Requirements Engineering (2020) 25:317–337
https://doi.org/10.1007/s00766-019-00325-w

ORIGINAL ARTICLE

Software requirements testing approaches: a systematic literature review

Jemison dos Santos[1] · Luiz Eduardo G. Martins[1] · Valdivino A. de Santiago Júnior[2] · Lucas Venezian Povoa[3] · Luciana Brasil R. dos Santos[3]

Received: 18 December 2018 / Accepted: 2 August 2019 / Published online: 16 August 2019

Abstract
Testing a software system is an important step approach to ensuring quality, safety, and reliability in safety-critical systems (SCS). Several authors have published new approaches to improve the processes of testing safety requirements taking into consideration existing processes that seek to improve techniques and contribute positively with software developers. This article aims to investigate the main approaches to requirements testing, particularly focusing on safety requirements in the context of SCS. We investigated how these approaches have been developed and what contributions they provide to academia and industry. We evaluated the pros and cons of the approaches and how they related to the joint work of requirements engineers and testers. We performed a systematic literature review (SLR), selecting 53 papers published between 1990 and 2018. Our research was conducted according to the guidelines proposed by Kitchenham and Biolchini. The results of this SLR point out to the new research related to the software and safety-critical systems testing. The results show issues in the integration of requirements engineers with the application test team and gaps in the approaches found, particularly in the applications of the techniques in the industry setting. Moreover, several approaches are presented to solve problems and help to prevent future problems. The results of this research point to the main approaches to requirements testing and their use in academia and industry, as well as the advantages and disadvantages. The shortcomings allow us to suggest new research in safety-critical systems in the scope of validation, verification, specification, and testing of safety requirements, as well as to integrate test teams with requirements engineers in order to get better results. Based on the results, we suggest future studies for improvements in the requirements testing techniques to improve the integration of safety requirements and test cases.

Keywords Requirements testing · Safety-critical systems · Safety requirements

✉ Jemison dos Santos
jemison321@gmail.com

Luiz Eduardo G. Martins
martinsleg@hotmail.com

Valdivino A. de Santiago Júnior
valdivino.santiago@inpe.br

Lucas Venezian Povoa
lucasvenez@gmail.com

Luciana Brasil R. dos Santos
lurebelo@gmail.com

[1] Department of Science and Technology, Federal University of São Paulo (UNIFESP), São José dos Campos, SP, Brazil

[2] Associated Laboratory for Computing and Applied Mathematics (LABAC), National Institute for Space Research (INPE), São José dos Campos, SP, Brazil

[3] Computer Science Department - Federal Institute for Education, Science, and Technology of São Paulo (IFSP), Caraguatatuba, SP, Brazil

Abbreviations
SCR Score according to quality assessment
CIT Amount of citations of paper

1 Introduction

One of the time-consuming activities in the software development cycle is testing [1]. It is one of the main ways of software quality assurance and aims to find the existence of faults [2]. The importance of the test observes throughout the literature where several records of accidents caused by software failures happen. There are several methodologies for requirements testing. However, the causes of an accident motivated by software failure can occur for several technical reasons [3]. With testing, it expects that the resulting product will be pleasing to the user and the developer, regardless of the methodology used [1].

In the industry setting, the requirements testing process tends to be expensive because it requires a rigorous methodology to meet rigorous objectives, making this process increasingly complex for both software systems and safety-critical systems. Critical systems are those that perform activities that can cause harm or risk to life or the environment [3]. We find this type of system in domains such as avionics, health, nuclear, automotive, aerospace, military, among others [2, 4–8].

Critical systems are complicated because they involve a variety of factors, such as the limits of analyst knowledge that can lead to software consistency and integrity failures, failures during system integration, quality assurance, traceability, completeness, ambiguities, and testability issues [2, 9]. For a safety-critical system (SCS), the adoption of an appropriate requirements testing methodology is imperative to ensure the effectiveness of certification verification [2].

Observing studies from 1990, new approaches to improving the generation and automation of test cases for software systems and SCS are noteworthy. These challenges include the development of safety requirements, development guides and much more, both in academia and industry [10–12].

The most significant cause of accidents caused by software is related to poorly created software requirements or requirements partially delivered to developers [13]. Therefore, requirements engineering plays an essential role in ensuring that the software objectives are well described and provide a natural understanding of the software, so that it can be analyzed, modeled, specified, verified, and correctly validated [14].

In order to know the state of the art and state of the practice regarding testability of requirements in general and particularly to testability of safety requirements, we conducted a systematic literature review (SLR)

We also are interested to know the application of these approaches in the industry and how this can promote the integration between software engineers and testers. To support this work, we take into account the following points of view: (1) approaches to analysis, verification, validation, inspection of software requirements; (2) how useful are such approaches to requirements testing and/or safety-critical software; (3) how relevant this is to industries; and (4) in what extent the approaches reported help in the integration of requirements engineers and software testers.

This paper is organized as follows: Section 2 presents background and related work. Section 3 presents the research methodology. The results and the analysis related to our research questions are presented in Sect. 4. Finally, we present our conclusions in Sect. 5.

2 Background and related work

This section presents some articles published in digital libraries described earlier in a systematic literature review, as mentioned in Sect. 1.

Software requirements testing is a significant activity that aims to validate whether the software product is functioning correctly and meets the requirements specified [15]. Testers are accustomed to applying a series of tests of diverse natures with diverse purposes, involving not only the functional tests of the application but also several other activities [1].

In addition to analyzing the performance and reliability of the system requirements, the requirements testing cooperates with the specification and analysis of user data and its application domain. This process is necessary to verify aspects such as profile creation, and the completeness of the documentation, in general, facilitate the design process [1, 15].

Software requirements testing is one of the requirements engineering processes that is applied to verify the consistency of the requirements raised during the requirements analysis process and whether they are testable. In this type of test, the testability of the requirement is examined, that is, testability verifies different probabilities and behavioral characteristics may cause the code to fail if something is incorrect in the requirement [1]. The software requirements test does not run the software; however, it analyzes the requirements of the system to avoid problems, unlike the requirements-driven test that performs test cases seeking compliance of the requirements with the software being built [15].

Requirement-driven testing is a non-design testing approach and can be used in design methodologies such as waterfall, agile, and scrum, focusing on aspects such as creating business requirements lists; requirement that is used to select test cases; and report the approval or disapproval of business requirements. It addresses issues such as validation that the requirements are correct, complete, unambiguous, and logically consistent, and about designing a set of test cases sufficient to ensure that the design and code fully meet those requirements [1, 15].

Within the context of requirements testing and requirements-driven testing, several approaches are sought in the literature that seeks to make these testing processes easier and also provides the scientific community with new contributions through new requirements testing and requirements-driven testing methodologies. Some approaches involve testing safety requirements; these requirements can lead to dangerous system crashes if they are not straightforward, complete, and concise.

There are some good practices that can be applied to requirements testing and requirements-oriented testing, such

as: requirements driven testing; features of risk-based analytical strategies with to perform a risk analysis to discern risk items and their impact on the application; increased efficiency of testing activities; alignment of testing processes with other organizational activities; and improves the value of testing for the organization.

In the context of requirements testing, it has been found that good practices have been applied to perform requirements testing, such as: methods for making full effectiveness using heuristic testing approaches to prevent ambiguity from occurring when multiple interpretations are plausible for the same requirement.

Software engineers tend to face high demands for reliable and robust software with short-term development [11]. They need methodologies that make the processes more agile and simplified. Ensuring that the software functions do not contribute to unintended system risks is the responsibility of the software system's security. The achievement takes place through a set of engineering and management activities in the safety-critical systems and software engineering domains to identify, analyze, design and track software mitigation and control of hazards and hazardous functions [16].

Martins and Gorschek [17] conducted an SLR in requirements engineering for safety-critical systems where they investigated which approaches have been proposed to elucidate, model, specify, and validate safety requirements in the SCS context, selecting 151 articles published between 1983 and 2014. They also used the Kitchenham and Charters' [18] guidelines. They aimed to encourage further research into the design of studies to improve the engineering requirements for SCS and suggested a research agenda for the community of researchers and advise for SCS professionals.

Vilela et al. [19] present an SLR in integration between requirements engineering and safety analysis, speaking about the requirement of the most sophisticated requirements engineering (RE) approaches, where they report that many accidents and catastrophes are related to safety because unmet, incomplete, or incomprehensible requirements are the leading causes of this fact. They aimed to investigate the proposed approaches to improve communication or integration between RE and safety engineering in the development of SCS. Their work proposed four taxonomies of hazard analysis, safety, safety-related information, and hazard information.

Gurbuz and Tekinerdogan [20] present a systematic mapping study in base tests of models for software safety. They aim to investigate the application domains on which the model-based safety test is performed, identify the current challenges and directions of the research, and identify possible solutions in that context.

They examine the current model-based testing approaches to software safety to conclude that the model-based test for safety is extensive and applied across multiple application domains. They show that model-based testing can provide significant benefits for software safety testing. Several solution directions identified, but additional research is critical to a reliable model-based testing approach to safety.

Haser et al. [21] performed an SLR according to Kitchenham and Charters' [18] guidelines and retrieved 83 relevant studies that identified three evaluation criteria to guide the testing process, static metrics, dynamic, and stochastic and random metrics to find and synthesize relevant primary studies to gain a comprehensive understanding of the current state of model-based integration testing. Their results show that there is an accumulated need for approaches in the model-based testing field that support non-functional requirements as they are gaining importance and the means to guide the integration testing process, especially in conjunction with automation.

Unterkalmsteiner et al. [22] in his study, he portrays the difficulty of a project succeeding when engineering is weak. He investigates the practice of using test cases as requirements in three companies to understand how test cases can help. Its results include many of benefits and challenges in using test cases to identify, validate, verify, track, and manage requirements. The findings provide insight into how the role of requirements can meet development, including challenges to consider.

3 Research methodology

This section will be composed of the empirical research model and the execution of the SLR. For the execution of the research methodology, the guideline proposed by Kitchenham and Charters [18] was used as the research methodology. Figure 1 shows a representation of the process proposed by Kitchenham for the SLR processes.

