study [22] is still valid. This study further suggests other approaches: location-based, machine learning-based, neural network-based, and empirical, which are empirical studies of TCP in certain domains, with specific guidelines or software.

Recently, two more literature reviews on TCP are published in 2021. Samad et al. [28] reviewed TCP in general, and Hasnain et al. [29] specifically reviewed TCP's functional requirements. Samad et al. reviewed 52 TCP articles in the 2007-2020 range. Like most studies on regression testing and, in particular, TCP, the RQ proposed in this study is a state-of-the-art of TCP technique, parameters, dataset or object software used, and metrics to verify TCP techniques. The parameters used in the study include cost, code coverage, and fault detection ability.

The study conducted by Hasnain et al. [29] focuses on TCP studies that utilize the functional requirements approach, with 35 article from 2009 to 2019. The study answered 7 RQs: state-of-the-art regarding functional requirements-based TCP, the key factors discussed in the TCP requirements-based study, the essential aspects considered for proposing the TCP approach, the crucial issues addressed in the TCP functional-requirements study, test case size and type of defect, metrics used, software under test (SUT), and whether these studies can be applied in the real world or not.

There are five surveys on both RT and TCP for specific purposes. The study [15] reviewed the trend of the TCP approach in web applications and analysed the qualitative assessment of web applications. The analysis was carried out on three web application sizes: small, medium, and large, and was analysed from two categories: simple and complex web applications. Meanwhile, a study has been conducted [31] to map the regression testing applications on web services. The mapping aims to identify gaps between existing studies and the future studies in each article reviewed. The study mapped several things: stakeholders, SUT and related standards, validation methods, and web services, as well as mapping to validation services.

Moreover, to review the use of TCP techniques in web services, a study [32] has identified statistical methods, metrics to validate the proposed technique, and issues relating to current TCP concerning web services. Furthermore, a study [33] reviewed the scope of TCP's application for continuous interaction (TCPCI) environment. Some important aspects were analysed, including problems in continuous integration (CI), sources of information (input resources) for TCP in TCPCI, evaluating measures using metrics in TCP, and analysis of research opportunities to guide future research.

A study by [34] analysed 98 articles to support the research. The authors analysed and mapped several aspects, including the techniques and the efforts to improve the test's scope. The authors also construct a taxonomy that allows researchers to consider the relevance and applicability of regression testing to specific industries. Table II presents the secondary studies, whether in the form of SLRs, surveys, or mapping, from 2010 to 2020, grouped by RT, TCP technique, and RT or TCP for specific purposes.

TABLE II. SECONDARY STUDIES IN REGRESSION TESTING (RT) AND TEST CASE PRIORITIZATION(TCP)

	#Study	Publication Year	Type of Studied	Year Coverage	#of Primary Studies	Other Information
RT	[21]	2010	Survey	1977-2009	159	
	[16]	2016	Survey	-	-	
	[2]	2016	Survey	2000-2014	25	
TCP	[23]	2012	SLR	1997-2011	65	
	[24]	2017	Mapping	1999-2016	108	
	[22]	2017	SLR	1999-2016	80	
	[5]	2018	Survey	-	-	
	[26]	2018	Survey	2001-2018	90	TCP approaches
	[25]	2018	SLR	2005-2016	32	Model-based TCP
	[30]	2019	Survey	1997-2016	191	
	[27]	2020	SLR	2017-2019	320	
	[28]	2021	SLR	2007-2020	52	
	[29]	2021	SLR	2009 to 2019	35	Functional requirement-based
RT / TCP for Specific Purpose	[31]	2014	Mapping	2000-2013	30	RT for Web Service
	[15]	2015	SLR	1995-2014	64	RT for Web Appl.
	[34]	2019	SLR	x-2016	98	RT in Industry-relevant
	[33]	2020	Mapping	1979-2020	35	TCP in Continuous Integration
	[32]	2020	SLR	2001-2017	65	TCP for Web Service

III. REVIEW METHOD

We adopted a Systematic Literature Review (SLR) strategy [35] as a method. SLR is a research method for conducting a literature review with systematic and regular steps. According to the method, Table III presents three stages of review: the initial or planning stage, the selection and review process, and the reporting of the resulting process.

A. *Research Question*

The research questions (RQs) are intended to find the techniques, approaches, and empirical experiences from many researchers to formulate an efficient way to process regression testing using TCP techniques and requirement-based TCP. The results of the SLR must be able to answer several questions in Table IV.

B. *Selecting and Review Process*

This section explains several stages of activities in implementing the SLR.

1) Literature resources: The articles used in this study are taken only from journals and proceeding. We selected the most common and influential database sources and the ones most widely used by researchers, as listed below:

a) IEEE Xplore
b) Science Direct
c) Springer
d) Semantic Scholar
e) Google Scholar

TABLE III. SYSTEMATIC LITERATURE REVIEW STAGE

SLR Phase	Steps
Planning	Formulating the research questions
Selecting and Review	Determining the data sources
	Determining search strings/keyword
	Applying inclusion and exclusion criteria
	Selecting, classifying, and analyzing the references.
Reporting	Presenting the SLR result

TABLE IV. LIST OF RESEARCH QUESTIONS

#RQ	Research Questions	Motivations
RQ1	What is state of the art for TCP in regression testing based on TCP approaches and techniques?	To discover the development of approaches in TCP study
RQ2	What is the software under test (SUTs) in TCP studies?	To identify the variation of SUTs in TCP studies. This will be useful for researchers to prepare the SUT carefully.
RQ3	What is the trend of metrics to measure TCP effectiveness?	To provide insight into how the effectiveness of approaches or techniques is measured.
RQ4	What is the state of the art of requirement-based TCP in literature?	To explore techniques or approaches studied in the requirements-based TCP.

2) Search string criteria: We formulated a string for the search process considering its relevance to the research question. Sometimes we used several words by combining them into a query for words with similar meanings, such as "technique," "approach," or "strategy." Furthermore, to emphasize a string, quotes are also used in a phrase, such as "regression testing" or "test case," so that search results can be more specific. The keywords for the query search string used are: "test case" AND (prioritization OR prioritize) AND (approach OR technique OR strategies) AND "regression testing."

3) Inclusion/Exclusion criteria: The next stage is selecting articles based on inclusion criteria (ICs) and exclusion criteria (ECs). Table V explains four inclusion and exclusion criteria.

4) Selection and quality assessment: We decided to choose papers published started in 2017 to answer the Research Questions. The reason is because studies conducted from 1999 to 2016 [22] and from 1997 to 2011[23] have been in detail reviewed, and researchers to date have widely cited the results.

Using the query stated in sub-section 3.2.2, we discovered 501 papers in the primary studies from various databases. These papers are published in both journals and proceedings. Next, we selected the papers using the inclusion and exclusion criteria as presented in Table V, resulted in 122 papers being selected. Furthermore, we conducted a quality assessment based on the following five parameters:

a) The objectives are clearly described.

b) The article clearly states the used approach or technique.

c) There is sufficient information about the software under test (SUT) as a research object.

d) The research design is appropriate to answer the research question.

e) Conclusions are stated clearly and measurably using one or more metrics.

The five above parameters must be "true," otherwise the paper will be excluded to obtain the expected quality. At this selection stage, 48 papers were finally listed. Fig. 1 describes the process of sources search and selection.

5) Data extraction process: The data extraction stage aims to collect data from selected papers, which is done by extracting information to answer the research question (RQs) defined. Table VI is a list of extraction parameters along with the research question to be answered.

C. Threats to Validity

There is a risk of threats to validity in the review survey even though careful measure has been taken care of throughout the survey. In this survey, there are two threats of validity as listed below:

1) There may be missing credible sources, which is beyond our knowledge. To minimize the threat, we search from the most common and influential database sources.

2) There is a possibility there exist relevant studies but are not captured by the keywords due to the differences in terminology and mentions. For this matter, we have searched, and test various search string combinations as stated in Section III.B.2.

