Information and Software Technology 147 (2022) 106902

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

A Systematic Literature Review on prioritizing software test cases using Markov chains

Gerson Barbosa [a,e], Érica Ferreira de Souza [b,*], Luciana Brasil Rebelo dos Santos [c], Marlon da Silva [d], Juliana Marino Balera [e], Nandamudi Lankalapalli Vijaykumar [e]

[a] Universidade Estadual Paulista (UNESP), Guaratinguetá, Brazil
[b] Universidade Tecnológica Federal do Paraná (UTFPR), Cornélio Procópio, Brazil
[c] Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP), Jacareí, Brazil
[d] Instituto Federal de Educação, Ciência e Tecnologia de São Paulo (IFSP), Campos do Jordão, Brazil
[e] Instituto Nacional de Pesquisas Espaciais (INPE), São José dos Campos, Brazil

ARTICLE INFO

Keywords:
Systematic Literature Review
Markov Chains
Test case prioritization

ABSTRACT

Context: Software Testing is a costly activity since the size of the test case set tends to increase as the construction of the software evolves. Test Case Prioritization (TCP) can reduce the effort and cost of software testing. TCP is an activity where a subset of the existing test cases is selected in order to maximize the possibility of finding defects. On the other hand, Markov Chains representing a reactive system, when solved, can present the occupation time of each of their states. The idea is to use such information and associate priority to those test cases that consist of states with the highest probabilities.
Objective: The objective of this paper is to conduct a survey to identify and understand key initiatives for using Markov Chains in TCP. Aspects such as approaches, developed techniques, programming languages, analytical and simulation results, and validation tests are investigated.
Methods: A Systematic Literature Review (SLR) was conducted considering studies published up to July 2021 from five different databases to answer the three research questions.
Results: From SLR, we identified 480 studies addressing Markov Chains in TCP that have been reviewed in order to extract relevant information on a set of research questions.
Conclusion: The final 12 studies analyzed use Markov Chains at some stage of test case prioritization in a distinct way, that is, we found that there is no strong relationship between any of the studies, not only on how the technique was used but also in the context of the application. Concerning the fields of application of this subject, 6 forms of approach were found: Controlled Markov Chain, Usage Model, Model-Based Test, Regression Test, Statistical Test, and Random Test. This demonstrates the versatility and robustness of the tool. A large part of the studies developed some prioritization tool, being its validation done in some cases analytically and in others numerically, such as: Measure of the software specification, Optimal Test Transition Probabilities, Adaptive Software Testing, Automatic Prioritization, Ant Colony Optimization, Model Driven approach, and Monte Carlo Random Testing.

1. Introduction

It is a fact, nowadays, that the question of software reliability is extremely relevant, given the dependence society has on software systems. Quality of all such produced software systems must be ensured, and hence Verification & Validation (V&V) activities play a major role to be employed to achieve this goal [2]. Moreover, software testing is the most used V&V activity in practice [3,4]. The goal of any software testing effort is to find defects[1] in the software product, in this case very related to defects (faults) in the source code.

Testing activity ensures the conformity of the software product with the requirements defined by the customer, through a systematic execution. However, in order to entirely achieve this goal, the ideal approach is to exhaustively test the software, which is impractical. In order to cope up with this impracticality, techniques are designed to use

* Corresponding author.
E-mail addresses: gerson.barbosa@unesp.br, gerson.barbosa@inpe.br (G. Barbosa), ericasouza@utfpr.edu.br (É.F. de Souza), lurebelo@ifsp.edu.br (L.B.R. dos Santos), marlon.silva@ifsp.edu.br (M. da Silva), juliana.balera@inpe.br (J.M. Balera), vijay.nl@inpe.br (N.L. Vijaykumar).

[1] Defect is an incorrect step, process, or data definition, for example, an incorrect instruction in a computer program — IEEE Std 610.12-1990 [1].

https://doi.org/10.1016/j.infsof.2022.106902
Received 21 September 2021; Received in revised form 28 February 2022; Accepted 4 March 2022
Available online 18 March 2022

only a small subset of the input domain, but with a high probability of revealing the presence of defects, if they exist [3]. Therefore, it is essential to adopt a consolidated methodology that makes the test activity viable and effective. Some approaches have already been proposed [5], such as Test Case Selection (TCS), which selects a subset of the test cases for execution according to a specific objective; Test Suite Reduction (TSR), which reduces the size of the test suite; and Test Case Prioritization (TCP), which prioritizes test cases to be executed. The differential of TCP concerning TSR and TCS is that TCP uses the entire test suite. Every single test case will be considered in prioritization, thus reducing the risk of omitting some type of defect. As the main interest of this study is TCP, the paper will focus on its approaches.

TCP presents techniques that propose to order test cases based on a defined criterion, which can be fault detection rate, coverage rate, or probability of execution history, among others. Khatibsyarbini et al. [6] presented a Systematic Literature Review (SLR) in which they surveyed the main techniques employed in TCP, being Search-based the most widely employed [7,8], Coverage-based [9,10] and Fault-based [11], totaling together 53% as the main techniques used in the studies returned. According to the same study, one of the least explored techniques was Model based [12], totaling about 4% of the papers found, which is indicative of little exploitation of the technique in TCP. Besides, within Model-based techniques approaches, there are no specific references to approaches using Markov Chains. Markov Chains are usually applied on statistical testing for software usage [13], defining a stochastic model that is capable of modeling multiple probability distributions corresponding to estimated usage patterns. The idea is to treat software testing as a stochastic process. Whittaker and Thomason [13] adopted elements and defined a probabilistic relationship between these elements. Another approach that is found very much in the literature, is Controlled Markov Chains (CMC) to Software Testing [14]. The authors use the concept of adaptive testing applied for software reliability improvement with detected software defects being removed from the SUT during Software Testing. Thus, at each cycle (loop), the probabilities are updated to select or generate the next test cases.

The software can be modeled as a finite state discrete-parameter Markov chain [15]. Markov chain may represent the software system, considering its available states, while the arcs indicate transitions among states and are assigned probabilities that refer to the probability of a state to move to another. With these probabilities, Markov chains can be solved to obtain steady-state probabilities. Steady-state probabilities represent the percentage of time occupied by each state. This characteristic can be applied for TCP, using the probabilities to define priorities of what states are more active than others. So, test cases going through states which have high probabilities, tend to higher usage of those specific paths. Therefore, if one must choose test cases, it is interesting to exercise the paths that contain states with a high percentage of occupation. Once these characteristics are applied in TCP, some benefits can be achieved, such as those that can be seen in [16,17].

The objective of this work is to conduct a Systematic Literature Review (SLR) in order to understand how Markov Chains have been applied on TCP. Aspects such as approaches, developed techniques, programming languages, analytical and simulation results, and validation tests were investigated. The major contribution of this present work is to gather and explore the main test case prioritization techniques using Markov Chains that were and are being used now. Because of saving time and money, test cases prioritized according to a defined specification become essential. In addition, we believe that these SLR results can help to identify a body of knowledge to support future research in Markov Chains and TCP, providing a basis for other researchers as well as students who consider learning about and contributing to this area.

The remainder of this paper is structured as follows. Section 2 brings together the background to this paper, explaining Markov Chains, Software Testing, and Prioritization. Then, Sections 3 and 4 discuss the research method and Selection Process applied to perform the SLR. Data Extraction and Synthesis is explained in Section 5, analyzing the results obtained according to the research questions defined for this study. Section 6 reports a general discussion to highlight some research points, their implications, and limitations. Section 7 presents the search for related works. Lastly, conclusions and future directions for this research are presented in Section 8.

2. Background

In this section, the main concepts of this study are discussed.

2.1. Markov Chains

Stochastic processes play a major role in a significant number of applications. Markov Chains can be considered as fundamental to deal with such stochastic processes. They can be defined as probability models where results from successive experiments are dependent on each other such that a given experiment depends only on its immediate past. The underlying idea is the so-called Markov Property, also known as memory-less property, in which predictions on stochastic processes can see the future that is independent of the past and it is enough to know just the present. In other words, past and present are independent if the present is known. Markov Chains are capable of modeling uncertainty in many real-world systems that evolve in time [18–20].