The need to execute this SLR (Fig. 1—Step 1) was due to the interest of knowing the main approaches of analysis and testing of software requirements, as well as its application in academia and industry. We also tried to identify which approaches the teams testing and engineering requirements and what are the pros and cons reported in the literature.

3.1 Research questions

The main focus of this systematic review of the literature is to raise the approaches related to requirements testing and SCS, according to the main approaches to describe the integration of these techniques between academy and industry, and the relationship between engineers and testers in the development. The creation of the research questions was made in Step 2—Fig. 1, and they are described in Table 1.

Future discussions will be based on data from each primary study. These RQs were elaborated to obtain state of

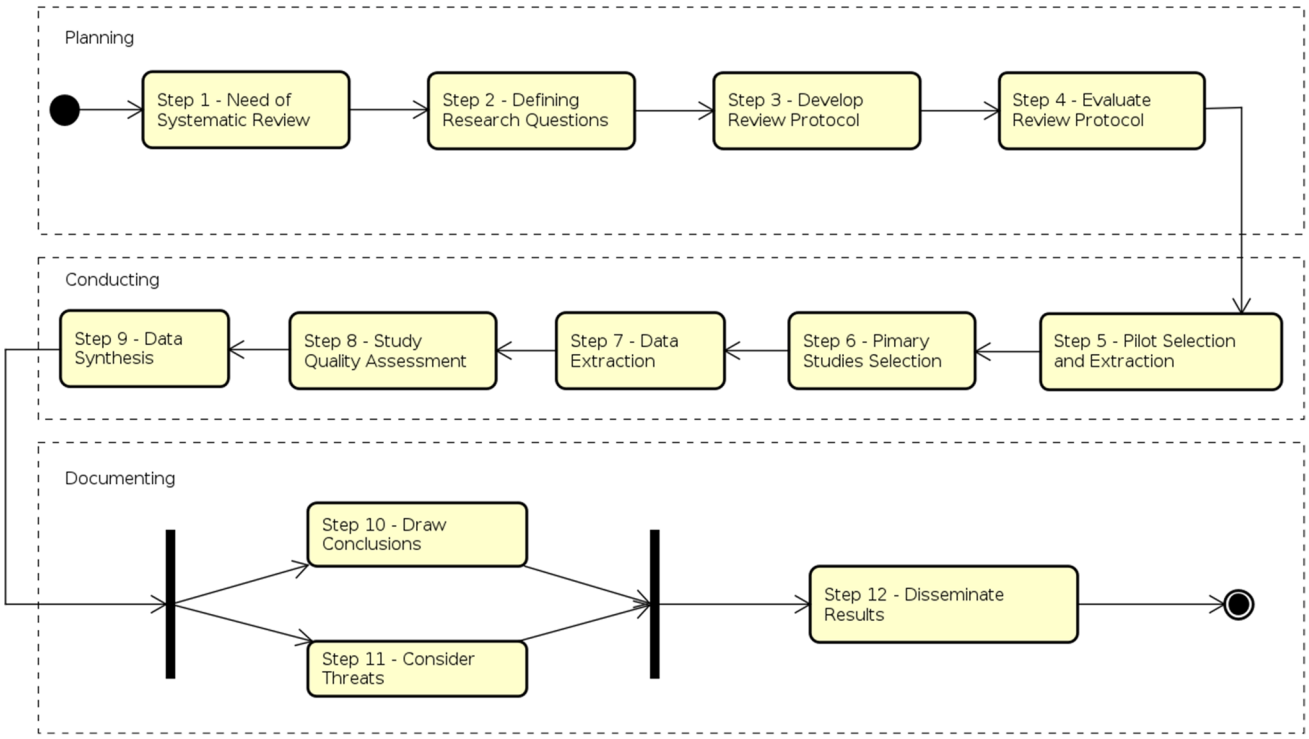

Fig. 1 Steps of a systematic literature review

Table 1 Research questions

ID	Research question	Motivation
RQ1	What are the main approaches proposed in the literature to test requirements?	To get the state-of-the-art requirements testing
RQ1.1	How do the approaches to requirements testing are applied in safety-critical requirements?	To verify how these approaches are addressing the issues in testing safety requirements
RQ1.2	What are the pros and cons of these approaches to test safety requirements?	To identify the benefits and gaps in using such approaches and to suggest and implement improvements
RQ1.3	How much are these approaches used for the industry practitioners?	To verify the validity of these approaches in the industry setting
RQ1.4	Is there evidence of integration between requirement engineers and testers in the approach?	To understand what extent these approaches improve the communication between requirements engineers and testers

the art at requirements testing that since 1990 with the software engineering has been gaining prominence.

3.2 Search strategy

The search strategy was automated in digital libraries by using search strings, which will be formed for each one based on the advanced search options. The strings will be adjusted, so they return the most relevant results for this work. The SLR protocol (Fig. 1—Step 3) follows the PICOC criteria suggested by Kitchenham and Charters [18]:

- *Population* Published papers related to software requirement testing or safety-critical systems testing or requirements analysis, specification and assessment;
- *Intentions* Collect existing approaches or new approaches in requirements testing and verify the methodology used for testing in academy or industry and how this interferes in the relationship between the engineer and the tester;
- *Comparison* Collect and list significant requirements for testing and safety-critical systems;
- *Outcomes* Identification of the most used approaches and their areas of application, relevance in the industry, adequacy of professionals, requirements engineering, impacts on critical systems, key points of each method,

Table 2 Search strings

Source	String
IEEE Xplore *and* Springer Link	(requirement*) AND (software OR system) AND (test* OR verif* OR valid* OR inspec*)
ACM DL *and* Science Direct	(requirement) AND (software OR system) AND (test OR verif OR valid OR inspec)

Table 3 Inclusion and exclusion criteria

#	Inclusion criteria
1	Primary studies
2	Surveys and papers from conferences and journals
3	Studies from 1990 onwards
4	Studies that report approaches of testing requirements
5	Studies that report approaches of testing safety requirements
#	Exclusion criteria
1	Papers with less than three pages
2	Papers not in English
3	Secondary studies
4	Duplicated studies (only one copy of each study must be included)
5	Redundant paper of the same authorship
6	Studies whose focus not on testing safety requirements
7	Studies irrelevant to the research, taking into account the research questions

impacts on industry, and more used approaches and more suitable;

- *Context* The context of the research involves articles related to the testing of software requirements and safety-critical systems.

Consequently, the searches were carried out in the digital databases of IEEE Explore,[1] ACM Digital Library,[2] Springer Link,[3] and Science Direct,[4] with the base search string:

requirement AND (software OR system) AND (test OR verif OR valid OR inspec).

Thus, filters added in the search aim for greater relevance in the results obtained, and these comprise Sect. 3.3. To manage the papers, we use Mendeley[5] software. Table 2 shows the search strings used in each database

[1] https://ieeexplore.ieee.org.

[2] https://dl.acm.org.

[3] https://link.springer.com.

[4] https://www.sciencedirect.com.

[5] https://mendeley.com.

3.3 Inclusion and exclusion criteria

Table 3 is composed of inclusion and exclusion criteria. These criteria were applied, so that all papers selected are within the context of our research questions. We chose papers from 1990 onwards, with more than three pages and in English to apply these criteria.

From the definition of these criteria, the search in the bases mentioned above was carried out and a total of 1316 papers were obtained. These papers were inserted into the Mendeley for further analysis. In Table 4, we show the totals for each of the four analysis phases of the primary studies are presented.

The first phase of selection is the search of the primary studies in the databases with the use of the string, where each base the search was carried out and the selected filters were applied, resulting in a total of 1316 papers.

In the second selection phase, duplicate articles were removed, those containing less than three pages and those not written in English, as well as the application of other filters available in the databases. This phase resulted in a total of 164 papers.

In the third selection phase, the titles, abstracts, and results of the studies were read, applying the inclusion and exclusion criteria to eliminate those works that do not focus on the research objective. The total of 90 papers of this phase was inserted in the Mendeley software.

In the fourth selection phase, a superficial reading of the papers is performed, observing the inclusion and exclusion criteria in other sections of the paper; in this phase, it is possible to exclude some papers that have gone through the previous selection phase and did not contain the RQs. Thus, the process of selecting the primary studies with 53 papers enabled for data extraction is concluded. The articles selected from each database can be seen in Table 5.

3.4 Data extraction and synthesis

To perform the data extraction of the studies enabled in this phase, a spreadsheet was used to fill some attributes of the articles. Table 6 gives the properties of data extraction.

Table 4 Selection phases and selected article numbers

Digital library	Phase 1	Phase 2	Phase 3	Phase 4
ACM DL	817	31	31	7
IEEE Xplore	117	88	45	40
Science Direct	80	10	10	4
Springer Link	302	35	4	2
Total	1316	164	90	53

P1 The description of the approach has been extracted to help answer RQ1 and to elect all the major approaches to testing requirements and critical safety requirements found in the primary studies. With these data, it was possible to know the approaches that would compose the SLR;
P2 The motivation of the authors to carry out the studies that generated the work is essential since it is necessary to analyze the contextualization of the domain environment and justify the insertion of this paper in this SLR;
P3 The validation of the context of the works found is critical to qualify the primary study within the context of the research and to judge the need to be included in SLR;
P4 To introduce a response to RQ1.1, the need to know whether the approach covers requirements testing is to identify by whom it uses. The answer gave "yes," "no," or "partially" and then explained;
P5 P6 The research question RQ1.1 corresponds to all the primary studies collected. These are categorized and separated from those that only meet requirements and those that can meet safety requirements. Answering these research questions provided a focus for an approach that is specific to safety or can be improved to meet new needs;
P7 P8 RQ1.2 allows to give an opinion about the studies found and to list all the advantages and disadvantages of each study;
P9 In order to add value to the SLR, it was investigated how each selected study was validated to guarantee the consistency of this research and the integrity of the selected papers;
P10 This question has a fixed answer value: "yes" or "no." Yes: Represents that there is evidence about connections between requirements and test artifacts and is therefore justified. No: Represents that there is no link between requirements with artifact testing;
P11 Knowing why the author chose a specific approach to the work allows us to understand whether the search is following the same search line and how relevant the search is;
P12 The techniques used by the authors give a basic knowledge of the operation of each one of them. The advantages and disadvantages that each author found when working with them make us realize where improvements can be sought;
P13 P14 The scope of application of each work shows the diversity of areas in which we can apply approaches to testing software requirements, testing the requirement of safety-critical systems, and existing tools available for support as well as ideas for new approaches.