TABLE V. LIST OF INCLUSION: EXCLUSION CRITERIA

#ICs	Inclusion Criteria	#ECs	Exclusion Criteria
IC1	The document selected is an article from a journal or proceeding	EC1	Lecture note, book chapter
IC2	The articles taken are those related to the focus study in this research, whether explicitly proposing new approaches/techniques, or studies that examine the effectiveness of a technique, through comparisons, or empirical studies	EC2	Articles that discuss in the form of an overview of these concepts
IC3	The articles published 2017-2020	EC3	The articles published outside of 2017-2020
IC4	The articles written in English	EC4	The articles in languages other than English

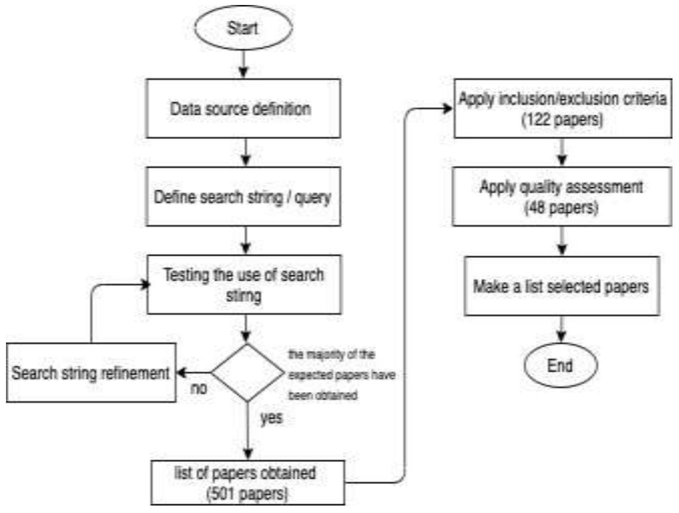
Start
Data source definition
Define search string / query
Testing the use of search string
Search string refinement
no
the majority of the expected papers have been obtained
yes
list of papers obtained (501 papers)
Apply inclusion/exclusion criteria (122 papers)
Apply quality assessment (48 papers)
Make a list selected papers
End

Fig. 1. Search and Selection Process.

TABLE VI. THE DATA EXTRACTION PARAMETERS

Research Question	Extraction Parameters
RQ1	TCP approaches and techniques
RQ2	SUT for empirical studies
RQ3	Metric used to measure the TCP effectiveness
RQ4	The strategies to implement the requirements-based TCP

IV. Results and Discussion

This section elaborates on the review results.

A. *Primary Studies Overview*

From the first search 501 articles were obtained from the databases. There are 235 journal articles, and 266 proceedings articles. Fig. 2 shows the distribution of articles obtained from 2017 to 2020. While Fig. 3 presents the comparison of the journal and proceeding in the first-round search.

When taking the inclusion and exclusion criteria into account, 122 papers were shortlisted, as shown in Table VII. Next, we filtered the shortlisted papers using the five quality assessment criteria, and only 48 were finalized (Table VIII). The detailed information of selected articles can be found in the Appendix.

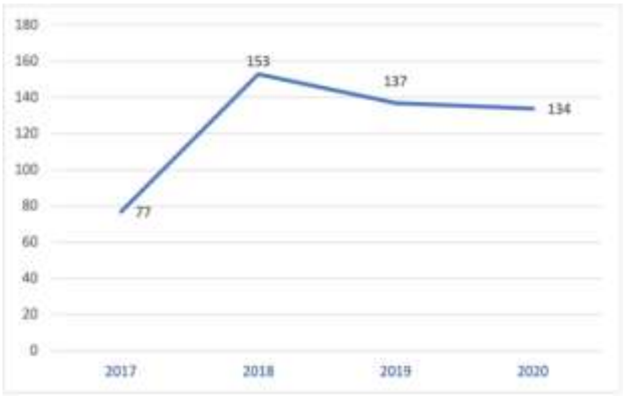

Fig. 2. First Search Results.

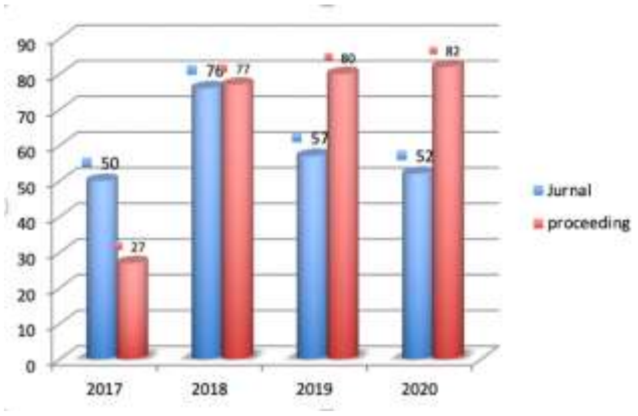

Fig. 3. Journal Articles and Proceedings Distribution.

TABLE VII. Total Articles during the Inclusion / Exclusion Selection

Year of Publication	Total articles selected	First-round	
		Included	Excluded
2020	134	39	95
2019	137	37	100
2018	153	28	125
2017	77	18	59
TOTAL	501	122	379

TABLE VIII. Total Articles during the Quality Assessment

Year	Result of the first round	Second round	
		Included	Excluded
2020	39	12	27
2019	37	16	21
2018	28	11	17
2017	18	9	9
TOTAL	122	48	73

The 48 primary studies consist of 31 journal articles and 17 proceedings, as illustrated in Fig. 4.

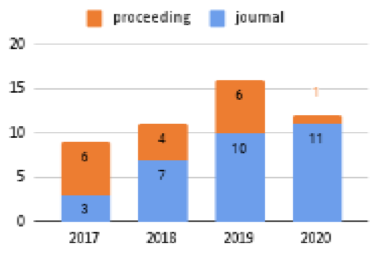

Fig. 4. Selected Papers through Two Rounds Selection.

Next, Fig. 5 shows the selected articles classification based on the origin (journal or proceeding) and quartiles in Scimago indexing. It is shown that 33.3% of articles are from Q1 journals, 18.8% are from Q2 journals, 8.3% are from Q3 journals, and 4.2 % are from Q4 journals.

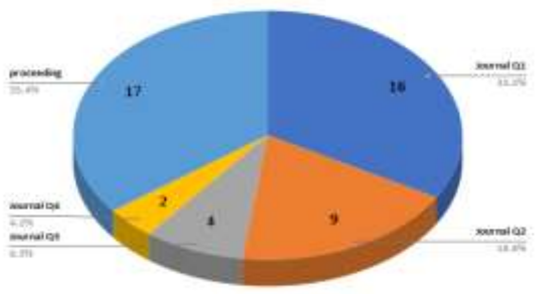

Fig. 5. Selected Papers Sources.

B. *Current Research Efforts to Improve TCP for Regression Testing*

This sub-section responds to RQ1, RQ2, and RQ3.

1) What is The State of the Art for TCP based on TCP Approaches and Techniques? (RQ1): The answer to RQ1 also covers the review done by Khatibsyarbini et al. [22] since it is essential to consider the improvement of TCP research before 2017. The significant discovery is the TCP taxonomy which portrays the regression testing types and some techniques in TCP. Fig. 6 shows the TCP approaches taxonomy proposed by [22] and portrays approaches added by [27]. Four items are added into the initial taxonomy: ' Location-based,' 'Machine-learning based,' 'Neural Network-based,' and 'Empirical.'

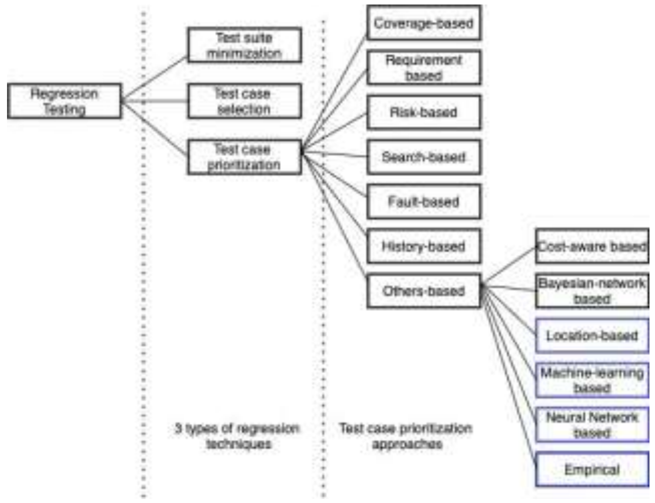

Fig. 6. Taxonomi of TCP Approach (Adapted from [22]).

Table IX presents the approaches in the TCP research during 2017-2020. While Fig. 7 visualizes the approaches distribution in the TCP research, it can be seen that several approaches are gaining popularity as they appeared in several researches.