Markov Chains can be modeled as Finite State Machines (FSM) [21] that are graphically illustrated by state-transition diagrams. The chains are modeled as a set of states and a set of transition arcs among the states. Transition arcs have labels that can be transition probabilities (Discrete-Time Markov Chain — DTMC) or transition rates (Continuous-Time Markov Chain — CTMC). In the case of DTMC, the system is observed at discrete points in time, and a change may occur from one state to another according to the probability defined on the corresponding transition arc leaving that state. When referring to CTMC, the state is changed only when an event occurs, i.e., it must be explicitly stimulated to change the present state to another. Such events are associated with transition rates that must follow an exponential distribution.

Both DTMC and CTMC can be solved using numerical methods [22] from which steady-state probabilities are obtained. These probabilities refer to performance metrics in the long run and such probability for each state represents the percentage of time occupied by each state. These steady-state probabilities will play an important role in prioritizing software test cases. Prioritization of test cases is the object of this study employing SLR, the main objective of the paper.

2.2. Software testing

Software Testing activities contribute to the quality of developed and implemented software codes. In this respect, Verification & Validation (V&V) plays an important role. V&V activities may be static (technical reviews, inspections, etc.) or dynamic (testing) [4]. Besides, V&V ensures that both the software model and the implemented product are in conformance according to the specification [3]. Testing software exhaustively is not feasible due to a large input domain. So, some techniques to select subsets must be employed bearing in mind the coverage aspect. Such techniques can be *Black-box Techniques* (test cases are generated from the input/output behavior, without the code) or *White-box Techniques* (code is necessary to generate the test cases and their assessment) [4]. The main interest will be on Black-box Techniques and several methods [3] may be employed to generate test cases as long as the specification is modeled utilizing FSM [21].

Many a time, depending on the method employed for generating test cases and if the software specification model is complex, a huge number of test cases will be generated. Unfortunately, in real world applications, despite the importance of exercising all the test cases on the implemented code, it is not always possible due to budget and time restraints. In cases such as these, there might be a need to devise a solution on which test cases should be exercised first.

2.3. Test case prioritization

Black box software test cases (generated by a tool from the specification, for example) must be exercised in the implemented software to have a verdict whether the implementation is in conformance with the specification. The exhaustive exercise of these cases in the implementation must consider time and available budget as well as human resources. Given that the number of software test cases generated can be large, demanding an impractical time for the execution and analysis of all cases, it is important to investigate alternatives to reduce the number of test cases, without losing the software quality and reliability. This aspect implies providing priorities for the choice of which cases should be exercised first and this depends on the perception and knowledge of the system by an independent testing team. In order to determine these priorities, the investigation of automatic and formal means of providing them for the choice of test cases is in order. Prioritizing test cases refers to choosing those cases that are more important based on some metric, but without decreasing the number of faults to be detected [23].

There are some initiatives to minimize the number of test cases or to prioritize these cases: neural networks [24]; Markovian process modulated by Markov [25]; indicators for planning tests using models that use time based on models of use of Markov Chains [26]; Monte Carlo simulation [27]; simulation to analyze coverage and cost in generating tests from Statecharts models [28]; analysis of distance functions [29]; methods to generate few and long test cases [30]; presentation of several prioritization techniques and search algorithms to prioritize test cases [31]; employment of data mining [32]; case-based reasoning [33].

Most of such solutions and approaches look for reducing the number of test cases while generating them utilizing some criteria. The question is what if there are already test cases generated. How to exercise just a portion of these already generated test cases, i.e., is it possible to classify the existing test cases? If so, depending on the time and resources, one can exercise only those test cases that have a higher priority than others that will not be exercised. Besides, this has an advantage, which is one can find out how many test cases can be exercised by calculating this number based on the available resources.

3. Research method

The research method conducted in this study was an SLR. SLR is a type of secondary study that uses a well-defined method to identify, analyze and interpret the available pieces of evidence in a way that is unbiased and (to a degree) repeatable. A secondary study is a study that reviews primary studies related to specific research questions to integrate/synthesize the evidence related to these questions [34]. The review was defined based on the guidelines given by Kitchenham and Charters [34] which involves three main phases: (i) **Planning:** refers to the pre-review activities and aims at establishing a review protocol defining the research questions, inclusion and exclusion criteria, sources of studies, search string and mapping procedures; (ii) **Conducting:** regards searching and selecting the studies, in order to extract and synthesize data from them; and (iii) **Reporting:** is the final phase and aims at writing up the results and circulating them to potentially interested parties.

In addition to the searches in the databases, backward snowballing from reference lists of Selected Studies was also applied. The reference lists of the studies can be analyzed looking for other relevant studies [34]. The process ends when no more relevant studies are found.

Next, we discuss the main protocol steps we performed for the SLR.

Table 1
Research questions and their rationales.

Nº	Research question	Rationale
RQ1	When and where the studies have been published?	The objective of this research question is to give an understanding of whether there are specific publication sources for these studies, and when they have been published.
RQ2	How Markov Chains have been applied to prioritize test cases?	This research question is to understand how Markov Chains have been applied on TCP. The main aspects investigated in this question are the approaches used in each study.
RQ3	What are the algorithms and/or tools to support TCP using Markov Chains in each study?	Highlights the main methodologies currently used to provide TCP supported by Markov Chains, such as developed techniques, programming languages, analytical and simulation results, and validation tests. This is useful for researchers and practitioners that intend to accomplish new initiatives of Markov Chains for TCP, as well as to guide future research towards new methods, tools, to fill the existing gaps.

Research questions. Table 1 presents the Research Questions (RQ) that this SLR aims to answer, as well as the rationale for considering them.

Considering the importance of having a well-structured research question to ensure greater range and specificity, we have defined the PICO (Population, Intervention, Comparison, and Outcomes) for this SLR. PICO is suggested for detailing the research question elements in order to support developing the review protocol [34–36]. The *population* is the group affected by the *interventions*, which refer to what is being investigated. The *comparison* is a reference parameter or a data set that initially exists. And finally, the *intervention* results are expressed by the *outcomes*. PICO of our SLR is presented as follows. PICO of our SLR is presented as follows.

Population: Researchers and *software testing* professionals who use or are interested in using *Markov Chains* in TCP.
Intervention: Applications of Markov Chains in TCP.
Comparison: It is not applicable.
Outcome: A summary of the main evidence that presents approaches to Markov Chains applications for TCP. We also want to reveal the main algorithms and/or tools to support TCP using Markov Chains.

Based on population and intervention, mainly, we identified the main areas to be investigated in this SLR, namely, "Software Testing" and "Markov Chains". The synonyms for each area were identified and separated into groups. Thus, these groups were considered in the development of the search string, as detailed below.

Search string. The search string considered in this research explores two areas — "Software Testing" and "Markov Chains" (see Table 1). To elaborate the search string, we started with an initial set of terms, and this set was iteratively improved until all relevant pre-selected studies were found. The pre-selected studies are part of a control group. We have two articles in the control group [16,17]. We used a control group to calibrate the search string, that is, when the control group studies were not retrieved, the string was calibrated (adjusted). In addition, for the elaboration of the search string, we only considered terms in English. This same criterion was used in the selection process (EC4 — "Study is not written in English").

Sources. Search was done in five electronic databases that were considered the most relevant according to Dyba et al. [37], namely:

Table 2
Areas and keywords.

Areas	Keywords
Software Testing	"Software Testing" OR "Software Test" OR "Test Case" OR "Test Sequence"
Markov Chains	"Markov Chains" OR "Markovian"
Search String: ("Software Testing" OR "Software Test" OR "Test Case" OR "Test Sequence") AND ("Markov Chains" OR "Markovian")	

IEEE Xplore (http://ieeexplore.ieee.org);
ACM Digital Library (http://dl.acm.org);
Scopus (http://www.scopus.com); and
ScienceDirect (http://www.sciencedirect.com);
SpringerLink (https://link.springer.com/).

Selection criteria. The selection criteria are organized in one inclusion criterion (IC) and seven exclusion criteria (EC).

The inclusion criterion is:

(IC1) Study must include TCP by employing Markov Chain;
(IC2) Studies published until July 2021.