Table 5 Selected primary studies

Digital library	Studies
ACM DL	[7, 14, 23–27]
IEEE Xplore	[4, 6, 8–11, 13, 28–60]
Science Direct	[5, 61–63]
Springer Link	[64, 65]

3.5 Study quality assessment

The qualification of the primary studies is a process of complementation of the inclusion and exclusion criteria, where weights 0, 0.5 and one, respectively, representing in this order "no," "partially," and "yes" are assigned. This score is intended to investigate whether there are different

Table 6 Extracted properties from primary studies

ID	Property	RQ
P1	Description of approach	RQ1
P2	Article motivation	Overview of the studies
P3	Validation Context	Overview of the studies
P4	Does the approach cover the requirements test?	Overview of the studies
P5	Is the approach appropriate for safety testing?	RQ1.1
P6	Is it possible to adapt the approach to testing safety requirements?	RQ1.1
P7	The pros of the approach	RQ1.2
P8	The cons of the approach	RQ1.2
P9	How was the approach validated?	Overview of the studies
P10	Does the approach bind requirements to the test artifacts?	RQ1.4
P11	Rations to choose this approach	Overview of the studies
P12	Techniques used to implement the approach	Overview of the studies
P13	Field of application	Overview of the studies
P14	Automated support	Overview of the studies

Table 7 Quality assessment

ID	Yes	No	Partially
QA1	53 (100%)	0 (0%)	0 (0%)
QA2	31 (58.49%)	0 (0%)	22 (41.50%)
QA3	49 (92.45%)	4 (7.55%)	0 (0%)
QA4	6 (11.32%)	47 (88.68%)	0 (0%)
QA5	1 (1.89%)	52 (98.11%)	0 (0%)
QA6	53 (100%)	0 (0%)	0 (0%)
QA7	44 (83.02%)	9 (16.98%)	0 (0%)
QA8	6 (11.32%)	47 (88.68%)	0 (0%)
QA9	15 (28.30%)	38 (71.70%)	0 (0%)

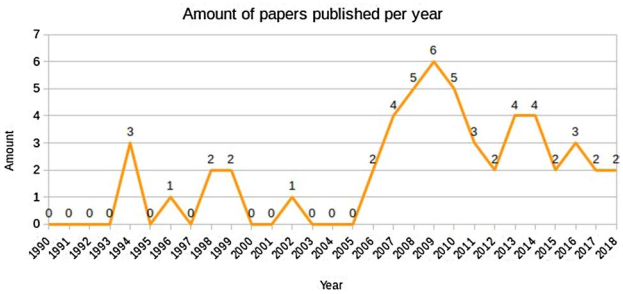

Fig. 2 Amount of papers published per year

explanations for the results of the primary studies and a form of individual weighting during the synthesis of results. However, guiding recommendations for future research, as described in the guide proposed by Kitchenham and Charters [18], the quantitative values of the quality assessment (QA) were made as follows:

QA1 Are the aims clearly defined?
QA2 Do they answer our research questions?
QA3 Are the approaches clearly defined?
QA4 Were the results compared to others? (If yes, were they obtained under similar circumstances?)
QA5 Are cons or bad results presented/discussed?
QA6 Is the environment clearly defined?
QA7 Is the approach useful to test safety requirements? (if yes, why?)
QA8 Did the approach involve requirements engineers and testers? (if yes, how?)
QA9 Is there evidence of use by industry practitioners? (if yes, how?)

The results for each question are presented in Table 7.

QA1 was proposed to verify whether the study has the potential to be inserted in this SLR, those marked with "yes" will be analyzed, and those that get a "no" answer will be excluded. The amount of "yes" was 100%. QA2 obtained a total of 31 "yes" and 22 "partial" answers, considering that papers that answered "yes" will have a greater emphasis on the analysis of results because they present requirements test approaches, those that have been "partially" analyzed less stringent. The QA3 points out the papers that presented in detail the approaches used, being 92.45% "yes" and 7.55% "no" and shows that several works do not present in an obvious clear way what was used or proposed. QAs 4 and 5 show that most authors do not discuss in the paper the results not considered suitable or the disadvantages found in the processes. The QA6 was introduced in the form to guarantee the understanding of the application of the approaches in the environment proposed by each author. QA7 is specific in testing critical requirements and creates a group within the primary studies obtained for further analysis. QA8 was proposed to visualize the situation of integration of test teams with requirements engineers, and finally, QA9 demonstrates the usefulness of approaches within the industry.

4 Result and analysis

Data were synthesized based on the data extracted from the primary studies and transferred to a spreadsheet. The analysis of the data was the process of verification of the information collected, through which they have discovered the types of requirements test approaches used by the authors, processes, and methodologies commonly used by the industry as well as their fields of application.

The evaluation of the quality of the primary studies collaborated to increase the credibility of the conclusions and for the synthesis to be coherent. The process of synthesis and analysis of the data is used to answer the research questions according to the information extracted from the articles in the extraction process.

The 53 articles selected (see "Appendix 1") that met the inclusion criteria presented approaches for verification, validation, inspection, and testing of software system requirements. The search for primary studies comprises the period from 1990 to March 2018, where the number of articles published per year can be seen in Fig. 2.

It is possible to notice that between the period of 1990 and 2006, the oscillation of the publications in the leading conferences and periodicals was of up to three articles per year. After 2006 the maximum amount of published articles doubles in 2009, it is common to see them continue to grow in the coming years. It is expected to continue to grow in the coming years, as technological advances increasingly demand software systems.

The primary studies come from several countries, such as Belgium (1), Brazil (1), Canada (3), China (8), Denmark (1), France (3), Germany (3), Italy (1), Korea (2), Malaysia (1), Morocco (1), Norway (2), Pakistan (1), Poland (1), Romania

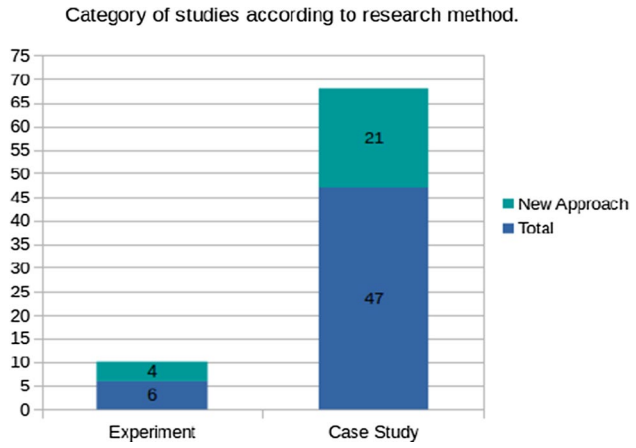

Fig. 3 Number of categories according to the research method

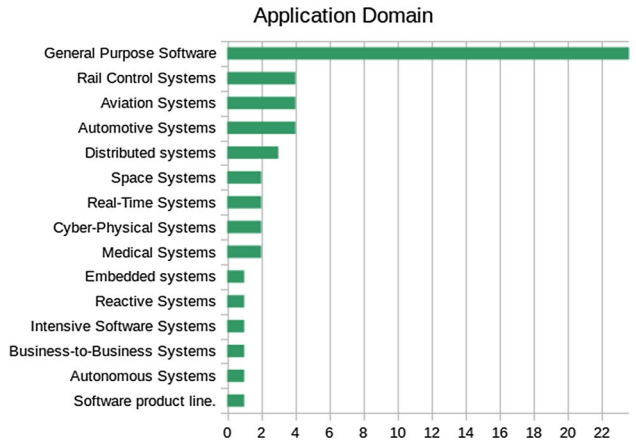

Fig. 4 Number of primary studies according to the application domain

(1), Saudi Arabia (1), Spain (1), Sweden (1), Taiwan (1) United Kingdom (3), and United States of America (16).

As stated in the presented data, it was possible to categorize the primary studies according to the research method as shown in Fig. 3.

Figure 3 shows the total number of primary studies found by search category and indicates the number of new approaches in that medium. The case studies amount to 88.67% of the total, within those indicative new approaches correspond to 44.68%. Experiments are 11.33% of the primary studies, with 66.67% proposing new approaches.

The primary studies report the application of software requirements testing in several application domains as shown in Fig. 4.

Most of the selected studies refer to general-purpose software, i.e., the authors describe the approach with a focus on methodologies and tools used, presenting case studies and experiments directly on sets of requirements [4, 6, 8, 10, 11, 13, 27, 28, 31–33, 40, 41, 45, 47, 48, 50, 53, 57, 59–62, 64, 66]. Studies on railway control systems include software such as the European rail traffic management system [7, 36], railroad locking systems [37], and Chinese train control [9]. Aviation systems are concerned with specifying requirements following the guidance of development standards [54] and onboard air systems [26, 46, 65]. Automotive systems have studies on the integrity of software [67], software of road vehicles [51], and automotive embedded software [42]. For distributed systems, telephone systems [29, 58] and computer supported cooperative work system [14] are topics found. In space systems, there are studies such as space shuttle [23] and nanosatellite [63]. In real-time systems, their functions are constrained by response time limits, such as a gas burner system [49] and a temporal logic language for real-time executable system specification (TRIO) [25]. In cyber-physical systems, there is a verification of requirements violation using formal methods [24, 30, 35]. In medical systems, one has an insulin infusion pump software [56] and medical devices from Siemens [38]. In Embedded systems, a requirements test was found on a temperature module in control software for a launch pad [43]. We also saw software requirements testing [34], B2B software [39], and autonomous systems that are software sensitive to changes in their environment, such as intelligent residences or adaptive systems and the product line of software [39].