TABLE IX. APPROACHES IN THE TCP RESEARCH

Approaches	**Research**
Risk based	[36][10]
Search based	[37] [38] [39] [40] [41] [42][43]
Fault based	[44] [45] [46] [47] [48]
Model based	[49][50] [46][51]
Modification based	[52][53][54]
Coverage based	[55] [56] [57] [58] [59] [60]
Similarity based	[61] [62][63] [64] [65][66] [67]
Requirement based	[68][69][70] [13][71]
User Interface based	[72]
History based	[73] [74]
Mutation based	[75]
Hybrid (combining more than 1 method)	[76][20][77] [53] [75]

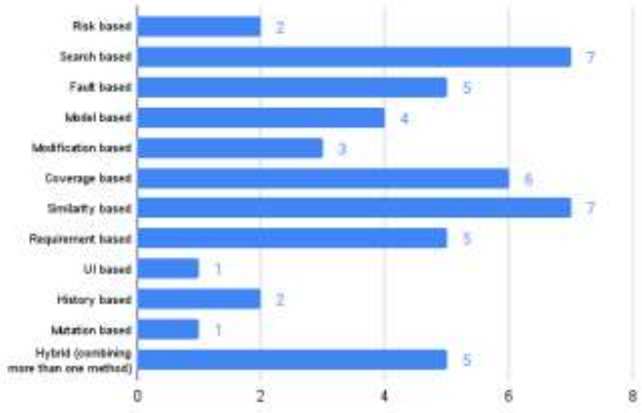

Fig. 7. The Trend of TCP Approaches.

In comparison to the approaches proposed by [22] and [27], we discover several other approaches through our survey: modification-based, user interface-based, model-based, mutation-based, and similarity-based. This discovery shows that researchers are still exploring and improving ways to better TCP by introducing more approaches.

Input resource, technique, and algorithm determination are essential to implement TCP [5]. Referring to the literature, there is no dominant technique or algorithm for implementing TCP. After we identified the TCP approaches, we then identified the techniques that researchers used in their study. Each researcher executes the chosen technique based on specific analysis and considerations. Some of the algorithms used include Greedy and Additional Greedy for search-based TCP [37], [40], Firefly Algorithm [38], [78], Neural Network Classifier [44], Ant Colony Optimization [55], [70], FAST Algorithm [79], Support Vector Machine/SVM [80], Genetic [42], [59], [76], [81], Fuzzy Expert [77], Dynamic Programming [45], Recommender System [58], Clustering Technique [73], [82], [83], Particle Swarm Optimization [61], Natural Language Processing (NLP) [74], and Bat-inspired Algorithm [48].

Based on our survey, we found that some researchers used more than one approach or technique in their study. For example, a study combined estimated risk value, coverage information, and fault detection [53]. Another study compared the mutation-based and diversity-aware [75]. Finally, there is also a study that looked into requirement and risk-based [84].

2) What is The Software under Test (SUT) in TCP Studies? (RQ2): Software or system under test (SUT) is a complete system as the object or target of testing. A well-structured and centralized SUT infrastructure can gradually build knowledge [85]. In this study, SUTs for evaluation are diverse. We classify the utilization of SUTs based on five categories: 1) researchers build their SUTs using open source from public resources, such as Github or other sources. In this case, the researchers design the fault and test cases for a specific purpose; 2) researchers utilize the SUT from the dataset or repository such as SIR, Defects4J, or others. In this case, researchers only need to explore and directly use the SUT from the repository; 3) researchers use the software from the industry as the cases with scale variations; 4) researchers build a software, create some faults and some test cases; 5) Others SUTs. Table X shows the distribution of SUT utilization according to the five categories.

The SIR and Defect4J repositories are still widely used as sources for SUTs. Besides, many researchers build and open-source SUT as a research object. Fig. 8 illustrates the distribution of SUT usage according to the five classifications described in Table X.

TABLE X. UTILIZATION OF SOFTWARE UNDER TEST (SUT)

SUT	Research
Building SUT by utilizing open source from public sources such as Github or the others	[86][20][1][45][41][58][60][48][42]
Utilizing SUT from the available database such as software - artifact infrastructure repository (SIR) and Defect4J	[37] [38] [39] [44] [49] [52] [55] [56] [40] [62] [77] [83] [66] [87][10] [59] [79] [43] [73] [61] [53] [64]
Real case from the industry	[49][49][20][1][46][58][51][60][47][42] [74]
Software, faults, and test-cases developed by the researcher	[82]
Others software	[36][71][76][69][81][70][13][71][42]

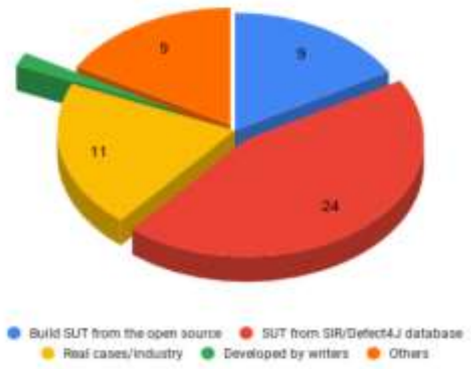

Fig. 8. Distribution of SUT Utilization.

3) What is the trend of metrics to measure the TCP effectiveness? (RQ3): In TCP studies, researchers generally aim to present the effectiveness of the developed techniques. In this regard, some metrics are known, as shown in Table XI. To answer what is the trend of metrics utilization to measure the TCP effectiveness, we identify metric utilization in all studies.

Several studies utilize more than one metric in their research. As in previous studies [22] and [23], the average percentage fault detection (APFD) was used dominantly in many TCP studies, while other metrics are spread out in less specific numbers. Table XII shows metric utilization in the studies, and Fig. 9 visualizes the distribution of metrics used.

C. What is the State of The Art of Requirement based-TCP (RQ4)?

Section 4.2 presents current research efforts to improve TCP for RT, while this section narrows the focus on current research efforts to improve requirements-based TCP.

Prior to conducting a review of the current research effort, we consider it is necessary to review the development of requirements-based TCP before 2017. Almost all TCP surveys that discuss requirements-based TCP start with prioritizing requirements for tests (PORT) [9] as the basis. The primary references for requirements-based TCP prior to 2017 are from studies [22] and [30]. The following is our exploration of the requirements-based TCP development including studies prior to 2017.

PORT is a value-driven approach to implementing TCP at the system level. Study[9] prioritizes the test-cases refer to four parameters: requirement volatility (RV), customer-assigned priority on requirements (CP), fault proneness of requirements (FP), and developer-perceived implementation complexity (IC). To determine the test case prioritization, each factor is carried out and given a score. For example, CP is rated with a range of 0-10, where 10 is the highest priority value. The evaluation result shows that the PORT technique can increase the detection rate of severe errors compared to executing a random test case. More specifically, CP is the most influential factor in increasing the PORT effectiveness and next IC. They used two metrics to measure the PORT effectiveness: the average severity of faults detected (ASFD) and total severity fault detection (TSFD).

TABLE XI. DESCRIPTION OF METRIC TO MEASURE THE TCP EFFECTIVENESS

Metric	Description
APFD	Average Percentage Fault Detection
APFDC	Average Percentage Fault Detection and Cost
Modified APFD	Modified Average Percentage Fault Detection
NAPFD	Normalize Average Percentage Fault Detection
EPS	Epsilon
ECC	Effectiveness of Change Coverage
PTRSW	Percentage of Total Risk Severity Weight
APCC	The average percentage of λ-wise combinations covered/ Average Percentage of Combinatorial Coverage
RP	The Average Relative Position
HMFD	The harmonic means of the rate of fault detection
APTC	Average percentage of test-point coverage
eAPWC	Enhanced average percentage of win-Cost coverage
NTE	The Number of Test to be Evaluated
HV	Hypervolume (HV) measures the volume in the objective space covered by the produced solutions with the range from 0 to 1 and a higher value of HV denotes a better performance of the algorithm
APSC	Average Percentage of Statement Coverage
Requirement coverage	The number of requirements covered by test
Code coverage	How many codes were executed while test performed? Can be in the form of number of line (line coverage), branch (branch coverage), or even path (path coverage).
Test case & patch diff.	Difference between the number of test cases and patches generated between approaches
Similarity	Test case similarity measures the distance between two test cases and returns a value within the range [0,1].
Prior-aware similarity	Prioritization-aware Test Case Similarity. Measures the average similarity of each of the test cases with its preceding test cases (i.e., test cases that were prioritized before)
Severity	Severity detection per test case execution (early detection of severe faults)

