Computers, Materials & Continua
DOI:10.32604/cmc.2021.014391
Article

Tech Science Press

Automated Test Case Generation from Requirements: A Systematic Literature Review

Ahmad Mustafa[1], Wan M. N. Wan-Kadir[1], Noraini Ibrahim[1], Muhammad Arif Shah[3,*], Muhammad Younas[2], Atif Khan[4], Mahdi Zareei[5] and Faisal Alanazi[6]

[1]Department of Software Engineering, School of Computing, Universiti Teknologi Malaysia Johor Bahru, Johor, 81310, Malaysia
[2]Department of Computer Science, Government College University, Faisalabad, 38000, Pakistan
[3]Department of Software Engineering, Pak-Austria Fachhochschule Institute of Applied Sciences and Technology, Haripur, 22620, Pakistan
[4]Department of Computer Science, Islamia College Peshawar, Peshawar, Pakistan
[5]Tecnologico de Monterrey, School of Engineering and Sciences, Zapopan, 45201, Mexico
[6]Department of Electrical Engineering, Prince Sattam Bin Abdulaziz University, Al Kharj, 11942, Saudi Arabia
[*]Corresponding Author: Muhammad Arif Shah. Email: arif.websol@gmail.com
Received: 17 September 2020; Accepted: 18 October 2020

Abstract: Software testing is an important and cost intensive activity in software development. The major contribution in cost is due to test case generations. Requirement-based testing is an approach in which test cases are derivative from requirements without considering the implementation's internal structure. Requirement-based testing includes functional and nonfunctional requirements. The objective of this study is to explore the approaches that generate test cases from requirements. A systematic literature review based on two research questions and extensive quality assessment criteria includes studies. The study identifies 30 primary studies from 410 studies spanned from 2000 to 2018. The review's finding shows that 53% of journal papers, 42% of conference papers, and 5% of book chapters' address requirements-based testing. Most of the studies use UML, activity, and use case diagrams for test case generation from requirements. One of the significant lessons learned is that most software testing errors are traced back to errors in natural language requirements. A substantial amount of work focuses on UML diagrams for test case generations, which cannot capture all the system's developed attributes. Furthermore, there is a lack of UML-based models that can generate test cases from natural language requirements by refining them in context. Coverage criteria indicate how efficiently the testing has been performed 12.37% of studies use requirements coverage, 20% of studies cover path coverage, and 17% study basic coverage.

Keywords: Test case generation; functional testing techniques; requirements-based test case generation; system testing; natural language requirement; requirements tractability; coverage criteria

1 Introduction

Software testing has a primary role in evaluating software quality and minimizing the project's cost and delivery in time. The developed system's quality can be measured by the customer's approval and satisfaction of the developed system. Software testing can be defined as [1] "a process of exercising or evaluating a system or system components by manual or automated means to verify that it satisfies specified requirements." The test cases are designed to check that all functional requirements are successfully incorporated into the system. These test cases validate the developed system against its specifications, and it assured that all requirements from the customer are successfully incorporated in the resulting system.

Furthermore, the testing process is divided into three subtasks, namely, test case generation, execution, and evaluation [2]. In this systematic literature review, we are focusing on test case generation techniques from requirements. According to IEEE Std 829, a test case a set of inputs, conditions, and expected results for under test systems. A test document states inputs, expected results, and a set of execution conditions for a test item [3]. The test phase is considered one of the most expensive tasks which take 40%–60% of the time, cost, and effort. Most of the time, software testing is planned with very few resources, without proper techniques or tools that may support it [4]. Often time delay in the software testing phase occurs due to the lack of resources [5]. Moreover, software testing is challenging for the software testing research community from academia and the software development industry [6].

Recently, automation in the testing process is considered to reduce time, costs, and effort. The automated case generations can be used to uncover faults and defects in artifacts during the early stage of software development [7]. The Software requirements specifications should be precise and unambiguous in order to support the automation process in software testing [8]. In the software development industry, the natural language (NL) is often used for requirements documentation because it is easy to understand, and the no additional training required to understand NL. After documenting the requirements, software design diagrams are modeled based on these requirements. These unified models are used in the development of the system. Manual generation of test cases from the requirements and design phase can be time-consuming and prone to human error [9].

The software requirements-based test (RBT) cases generation technique is widely used in the software development field to support software tester. The generated test cases with RBT guarantee that the software system is performing as per customer expectations. These tests provide a ground for functional testing of the system [10]. Functional testing is a critical factor for risk management, quality management, and delivery of the resulting system well in time. This systematic review pinpoints the possible challenges linked with test case generation techniques from requirements. Moreover, this paper attempts to bridge the gap in requirements and testing and areas to improve requirements and testing. However, the focus of the study on automated testing, and we try to find the answers to the following questions:

- What are the existing approaches to generate test cases from requirements?
- What are the challenges in requirements-based testing?

To answer these questions, we designed a systematic procedure to retrieve all the works related to requirements-based test cases generations. The rest is organized in the following sections: (2) The Material and Method—(3) Results and Discussion, (4) challenges in Requirements Based Testing, (5) Threat to validity, and (6) Conclusion.

2 Research Methodology

A Systematic Literature Review (SLR) is a method in Evidence-Based Software Engineering (EBSE) proposed by Kitchenham et al. [11]. First, we defined the review protocol, as shown in Fig. 1 in which we explained the objectives, research questions, search strategy, studies selection criteria, and search string definition as presented below.