In the following subsections, the results of each research question derive from descriptions and discussions.

4.1 RQ1: What are the main approaches proposed in the literature to test requirements?

The motivation for this research question was to raise the state-of-the-art requirements test. This research question was divided into four subquestions that are presented from Sects. 4.2 to 4.5 analyzing several aspects of this topic. The studies selected for this SRL can be seen in "Appendix 1."

Table 8 shows the types of requirements tests found in the primary studies

In model-based requirements testing (43.39%), the authors presented approaches that apply use case diagrams [8, 38, 44], activity diagrams [4, 10, 44, 61, 64], diagrams and sequence [7–9], class diagrams [36, 44, 61–63], finite state machines [39, 59], and state graph [27, 46].

Representing 3.77% is the requirements test based on the ModelicaML models, and this graphical modeling language belongs to an extended subset of the UML and allows the generation of executable code derived from requirements models [28, 30]. There is also a test of requirements based on time usage model; this is presented as a formal representation of the requirements specification [42, 51, 68]; and finally, the requirements test based on use case maps, they are used to capture and validate software requirements [29,

Table 8 Approaches to requirements testing found in the literature

Approaches	Amount
Requirements-based testing on UML models	23
Model-based requirements test ModelicaML	2
Requirements testing time usage models	2
Requirements-based testing on models UCM	2
Gray box-based requirements testing	2
Requirements-based testing on alpha–beta cutting procedure	1
Requirements testing behavior trees	1
Domain knowledge-based requirements test	1
Requirements-based testing on product test templates	1
Requirements testing based on test specification language	1
Requirements testing with the RADIX Tool	1
Requirements testing with bounded model checking tool	1
Requirements testing with role-based models	1
Linear temporal logic requirements test	1
Black box requirements test	1
Aspect-oriented requirements test	1
Requirements testing point of view	1
Requirements test driven by ratings and algorithms	1
Validation of requirements with pseudo-software	1
Algorithm-oriented requirements validation	1
Requirements verification with info tree	1
Requirements verification with formal methods	1
Requirements verification oriented by debug algorithm	1
Requirements verification by formal methods	1
Requirements verification via formal analysis	1
Verification and validation with a Bayesian belief network	1
Verification and validation with jUnit framework	1
Total	53

65]. The other approaches found to represent only 1.88% of the primary studies.

4.2 RQ1.1: How do the approaches to requirements testing are applied in safety-critical requirements?

The motivation for this research question was to see how these approaches are dealing with problems in testing safety requirements.

The analysis during the data extraction process of the relationship between the approach and critical security requirements occurred. In the data extraction, worksheet was marked with "yes" approaches that are suitable for testing critical safety requirements and "no" for approaches that have tested non-critical system requirements.

Figure 5 shows the total number of primary studies of the SLR separating them into two groups by test topic: primary studies reporting requirements testing approaches and

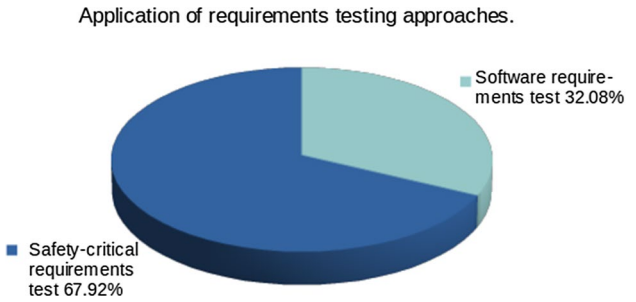

Fig. 5 Number of approaches to requirements testing and safety-critical requirements testing

primary studies reporting approaches to testing safety-critical systems requirements.

Among the approaches to testing software requirements (67.92%), some approaches can be adapted to the test of safety-critical requirements, and they are: [6, 8, 13, 29, 33, 39, 40, 48, 53, 58, 60].

The approach of Alves et al. [6] considered the dynamic behavior of the system in formal representation as state diagram assertions and validated using jUnit test scenarios. A set of safety-critical requirements related to the sequence of flight events is chosen to be formally specified, validated, and verified using the proposed formal approach.

Amyot et al. [29] present an approach based on the combined use of two notations: UCMs for causal scenarios that are used to capture and integrate critical security requirements and integrate UCMs to help avoid excessive interactions before generating prototypes; the formal specification language LOTOS.[6] UCM scenarios translated into high-level LOTOS specifications, which can be used to formally validate safety-critical requirements, including functional tests based on UCMs; they emphasize the most relevant, existing, and safety-critical functionalities of the system.

Farhat et al. [33] conducted a feasibility study of the use of aspects to test non-functional requirements NFRs, based on two categorizations of NFRs. The first categorization divides NFRs into four types, that is, functionally restrictive, restrictive, policy restrictive, and architecturally restrictive additives, and the second categorization divides NFRs into two types: operational and non-operational. These categorizations would serve as a starting point for developing frameworks or methodologies for testing safety-critical requirements with aspects.

Introducing a family of similarity-based test case selection techniques for sets of tests generated from state machines in a monitoring component in a safety-critical control system implemented in C++. Hemmati et al. [39] have developed an approach to select a subset of the generated test

[6] More information at http://cadp.inria.fr/man/lotos.html.

set, so that it can be run and analyzed within the constraints of time and resources while preserving to the utmost the fault-finding power of the original test suite.

Ibrahim et al. [8] developed an automatic generator for programming codes, with the concept of introducing an automatic tool for requirements testing, in which the tool is used to generate the test cases automatically, according to the system requirements. System requirements are transformed using use case diagrams, event flow, and sequence diagrams. Event flow and sequence diagrams are used to verify the consistency of use cases as well as the validity of test cases. Hasling et al. [38] approach uses a similar method to safety-critical systems.

A pseudo-software developed by Jwo and Cheng [40] as a conceptual framework for the development and validation of iterative requirements, which facilitates the broad participation of stakeholders, realizing the tangibility of the software under construction in the initial stage through simulation. The requirements recorded in highly readable forms, including templates for presentation and descriptions of free-form text for computational logic. It can be used for development and validation of safety-critical requirements in any method of software development, for safety-critical software or not.

Kelley [13] discusses a technique for automatically generating test cases from system requirements models (SpecTRM-RL models)—a methodology that can also apply to safety-critical systems because of similarity to other approaches that have achieved excellent results in MBT. SpecTRM-RL is a requirement specification language developed by Professor Nancy Leveson at MIT.[7] The goal was to develop algorithms to generate test cases and examine the effectiveness of these algorithms.

Raja [48] provides an overview of requirements validation techniques, such as requirements inspections, prototyping requirements, requirements testing, and point-of-view requirements validation. It highlights the pros and cons of these requirements validation techniques. The empirical methodologies presented can also be applied to safety-critical systems.

Straszak and Smialek [53] present the concept and tool requirements-driven software testing—ReDSeT, which allows the automatic generation of integrated tests based on different types of requirements. Tests expressed in the newly introduced test specification language (TSL). The basis for the generation of functional tests is detailed models of use cases. However, when combining different types of requirements, relationships between tests are created.

Yau [58] presents an object-oriented requirements specification (OORS) verification approach in software development for safety-critical embedded systems. The requirements specification generated by object-oriented analysis is described using a formal specification language transformed into an information tree. Thus, the completeness and consistency of the requirements specification expressed concerning the information tree checked by comparing it with the original requirement condition.

Yu [60] describes the implementation of an integrated approach to software development review, inspection, and testing based on well-identified software requirements. The safety-critical requirements traceability procedure described in the document is a systematic method to help scientists and engineers achieve the goal.

Authors such as Hasling et al. [38], Ibrahim et al. [8], and Kelley [13] show the use of support tools for testing safety-critical requirements based on models, and the tools generate test cases from use case models and activity diagrams, sequence diagrams, and finite state of the UML. There is also the approach of Wendland et al. [56] which presents an approach to how behavior trees can extended as test activities; these models follow the IEEE830 standard, making the models based on safety-critical requirements more complete and testable for the later generation of test cases.

These approaches are applicable to any application domain, since respecting the limits of each approach, the safety-critical requirements modeling allows the engineer to extract and execute test cases before the software implementation.

4.3 RQ1.2: What are the pros and cons of these approaches to test safety requirements?

The motivation for this research question is to identify the advantages and disadvantages of using the selected approaches and identify the type of application domain that each applying best.

In every methodology, there are advantages and disadvantages to consider, and there is no method that is good for any requirements test, nor a test that guarantees total confidence and safety for a safety-critical software system.

Requirements modeling helps better understand how the system works. The advantages of TBM are related to the application of tests in the early stages in the safety-critical software development cycle; reduction in test time and consequently cost reduction; reduction in ambiguities present in the requirements; creation of test cases automatically from requirements models; consumption of scarce resources; besides collaborating in the detection of problems with safety requirements during the modeling [8, 13, 38, 39, 61].

The use cases can be represented by decision trees [13, 39], domain ontologies [63], statecharts [42, 46], or UML and sysML diagrams [8, 10, 13, 38, 39]. It is also possible during the analysis phase to correct ambiguities and incompleteness of safety requirements, and this is one of

[7] http://www.mit.edu.

the biggest reasons for failures in safety-critical software projects. However, it requires a certain level of skill of modeling testers, and it also requires an initial effort to define which is the best model type to apply [38].

The test of safety requirements based on alpha–beta cutting procedure guarantees to test the redundancy and ambiguity of the safety requirement, particularly emphasizing the inspection of the function deficiency and verifying the testability of the safety requirement by using of requirements trees [45]. The main disadvantage is the need for specific knowledge to apply this approach.

The advantages of the approaches in the studies of Hammani [11] and Sarwar et al. [50] refer to verification of the inadequate modeling of non-functional requirements and how these safety requirements are neglected in the system usability assessment process and requirements testing. Aiello et al. [28] and Farhat et al. [33] describe a formal requirements specification, ambiguity reduction, and Sutcliffe and Gregoriades [55], which show a new view on model checking on system requirements based on non-functional requirements, enabling verification solutions based on simulation and improving work efficiency as the advantages found in their work.