TABLE XII. METRIC UTILIZATION IN THE TCP RESEARCH

Metric	Research
APFD	[36][38][39][44][49][56][40][57][86] [62][72][20][77][83][88][66][87][41] [50] [79][43][73][75][60][82][47][48] [61][53][74][64]
APFDc	[38][80][1][45][87][51][10][59]
Modified APFD	[44]
NAPFD	[37][58]
EPS	[39]
ECC	[39]
Execution Time	[38][52][55][56][40][80][76][20][42] [41] [65][48]
PTRSW	[36]
APTC	[70]
eAPWC	[70]
APCC	[57][64]
RP	[54]
HMFD	[86]
NTE	[60]
Requirement coverage	[13][42][68][69]
Code Coverage	[55][56]
Fault detected	[40][46][71]
Test case and patch diff. count	[52]
APSC	[59]
Similarity	[42]
Prioritization-aware-Similarity	[42]
Severity	[45]

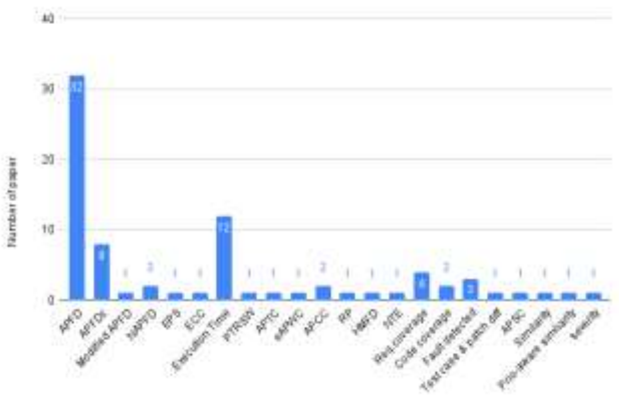

Fig. 9. Distribution of Metrics Utilization.

The following requirement-based TCP was introduced in 2009 [11], involving six factors: changes in requirements, customer assigned priority of requirements, fault impact, developer-perceived code implementation complexity, application flow, and usability. The authors divided the six factors into three factors for testing at the initial version stage and three factors for the regression testing stage. Furthermore, this study proposed a technique or steps to prioritize test cases using requirements-based factors. This stage information can be a reference for TCP requirement-based researchers.

A TCP through correlation of requirement and risk has been studied by Yoon et al. [89]. They reported TCP's risk-based testing (RBT) technique using defining risk items and estimated the risk exposure value derived from the requirement. The calculation of risk exposure value is determined based on requirement risk weight and the value of risk exposure. Specifically, they defined product risk items, which are expected to be helpful for the risk identification process. They also presented empirical studies comparing the effectiveness of their approach with other prioritization approaches. This empirical study shows that the utilization of risk exposure is promising in terms of effectiveness and can detect severe errors. This condition will have an impact on efforts to save time and costs.

In addition, Arafeen and Do [90] reported about TCP method using requirements-based clustering. They used a machine-learning algorithm to cluster the textual similarity among requirements. The clustering technique classified the distribution of words that co-occurs in their requirements. There are three tasks in this process: term-document matrix construction, term extraction, and k-means clustering. Their empirical study showed that the method could improve the effectiveness of TCP. Their empirical study showed that the method could improve the effectiveness of TCP.

Throughout years, several studies have been carried out to deal with requirements-risk in requirements-based TCP [8], [77], [91], [92]. These studies were seen as a series of efforts to further improve TCP based on requirement risk. Since PORT [9] was introduced, the researchers further explore the fuzzy expert system to prioritize test cases systematically [92] and later was investigated empirically with industry cases [77].

Many types of factors were utilized in the research conducted on test-case prioritization [8], such as utilized requirements modification status (RMS), requirements size (RS), requirements complexity (RC), and potential security threats (PST). Meanwhile, a study [77] reported four indicator risks to propose their approach, which are RC, fuzzification, a potential security risk (PSR), and requirements modification level (RML).

The other types of requirement risk factors was explained in [91], which proposed general steps to prioritize test: 1) estimating the risk and requirements correlation; 2) calculate the risk weight for all requirements; 3) calculate the exposure value; 4) evaluate additional factors for requirements prioritization; and 5) prioritize the requirements and test cases for all requirements.

The researchers utilized some risk factors to implement TCP, while the other researchers implemented the requirement-risk TCP for specific software. For example, a study has been carried out [36] to calculate the risk value from some parameters of requirement complexity, such as methods failure likelihood (MFL), method complexity (MC), change requirements (CR), methods failure impact (MFI), and method size (MS). The result of these calculations then used to determine the prioritized test suite. Meanwhile, study [93] utilized five aspects of risk to formulate their framework: risk item type, characteristic, measurement method, calculation procedure, and risk level.

A study [94] utilized the correlation of requirements to build the TCP technique. Their study calculated requirements priority (RP) based on customer-perceive priority (CP), development perceives priority (DP). RP of the i-th calculation in the formula RPi = $CP_i + DP_i$. They assumed that the CP and DP have equal weight. The authors claimed, this TCP technique was efficient on a small-scale study, and their method was better than the sorting process.

The discussion of requirements-based TCP since 2017 was begun with an analysis of research [71] which implement the TCP using requirement dependency with four parameters: test cases, test requirements, errors, and costs. On the other hand, they defined other elements related to functional requirements and requirements dependencies. Therefore, the authors use the algorithm to prioritize test cases considers the objectives of optimization, error detection, and cost.

The study [68] presents the requirement-dependency TCP by modeling the requirements and information of the test-related and their relationships with some aspects such as stakeholder affiliation, stakeholder's assigned priority, cost, time, risk, and business value. In prioritizing the test cases, they utilize the PageRank algorithm.

Study [69] utilized information coverage as an input resource. The authors proposed the use of complex test cases to test the requirements coverage. With complete coverage, the error detection rate also increased. At the same time, study [94] explained TCP's usage based on requirement correlations. When the testing process detects errors in a functional requirement, other correlated requirements may contain similar errors or other errors depending on the correlation between the two requirements. This study gives a better understanding of requirements correlation and its impact to be further explored in future TCP research. The parameters for the prioritization process use customer priority (CP) and developer priority (DP). Both of which are assessed by humans to produce a requirement priority (RP) as the initialization stage for the test case prioritization process.

In 2019, a study [77] utilized requirements risks in requirements-based TCP which introduced the fuzzy logic to reduce the humans' role in estimating risk factors for prioritizing test cases. This study is a continuation of the previous requirements-risk survey, which started in 2014[91].

We investigate more on TCP using requirement dependencies researched by two studies [68] and [71]. These studies are essential to explore because requirements-based TCP research focuses on the use of information in requirements, such as the interactions between requirements that influence the feasibility of the functionalities. This interaction is known as requirements dependency. The study [71] compared the cost-effectiveness of testing between the Greedy Method and the Genetic Algorithm (GA). The study prioritizes the GA to form a test suite that ensures all the defined requirements and has the lowest cost and highest fault detection.

TABLE XIII. THE STUDY ON REQUIREMENT-BASED TCP

Studies	Requirement-information	Software Under Test (SUT)	Metric Used
[9]	requirements volatility, customer priority (CP), Implementation Complexity (IC), and requirement's fault proneness.	Four projects developed by students in the advanced graduate	ASFD, TSFD,
[11]	Customer assigned priority of requirements, developer-perceived, code, IC, change requirement (CR), application flow, fault impact, and usability	Five projects (Phase1); Project with 5000 LOC (Phase2); Industrial Case, Cosmosoft Technologies Limited (Phase3)	TSFD, ASFD, TTEI, ATEI
[89]	Requirement risk	Program from Siemens	APFD
[90]	Requirement clustering	Java programs containing multiple versions (two program)	APFD
[91]	Requirement risk	Open-source and capstone project.	APFD
[8]	Requirement risk	Enterprise-level IBM analytics application.	APFD
[92]	Requirement risk	Open-source and the industrial (one program)	APFD
[94]	Requirement risk	Industrial case study	APFD
[71]	Requirement dependency	A synthetic case study	APFD
[68]	Requirement dependency	Small example case	Requirement coverage
[69]	Requirement coverage	Own case study	APFD
[77]	Requirement risk	Industrial application	APFD

Meanwhile, Abbas et al. [68] made a requirement dependency meta-model on non-functional requirements and performed TCP using the Page Rank Algorithm. This requirement dependency value was used as an addition to the priority ranking weight for these requirements. In summary, Table XIII presents the requirement-based TCP research conducted to date.