The exclusion criteria are:

(EC1) Study has no abstract;
(EC2) Abstract or extended abstract without Full Text;
(EC3) Study is not a Primary Study. The studies considered editorials, summaries of keynotes, tutorials, posters were excluded;
(EC4) Study is not written in English;
(EC5) Study is a copy or an older version of another publication already considered. In these cases, the most current version is considered;
(EC6) No access to the Full Paper; and
(EC7) Study mentions Markov Chains but not employed for TCP.

Data storage. The publications returned in the searching phase were cataloged and stored appropriately. We used a data extraction form to gather all relevant data from the identified studies and manage the selection process. Some of the main information presented in this form was followed by a protocol, identifier (id) for each returned study, bibliographic reference, and answers to research questions. This catalog helps us in the data extraction and synthesis procedures and may be used by potentially interested, for example, for updating or replication. A summarized version of this form is available at https://doi.org/10.5281/zenodo.5783881.

Assessment. Before conducting the SLR, we tested the review protocol. This test was conducted to verify its feasibility and adequacy, based on a pre-selected set of studies considered relevant to our investigation (control group). SLR process was conducted by four authors and the other two carried out the validation. In addition, the results of each reviewer were then compared to detect possible bias. Kappa coefficient [38] was also performed to measure the level of agreement between the results obtained from the reviewers in the selection process. Kappa coefficient is a statistical measure of inter-rater agreement for qualitative items. It is a more robust measure than simple percent agreement calculation since it considers the agreement occurring by chance. A Kappa coefficient value of 1 represents total agreement. Values close to 0 indicate no agreement. For calculating the Kappa coefficient, we considered the result set from Stage 2 (application selection criteria), which contains 94 papers. So, in our case, the overall kappa coefficient obtained was 0.67, whose interpretation, according to Landis and Koch [38], is to have a substantial agreement.

4. Selection process

Initially, the search string presented in Table 2 was applied to the electronic databases, considering three metadata fields: title, abstract, and keywords. The search string went through syntactic adaptations according to the particularities of each source. We considered the studies published until July 2021 (IC2). The main stages performed in this SLR are shown in Fig. 1.

As a result of searching the selected sources, a total of 480 publications were returned, out of which 29 were from IEEE Xplore, 10 from ACM, 47 from ScienceDirect, 382 from Scopus, and 12 from SpringerLink. In the 1st stage duplicated studies were eliminated, resulting in 368 studies (reducing approximately 23.3%). In the 2nd stage, the selection criteria (inclusion and exclusion criteria) were applied for title, abstract, and keywords, leading to 94 studies (reducing approximately 74.5%). In the 3rd stage, the selection criteria were applied considering the full text, resulting in a set of 10 studies (reducing approximately 89.4%).

The objective of this SLR is to summarize how Markov Chains approaches have been applied on TCP. Many studies returned by the search string use Markov Chains in the context of Software Testing. However, in the 3rd stage of the selection process, most studies were removed by the exclusion criterion EC7, since although the study addressed Markov Chains in Software Testing, the study focus was not applied to prioritize test cases. Hence, 78 studies were removed by criterion EC7, since Markov Chains was applied in other contexts, for example, in the work [39] it is a proposed test case generation method to evaluate the performance of mobile applications. In this work, Markov Chains are used to evaluate the performance of applications. In the case of [40], the work is on Partition Testing using Markov Chain. And as an example of a case we often encounter, the study [41] applies system reliability using Markov Chain usage models. However, none of these works use Markov Chains for TCP.

Over these 10 studies that remained in 3rd stage, we performed backward snowballing in 4th stage. In the snowballing process, we looked at all the references from the 10 studies selected (306 references) and we applied the selection criteria in the title, abstract, and keywords, leading to 6 studies. The selection criteria were again applied to the 6 studies considering the full text analysis, resulting in 2 studies.

As a result, we got to 12 studies to be analyzed (10 from the sources and 2 from backward snowballing). Table 3 summarizes the stages and their results, showing the progressive reduction of the number of studies throughout the selection stages. Table 4 presents the 12 selected studies.

5. Data extraction and synthesis

As already mentioned previously, after applying the inclusion and exclusion criteria, 12 studies remain to be explored in this review (see Table 4). With these selected studies, we analyzed each one in order to answer the research questions presented in Table 1.

5.1. (RQ1) When and where the studies have been published?

In order to offer a general view of the efforts in the area investigated, a distribution of the 12 selected studies is shown in Table 5. Table 5 includes the type, where the studies were published, country of the institution of the first author, and the distribution of the 12 selected studies over the years. 10 different publication sources were identified. It is worth pointing out that four studies were published in the Journal *Information and Software Technology*, showing that it can be a well-established forum for discussing the topic.

Regarding the year of publication of the studies, it has spread over the last two decades (from 2000 to 2017). This indicates that prioritization of test cases, regardless of the applications and techniques used, has only been recently employed. Among them, the most recent are papers A_6 and A_7, both published in 2017. And the oldest papers A_1, A_2, A_{12} and A_{13} in 2000. Although the search was carried out until July 2021, the last prioritization study using Markov Chain returned is from 2017, which shows an exploration gap in the last four years.

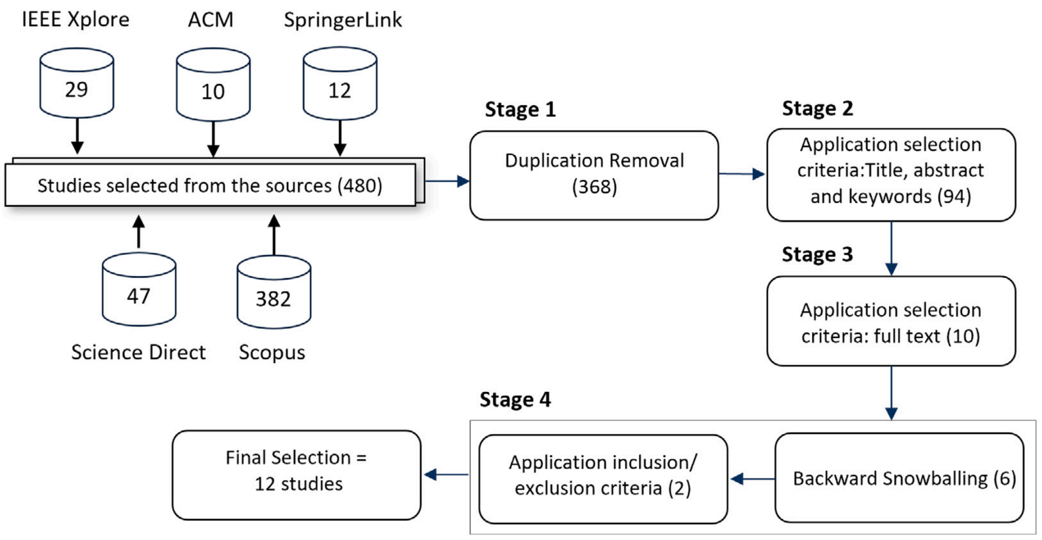
IEEE Xplore
29
ACM
10
SpringerLink
12
Studies selected from the sources (480)
47
Science Direct
382
Scopus
Stage 1
Duplication Removal (368)
Stage 2
Application selection criteria:Title, abstract and keywords (94)
Stage 3
Application selection criteria: full text (10)
Stage 4
Backward Snowballing (6)
Application inclusion/ exclusion criteria (2)
Final Selection = 12 studies

Fig. 1. Search and selection SLR process.

Table 3
Results from the selection stages.

Stage	Applied criteria	Analyzed content	Initial number of studies	Number of excluded studies	Final number of studies
1st	Duplicate removal	Title, abstract, keywords	480	112 (23.3%)	368
2nd	IC1, EC1–EC4	Title, abstract, keywords	368	274 (74.5%)	94
3rd	IC1, EC5–EC7	Full text	94	84 (89.4%)	**10**
4th (a)	Snowballing, IC1, EC1–EC4	Title, abstract, keywords	(306) References from 10 studies	296 (96.7%)	6
4th (b)	Snowballing, IC1, EC5–EC7	Full text	6	4 (66.7%)	**2**
Final Result: 10 (electronic databases) + 2 (backward snowballing) = 12 selected studies					

Table 4
Selected studies.