The unit test is advantageous by allowing greater test coverage, preventing regression, encouraging refactoring, and avoiding long debugging sessions on safety requirements [32].

With white box techniques, it is possible to execute essential parts of the program and be able to find useful values for the inputs, and on the other hand, the paths to be executed can be infinite; one can also stop to execute some ways as the automation is difficult [29, 32, 41, 50].

Dudila and Letia [32] and Seo and Choi [41] state that the black box test can be used in any test phase and applies to every programming paradigm and is effective in detecting errors. As with other methods, it has disadvantages because it depends on a good specification of requirements and does not guarantee the execution of essential parts of the system.

The main advantages and disadvantages found in primary studies, and it is available in "Appendix 2."

4.4 RQ1.3: How much are these approaches used for the industry practitioners?

The motivation for this research question is to verify the validation of these approaches in the industry scenario. To classify the primary studies, we identified the studies that report the application of approaches within the industrial sector for studies that did not leave specific the sector of application, aiming to evidence the application of the approach used in the industrial or academic area.

Table 9 describes a representation of the number of articles found for each sector and the definition according to the opinion provided by the authors or according to the application sector, academic, or industrial.

From 53 primary studies of SLR, 16 studies apply approaches for the industrial scope in the testing of software requirements, and these studies are applied in industry as described by the authors. There are also 37 studies that do not present evidence of use in the industry and therefore considered not relevant for the industry by the authors.

Observing the researchers participating in the primary studies collected 50% of the work was carried out by academic researchers who contributed to industrial approaches. [4, 6, 30, 37, 39, 47, 51, 68]. Studies by academics and industry professionals that demonstrate evidence of relevance for the industrial sector add up to 43.75% [8, 32, 40, 46, 53, 59, 65], and finally, one study (6.25%) contains an approach developed only by industry professionals.

When comparing the sector of application of the primary studies, it is possible to see in Fig. 6 that the publications in the industrial sector started in the year 2007 through the studies of Ibrahim et al. [8], Jwo and Cheng [40], and Gao et al. [46] at least on the bases that were used in SLR (see Table 4). On the other hand, it is seen that the publications in the academic branch began in the year 1993 with the study of Ravn et al. [49].

Ibrahim et al. [8] contributed to the industry with the development of a programming code generator that performs requirements testing based on the test cases generated from the software requirements. To validate their work, Ibrahim et al. [8] tested it in a beverage vending machine system, and he pointed out that by applying the approach, it was possible to reduce the cost of the testing phase and reduce the software production time. Dudila and Letia [32] and Kesserwan et al. [65] argue that requirements testing has become a strenuous activity as systems become more complex to meet growing needs.

Considering the test of software requirements to control trains and lines, Han et al. [37] and Yu et al. [59] present approaches for testing safety-critical systems requirements by describing requirements in formal specifications and model-based testing, respectively.

In the automotive field, Siegl et al. [51] describe requirements through models, analyzes and validates them for the derivation of test cases. The execution of the tests guarantees the system reduction in failures and generates a basis of acceptance criteria for validation of the system. In Siegl et al. [68], the approach developed by Siegl et al. [51] is applied in a German automotive system but with the development of a framework that assists it. The author explains that the approach can collaborate with projects in the

Table 9 Approaches that present evidence of importance to the industry

Study	The criterion of importance to the industry
Ali, 2010	The approach can be useful in developing better contextual goal models, identifying additional inconsistencies and conflicts that are difficult to detect through an entirely based manual approach only in the skills of the professionals
Ali, 2013	The technique applies in two case studies with spatial systems. Besides, the authors make references to government systems and systems that have a high cost for development
Bouskela, 2015	Applies a language primarily used in avionics and aerospace that is especially effective for model-based analysis and specification of complex embedded real-time systems
Cimatti, 2012	The approach was the basis of industrial design to validate the specification of requirements of the European Train Control System (ETCS)
Dudila, 2013	The technique used by the authors can replicate in other projects developed by industries because they have a great scope in application domains
Han, 2016	The authors developed and applied the approach within the industry
Hemmati, 2013	The results based on two industrial case studies in embedded systems show benefits and an improvement in test performance when using a similarity-based approach
Ibrahim, 2007	The automatic test case generator develops with the aim of reducing the generation of test cases from the system requirements and cost reduction in the testing process, so it is essential for the industry to apply in different fields of application
Jwo, 2007	Paper presents pseudo-software, a conceptual framework for the development, and validation of iterative requirements, which facilitates the full participation of stakeholders, realizing the tangibility of the software under construction in stage through simulation. This approach will help reduce software costs by replicating better development requirements
Kesserwan, 2017	The validation of the methodology conducted by studying the effectiveness of the test cases generated in the industrial case study regarding path coverage
Gao, 2007	This approach proposes a modeling method that is suitable to establish the requirements model for software-intensive avionics
Mirarab, 2008	The authors refine a method already used by the industry and improve it within the proposed framework
Siegl, 2010	An approach to testing model-based requirements for automotive systems
Siegl, 2011	An automated approach to requirements testing in automotive systems
Straszak, 2014	The approach is essential for the industry because it is automatic, advantageous, and less expensive
Yu, 2009	An approach to testing the requirements of safety-critical systems is that model-based testing in the railway operations system

aviation domain according to the degree of complexity of both.

Gao et al. [46] propose a modeling method to establish the requirements model for software-intensive avionics.[8] The proposed method makes an abstraction for common avionic system characters, including data, receiving, sending, scenes, events, conditions, and period. Using the requirement model generated by the method and in combination with a particular test case generation strategy, the test cases and the adjacent environment simulation models of the system under test create automatically.

Although 30.18% of the primary studies have found evidence of contribution to the industry with approaches developed within the industry or with case studies on industrial software, it is observable that interest in testing software requirements has been growing and consequently yielding good results. Meanwhile, several approaches are expected to emerge and the number of publications to increase.

4.5 RQ1.4: Is there evidence of integration between requirement engineers and testers in the approach?

The motivation for this research question is to understand to what extent these approaches improve communication between engineers and testers. With SLR, we note how little is there in integration between requirements engineers and system testers when it comes to testing software requirements and safety-critical requirements.

The approaches that report or evidence the interaction between engineers and testers correspond to 11.32% of the primary studies [41, 46, 47, 51, 57, 59]. Figure 7 shows the graphical representation of this quantity in the total number of SLR studies.

The approaches of Gao et al. [46], Seo and Choi [41], and Siegl et al. [51] perform the requirements modeling and then perform the test performed by the team of testers. Although the integration may seem simple or visible to the requirements testing context, it does not occur in other works

[8] The term avionics comes from aviation electronics. Avionics are designated aircraft navigation and communication systems, autopilot, and flight control systems, also include non-pilot onboard electrotechnical systems such as passenger video systems.

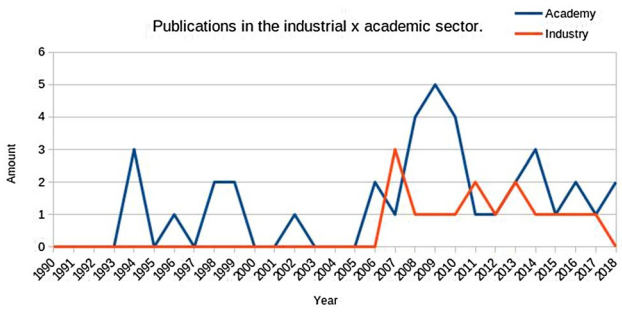

Fig. 6 Index of publications of papers in the industrial and academic sectors

such as Hasling et al. [38], Straszak and Smialek [53], and Wendland et al. [56].

In the approach of Gao et al. [46], engineers model software requirements in the tool proposed by the approach and the testers perform the tests. In Yu et al. [59], integration is implicit, but existence is noted for resembling other approaches such as Gao et al. [46] and Seo and Choi [41].

The authors of the primary studies found in the SLR do not provide details of the interaction process between requirements engineers and software testers, but it is possible to see that although this integration is not detailed. The other primary studies do not mention the integration of testers and engineers, and in addition to having no evidence of the presence of the tester, it is possible to note that this is a gap to be filled.

Perhaps because test teams are often outside the company for which the software product is developed, or even because managers do not unify test groups with development because testing can be biased. Methodologies that offer the possibility of integration of engineers and requirements testers throughout the development process and provide improvements for the final product and the company that develops it.

5 Threats to validity

The main characteristic of an SLR is the rigor to the protocol of its execution. With this in mind, some aspects raised that could pose risks to the results obtained and compromise the conclusions made.

Digital libraries in which the researcher went are the first threat to validity since other collections contain studies related to the subject. Digital libraries integrate the research chosen for the importance they represent in the computational environment and are pointed out as the main ones in the academic environment.

Due to a large number of articles found, it would not be possible to insert more papers due to the deadline and number of review participants. Search strings are also threats to validity because some terms that may not be involved or well defined with the theme or the lack of any that would make the result different from the search performed, although the results presented have the content appropriate.

The selection of studies is another point indicated as a threat to validity. Inclusion and exclusion criteria may have excluded some studies that contained interesting information for RQS. The criteria involved other aspects such as the authors' level of knowledge during the search, the writing may be ambiguous, and some critical information does not seem to fit the criteria.

The way the results presented in each selected primary study may influence the results of the SLR, and an article with confusing or disoriented results influences the extraction of the data, which in turn can ignore during the evaluation process. The evaluation of the quality of the articles can also be identified as a threat because the issues formulated according to the degree of knowledge of the SLR authors.

6 Research agenda

Motivated by the results of this SLR, we suggest a research agenda where some issues raised during the SLR can investigate in the future:

1. Which are the main approaches that are appropriate to deal with the testability of safety requirements and which can also deal with other types of requirements (functional, performance)?
2. How can the use of artificial intelligence support the testability of safety requirements?
3. How to promote greater integration between test and requirements engineering teams? Is there any notation/method that facilitates this integration?