V. RESEARCH FINDING

Regression testing is a crucial stage in the software development process especially in the era of Agile development. TCP appears as the most popular technique in regression testing due to testing efficiency. Even if testing must be stopped for some reason, the high-priority test cases have found the essential faults.

We have conducted a rigorous study through searchers from reputable resources and carefully filtered the findings through a quality assessment to review current efforts on TCP for RT. In RQ1, we found that the existing TCP approaches presented by [22] and later added by [27] are still of interest to researchers. We discover five new approaches through the review: modification-based, user interface-based, model-based, mutation-based, and similarity-based. The discovery shows that the taxonomy is still progressing and further explored by researchers. In terms of the technique for TCP implementation, we discover that there is no dominant algorithm used. Researchers have their reasons for choosing a technique to implement TCP based on specific analysis and motivations. However, it is noticeable that the Greedy and Additional-Greedy Algorithm, which are classic search-based TCP algorithms, are still in demand. Likewise, Genetic Algorithms are also popular. On the other hand, we discover that Machine Learning Algorithms seem to be growing in use and gaining popularity.

As for RQ2, we have investigated that SIR is an SUT repository database with an excellent and complete SUT collection. However, although many researchers still use it, some studies use SUT for empirical studies. Through the review, we classify SUTs into six categories: SUTs built from open-source software, SUTs from repositories such as SIR and Defects, SUTs from industrial cases, SUTs constructed by researchers.

Answering RQ3, in terms of using metrics to measure the effectiveness of TCP, we did not find any significant progress. Some researchers add aspects of measurement on a fixed basis to the APFD. In this case, the APFD is dominantly used.

Finally, on RQ4, referring to reviews carried out by previous surveys, we found the overall development on the use of requirements-based TCP. Although requirements-based TCP is not as popular as many other approaches, it can be further developed. For example, referring to our survey, we found that requirement risk TCP introduced by [77], requirement dependency TCP by [71] and [68]. It is proven that the research conducted has allowed growing further and improving the effectiveness. As stated by [22], one of the advantages of requirement-based TCP is the privilege of utilizing information from requirements. Therefore, the preparation of test cases can be done earlier to save time in the testing process.

VI. Conclusion

This paper presents the SLR result on test case prioritization for regression testing. The study's main objective was to get the state of the art on TCP for regression testing in 2017-2020. Furthermore, this study also investigates the TCP explicitly based on requirements since the TCP was first introduced until 2020.

We found more TCP approaches not mentioned in other surveys, which have opportunities for further research. The new TCP approaches are modification-based, user interface-based, model-based, mutation-based, and similarity-based. In the use of SUT, it appears that there are more diverse variations of SUT. Even so, the utilization of SUT repositories such as SIR and Defect4J is still in great demand. We also discover that APFD is still a very dominant metric, and almost no specific new metrics are found.

For future work, it is beneficial to explore the utilization of requirements to improve TCP effectiveness. We view that requirement-based utilization will help prepare test cases earlier, so the testing process can be more efficient. Requirement risk is an essential aspect of a requirement that is considered in the test case prioritization development. Meanwhile, the dependency between requirements is a crucial issue to consider in software development, so it can be one of the factors for prioritizing test cases. We cannot ignore the relationship between requirements in the software development process, including in the testing stage. Requirement-based TCP still has many opportunities for improvement. We believe that there are many attributes in requirements that can improve TCP effectiveness.

VII. Acknowledgment

Thank you to Universiti Teknikal Malaysia Melaka for providing supports and resources for this research.

References

[1] Pradhan, S. Wang, S. Ali, T. Yue, and M. Liaaen, “Employing rule mining and multi-objective search for dynamic test case prioritization,” J. Syst. Softw., vol. 153, pp. 86–104, 2019, doi: 10.1016/j.jss.2019.03.064.

[2] [2]R. H. Rosero, O. S. Gómez, and G. Rodríguez, “15 Years of Software Regression Testing Techniques - A Survey,” Int. J. Softw. Eng. Knowl. Eng., vol. 26, no. 5, pp. 675–689, 2016, doi: 10.1142/S0218194016300013.

[3] Sebastian Ulewicz and Birgit Vogel-Heuser, “Industrially Applicable System Regression Test Prioritization in Production Automation,” 2 IEEE Trans. Autom. Sci. Eng., pp. 1545–5955, 2018.

[4] S. Souto and M. d’Amorim, “Time-space efficient regression testing for configurable systems,” J. Syst. Softw., vol. 137, pp. 733–746, 2018, doi: 10.1016/j.jss.2017.08.010.

[5] H. Hemmati, Advances in Techniques for Test Prioritization, 1st ed., vol. 112. Elsevier Inc., 2019.

[6] D. Hao, L. Zhang, and H. Mei, “Test-case prioritization : achievements and challenges,” pp. 1–9, 2016.

[7] M. J. Arafeen and H. Do, “Test case prioritization using requirements-based clustering,” Proc. - IEEE 6th Int. Conf. Softw. Testing, Verif. Validation, ICST 2013, pp. 312–321, 2013, doi: 10.1109/ICST.2013.12.

[8] H. Srikanth, C. Hettiarachchi, and H. Do, “Requirements based test prioritization using risk factors: An industrial study,” Inf. Softw. Technol., vol. 69, pp. 71–83, 2016, doi: 10.1016/j.infsof.2015.09.002.

[9] J. Srikanth, H, Williams, L, Osborne, “System Test Case Prioritization of New and Regression Test Cases,” in IEEEE International Symposium on Empirical Study, 2005, pp. 64–73.

[10] Y. Wang, Z. Zhu, B. Yang, F. Guo, and H. Yu, “Using reliability risk analysis to prioritize test cases,” J. Syst. Softw., vol. 139, pp. 14–31, 2018, doi: 10.1016/j.jss.2018.01.033.

[11] R. Krishnamoorthi and S. A. Sahaaya Arul Mary, “Requirement based system test case prioritization of new and regression test cases,” Int. J. Softw. Eng. Knowl. Eng., vol. 19, no. 3, pp. 453–475, 2009, doi: 10.1142/S0218194009004222.

[12] V. Garousi, R. Özkan, and A. Betin-Can, “Multi-objective regression test selection in practice: An empirical study in the defense software industry,” Inf. Softw. Technol., vol. 103, pp. 40–54, 2018, doi: 10.1016/j.infsof.2018.06.007.

[13] S. Ulewicz, B. Vogel-heuser, and S. Member, “Industrially Applicable System Regression Test Prioritization in Production Automation,” IEEE Trans. Autom. Sci. Eng., pp. 1–13, 2018.

[14] G. Rothermel, R. H. Untcn, C. Chu, and M. J. Harrold, “Prioritizing test cases for regression testing,” IEEE Trans. Softw. Eng., vol. 27, no. 10, pp. 929–948, 2001, doi: 10.1109/32.962562.

[15] A. Zarrad, “A Systematic Review on Regression Testing for Web-Based Applications,” J. Softw., vol. 10, no. 8, pp. 971–990, 2015, doi: 10.17706/jsw.10.8.971-990.

[16] H. Do, “Recent Advances in Regression Testing Techniques,” Adv. Comput., vol. 103, pp. 53–77, 2016, doi: 10.1016/bs.adcom.2016.04.004.

[17] N. Sharma, Sujata, and G. N. Purohit, “Test case prioritization techniques ‘an empirical study,’” 2014 Int. Conf. High Perform. Comput. Appl. ICHPCA 2014, 2015, doi: 10.1109/ICHPCA.2014.7045344.

[18] P. E. Strandberg, D. Sundmark, W. Afzal, T. J. Ostrand, and E. J. Weyuker, “Experience Report: Automated System Level Regression Test Prioritization Using Multiple Factors,” Proc. - Int. Symp. Softw. Reliab. Eng. ISSRE, no. October, pp. 12–23, 2016, doi: 10.1109/ISSRE.2016.23.

[19] H. S. De Campos, C. A. De Paiva, R. Braga, M. A. P. Araujo, J. M. N. David, and F. Campos, “Regression tests provenance data in the continuous so ware engineering context,” ACM Int. Conf. Proceeding Ser., vol. Part F1306, no. September 2018, 2017, doi: 10.1145/3128473.3128483.