Study	Title	Reference
A_1	Measuring complexity and coverage of software specifications	[16]
A_2	Software dependability evaluation based on Markov usage models	[42]
A_3	Optimal Software Testing and adaptive Software Testing in the context of software cybernetics	[43]
A_4	Optimal and adaptive testing for software reliability assessment	[44]
A_5	Optimal test profile in the context of software cybernetics	[45]
A_6	Test suite prioritization for efficient regression testing of model-based automotive software	[17]
A_7	Statistical prioritization for software product line testing: an experience report	[46]
A_8	Automated generation of software testing path based on ant colony	[47]
A_9	A model driven approach for system validation	[48]
A_{10}	Reducing safety-critical software statistical testing cost based on importance sampling technique	[49]
A_{11}	Application of Markov chain Monte Carlo random testing to test case prioritization in regression testing	[50]
A_{12}	A Controlled Markov Chains Approach to Software Testing	[14]

Table 5
Sources and types of selected articles.

Study	Type	Publication source	Year	Country
A_1	Journal	Information and Software Technology	2000	USA
A_2	Journal	Performance Evaluation	2000	Austria
A_3	Journal	Information and Software Technology	2002	China
A_4	Journal	Information and Software Technology	2004	China
A_5	Conference	Second Asia-Pacific Conference on Quality Software	2001	China
A_6	Conference	International Conference on Software Analysis, Testing and Evolution	2017	Germany
A_7	Journal	Software and Systems Modeling	2017	France
A_8	Congress	Second International Congress on Technology, Communication and Knowledge	2016	Iran
A_9	Conference	International Systems Conference (SysCon)	2012	USA
A_{10}	Journal	Advanced Materials Research	2012	China
A_{11}	Journal	IEICE Transactions on Information and Systems	2012	USA
A_{12}	Journal	European Journal of Operational Research	2011	China

To better understand the possible relationship between the 12 selected studies, we created a citation relationship graph presented in Fig. 2. Fig. 2 highlights when a study cites another study, for example, study A_3 was cited by study A_{12}. Analyzing the citations presented in Fig. 2, we can see that there is no significant relationship between the 12 selected studies. It is believed that this happens because the nature of the test case prioritization applications in each study is quite different. The approach to TCP of each study is presented in Section 5.2.

An important aspect to better understand a research segment is where this type of work has been carried out. Table 4 shows the countries of the institutions in which the first author of each selected study is affiliated. Only 12 studies remained for final analysis, where

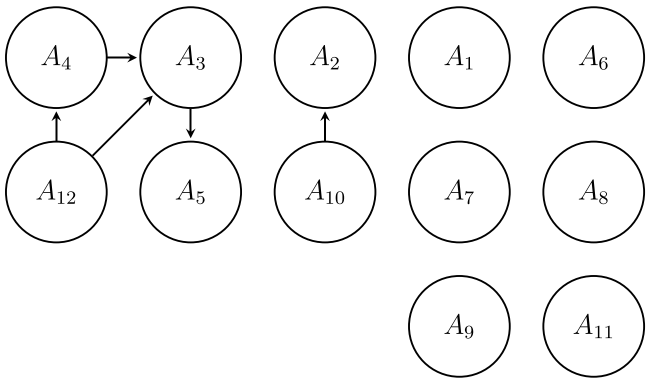

Fig. 2. Relationship between citations of selected studies.

Table 6
Distribution of articles by application.

Context	A_1	A_2	A_3	A_4	A_5	A_6	A_7	A_8	A_9	A_{10}	A_{11}	A_{12}
Usage model	✓	✓								✓		
Controlled MC			✓	✓	✓							✓
Model-based testing						✓		✓	✓			
Regression testing						✓					✓	
Statistical testing							✓			✓		
Random testing											✓	

institutions of these authors are in six different countries. Among them, the largest concentration is in China, containing five, followed by the USA with three in the final group of selected articles. The remainder of the distribution is made up of four more containing one article each.

The concern with testing has always been a concern of any device, be it technological, mechanical, or even organizational. However, the increase of such applications inserted in humanity in an interconnected way to life and survival, made that exhaustive testing was essential. Table 5 shows the year of publication of the selected studies. The first of them, published in 2000 [16], demonstrates the beginning of the concern with prioritizing the countless number of tests of any tool used by humanity at the beginning of the new millennium. Likewise, we can see a trend in the first decade (half of the selected works). This beginning of concern with the main tests to be carried out within increasingly complex devices is noticeable in different regions of the globe (see Table 5). Whether they are related to each other or not (see Fig. 2). This aspect demonstrates independence linked to the need for the topic. In the second decade, the same occurs. That is, different approaches concerned with prioritizing test cases in different regions of the planet are also noticeable in Table 5. Fig. 2 shows us how the concern with prioritizing test cases arose from the very need of the time, that is, without so much influence from previous studies, as we can see by only four citations between them that exist in the works, which demonstrates that the trend occurred independently around the world.

5.2. (RQ2) How Markov Chains have been applied to prioritize test cases?

To understand key initiatives of using Markov Chains in the priority test cases context, we investigate which approaches have been proposed or applied in the selected studies. Table 6 shows the main contexts in which test case prioritization using Markov Chains are inserted. It shows us how this subject has resources to be useful in different applications.

The following is a brief presentation of the approach addressed by each study. We separate the articles according to the main context in which they are inserted. Some papers are inserted in more than one context, so we will separate them according to the main one.

5.2.1. Usage model

Coverage tests, using Markov Chain usage models, explore test sequences through the intended use of software, rather than testing a specific code implementation. In A_1 [16], Markov Chains are used to cover the states and arcs of a model. Each possible path in the model, that is, traveling from state to state through the arcs that connect them, has a probability of occurring. From the perspective of prioritization, the authors seek to find, among the infinity of a model's test cases, several test cases that cover all the model's behavior employing the probability of the test cases to occur. This idea is an example of the popular *Chinese postman* problem for targeted graphs [51].

Still, in the context of Markov Chain usage models, the study A_2 [42] developed a technique for computing optimal test transition probabilities. The author performs the optimization of test cases using three measures: *(i) risk; (ii) safety; and (iii) reliability*. This information is previously added in a probabilistic way, such as a probability of failure and loss. Thus, the study statistically prioritizes test cases using these new parameters.

5.2.2. Controlled MC

In study A_3 [43], the author explores **Controlled Markov Chains (CMC)** in the context of cybernetics software. This context explores the interplay between software and control, with an attempt to apply cybernetic or control-theoretical approaches to solve Software Engineering problems. The author also introduces an adaptive software testing strategy for state transitions to behave like a Markov chain. This work presents a potential solution for prioritizing test cases, addressing the problem as a control problem in the context of cybernetics software. Still, in the same context of **CMC**, the study A_4 [44] following an optimal software test idea, shows that the **CMC** approach is also applicable to dealing with optimal software testing problems for software reliability assessment. Then, the code of the software under test is frozen and the number of prioritized test cases is given. The study A_5 is the first of this trilogy of applications in the context of **CMC**. It shows the initial concepts used in these two previous ones (A_3 and A_4). The analyses of this paper suggest that nonhomogeneous Markov models should also be considered for software operational profile modeling.

Again inserted in the context of CMC, the study A_{12} [14] performs, among other things, prioritization of test cases. Using the same methodology as in A_3, A_4, and A_5, the author presents a new case study of adaptive testing and discusses why the **CMC** approach or control-theoretic approach can work in practice.

5.2.3. Model-based testing

The study A_8 [47] presents a proposed solution based on the ant colony optimization algorithm and **model-based tests**. This algorithm is a probability-based heuristic, created to solve computational problems that involve searching for paths in graphs. This algorithm was inspired by observing the behavior of ants when they leave their colony to find food [52]. With this approach, the study that is based on model-based tests, the software under test is considered as a module and this optimization with Markov Chains is used to produce prioritized test paths. With this, the coverage of the model's behavior is complete, and the cost of the test is reduced.

It is considered as a system validation, a series of checks to verify the functioning/compliance of an application with its purpose and functions. Within this context, in the study A_9 [48], the authors propose a methodology in which the usage of a system is modeled through a Markov Modulated Markov Process (MMMP). With these modeled diagrams, first, the test cases are generated, and then these cases are prioritized. The state transition probabilities are estimated using the Expectation–Maximization algorithm [53]. The prioritization of test cases is done following what is described in [23]. That is, considering that a system is modeled in a probabilistic sense, an example of a prioritization criterion depends on the probability of a generated test sequence being part of the usage.