Concerning systems of systems (SoS), the approach found that the traditional approaches of independent validation and verification focus on ensuring the satisfaction of the requirements by the software delivered and reason about the security of the software within the context of the system in which the software is running. This approach stresses the need to know the approaches that deal with the system stability of system requirements and to address existing gaps in this domain of application.

The results of the SLR are expected to encourage researchers to seek improvements to these approaches or to

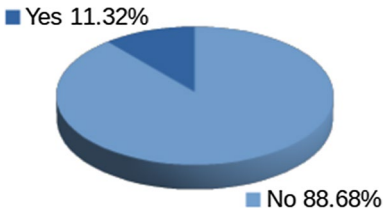

Fig. 7 Number of approaches that promote integration between requirements engineers and testers

create new approaches that integrate benefits in requirements testing and testing of critical safety requirements.

7 Conclusion

We presented an SLR with thematic in the test of critical safety requirements and test of non-critical software requirements. Several approaches that relate test requirements, verification, and validation were found in the literature and went through some phases of analysis and classification to answer the research questions proposed for SLR. Relevant finds identified along this review are highlighted as the following:

7.1 Approaches to testing safety requirements

RSL has selected several approaches that have highlighted the use of techniques for testing SSC requirements, and these approaches show techniques that mainly involve the use of templates for the creation and execution of test cases. Covering most of the selected approaches, model-based testing was the primary method used by the authors and using the UML as the main applied modeling language. Another interesting finding was the different types of application domains for the modeling of requirements for the derivation of test cases, from rail control software to space system software. The diversity of approaches found shows that the interest of researchers in the area of requirements testing is growing, regardless of whether it is critical or not, and the results presented by the authors and described in the subsections of the research questions are essential for the academic and industrial segments.

7.2 Use of approaches in the industry

Due to the few primary studies that have presented evidence of relevance to the industrial sector, it is possible to note that more work involving the industry still needs to be developed. The case studies and experiments developed in partnership with the sector presented good results and gave good suggestions for the future work. The data analyzed show that it is not very common to find studies published only by professionals of the industrial sector, and the reason for this occurrence is subject to approach in future works.

7.3 Advantages and disadvantages of approaches

Concerning the advantages and disadvantages belonging to the approaches, the authors showed the advantages through their results and conclusions but did not point out bad results or disadvantages that they obtained in their research. The disadvantages, since they highlighted in the studies, would be valuable information both to analyze the gaps in each methodology and to suggest improvements.

7.4 Integration between requirements engineering and testers

The literature found on this subject, most of the approaches do not promote this integration between teams, and few studies report some integration between testers and requirements engineers. It is notable that there is a great need to exert this practice more because the authors of these reports show good results of this integration, such as reduction in costs and time of software development, higher quality in functional tests, acceptance tests, interface tests, and especially the requirements test. The studies also point to the participation of testers at the beginning of development in parallel where testers establish test plans in conjunction with requirements engineers, and this partnership between teams promotes greater safety and quality in development as a whole.

However, defining which approach to use for requirements testing requires knowledge and skill of the requirements engineer and the tester. It turned out that inserting the tester into the planning stage of the software is a good practice to reduce development time and cost. We found few approaches that are really applied to the industry setting. However, requirements testing can be applied throughout the development life cycle.

Acknowledgements I would thank CAPES (Coordination of Improvement of Higher Level Personnel) for having subsidized 11 months of the pro-grad program in computer science at UNIFESP.

Appendix 1

See Table 10.

Table 10 List of selected primary studies

Author	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	SCR	CIT
Ahmed and Tripathi [14]	1	1	1			1	1			5	6
Aiello et al. [28]	1	0.5	1			1	1			4.5	0
Ali and Moawad [4]	1	1	1			1				4	1
Ali et al. [5]	1	1	1	1	1	1			1	7	28
Alves et al. [6]	1		1			1	1		1	5	2
Amyot et al. [29]	1					1	1			3	12
Andrade et al. [10]	1		1			1				3	36
Bouskela et al. [30]	1					1	1		1	4	2
Cimatti et al. [7]	1	1	1			1	1		1	6	7
Crow and Di Vito [23]	1	1	1			1	1			5	9
Dalal et al. [31]	1	1	1			1	1			5	3
Dokhanchi et al. [24]	1	1	1			1	1			5	0
Dudila and Letia [32]	1	1	1			1	1		1	6	0
El-Attar and Hezam [64]	1	0.5	1	1		1	1			5.5	2
Farhat et al. [33]	1	0.5	1			1	1			4.5	3
Felder and Morzenti [25]	1	0.5				1				2.5	8
Foster and Helm [34]	1	0.5	1			1	1			4.5	2
Fraser and Ammann [35]	1	0.5	1			1	1			4.5	2
Ghazel et al. [36]	1	0.5	1			1	1			4.5	0
Gutiérrez et al. [61]	1	1	1			1	1			5	4
Hammani [11]	1	1	1	1		1	1			6	4
Han et al. [37]	1	1	1			1	1		1	6	1
Hasling et al. [38]	1	0.5	1			1	1			4.5	15
Heitmeyer et al. [26]	1	1	1			1	1			5	173
Hemmati et al. [39]	1	0.5	1			1			1	4.5	33
Ibrahim et al. [8]	1	0.5	1			1			1	5.5	2
Jwo and Cheng [40]	1	0.5	1	1		1			1	5.5	1
Kelley [13]	1	1	1			1				4	3
Kesserwan et al. [65]	1	0.5	1	1		1	1		1	6.5	0
Seo and Choi [41]	1	0.5	1			1	1	1		5.5	3
Lee et al. [42]	1	1	1			1				4	0
Lei and Wang [43]	1	0.5	1			1	1			4.5	0
Yin et al. [44]	1		1			1	1			4	0
Liu et al. [45]	1	0.5	1			1	1			4.5	0
Gao et al. [46]	1	0.5	1			1	1	1	1	6.5	1
Mirarab et al. [47]	1	0.5	1			1	1	1	1	6.5	3
Ober et al. [62]	1	1	1			1	1			5	23
Raja [48]	1	1	1			1	1			5	3
Ravn et al. [49]	1	0.5				1	1			3.5	52
Sarwar et al. [50]	1	0.5	1			1	1			4.5	1
Schneider et al. [27]	1	0.5	1			1	1			4.5	34
Siegl et al. [51]	1	1	1			1	1	1	1	7	8
Siegl et al. [52]	1	0.5	1			1	1			4.5	1
Stachtiari et al. [63]	1	0.5	1			1	1			4.5	0
Straszak and Smiałek [53]	1	1	1			1	1		1	6	0
Sun et al. [54]	1	1	1			1	1			5	0
Sutcliffe and Gregoriades [55]	1	1	1			1	1			5	12
Wendland et al. [56]	1	1	1			1	1			5	5
Tang et al. [9]	1		1			1	1			4	0
Yang et al. [57]	1	1	1			1	1	1		6	0
Yau [58]	1		1			1	1			4	0
Yu et al. [59]	1		1			1	1			4	7
Yu [60]	1	1	1	1		1	1	1	1	8	1

Appendix 2

See Table 11.

Table 11 Pros and cons

Study	Advantage	Disadvantage
[14]	The author provides the correctness and consistency of a project specification. The approach is used to ensure no violation of confidential security requirements when policy compliance roles distributed to participants	Assigning property privileges in design can result in the violation of some critical requirement
[28]	Reduction in ambiguity and increase in accuracy, due to the well-defined syntax and semantics of the formal language adopted; Improving the efficiency of collaborative work between system manufacturers and vendors such as property models provides a shared representation and reference system requirements that can guide the testing and early validation of system and subsystem interactions	Requires knowledge in FORM-L to better understand the approach code
[4]	Establishes clear steps to run the test process in both phases (domain engineering phase and application engineering phase)	The article shows the new approach, but does not present a use case where the approach is applied
[5]	Provides a systematic process that guides the construction and analysis of contextual goal models; as a result of the evaluation presented by the author, the automated analysis discovers inconsistencies and deadly conflicts that are not recognizable by requirements engineers who develop contextual goal models	The application of the approach occurred in a single case study. More case studies are needed to generalize the conclusions; Authors approach has proven to be well-scaled with small and medium models of goal models while requiring additional optimization to handle large models
[6]	The V&V process has full support in computer-aided (StateRover) hardware. The effective use of requirements checking accomplishes by constructing correct and complete property assertions	Testing a critical safety system can be exceptionally difficult with traditional verification and validation techniques. However, the final implementation test may not be necessary to provide requirements validation because of the difficulty of ensuring coverage of test cases for all possible scenarios of failure
[29]	During integration, some avoided interactions ensure separate and complete preconditions, composing plug-ins into stubs according to the intent of the resources	There is the talk of the approach, but the development is not detailed
[10]	Estimates embedded software execution time, power consumption, and verification of system properties in the early stages of the development life cycle	It requires a high-level knowledge
[30]	Based on the central idea that teams with different specialties should be able to cooperate using modeling and simulation to build complex, secure and reliable systems. There must be a clear separation between requirements, design, and modeling of the physical system, which traceability must meet. The use of formal models helps remove ambiguities and omissions, and models should be easily readable by design and operation engineers and backed by efficient industrial tools	Authors use a high-level language
[7]	Simple enough to allow use by non-specialists at formal methods	It generates time effort for language learning
[23]	The most lasting contribution of the four case studies described was the development of reusable strategies and a clarification of the usefulness of formal methods techniques across a broad spectrum of maturity levels	An expert in the field of application is required to use the approach
[31]	The four case studies presented provide details and results of the application of large-scale combinatorial test techniques to various applications	The authors could give more details about the case studies because there is only one explanation of how they occurred
[1]	Enhances the elicitation process by providing feedback to users on validity, redundancy, and voidness issues	The specification presented is not integrated with the tool if integration could simplify development
[32]	The minimization of debugging effort at a later time in software development	The unit test can have biased results according to who performs it