[20] J. Anderson, M. Azizi, S. Salem, and H. Do, “On the use of usage patterns from telemetry data for test case prioritization,” Inf. Softw. Technol., vol. 113, pp. 110–130, 2019, doi: 10.1016/j.infsof.2019.05.008.

[21] S. Yoo and M. Harman, “Regression testing minimization, selection and prioritization: A survey,” Softw. Test. Verif. Reliab., vol. 22, no. 2, pp. 67–120, 2012, doi: 10.1002/stv.430.

[22] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, and R. Tumeng, “Test case prioritization approaches in regression testing: A systematic literature review,” Inf. Softw. Technol., vol. 93, no. June 2018, pp. 74–93, 2018, doi: 10.1016/j.infsof.2017.08.014.

[23] Y. Singh, A. Kaur, B. Suri, and S. Singhal, “Systematic literature review on regression test prioritization techniques,” Inform., vol. 36, no. 4, pp. 379–408, 2012, doi: 10.31449/inf.v36i4.420.

[24] H. S. De Campos Junior, M. A. P. Arajo, J. M. N. David, R. Braga, F. Campos, and V. Ströele, “Test case prioritization: A systematic review and mapping of the literature,” ACM Int. Conf. Proceeding Ser., no. August 2018, pp. 34–43, 2017, doi: 10.1145/3131151.3131170.

[25] M. L. Mohd Shafie and W. M. N. Wan Kadir, “Model-based test case prioritization: A systematic literature review,” J. Theor. Appl. Inf. Technol., vol. 96, no. 14, pp. 4548–4573, 2018.

[26] R. Mukherjee and K. S. Patnaik, “A survey on different approaches for software test case prioritization,” J. King Saud Univ. - Comput. Inf. Sci., 2018, doi: 10.1016/j.jksuci.2018.09.005.

[27] M. D. C. De Castro-Cabrera, A. García-Dominguez, and I. Medina-Bulo, “Trends in prioritization of test cases: 2017-2019,” Proc. ACM Symp. Appl. Comput., pp. 2005–2011, 2020, doi: 10.1145/3341105.3374036.

[28] A. Samad, H. Mahdin, R. Kazmi, and R. Ibrahim, “Regression Test Case Prioritization: A Systematic Literature Review,” Int. J. Adv. Comput. Sci. Appl., vol. 12, no. 2, pp. 655–663, 2021, doi: 10.14569/IJACSA.2021.0120282.

[29] M. Hasnain, M. F. Pasha, I. Ghani, and S. R. Jeong, Functional Requirement-Based Test Case Prioritization in Regression Testing: A Systematic Literature Review, vol. 2, no. 6. Springer Singapore, 2021.

[30] Y. Lou, J. Chen, L. Zhang, and D. Hao, A Survey on Regression Test-Case Prioritization, 1st ed., vol. 113. Elsevier Inc., 2019.

[31] D. Qiu, B. Li, S. Ji, and H. Leung, “Regression testing of web service: A systematic mapping study,” ACM Comput. Surv., vol. 47, no. 2, 2014, doi: 10.1145/2631685.

[32] M. Hasnain, I. Ghani, M. F. Pasha, C. H. Lim, and S. R. Jeong, “A Comprehensive Review on Regression Test Case Prioritization Techniques for Web Services,” KSII Trans. Internet Inf. Syst., vol. 14, no. 5, pp. 1861–1885, 2020, doi: 10.3837/tiis.2020.05.001.

[33] J. A. Prado Lima and S. R. Vergilio, “Test Case Prioritization in Continuous Integration environments: A systematic mapping study,” Inf. Softw. Technol., vol. 121, p. 106268, 2020, doi: 10.1016/j.infsof.2020.106268.

[34] N. bin Ali et al., On the search for industry-relevant regression testing research, vol. 24, no. 4. 2019.

[35] B. Kitchenham and P. Brereton, “A systematic review of systematic review process research in software engineering,” Inf. Softw. Technol., vol. 55, no. 12, pp. 2049–2075, 2013, doi: 10.1016/j.infsof.2013.07.010.

[36] H. Jahan, Z. Feng, and S. M. H. Mahmud, “Risk-Based Test Case Prioritization by Correlating System Methods and Their Associated Risks,” Arab. J. Sci. Eng., vol. 45, no. 8, pp. 6125–6138, 2020, doi: 10.1007/s13369-020-04472-z.

[37] R. Wang, Z. Li, S. Jiang, and C. Tao, “Regression Test Case Prioritization Based on Fixed Size Candidate Set ART Algorithm,” Int. J. Softw. Eng. Knowl. Eng., vol. 30, no. 3, pp. 291–320, 2020, doi: 10.1142/S0218194020500138.

[38] W. Su, Z. Li, Z. Wang, and D. Yang, “A Meta-heuristic Test Case Prioritization Method Based on Hybrid Model,” Proc. - 2020 Int. Conf. Comput. Eng. Appl. ICCEA 2020, pp. 430–435, 2020, doi: 10.1109/ICCEA50009.2020.00099.

[39] S. Mondal and R. Nasre, “Hansie: Hybrid and consensus regression test prioritization,” J. Syst. Softw., vol. 172, no. October, 2021, doi: 10.1016/j.jss.2020.110850.

[40] J. Chi et al., “Relation-based test case prioritization for regression testing,” J. Syst. Softw., vol. 163, 2020, doi: 10.1016/j.jss.2020.110539.

[41] M. Khanna, N. Chauhan, and D. K. Sharma, “Search for prioritized test cases during web application testing,” Int. J. Appl. Metaheuristic Comput., vol. 10, no. 2, pp. 1–26, 2019, doi: 10.4018/IJAMC.2019040101.

[42] A. Arrieta, S. Wang, U. Markiegi, G. Sagardui, and L. Etxeberria, “Employing Multi-Objective Search to Enhance Reactive Test Case Generation and Prioritization for Testing Industrial Cyber-Physical Systems,” IEEE Trans. Ind. Informatics, vol. 14, no. 3, pp. 1055–1066, 2018, doi: 10.1109/TII.2017.2788019.

[43] M. Azizi and H. Do, “Graphite: A Greedy Graph-Based Technique for Regression Test Case Prioritization,” in 2018 IEEE International Symposium on Software Reliability Engineering Workshops (ISSREW), 2018, pp. 245–251, doi: 10.1109/ISSREW.2018.00014.

[44] M. Mahdieh, S. H. Mirian-Hosseinabadi, K. Etemadi, A. Nosrati, and S. Jalali, “Incorporating fault-proneness estimations into coverage-based test case prioritization methods,” Inf. Softw. Technol., vol. 121, no. January, p. 106269, 2020, doi: 10.1016/j.infsof.2020.106269.

[45] M. Khanna, A. Chaudhary, A. Toofani, and A. Pawar, “Performance Comparison of Multi-objective Algorithms for Test Case Prioritization During Web Application Testing,” Arab. J. Sci. Eng., vol. 44, no. 11, pp. 9599–9625, 2019, doi: 10.1007/s13369-019-03817-7.

[46] I. Hajri, A. Goknil, F. Pastore, and L. C. Briand, “Automating system test case classification and prioritization for use case-driven testing in product lines,” Empir. Softw. Eng., vol. 25, no. 5, pp. 3711–3769, 2020, doi: 10.1007/s10664-020-09853-4.

[47] S. Nayak, C. Kumar, and S. Tripathi, “Enhancing Efficiency of the Test Case Prioritization Technique by Improving the Rate of Fault Detection,” Arab. J. Sci. Eng., vol. 42, no. 8, pp. 3307–3323, 2017, doi: 10.1007/s13369-017-2466-6.

[48] M. M. Öztürk, “A bat-inspired algorithm for prioritizing test cases,” Vietnam J. Comput. Sci., 2017, doi: 10.1007/s40595-017-0100-x.

[49] K. W. Shin and D. J. Lim, “Model-based test case prioritization using an alternating variable method for regression testing of a UML-based model,” Appl. Sci., vol. 10, no. 21, pp. 1–23, 2020, doi: 10.3390/app10217537.

[50] J. F. S. Ouriques, E. G. Cartaxo, and P. D. L. Machado, “Test case prioritization techniques for model-based testing: a replicated study,” Softw. Qual. J., vol. 26, no. 4, pp. 1451–1482, 2018, doi: 10.1007/s11219-017-9398-y.