5.2.4. Regression testing

The study A_6 [17] is motivated by automotive software applications. The authors introduce a new method for automatic prioritization of test cases of an original test suite for efficient **regression testing**. The test suite prioritization method is based on two key principles: (i) *Fault activation analysis* — A test case should stimulate an error in an updated block; and (ii) *Error propagation analysis* — the most important is that the stimulated error should propagate to a code location where it can be detected. This second part is the most important, considering that this is where the prioritization of test cases through Markov Chains occurs. In this part, the authors use software (ErrorPro^{TM} [54]) that automatically creates a set of discrete-time Markov Chain (DTMC) models using the *Dual-graph Error Propagation Model (DEPM)* and computes the mean number of errors that will reach the selected data storages, associated with the assertions.

5.2.5. Statistical testing

Software Product Lines (SPL) are families of systems that have several characteristics in common, but each member of that family has its own features. Thus, within an SPL family, given the huge number of test cases generated by the combinations of these characteristics, the practice of software tests that cover all of them becomes impracticable. Thus, a prioritization of test cases naturally becomes essential for the main characteristics of these products to be tested. In A_7 [46], the authors assess the integration of usage models through Markov Chains to derive **statistical test** approaches for SPLs. Thus, this approach allows cases to find the most probable and the rarest tests in a specification.

In the study, A_{10} [49], prioritization of test cases is done by optimizing the transition probabilities of the Markov Chain. The technique is based on Importance Sampling (IS). In statistics, importance sampling is a general technique for estimating properties of a particular distribution, while only having samples generated from a different distribution than the distribution of interest. The method proposed in the study is efficient in reducing the cost of the **statistical test** of security-critical software and can also produce an impartial reliability test result. The results of the field test show that the method reduces the cost of the test while producing unbiased estimates of software reliability.

5.2.6. Random testing

The A_{11} [50] proposes the test case prioritization in regression testing. The motivation of the proposal is the big problem found in all studies, a large amount of tests suite to be executed which causes a great operational cost, now in regression testing. Thus, highlighting the importance of prioritizing test cases so that correct analysis is feasible. The paper discusses the applicability of Markov chain Monte Carlo **random testing** (MCMC-RT) to the test case prioritization, which is an alternative random-coverage-based algorithm. The basic idea of MCMC-RT is to estimate the distribution of fault location from the past outcomes of software test execution, and, as a result, it leads to test cases as evenly as possible across the input domain.

5.3. (RQ3) What are the algorithms and/or tools to support TCP using Markov Chains in each study?

The main objective of this research question is to bring to light the main methodologies used in the application of Markov Chains for prioritizing test cases. Table 7 compiles the main information related to this goal. Information refers to developed techniques, programming languages, analytical and simulation results, and validation tests. An important piece of information that could be present in the table would be an open-source application that was developed. However, none of the studies produced one.

From the second column of the table (“Context”), we can infer that 1/3 of the studies refer to Controlled Markov Chains (Controlled MC) (A_3, A_4, A_5 and A_{12}). In these cases, state transition probabilities are functions of control inputs, making this the most present approach. Three studies have more than one main context in which (A_6, A_{10} and A_{11}). Finally, continuing the second column, the least used approach, present in only one study (A_7), is Statistical Testing.

Except for A_5, A_7, and A_{10}, all others have generated some new techniques in which Markov Chains are used in the test case prioritization process. The techniques are shown in the third column of the table (“Developed Techniques”). The application of Markov Chains plays a significant role within each technique. As a major highlight, the study A_6 has the most directly applied technique, where automation of test cases prioritization is developed.

Most of the studies do not indicate the programming language used in their development. Only three indicate this information: A_3, A_6 and A_9. In A_3 and A_9, MATLAB is used and A_6 uses MATLAB Simulink. With this information, we can better identify how the generated techniques were developed, in terms of paradigms, for example. However, we can see that MATLAB is the most used for this use of Markov Chains in test cases prioritization.

The last three columns of the table bring us how the results were generated (“Analytical results”, “Simulation results”, and “Validation tests”). Only studies A_7, A_8, and A_{11} did not generate results analytically. This demonstrates a lot of mathematical appeal in most cases. On the other hand, studies A_5 and A_{12} were the only ones that did not use some numerical form to demonstrate their results. This combination of analytical and numerical results and in most cases demonstrates how complete the results are. And finally, which brings even more assurance and confidence in the techniques developed and, in the results presented, only study A_5 does not present a validation test.

6. Discussions

Model Based Testing can develop test cases, based on specification, in the early phases much before the software is ready. These test cases must be exercised on the implemented software to provide a verdict of whether the software is in conformance with its specification. Depending on the criteria (to generate the tests), the number of test cases can be large. Depending on the time and budget restraints, it is not always possible to exercise the entire set. In such cases, TCP can play an important role. What are the strategies to understand which test cases must be run before the others, i.e., which should be exercised based on some priority?

The approach the paper described here discusses the use of Markov Chains. The idea originated because of the tool WEB-PerformCharts [55, 56] developed by some of the authors. The tool has two options to be used. The first one is to generate test cases given a software model represented as a Finite State Machine [21] or Statecharts [57]. The second refers to steady-state probabilities by solving a Markov Chain. In both cases, the model can be represented as a Finite State Machine (FSM) or Statecharts. When the representation is modeled in Statecharts, there is a translation to FSM. The tool was heavily used to generate test cases for space applications within the National Institute for Space Research (INPE).[2] For the option of test cases, several criteria, such as Transition Tour, Switch Cover, H-Switch Cover, Distinguished Sequence, Unique Input Output are implemented and used [58–61]. Depending on the criteria, several test cases are generated. If the same FSM is used with stochastic transition events or transition probabilities, it is possible to determine the steady-state probabilities by solving a Markov Chain. One must remember the memory-less property of Markov Chains and therefore, the labels on the transition arcs must follow an exponential distribution. These steady-state probabilities can in turn be used to prioritize the test cases as they represent the percentage of time each state is active.

Among the 12 resulting studies, most of them are from Journals with publications years between 2000 and 2017 which can be considered

[2] www.gov.br/inpe.

Table 7
Information on the methodology used in the development of works on the use of Markov Chains for prioritizing test cases.

ID	Context	Developed technique	Programming language	Analytical results	Simulation results	Validation tests
A_1	Usage model	Measure of the complexity of a software specification	–	✓	✓	✓
A_2	Usage model	Optimal test transition probabilities in a Markov software usage model	–	✓	✓	✓
A_3	Controlled MC	Adaptive software testing	MATLAB	✓	✓	✓
A_4	Controlled MC	Adaptive software testing (extended)		✓	✓	✓
A_5	Controlled MC	–	–	✓	–	–
A_6	Model-based testing, Regression testing	Automatic prioritization of test cases	MATLAB Simulink	✓	✓	✓
A_7	Statistical testing	–	–	–	✓	✓
A_8	Model-based testing	Ant colony optimization algorithm and model-based testing	–	–	✓	✓
A_9	Model-based testing	Model driven approach for system validation	MATLAB	✓	✓	✓
A_{10}	Usage model, Statistical testing	–	–	✓	✓	✓
A_{11}	Regression testing, Random testing	Markov Chain Monte Carlo Random Testing	–	–	✓	✓
A_{12}	Controlled MC	–	–	✓	–	✓

that prioritization of tests cases as recent. Among the journals, more papers were submitted to IST. One can observe a weak relationship among the Selected Studies as the applications to prioritize test cases are very distinct. The analysis also pointed out that publications are from 8 countries. The analysis of the studies also indicated that the approach proposed in the paper described here is quite different from other publications. The paper also shows the publications that refer to important information on developed techniques or tools, programming languages used to develop the technique, whether results used simulation or calculated analytically, etc. While evaluating the context used by the studies, we observe that most of the studies employed Controlled Markov Chains, and the least used context was Statistical Testing. All the Selected Studies employed Markov Chain in some context but one that automated the prioritization of test cases was directly related to our paper. Concerning programming languages, MATLAB was the most used. In terms of results, most depended on simulation and not analytical calculations.