Table 11 (continued)

Study	Advantage	Disadvantage
[64]	Allows to view system requirements differently and treat them for problems not yet seen	If the user has no experience with software engineering, the results will not be ideal
[33]	Tests non-functional requirements using aspects	There are two significant weaknesses when using aspects that are the inability of the aspect code to be woven at all points of execution and the lack of direct support for interlacing aspects with other aspects
[25]	The TRIO specification correctly captures and formalizes the requirements for the specified system, which in the early stages of the development process are based primarily on subjective user expectations or simple descriptions in natural language	Authors use a high-level language
[34]	Can perform a detailed evaluation using less time and resources between the requirements validation and system verification phases	Authors could have deepened the system requirements validation issues and seek new validations
[35]	Know two essential properties of test cases when testing against requirements: scope and propagation of property violations	It requires knowledge of linear temporal logic
[36]	Establishes the basis for a generic approach to the verification of temporal requirements of complex systems and develops software tools to implement the methodology	As the author reports, they could yield more fruitful results in information
[61]	Automation of the generation of functional test cases from software requirements, reducing effort and time in this process	Automatic generation of test cases is performed based on use cases, and applying some assumptions is imposed restrictions on functional requirements, format, and context. The design requirements are as a set of interactions between the system under test and a group of external actors
[37]	Extracts a set of general safety requirements from a variety of sources and sorts them by different characteristics. It includes safety requirements extracted from the existing relevant functional safety requirements to describe the dynamic aspect of the system	The application executing the approach is not fully developed
[38]	The approach aims to ensure the testability of software requirements	Testers need to be trained to create tests using models, rather than merely defining test scenarios
[26]	A formal analysis technique for automatic error detection such as non-deterministic type errors, missing cases, and circular definitions in requirements specifications	Needs to be knowledgeable about formal methods
[39]	Leads to significant savings concerning many test cases that do not need to execute	The approach does not test safety requirements
[8]	Generates the test case automatically	The approach does not consider non-functional requirements
[40]	Readability is the most relevant context that stakeholders can manipulate the pseudo-software as the actual software would do	Although some success achieved in applying pseudo-software, additional work is needed to make it more complete, for example, by adding features to manage requirements changes
[13]	Automating the generation of test cases saves resources; generating test cases is a time-consuming task, and test cases are generated before any code implosion, which will allow developers to use test cases as they develop code. Reducing the number of iterations between development and testing, saving even more resources	Generating test cases is a time-consuming task
[65]	The main advantage is to be able to generate tests from models generated with the requirements of the system, and this promotes to the engineer gain of time, ease and safety	Cons are the result, although it is relevant, there would be more examples in other areas of application of the tests performed by the authors
[41]	Know the properties of the five approaches presented	The author does not detail the operation of each approach, only briefly explains and compares them
[42]	The approach has advantages against the conventional process of software development, especially for to error correction, verification, and validation. The model-driven software development process has driven by shorter product development cycles, increased software complexity, reduced product quality expectation, and reduced cost	The model-based approach requires a suitable software model as well as a simulation tool or program and also requires a lot of additional cost and labor

Table 11 (continued)

Study	Advantage	Disadvantage
[43]	Validates the implementation of the software running on the target machine according to the requirement, i.e., the possibility of obtaining full coverage of the requirement	Model-driven testing requires a specific skill of the tester because knowledge about the technologies involved in the testing process is needed
[44]	Helps developers find errors sooner and makes the development process more efficient and economical	Authors use a high-level language
[45]	Can avoid the irregularity of the requirement by abstracting the shortage and ensure the success of the requirements test	Require skills with fractioning requirements and matching algorithms
[46]	The approach creates test cases automatically based on state graphs	It requires a high level of specific knowledge
[47]	Automates and formalizes the activities of the test processes	Unable to test critical safety requirements
[62]	Validate models UML by model simulation and verification, based on a mapping to an automaton-based model (communicating timed extended automata)	The authors say that although they already have UML 2.0 available, they have used version 1.4 but intend to update the search in the future
[48]	Ensures the elimination of unwanted requirements, and the test cases produced can be used in the final test of the system	The disadvantage of requirements testing is that it involves costs. For small businesses with a relatively smaller number of people, it may not be useful. Likewise, requirements testing requires experienced testers and requirements engineers. Small businesses may not have such requirements concerning information technology strength experienced. Besides, small businesses may not provide professional training to new people for requirements testing
[49]	It demonstrates how mathematical reasoning is used in verifying that the designs satisfy the requirements and in proving that a more detailed distributed design satisfies a centralized abstract design	Authors use a high-level mathematical language
[50]	Tests usability in a quantitative way, making it easy for the development organization to assess how much a particular system is usable; moreover, potential system failures can also be detected quickly, which serves as a basis for improving the system. Tests usability in a quantitative way, making it easy for the development organization to assess how much a particular system is usable; moreover, potential system failures can also be detected quickly, which serves as a basis for improving the system	The approach is lacking in clarity and detail
[27]	The approach identifies requirements with errors and malfunctioning components	There are several explanations not understood at work due to a high-level language
[51]	Deficiencies and ambiguities in the specification of requirements can be identified and clarified during modeling	The approach is not clear enough in the article to list a disadvantage
[63]	Early validation of requirements aims to reduce the need for costly validation testing and corrective measures in late stages of development	Requires the engineer to have high knowledge in the oriented derivation of formal properties
[53]	Can perform the acceptance test based on requirements models	The approach has a high cost and does not allow finding subjective defects because it is looking for expected errors
[54]	Reduct of test effort	It was demonstrated only for orientation DO-178
[55]	Introduces a new view on model verification in system requirements using a Bayesian belief network (BBN) technology to incorporate theoretically motivated predictions of human error and system reliability	Requires specific knowledge of BBN
[56]	The advantage pointed out by the author is the elicitation of the test requirements, analyzing the tree of integrated behavior from a tester. In doing so, the step to test the specifications becomes more straightforward because the information relevant to the specification of the test cases is present and does not need the graduated determination in any way	There is little literature for this approach

Table 11 (continued)

Study	Advantage	Disadvantage
[9]	Can be fully automated for testing	It is not clear that this approach is adaptable to other case studies besides that presented by the author
[57]	Covers all software testing processes	Demands specific knowledge of the approach
[58]	Can transform the object-oriented requirements specification into natural language	The author could have applied in a real system for testing
[59]	The structure maps each possible input and output and test criterion that is related to the functional safety of the critical safety system and can also relate to a well-defined set of system failures	The approach does not provide details in the article
[60]	Reduct of cost for troubleshooting software and bugs	The technology developed is already outdated and needs improvements in processes and methods

References

1. Delamaro M, Jino M, Maldonado J (2017) Introdução ao teste de software, vol 1. Elsevier, Brasil
2. Lau MF, Yu YT (2005) An extended fault class hierarchy for specification-based testing. ACM Trans Softw Eng Methodol 14(3):247–276
3. Leveson NG (1995) Safeware: system safety and computers. ACM, New York
4. Ali MM, Moawad R (2010) An approach for requirements based software product line testing. In: 2010 the 7th international conference on informatics and systems (INFOS), 1 March 2010
5. Ali R, Dalpiaz F, Giorgini P (2013) Reasoning with contextual requirements: detecting inconsistency and conflicts. Inf Softw Technol 55(1):35–57
6. Alves MCB, Drusinsky D, Shing M-T (2011) A practical formal approach for requirements validation and verification of dependable systems. In: 2011 Fifth Latin- American Symposium on Dependable Computing Workshops. IEEE, pp 47–51
7. Cimatti A, Roveri M, Susi A, Tonetta S (2012) Validation of requirements for hybrid systems. ACM Trans Softw Eng Methodol 21(4):1–34
8. Ibrahim R, Saringat MZ, Ibrahim N, Ismail N (2007) An automatic tool for generating test cases from the system's requirements. In: 7th IEEE international conference on computer and information technology (CIT 2007). IEEE, pp 861–866
9. Tang W, Ning B, Xu T, Zhao L (2010) Scenario-based modeling and verification for CTCS-3 system requirement specification. In: 2010 2nd International conference on computer engineering and technology, vol 1. IEEE, pp V1-400–V1-403
10. Andrade E, Maciel P, Gustavo C, Bruno N (2009) A methodology for mapping SysML activity diagram to time petri net for requirement validation of embedded real-time systems with energy constraints. In: 2009 3rd international conference on digital society. IEEE, pp 266–271
11. Hammani FZ (2014) Survey of non- functional requirements modeling and verification of Software Product Lines. In: 2014 IEEE 8th international conference on research challenges in information science (RCIS), pp 1–6
12. Sharma A, Kushwaha SD (2011) A metric suite for early estimation of software testing effort using requirement engineering document and its validation. In: 2011 2nd International conference on computer and communication technology, ICCCT-2011, pp 373–378
13. Kelley K (2009) Automated test case generation from correct and complete system requirements models. In: 2009 IEEE aerospace conference. IEEE, pp 1–10
14. Ahmend T, Tripathi AR (2007) Specification and verification of safety requirements in a programming model for decentralized CSCW systems. ACM Trans Inf Syst Saf 10(2):7
15. Sommerville I (2011) Engenharia de software. In: Tradução Ivan Bosnic e Kalinka G. de O. Goncçalves; revisão técnica Kechi Hirama, 9 edition. Pearson Prentice Hall, São Paulo
16. Driskell SB, Murphy J, Michael BJ, Shing MT (2010) Independent validation of software safety requirements for systems of systems. In: 2010 5th international conference on system of systems engineering, SoSE 2010
17. Martins LEG, Gorschek T (2016) Requirements engineering for safety-critical systems: a systematic literature review. Inf Softw Technol 75:71–89
18. Kitchenham B, Charters S (2007) Guidelines for performing systematic literature reviews in software engineering. Engineering 2:1051