[51] T. Zhang, X. Wang, D. Wei, and J. Fang, “Test Case Prioritization Technique Based on Error Probability and Severity of UML Models,” vol. 28, no. 6, pp. 831–844, 2018, doi: 10.1142/S0218194018500249.

[52] Y. Venugopal, P. Quang-Ngoc, and L. Eunseok, “Modification point aware test prioritization and sampling to improve patch validation in automatic program repair,” Appl. Sci., vol. 10, no. 5, pp. 1–14, 2020, doi: 10.3390/app10051593.

[53] W. Fu, H. Yu, G. Fan, X. Ji, and X. Pei, “A Regression Test Case Prioritization Algorithm Based on Program Changes and Method Invocation Relationship,” Proc. - Asia-Pacific Softw. Eng. Conf.

APSEC, vol. 2017-Decem, pp. 169–178, 2018, doi: 10.1109/APSEC.2017.23.

[54] H. Wang, M. Yang, L. Jiang, J. Xing, Q. Yang, and F. Yan, "Test Case Prioritization for Service-Oriented Workflow Applications: A Perspective of Modification Impact Analysis," IEEE Access, vol. 8, pp. 101260–101273, 2020, doi: 10.1109/ACCESS.2020.2998545.

[55] M. K. Pachariya, "Building Ant System for Multi-Faceted Test Case Prioritization: An Empirical Study," Int. J. Softw. Innov., vol. 8, no. 2, pp. 23–37, 2020, doi: 10.4018/IJSI.2020040102.

[56] R. Huang, Q. Zhang, D. Towey, W. Sun, and J. Chen, "Regression test case prioritization by code combinations coverage," J. Syst. Softw., vol. 169, p. 110712, 2020, doi: 10.1016/j.jss.2020.110712.

[57] R. Huang et al., "Abstract Test Case Prioritization Using Repeated Small-Strength Level-Combination Coverage," IEEE Trans. Reliab., vol. 69, no. 1, pp. 349–372, 2020, doi: 10.1109/TR.2019.2908068.

[58] M. Azizi, "A Collaborative Filtering Recommender System for Test Case Prioritization in Web Applications," pp. 1560–1567, 2018.

[59] D. Di Nucci, A. Panichella, A. Zaidman, and A. De Lucia, "A Test Case Prioritization Genetic Algorithm Guided by the Hypervolume Indicator," IEEE Trans. Softw. Eng., vol. 46, no. 6, pp. 674–696, 2020, doi: 10.1109/TSE.2018.2868082.

[60] R. Matinnejad, S. Nejati, L. C. Briand, and T. Bruckmann, "Test Generation and Test Prioritization for Simulink Models with Dynamic Behavior," IEEE Trans. Softw. Eng., vol. 45, no. 9, pp. 919–944, 2019, doi: 10.1109/TSE.2018.2811489.

[61] M. Khatibsyarbini, M. A. Isa, and D. N. A. Jawawi, "A hybrid weight-based and string distances using particle swarm optimization for prioritizing test cases," J. Theor. Appl. Inf. Technol., vol. 95, no. 12, pp. 2723–2732, 2017.

[62] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, H. N. A. Hamed, and M. D. Mohamed Suffian, "Test Case Prioritization Using Firefly Algorithm for Software Testing," IEEE Access, vol. 7, pp. 132360–132373, 2019, doi: 10.1109/access.2019.2940620.

[63] X. Lei, M. Huaikou, Z. Weiewei, and C. Shaojun, "An Empirical Study on Clustering Approach Combining Fault Prediction for Test Case Prioritization," in International Conference on Information Systems, 2017, pp. 815–820, doi: 10.1109/ICIS.2017.7960105.

[64] R. Huang, Y. Zhou, W. Zong, D. Towey, and J. Chen, "An Empirical Comparison of Similarity Measures for Abstract Test Case Prioritization," pp. 3–12, 2017, doi: 10.1109/COMPSAC.2017.271.

[65] B. Miranda, E. Cruciani, R. Verdecchia, and A. Bertolino, "FAST approaches to scalable similarity-based test case prioritization," Proc. - Int. Conf. Softw. Eng., vol. 2018-Janua, pp. 222–232, 2018, doi: 10.1145/3180155.3180210.

[66] S. A. Halim, D. N. A. Jawawi, and M. Sahak, "Similarity distance measure and prioritization algorithm for test case prioritization in software product line testing," J. Inf. Commun. Technol., vol. 18, no. 1, pp. 57–75, 2019, doi: 10.32890/jict2019.18.1.4.

[67] A. D. Shrivathsan et al., "Novel fuzzy clustering methods for test case prioritization in Software Projects," Symmetry (Basel)., vol. 11, no. 11, pp. 1–22, 2019, doi: 10.3390/sym11111400.

[68] M. Abbas, I. Inayat, M. Saadatmand, and N. Jan, "Requirements Dependencies-Based Test Case Prioritization for Extra-Functional Properties," in 2019 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), 2019, pp. 159–163, doi: 10.1109/ICSTW.2019.00045.

[69] R. Butool, A. Nadeem, M. Sindhu, and O. u. Zaman, "Improving Requirements Coverage in Test Case Prioritization for Regression Testing," in 2019 22nd International Multitopic Conference (INMIC), 2019, pp. 1–6, doi: 10.1109/INMIC48123.2019.9022761.

[70] W. Zhang, Y. Qi, X. Zhang, B. Wei, M. Zhang, and Z. Dou, "On Test Case Prioritization Using Ant Colony Optimization Algorithm," in 2019 IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems (HPCC/SmartCity/DSS), 2019, pp. 2767–2773, doi: 10.1109/HPCC/SmartCity/DSS.2019.00388.

[71] A. Vescan, C. Șerban, C. Chisăliță-Crețu, and L. Dioșan, "Requirement dependencies-based formal approach for test case prioritization in regression testing," Proc. - 2017 IEEE 13th Int. Conf. Intell. Comput. Commun. Process. ICCP 2017, no. September 2017, pp. 181–188, 2017, doi: 10.1109/ICCP.2017.8117002.

[72] Z. Yu, F. Fahid, T. Menzies, G. Rothermel, K. Patrick, and S. Cherian, "TERMINATOR: Better automated UI test case prioritization," ESEC/FSE 2019 - Proc. 2019 27th ACM Jt. Meet. Eur. Softw. Eng. Conf. Symp. Found. Softw. Eng., pp. 883–894, 2019, doi: 10.1145/3338906.3340448.

[73] M. Abdur, M. Abu, and M. Saeed, "Prioritizing Dissimilar Test Cases in Regression Testing using Historical Failure Data," Int. J. Comput. Appl., vol. 180, no. 14, pp. 1–8, 2018, doi: 10.5120/ijca2018916258.

[74] Y. Yang, X. Huang, X. Hao, Z. Liu, and Z. Chen, "An Industrial Study of Natural Language Processing Based Test Case Prioritization," in 2017 IEEE International Conference on Software Testing, Verification and Validation (ICST), 2017, pp. 548–549, doi: 10.1109/ICST.2017.66.

[75] D. Shin, S. Yoo, M. Papadakis, and D. H. Bae, "Empirical evaluation of mutation-based test case prioritization techniques," Softw. Test. Verif. Reliab., vol. 29, no. 1–2, pp. 1–28, 2019, doi: 10.1002/stvr.1695.

[76] D. B. Mishra, R. Mishra, A. A. Acharya, and K. N. Das, Test case optimization and prioritization based on multi-objective genetic algorithm, vol. 741. Springer Singapore, 2019.

[77] C. Hettiarachchi and H. Do, "A Systematic Requirements and Risks-Based Test Case Prioritization Using a Fuzzy Expert System," Proc. - 19th IEEE Int. Conf. Softw. Qual. Reliab. Secur. QRS 2019, pp. 374–385, 2019, doi: 10.1109/QRS.2019.00054.

[78] M. Khatibsyarbini, M. A. Isa, D. N. A. Jawawi, H. N. A. Hamed, and M. D. M. Suffian, "Test Case Prioritization Using Firefly Algorithm for Software Testing," IEEE Access, vol. 7, pp. 132360–132373, 2019, doi: 10.1109/ACCESS.2019.2940620.