Our focus has always been, Model Based Testing (MBT) and it is very well established within the testing community. Maybe it is worth exploring any new suggestions of methods to represent reactive systems. Our knowledge lies in FSM and Statecharts. Most of our examples come from Space Applications due to the mission of the space institute of one of the authors. However, it would be good if there were more practical approaches along with free software systems or frameworks available to the general interested public. We intend to make our tool WEB-PerformCharts available to anyone willing to explore to generate steady-state probabilities and/or test cases.

Markov chains have been in use to model performance in a wide variety of applications such as traffic flows, communications networks, genetic engineering, and queueing systems. The same is true when the application is a software system. The potential of using Markov chains in other areas can greatly benefit software systems as well. Even though an important contribution Whittaker and Thomason [13] dates in the 1990s, one can say that associating Markov chains with software testing is not yet very consolidated. This is one of the reasons for conducting a Systematic Literature Review on this topic. As shown in the analysis of the resulting literature, not much has been done.

This opens some avenues for those researchers interested in exploring this topic. We intend to explore the ideas presented in the selected studies as well as our own to contribute to our computational tool, WEB-PerformCharts to include test case prioritization. This approach may be attractive to companies that develop software and come up with budget restrictions putting pressure to deliver a software product with quality and duly validated. The first idea is to use the existing test cases already generated from our tool for certain space application systems and model a Markov Chain of these systems. With the steady-state probabilities, we must develop an algorithm of ordering the test cases based on those that traversed through the state with higher probabilities.

6.1. Threats to validity

SLR presented in this paper has some limitations. We discussed the validity concerning the four groups of common threats to validity [62]: internal validity, construct validity, external validity, and conclusion validity.

Internal Validity. The study selection was initially performed by a part of the authors (four authors), and thus some subjectivity could have been embedded. To reduce this subjectivity, the other two authors performed selection in a sample (approximately 30%) and then compared to detect possible bias. In addition, an analysis of the degree of conformance was performed to measure the level of agreement between the results obtained from the reviewers in the selection process. Kappa coefficient was calculated (Section 3). This analysis considers the agreement occurring by chance, so it is a more robust measure than a simple percent agreement calculation. For calculating the kappa coefficient, we considered the result set from Stage 2 which contains 92 papers. In our case, the overall kappa coefficient obtained was 0.67. According to Landis and Koch [38], this value is considered a substantial agreement. The value obtained in the Kappa calculation demonstrates that the reviewers are qualified to evaluate the characteristics of interest in this SLR, that is, the reviewers had no difficulty in verifying whether the study was dealing with TCP approaches using Markov Chains.

Once we retrieved the papers, we had to decide if the studies meet our selection criteria (IC and EC). Some studies were the subject of intense debate. 78 studies were discussed among the authors and removed based on criterion EC7. As mentioned in the selection process

Table 8
Tertiary studies — Areas and Keywords.

Areas	Keywords
Software Testing	"Software Testing" OR "Software Test" OR "test case" OR "test cases" OR "test sequence"
Markov Chains	"Markov Chains" OR "Markovian"
Secondary Study	"Systematic Review", "Literature Review", "Systematic Mapping", "Mapping Study", "Systematic Map", "Meta-analysis", "Survey" and "Literature Analysis"
Search String: ("Software Testing" OR "Software Test" OR "Test Case" OR "Test Cases" OR "Test Sequence") AND ("Markov Chains" OR "Markovian") AND ("Systematic Review" OR "Literature Review" OR "Systematic Mapping" OR "Mapping Study" OR "Systematic Map" OR "Meta-analysis" OR "Literature Analysis")	

(Section 4), although the study addressed Markov Chains in Software Testing, the study focus was not applied to prioritizing test cases.

Construct Validity. Some terminological problems in the search strings or the number of electronic databases used may have led to missing some primary studies. The exclusion of other sources makes the review more repeatable, but possibly some valuable studies may have been left out of our analysis. Although this limitation exists, we believe that the studies discussed in this SLR provide a snapshot of empirical research on outcomes and impacts of existing research on Markov Chains and TCP. Even though, to minimize these problems, we performed backward snowballing in the Selected Studies. The backward snowballing allowed complementing the selected studies with potentially other studies indexed in different sources.

External Validity. Although Springer's electronic database is an important source of studies on the topics discussed in this SLR, we only had access to studies considered content type equal to "Article" and "Conference paper". Studies that have been published in "Chapter" format are restricted. This fact may also have led to missing some primary studies. However, as discussed in construct validity, we believe that the discussed selected studies provide an important vision of empirical research on Markov Chains and TCP.

Conclusions Validity. In the same way, as during the study selection stage, some subjectivity may have occurred in the data extraction. To reduce this subjectivity, the data extraction was performed by more than one author considering different samples (approximately 30%) and later compared to detect possible bias.

7. Related work

In this paper, we presented an SLR, i.e. a secondary study that is based on analyzing research papers (primary studies) [34]. Our SLR aims to identify and classify all research related to understanding how Markov Chains have been applied on TCP. Hence, before accomplishing the secondary study presented in this paper, we performed a tertiary study looking for other secondary studies investigating the same research topic, that is, Software Testing and Markov Chains. Tertiary studies are considered as a review that focuses only on secondary studies (SLR or Mapping Studies), i.e., it is a review about other secondary studies [34].

In this tertiary study, we used the search string shown in Table 8, which was applied in three metadata fields (title, abstract, and keywords).

We applied the search string in the following scientific databases: IEEE Xplore, ACM Digital Library, Scopus, ScienceDirect, and SpringerLink. ACM Digital Library was the only database without returning any study. In the other databases, we found 5 results in IEEE Xplore, 9 in ScienceDirect, and 13 in Scopus. Nevertheless, not even one of the results was related to the goal of the tertiary study, i.e., related to SLRs or mapping studies about Markov Chains and Software Testing simultaneously.

As we did not find any secondary study addressing Markov Chains and TCP, we decided to investigate secondary studies that deal with Markov Chains and TCP separately. We started looking for secondary studies in TCP using the following search string: *("Test Case Prioritization") AND ("Systematic Review" OR "Literature Review" OR "Systematic Mapping" OR "Mapping Study" OR "Systematic Map" OR "Meta-analysis" OR "Literature Analysis")*. The string was applied in three metadata fields (title, abstract, and keywords). The same five electronic databases were searched. A total of 32 were identified. After eliminating duplication and applying the selection criteria, we reached 18 papers presenting secondary studies on TCP. We analyze the authors' different interests in summarizing evidence on TCP. We grouped these interests in the following categories: (i) Approaches (3 studies); (ii) Regression Testing Techniques (5 studies); (iii) Model-based Testing (2 studies); (iv) Web Services (2 studies); (v) Genetic algorithms (2 studies); (vi) Event sequences (2 studies); Requirements (2 studies); and Others (3 studies), including Continuous Integration environments (TCPCI), Evolutionary Algorithm, Industrial relevance, and Applicability. In some cases, we classified the studies in more than one category since the study covered more than one interest.

Now, we also looked for secondary studies in Markov Chain, we used the following search string: *("Markov Chain") AND ("Systematic Review" OR "Literature Review" OR "Systematic Mapping" OR "Mapping Study" OR "Systematic Map" OR "Meta-analysis" OR "Literature Analysis")*. The same five electronic databases were searched, and 37 studies were returned. Duplications were eliminated, resulting in 32. Two papers were presenting secondary studies on Markov Chains. In one of these papers [63], the SLR conducted was about the application of Hidden Markov Models (HMMs) in the field of sentiment analysis. HMM is a Markov-based probabilistic graphical model, and the respective literature review analyzed use cases applied to sentiment classification. The other paper [64] studies the Business Processes Modeling applied to Small Medium Enterprises (SME). It relates analytical management of computational techniques from a systematic review about Markov chain modeling and other methods such as the game-theoretic analysis, the probabilistic modeling, and the Cognitive Maps methodology. However, the model proposed as a result of the SLR, in this case, is Petri nets-based.

Based on the results of these two investigations, one can say that there is a diversity of secondary studies in TCP and Markov Chains. However, when we combined these two areas, no secondary studies were identified.