19. Vilela J, Castro J, Martins LEG, Gorschek T (2017) Integration between requirements engineering and safety analysis: a systematic literature review. J Syst Softw 125:68–92
20. Gurbuz HG, Tekinerdogan B (2018) Model-based testing for software safety: a systematic mapping study. Softw Quality J 26(4):1327–1372
21. Häser F, Felderer M, Ruth B (2014) Soft-ware paradigms, assessment types and non-functional requirements in model-based integration testing: a systematic literature review. In: Proceedings of the 18th international conference on evaluation and assessment in software engineering, EASE'14, New York, NY, USA. ACM, pp 29:1–29:10
22. Unterkalmsteiner M, Feldt R, Gorschek T (2014) A taxonomy for requirements engineering and software test alignment. ACM Trans Softw Eng Methodol (TOSEM) 23(2):16:1–16:38
23. Crow J, Di Vito B (1998) Formalizing space shuttle software requirements: four case studies. ACM Trans Softw Eng Methodol 7(3):296–332
24. Dokhanchi A, Hoxha B, Georgios F (2017) Formal requirement debugging for testing and verification of cyber-physical systems. ACM Trans Embed Comput Syst 17(2):1–26
25. Felder M, Morzenti A (1992) Validating real-time systems by history- checking TRIO specifications. In: Proceedings of the 14th international conference on software engineering—ICSE'92, vol 3, New York, USA. ACM Press, pp 199–211
26. Heitmeyer CL, Jeffords RD, Labaw BG (1996) Automated consistency checking of requirements specifications. ACM Trans Softw Eng Methodol 5(3):231–261
27. Schneider F, Easterbrook SM, Callahan JR, Holzmann GJ (1998) Validating requirements for fault tolerant systems using model checking. In: Proceedings of IEEE international symposium on requirements engineering: RE'98. IEEE Comput. Soc, pp 4–13
28. Aiello F, Garro A, Lemmens YA, Stefan D (2017) Simulation-based verification of system requirements: an integrated solution. In: 2017 IEEE 14th international conference on networking, sensing and control (ICNSC), vol 1. IEEE, pp 726–731
29. Amyot D, Logrippo L, Buhr RJA, Gray T (1999) Use case maps for the capture and validation of distributed systems requirements. In: Proceedings IEEE international symposium on requirements engineering (Cat.No.PR00188), pp 44–53
30. Bouskela D, Nguyen T, Audrey J (2015) Towards a rigorous approach for verifying cyber-physical systems against requirements. In: 2015 IEEE electrical power and energy conference (EPEC). IEEE, pp 250–255
31. Dalal SR, Jain A, Karunathi N, Leaton JM, Patton CM, Lott GC, Horowitz BM (2005) Model based testing in practice at microsoft. Electr Notes Theor Comput Sci 111:5–12
32. Dudila R, Letia IA (2013) Towards combining functional requirements tests and unit tests as a preventive practice against software defects. In: 2013 IEEE 9th international conference on intelligent computer communication and processing (ICCP). IEEE, pp 279–282
33. Farhat S, Simco G, Mitropoulos FJ (2010) Using aspects for testing nonfunctional requirements in object-oriented systems. In: Proceedings of the IEEE SoutheastCon 2010 (SoutheastCon). IEEE, pp 356–359
34. Foster GJ, Helm AL (2010) A unified approach to requirements validation and system verification. In: 2010 IEEE international systems conference, IEEE, pp 404–408
35. Fraser G, Ammann P (2008) Reachability and propagation for LTL requirements testing. In: 2008 the 8th international conference on quality software. IEEE, pp 189–198
36. Ghazel M, Masmoudi M, Toguyeni A (2009) Verification of temporal requirements of complex systems using UML patterns, application to a railway control example. In: IEEE international conference on system of systems engineering. SoSE2009, pp 1–6
37. Han L, Liu J, Zhou T, Sun J, Chen X (2016) Safety requirements specification and verification for railway interlocking systems. In: 2016 IEEE 40th annual computer software and applications conference (COMPSAC), vol 1. IEEE, pp 335–340
38. Hasling B, Goetz H, Beetz K (2008) Model based testing of system requirements using UML use case models. In: 2008 international conference on software testing, verification, and validation. IEEE, pp 367–376
39. Hemmati H, Arcuri A, Briand L (2013) Achieving scalable model-based testing through test case diversity. ACM Trans Softw Eng Methodol 22(1):1–42
40. Jwo J-S, Cheng YC (2007) Pseudo software: a new concept for iterative requirement development and validation. In: 14th Asia-Pacific software engineering conference (APSEC'07). IEEE, pp 105–111
41. Seo KI, Choi EM (2006) Comparison of five black-box testing methods for object- oriented software. In: 4th International conference on software engineering research, management and applications (SERA'06). IEEE, pp 213–220
42. Lee K-H, Min P-G, Cho J-H, Lim D-J (2012) Model-driven requirements validation for automotive embedded software using UML. In: 8th International conference on computing technology and information management (ICCM), vol 1, pp 46–50
43. Lei H, Wang Y (2016) A model-driven testing framework based on requirement for embedded software. In: 2016 11th International conference on reliability, maintainability and safety (ICRMS). IEEE, pp 1–6
44. Yin L, Liu J, Li X (2009) Validating requirements model of a B2B system. In: 2009 8th IEEE/ACIS international conference on computer and information science. IEEE, pp 1020–1025
45. Liu G, Huang S, Piao X (2008) Study on requirement testing method based on alpha-beta cut-off procedure. In: 2008 international conference on internet computing in science and engineering. IEEE, pp 396–402
46. Gao M, Zhong D, Lu M, Yin Y (2007) Research on test requirement modeling for software-intensive avionics and the tool implementation. In: 2007IEEE/AIAA 26th digital avionics systems conference. IEEE, pp 6.D.2 1–6.D.2 10
47. Mirarab S, Ganjali A, Ladan T, Li S, Liu W, Morrissey M (2008) A requirement-based software testing framework: an industrial practice. In: 2008 IEEE international conference on software maintenance, pp 452–455
48. Raja UA (2009) Empirical studies of requirements validation techniques. In: 2009 2nd International conference on computer, control and communication. IEEE, pp 1–9
49. Ravn AP, Rischel H, Hansen KM (1993) Specifying and verifying requirements of real- time systems. IEEE Trans Softw Eng 19(1):41–55
50. Sarwar T, Habib W, Arif F (2013) Requirements based testing of software. In: 2013 2nd International conference on informatics and applications (ICIA). IEEE, pp 347–352
51. Siegl S, Hielscher K-S, German R (2010) model based requirements analysis and testing of automotive systems with timed usage models. In: 2010 18th IEEE international requirements engineering conference. IEEE, pp 345–350
52. Siegl S, Hielscher K-S, German R, Berger C (2011) Automated testing of embedded automotive systems from requirement specification models. In: LATW 2011—12th IEEE Latin- American Test Workshop
53. Straszak T, Smialek M (2014) Automating acceptance testing with tool support. In: Automating acceptance testing with tool support, vol 2, pp 1569–1574
54. Sun Y, Brain M, Kroening D, Hawthorn A, Wilson T, Schanda F, Jimenez FJG, Daniel S, Bryan C, Broster I (2017) Functional requirements-based automated testing for avionics. In: 2017 22nd

international conference on engineering of complex computer systems (ICECCS), vol 1. IEEE, pp 170–173
55. Sutcliffe A, Gregoriades A (2002) Validating functional system requirements with scenarios. In: Proceedings IEEE joint international conference on requirements engineering, vol 2002—Janua. IEEE Comput. Soc, pp 181–188
56. Wendland M, Schieferdecker I, Vouffo-Feudjio A (2011) Requirements-driven testing with behavior trees. In: 2011 IEEE fourth international conference on software testing, verification and validation workshops. IEEE, pp 501–510
57. Yang C, Hu X, Zhu Y, Huang M (2014) The domain knowledge-based software test model researches. In: 2014 10th International conference on reliability, maintainability and safety (ICRMS). IEEE, pp 323–328
58. Yau SS (1994) An approach to object-oriented requirements verification in software development for distributed computing systems. In: Proceedings eighteenth annual international computer software and applications conference (COMPSAC 94), pp 96–102
59. Yu G, Xu ZW, Du JW (2009) An approach for automated safety testing of safety-critical software system based on safety requirements. In: 2009 International forum on information technology and applications, vol 3. IEEE, pp 166–169
60. Yu WD (1994) Verifying software requirements: a requirement tracing methodology and its software tool-RADIX. IEEE J Sel Areas Commun 12(2):234–240
61. Gutiérrez JJ, Escalona MJ, Mejías M (2015) A model-driven approach for functional test case generation. J Syst Softw 109:214–228
62. Ober I, Graf S, Ober I (2006) Validating timed UML models by simulation and verification. Int J Softw Tools Technol Transf 8(2):128–145
63. Stachtiari E, Mavridou A, Panagiotis P, Bliudze S, Sifakis J (2018) Early validation of system requirements and design through correctness-by-construction. J Syst Softw 145:52–78
64. El-Attar M, Hezam AA-G (2016) Using safety robustness analysis for early-stage validation of functional safety requirements. Requir Eng 21(1):1–27
65. Kesserwan N, Dssouli R, Bentahar J, Stepien B, Labrèche P (2019) From use case maps to executable test procedures: a scenario-based approach. Softw Syst Model 18(2):1543–1570
66. Saito S, Takeuchi M, Yamada S, Aoyama M (2014) RISDM: a requirements inspection systems design methodology: Perspective-based design of the pragmatic quality model and question set to SRS. In: 2014 IEEE 22nd international requirements engineering conference (RE). IEEE, pp 223–232
67. Khosrowjerdi H, Meinke K, Andreas R (2018) Virtualized-fault injection testing: a machine learning approach. In: 2018 IEEE 11th international conference on software testing, verification and validation (ICST). IEEE, pp 297–308
68. Siegl S, Hielscher K-S, Reinhard G, Christian B (2011) Automated testing of embedded automotive systems from requirement specification models. In: 2011 12th Latin American Test Workshop (LATW). IEEE, pp 1–6