[79] B. Miranda, E. Cruciani, R. Verdecchia, and A. Bertolino, "FAST Approaches to Scalable Similarity-Based Test Case Prioritization," in 2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE), 2018, pp. 222–232, doi: 10.1145/3180155.3180210.

[80] Z. Yu and T. Menzies, "TERMINATOR : Better Automated UI Test Case Prioritization."

[81] D. Kumar and Y. Sandip, "Regression test case selection and prioritization for object oriented software," Microsyst. Technol., vol. 7, 2019, doi: 10.1007/s00542-019-04679-7.

[82] S. C. Lei Xiao, Huaikou, Weiwei Zhuang, "An Empirical Study on Clustering Approach Combining Fault Prediction for Test Case Prioritization," in 2017 IEEE/ACIS 16th International Conference on Computer and Information Science (ICIS), 2017, pp. 815–820, doi: 10.1109/ICIS.2017.7960105.

[83] A. D. Shrivathsan, K. S. Ravichandran, R. Krishankumar, V. Sangeetha, and S. Kar, "Novel Fuzzy Clustering Methods for Test Case Prioritization in Software Projects," pp. 1–22.

[84] C. Hettiarachchi and H. Do, "A Systematic Requirements and Risks-Based Test Case Prioritization Using a Fuzzy Expert System," in 2019 IEEE 19th International Conference on Software Quality, Reliability and Security (QRS), 2019, pp. 374–385, doi: 10.1109/QRS.2019.00054.

[85] H. Do, S. Elbaum, and G. Rothermel, "Supporting controlled experimentation with testing techniques: An infrastructure and its potential impact," Empir. Softw. Eng., vol. 10, no. 4, pp. 405–435, 2005, doi: 10.1007/s10664-005-3861-2.

[86] H. Wang, M. Yang, L. Jiang, J. Xing, Q. Yang, and F. Yan, "Test Case Prioritization for Service-Oriented Workflow Applications: A Perspective of Modification Impact Analysis," IEEE Access, vol. 8, pp. 101260–101273, 2020, doi: 10.1109/ACCESS.2020.2998545.

[87] D. S. Silva, R. Rabelo, P. S. Neto, R. Britto, and P. A. Oliveira, "A test case prioritization approach based on software component metrics," Conf. Proc. - IEEE Int. Conf. Syst. Man Cybern., vol. 2019-Octob, no. August, pp. 2939–2945, 2019, doi: 10.1109/SMC.2019.8914670.

[88] D. K. Yadav and S. Dutta, "Regression test case selection and prioritization for object oriented software," Microsyst. Technol., vol. 26, no. 5, pp. 1463–1477, 2020, doi: 10.1007/s00542-019-04679-7.

[89] M. Yoon, E. Lee, M. Song, and B. Choi, "A Test Case Prioritization through Correlation of Requirement and Risk," J. Softw. Eng. Appl., vol. 05, no. 10, pp. 823–835, 2012, doi: 10.4236/jsea.2012.510095.

[90] Arafeen and H. Do, "Test Case Prioritization using Requirement-Based Clustering," in 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, 2013, pp. 488–492, doi: 10.1109/ICST.2013.12.

[91] C. Hettiarachchi, H. Do, and B. Choi, "Effective regression testing using requirements and risks," Proc. - 8th Int. Conf. Softw. Secur. Reliab. SERE 2014, pp. 157–166, 2014, doi: 10.1109/SERE.2014.29.

[92] C. Hettiarachchi, H. Do, and B. Choi, "Risk-based test case prioritization using a fuzzy expert system," Inf. Softw. Technol., vol. 69, pp. 1–15, 2016, doi: 10.1016/j.infsof.2015.08.008.

[93] M. Felderer, C. Haisjackl, V. Pekar, and R. Breu, "A risk assessment framework for software testing," Lect. Notes Comput. Sci. (including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), vol. 8803 Lever, pp. 292–308, 2014, doi: 10.1007/978-3-662-45231-8_21.

[94] T. Ma, H. Zeng, and X. Wang, "Test case prioritization based on requirement correlations," 2016 IEEE/ACIS 17th Int. Conf. Softw. Eng. Artif. Intell. Netw. Parallel/Distributed Comput. SNPD 2016, no. 61170044, pp. 419–424, 2016, doi: 10.1109/SNPD.2016.7515934.

[95] W. Zhang, Y. Qi, X. Zhang, B. Wei, M. Zhang, and Z. Dou, "On test case prioritization using ant colony optimization algorithm," Proc. - 21st IEEE Int. Conf. High Perform. Comput. Commun. 17th IEEE Int. Conf. Smart City 5th IEEE Int. Conf. Data Sci. Syst. HPCC/SmartCity/DSS 2019, pp. 2767–2773, 2019, doi: 10.1109/HPCC/SmartCity/DSS.2019.00388.

Appendix

The Selected Articles

Study	Publication type	Publication year	Publisher
[36]	Journal	2020	Arabian Journal for Science and Engineering
[37]	Journal	2020	International Journal of Software Engineering and Knowledge Engineering
[38]	Conference paper	2020	International Conference on Computer Engineering and Application (ICCEA)
[39]	Journal	2020	Journal of Systems and Software
[44]	Journal	2020	Information and Software Technology
[49]	Journal	2020	Applied Sciences Multidisciplinary Digital Publishing Institute (MDPI)
[52]	Journal	2020	Applied Sciences Multidisciplinary Digital Publishing Institute (MDPI)
[55]	Journal	2020	International Journal of Software Innovation
[56]	Journal	2020	The Journal of Systems & Software
[40]	Journal	2020	The Journal of Systems and Software
[57]	Journal	2020	Journal of LaTeX Class Files
[54]	Journal	2020	IEEE Access
[78]	Journal	2020	IEEE Access
[68]	Conference paper	2019	IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)
[72]	Conference paper	2019	Association of Computing Machinery (ACM)
[76]	Conference Paper	2019	International Conference on Harmony Search Algorithm (ICHSA)
[20]	Journal	2019	Information and Software Technology
[69]	Conference paper	2019	IEEE 2019 22nd International Multitopic Conference (INMIC)
[77]	Conference paper	2019	IEEE 19th International Conference on Software Quality, Reliability and Security (QRS)
[1]	Journal	2019	The Journal of Systems and Software
[83]	Journal	2019	Symmetry Multidisciplinary Digital Publishing Institute (MDPI)
[81]	Journal	2019	Microsystems Technologies
[66]	Journal	2019	Journal of Information and Communication Technology
[95]	Conference paper	2019	IEEE 21st International Conference on High Performance Computing and Communications; IEEE 17th. International Conference on Smart City; IEEE 5th International Conference on Data Science and Systems
[45]	Journal	2019	Arabian Journal for Science and Engineering
[87]	Conference paper	2019	IEEE International Conference on Systems, Man and Cybernetics (SMC)
[46]	Journal	2019	Empirical Software Engineering
[41]	Journal	2019	International Journal of Applied Metaheuristic Computing
[58]	Symposium paper	2018	Symposium on Applied Computing
[51]	Journal	2018	International Journal of Software Engineering
[50]	Journal	2018	Software Quality Journal
[10]	Journal	2018	The Journal of Systems and Software

[59]	Symposium paper	2018	Symposium on Search-Based Software Engineering 2015
[79]	Conference paper	2018	IEEE 40th International Conference on Software Engineering
[43]	Conference paper	2018	IEEE International Symposium on Software Reliability Engineering Workshops
[13]	Journal	2018	IEEE Transactions on Automation Science And Engineering
[73]	Journal	2018	International Journal of Computer Applications
[75]	Journal	2018	Software Testing, Verification and Reliability
[60]	Journal	2018	IEEE Transactions on Software Engineering
[82]	Conference paper	2017	International Conference on Information Systems
[71]	Conference paper	2017	IEEE International Conference on Intelligent Computer Communication and Processing (ICCP)
[47]	Journal	2017	Arabian Journal for Science and Engineering
[48]	Journal	2017	Vietnam Journal of Computer Science
[42]	Journal	2017	IEEE Transactions on Industrial Informatics
[61]	Journal	2017	Journal of Theoretical and Applied Information Technology
[53]	Conference paper	2017	Asia-Pacific Software Engineering Conference
[74]	Conference paper	2017	IEEE International Conference on Software Testing, Verification and Validation
[64]	Conference paper	2017	IEEE 41st Annual Computer Software and Applications Conference