8. Conclusions

In this paper, we have reported the results of an SLR on TCP using Markov Chains. From the defined search string, a total of 468 studies obtained from five databases were returned (IEEEXplore, ACM, ScienceDirect, SpringerLink, and Scopus), of which only 12 persisted until the last selection step. For this study, 3 research questions were defined, whose objective is not only to clarify the technologies and methodologies involved in the application of Markov Chains and the prioritization of test cases but also to understand how research related to this field is being distributed and studied by the around the world.

The main objective of this work was to investigate all the approaches and applications related to this topic, so that, based on the correlations between the studies found, we could trace possible future directions regarding this topic. Our motivation for this was the fact that Test Case Prioritization is a profitable field from the point of view of the entire software development cycle, as its application can be useful in many ways, such as saving time and budget since the goal is to optimize the process of software testing, which often consumes a good deal of effort on the part of the development team.

Overall, the major contribution of this study was to elucidate the context of the application of this subject. However, based on data extracted from each of the 12 studies analyzed, we found that there is no strong relationship between any of the studies. This idea is based on several findings throughout the conduct of this SLR, among them, we can mention the fact that the studies were conducted independently of each other: although these works have used Markov Chains at some stage of prioritization, they are not related to each other. Furthermore, we realized with our results that the applications of Markov Chains in the prioritization of test cases vary a lot, since their motivation concerning the developed processes.

In a general way, we can summarize the findings of this research below:

- Regarding the distribution of research related to this topic around the world, the research centers dedicated to this subject are concentrated in 3 of the 6 continents. In addition, from a temporal point of view, activities were concentrated between 2000 and 2017, and there was an interval without any publication between 2003 and 2011. However, with this SLR's data, there is a gap time since the publication of the last study analyzed, in 2017.
- Concerning the fields of application of this subject, 6 forms of approach were found: Controlled MC, Usage Model, Model-Based Test, Regression Test, Statistical Test, and Random Test. Among them, the most used were Usage Models, Controlled CM, and Test-Based Models. Of these works, about 67% developed some technique in their research. This demonstrates how these approaches are directed to different applications.
- Regarding the technologies used, not all studies have detailed this point. However, based on the extracted data, the approaches used were well diversified: Measure of the software specification, Optimal Test Transition Probabilities, Adaptive Software Testing, Automatic Prioritization, Ant Colony Optimization, Model Driven approach, and Monte Carlo Random Testing

Based on the analysis of all 12 studies, we realized the benefits of applying Markov Chains for TCP. Thus, as a proposal for future work, we will develop a test case generation and prioritization technique using Markov Chains. Our technique in the initial testing phase will use the probabilities of each test sequence to perform the ranking.

Finally, we believe that results from this SLR can strongly help to identify a body of knowledge to support future research. Learning as much as possible from other domains related to the topic and providing a basis for other researchers as well as students who consider learning about and contributing to this area.

CRediT authorship contribution statement

Gerson Barbosa: Conceptualization, Formal analysis, Validation, Resources, Writing – original draft. **Érica Ferreira de Souza:** Conceptualization, Methodology, Supervision, Formal analysis, Validation, Resources, Writing – original draft, Project administration. **Luciana Brasil Rebelo dos Santos:** Conceptualization, Formal analysis, Validation, Resources, Writing – original draft, Writing – review & editing. **Marlon da Silva:** Conceptualization, Formal analysis, Validation, Resources, Writing – original draft, Writing – review & editing. **Juliana Marino Balera:** Conceptualization, Formal analysis, Validation, Resources, Writing – original draft. **Nandamudi Lankalapalli Vijaykumar:** Conceptualization, Formal analysis, Validation, Resources, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Dr. Érica Ferreira de Souza is supported by a research grant from the Brazilian funding agency CNPq, Brazil, with reference number: 432247/2018-1.

References

[1] I. Eee, Standard Glossary of Software Engineering Terminology, IEEE Software Engineering Standards & ollection, IEEE, 1990, pp. 610–612.
[2] C. Baier, J.-P. Katoen, Principles of Model Checking, MIT Press, 2008.
[3] M. Delamaro, J. Maldonado, M. Jino, Introdução ao Teste de Software, Rio de Janeiro, RJ, 2007.
[4] A. Mathur, Foundations of Software Testing, Seventh Impression, Pearson Education, New York, 2012.
[5] C. Catal, D. Mishra, Test case prioritization: a systematic mapping study, Softw. Qual. J. 21 (3) (2013) 445–478.
[6] M. Khatibsyarbini, M.A. Isa, D.N. Jawawi, R. Tumeng, Test case prioritization approaches in regression testing: A systematic literature review, Inf. Softw. Technol. 93 (2018) 74–93.
[7] F.M. Nejad, R. Akbari, M.M. Dejam, Using memetic algorithms for test case prioritization in model based software testing, in: 2016 1st Conference on Swarm Intelligence and Evolutionary Computation (CSIEC), 2016, pp. 142–147.
[8] Y. Lou, D. Hao, L. Zhang, Mutation-based test-case prioritization in software evolution, in: 2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE), 2015, pp. 46–57.
[9] D. Di Nardo, N. Alshahwan, L. Briand, Y. Labiche, Coverage-based regression test case selection, minimization and prioritization: A case study on an industrial system, Softw. Test. Verif. Reliab. 25 (4) (2015) 371–396.
[10] D. Di Nardo, N. Alshahwan, L. Briand, Y. Labiche, Coverage-based test case prioritisation: An industrial case study, in: 2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, 2013, pp. 302–311.
[11] G. Rothermel, R.H. Untch, C. Chu, M.J. Harrold, Test case prioritization: an empirical study, in: Proceedings IEEE International Conference on Software Maintenance - 1999 (ICSM'99). 'software Maintenance for Business Change' (Cat. No.99CB36360), 1999, pp. 179–188.
[12] B. Korel, G. Koutsogiannakis, 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.
[13] J.A. Whittaker, M.G. Thomason, A Markov chain model for statistical software testing, IEEE Trans. Softw. Eng. 20 (10) (1994) 812–824.
[14] K.-Y. Cai, A controlled Markov chains approach to software testing, Softw. Eng.: Int. J. (SeiJ) 1 (1) (2011) 38–59.
[15] S.K. Khatri, K. Kaur, R. Datta, Testing Apache OpenOffice Writer using statistical usage testing technique, Int. J. Syst. Assur. Eng. Manag. 6 (1) (2015) 3–17.
[16] G. Walton, J. Poore, Measuring complexity and coverage of software specifications, Inf. Softw. Technol. 42 (12) (2000) 859–872.
[17] A. Morozov, K. Ding, T. Chen, K. Janschek, Test suite prioritization for efficient regression testing of model-based automotive software, in: 2017 International Conference on Software Analysis, Testing and Evolution (SATE), IEEE, 2017, pp. 20–29.
[18] H.C. Tijms, H.C. Tijms, Stochastic Models: An Algorithmic Approach, Vol. 303, Wiley, New York, 1994.
[19] H. Tijms, A First Course in Stochastic Models, Wiley, 2003.
[20] E. Cinlar, Introduction to Stochastic Processes, Prentice-Hall, Englewood Cliffs, New Jersey, 1975, p. 420.
[21] D. Lee, M. Yannakakis, Principles and methods of testing finite state machines-a survey, Proc. IEEE 84 (8) (1996) 1090–1123.
[22] Y. Saad, Numerical Methods for Large Eigenvalue Problems, Manchester University Press, 1992.
[23] S. Elbaum, A.G. Malishevsky, G. Rothermel, Test case prioritization: A family of empirical studies, IEEE Trans. Softw. Eng. 28 (2) (2002) 159–182.
[24] N. Gökçe, F. Belli, M. Eminli, B.T. Dincer, Model-based test case prioritization using cluster analysis: a soft-computing approach, Turk. J. Electr. Eng. Comput. Sci. 23 (3) (2015) 623–640.
[25] A. Kashyap, T. Holzer, S. Sarkani, T. Eveleigh, Model based testing for software systems: an application of markov modulated markov process, Int. J. Comput. Appl. 46 (14) (2012) 13–20.
[26] S. Siegl, K.-S. Hielscher, R. German, C. Berger, Formal specification and systematic model-driven testing of embedded automotive systems, in: 2011 Design, Automation & Test in Europe, IEEE, 2011, pp. 1–6.
[27] P. Brémaud, Markov Chains: Gibbs Fields, Monte Carlo Simulation, and Queues, Vol. 31, Springer Science & Business Media, 2013.
[28] L.C. Briand, Y. Labiche, Y. Wang, Using simulation to empirically investigate test coverage criteria based on statechart, in: Proceedings. 26th International Conference on Software Engineering, IEEE, 2004, pp. 86–95.
[29] A.E.V.B. Coutinho, E.G. Cartaxo, P.D. de Lima Machado, Analysis of distance functions for similarity-based test suite reduction in the context of model-based testing, Softw. Qual. J. 24 (2) (2016) 407–445.

[30] A.T. Endo, A. Simao, Evaluating test suite characteristics, cost, and effectiveness of FSM-based testing methods, Inf. Softw. Technol. 55 (6) (2013) 1045–1062.
[31] A.G. Raiyani, S.S. Pandya, Proritization technique for minimizing number of test cases, Int. J. Softw. Eng. Res. Pract. 1 (2011).
[32] K. Muthyala, R. Naidu, A novel approach to test suite reduction using data mining, Indian J. Comput. Sci. Eng. 2 (3) (2011) 500–505.
[33] S. Roongruangsuwan, J. Daengdej, Test case reduction methods by using CBR, in: International Workshop on Design, Evaluation and Refinement of Intelligent Systems (DERIS2010), 2010, p. 75.
[34] B.A. Kitchenham, S. Charters, Guidelines for performing Systematic Literature Reviews in Software Engineering, Technical Report EBSE 2007-001, Keele University and Durham University, UK, 2007.
[35] M. Pai, M. McCulloch, J. Gorman, N. Pai, W. Enanoria, G. Kennedy, P. Tharyan, J.M. Colford, Systematic reviews and meta-analyses: an illustrated, step-by-step guide, Natl. Med. J. India 17 (2) (2004) 89.
[36] J. Biolchini, P. Mian, A. Natali, G. Travassos, Systematic Review in Software Engineering, Technical Report ES 679/05, COPPE/UFRJ, Brazil, 2005.
[37] T. Dyba, T. Dingsoyr, G. Hanssen, Applying systematic reviews to diverse study types: An experience report, in: International Symposium on Empirical Software Engineering and Measurement, 2007, pp. 225–234.
[38] J.R. Landis, G.G. Koch, The measurement of observer agreement for categorical data, Biometrics (1977) 159–174.
[39] M. Al-tekreeti, K. Naik, A. Abdrabou, M. Zaman, P. Srivastava, Test generation for performance evaluation of mobile multimedia streaming applications, in: MODELSWARD, 2018, pp. 225–236.
[40] C.-a. Sun, H. Dai, H. Liu, T.Y. Chen, K.-Y. Cai, Adaptive partition testing, IEEE Trans. Comput. 68 (2) (2018) 157–169.
[41] L. Lin, Y. Xue, F. Song, A simpler and more direct derivation of system reliability using markov chain usage models, in: Proceedings of the 29th International Conference on Software Engineering and Knowledge Engineering, KSI Research Inc. and Knowledge Systems Institute Graduate School, 2017, pp. 462—466.
[42] W.J. Gutjahr, Software dependability evaluation based on Markov usage models, Perform. Eval. 40 (4) (2000) 199–222.
[43] K.-Y. Cai, Optimal software testing and adaptive software testing in the context of software cybernetics, Inf. Softw. Technol. 44 (14) (2002) 841–855.
[44] K.-Y. Cai, Y.-C. Li, K. Liu, Optimal and adaptive testing for software reliability assessment, Inf. Softw. Technol. 46 (15) (2004) 989–1000, Third International Conference on Quality Software: QSIC 2003.
[45] K.-Y. Cai, Optimal test profile in the context of software cybernetics, in: Proceedings Second Asia-Pacific Conference on Quality Software, 2001, pp. 157–166.
[46] X. Devroey, G. Perrouin, M. Cordy, H. Samih, A. Legay, P.-Y. Schobbens, P. Heymans, Statistical prioritization for software product line testing: an experience report, Softw. Syst. Model. 16 (1) (2015) 153–171.
[47] F. Sayyari, S. Emadi, Automated generation of software testing path based on ant colony, in: 2015 International Congress on Technology, Communication and Knowledge (ICTCK), IEEE, 2015, pp. 435–440.
[48] A. Kashyap, W.J.J. Roberts, S. Sarkani, T.A. Mazzuchi, A model driven approach for system validation, in: 2012 IEEE International Systems Conference SysCon 2012, IEEE, 2012, pp. 1–7.
[49] J. Yan, C.H. Deng, M.L. Ji, Reducing safety-critical software statistical testing cost based on importance sampling technique, Adv. Mater. Res. 433–440 (2012) 4691–4697.
[50] B. Zhou, H. Okamura, T. Dohi, Application of Markov Chain Monte Carlo random testing to test case prioritization in regression testing, IEICE Trans. Inf. Syst. E95.D (9) (2012) 2219–2226.
[51] A. Gibbons, Algorithmic Graph Theory, Cambridge University Press, 1985.
[52] M. Dorigo, K. Socha, An introduction to ant colony optimization, in: Handbook of Metaheuristics, Vol. 26, 2006.
[53] B. Regnell, P. Runeson, C. Wohlin, Towards integration of use case modelling and usage-based testing, J. Syst. Softw. 50 (2) (2000) 117–130.
[54] A. Morozov, R. Tuk, K. Janschek, ErrorPro: Software tool for stochastic error propagation analysis, in: 1st International Workshop on Resiliency in Embedded Electronic Systems, Amsterdam, the Netherlands, 2015, pp. 59–60.
[55] A.O. Arantes, N.L. Vijaykumar, V.A. de Santiago Junior, D. Guimarães, WEB-PerformCharts: a collaborative web-based tool for test case generation from Statecharts, in: Proceedings of the 10th International Conference on Information Integration and Web-Based Applications & Services, 2008, pp. 374–381.
[56] V. Santiago, N.L. Vijaykumar, D. Guimarães, A.S. Amaral, E. Ferreira, An environment for automated test case generation from statechart-based and finite state machine-based behavioral models, in: 2008 IEEE International Conference on Software Testing Verification and Validation Workshop, IEEE, 2008, pp. 63–72.
[57] D. Harel, Statecharts: A visual formalism for complex systems, Sci. Comput. Program. 8 (3) (1987) 231–274.
[58] E.F. Souza, V.A. Santiago Júnior, D. Guimaraes, N.L. Vijaykumar, Evaluation of test criteria for space application software modeling in statecharts, in: International Conference on Computational Intelligence for Modelling Control and Automation, 2008, pp. 157–162.
[59] A.O. Arantes, V.A. Santiago Júnior, N.L. Vijaykumar, E.F. Souza, Tool support for generating model-based test cases via web, Int. J. Web Eng. Technol. IiWAS 9 (1) (2014) 62–96.
[60] E.F. Souza, V.A. Santiago Júnior, N.L. Vijaykumar, H-Switch Cover: a new test criterion to generate test case from finite state machines, Softw. Qual. J. 25 (2) (2017) 373–405.
[61] M.M. Mariano, E.F. Souza, A.E. Endo, N.L. Vijaykumar, Comparing graph-based algorithms to generate test cases from finite state machines, J. Electron. Test.-Theory Appl. 1 (2020) 1–19.
[62] X. Zhou, Y. Jin, H. Zhang, S. Li, X. Huang, A map of threats to validity of systematic literature reviews in software engineering, in: 23rd Asia-Pacific Software Engineering Conferencees, 2016, pp. 153–160.
[63] V. Odumuyiwa, U. Osisiogu, A systematic review on hidden Markov models for sentiment analysis, in: 2019 15th International Conference on Electronics, Computer and Computation (ICECCO), 2019, pp. 1–7.
[64] D.A. Karras, R.C. Papademetriou, A systematic review of analytical management techniques in business process modelling for SMEs beyond what-if-analysis and towards a framework for integrating them with BPM, in: Proceedings of the Seventh International Symposium on Business Modeling and Software Design – BMSD, 2017, pp. 99–110.