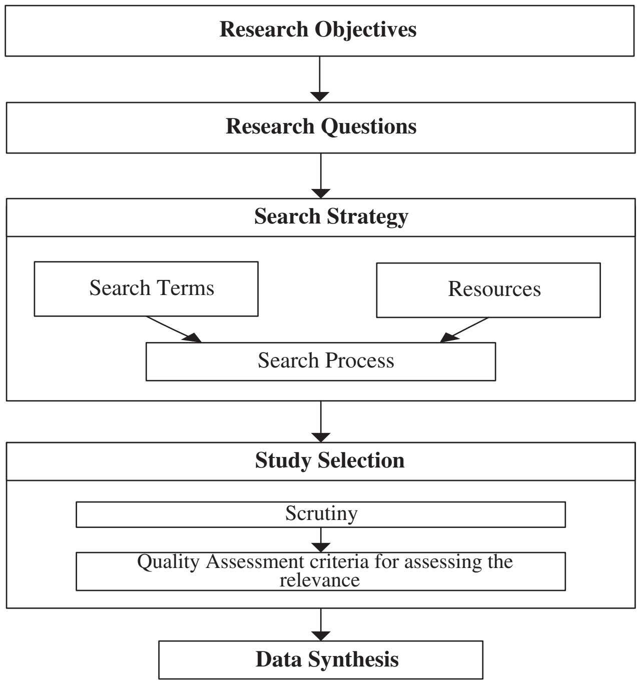

Figure 1: Phases of the review study

2.1 Review Protocols

A systematic review protocol defines the reason, speculation, and arranging techniques for the review. It was planned before a review is started and used as a guide to carry out the review. Fig. 1. shows the phases of review, such as objectives, research questions, search strategy, study selection, and data analysis.

2.2 Research Objectives

After selecting the domain, the following aim of this review study are stated below:

- Identify the existing proposed solution that addresses the automated test cases generations of functional testing from requirements.
- Analyze on the base of quality assessment criteria indicators if current approaches offer an appropriate solution for a real project and improve the quality of test phase with reduction of cost and time to market, requirement coverage, and dynamic change management methodology.
- Identify the gaps in existing approaches.
- Propose future works about the enhancement of the automated process of functional test case generation from requirements.

2.3 Research Questions

The research questions are an essential part of the systematic review, as suggested by Kitchenham et al. [11]. To achieve the objectives mentioned in (Section 2.2), we identify the following research questions:

- What are the current approaches to generate test cases from requirements?
- What are the challenges in requirements-based testing?

2.4 Search Terms

From our two research questions, we derived the following keywords: "functional test cases generation, functional testing, functional testing techniques, requirements-based test cases generation, requirements-based testing techniques, system testing, requirements change effect on software testing, traceability of requirements, coverage criteria." A search string was constructed using relevant terms based on research questions

2.5 Searching Strategy

The SLR procedure advises searching from several electronic sources [12]. During phase 1 of the search, we search from Scopus, a web of science, and google scholar. We read the abstract of studies. After reading the studies at the 2nd phase, we downloaded the related studies from the electronic databases, as shown in Fig. 2.

Fig. 2. shows a total of 410 studies found related to the study area during the publishing years 2000–2018. These studies are published in different digital Libraries such as ACM, IEEE, Science Direct, Springer, and other digital libraries. Fig. 2. shows the number of published papers in the journal, conference, and book chapter on test case generation. Fig. 3. shows that the research area under observation is getting the attention of researchers.

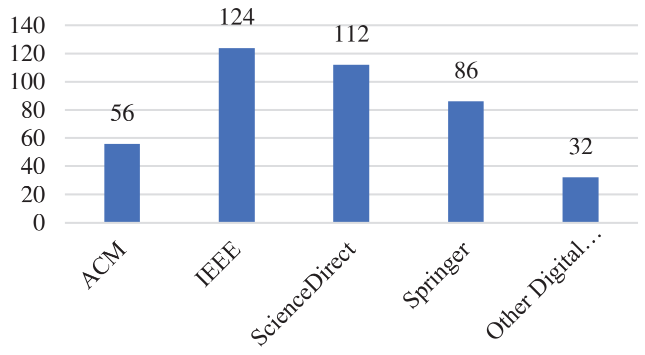

Figure 2: Number of studies digital libraries wise

2.6 Studies Selection Process

For the selection of appropriate studies, many criteria were developed. The studies that met the following conditions were included in primary studies.

- Research papers
- Studies that contains system test case generation process.
- Studies that proposed an approach or methodology for system-level test case generation.
- Papers written in the English language.

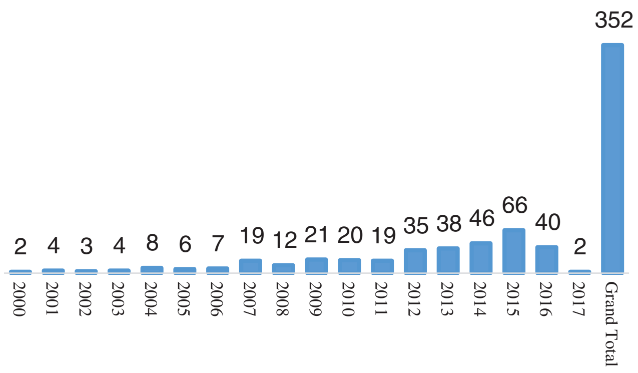

Figure 3: The accumulated number of publications per year

The following types of paper were excluded:

- Studies those are not addressed requirements-based test case generations.
- Studies that are not addressed problems/issues/challenges in requirements-based testing.

Fig. 5 shows the number of search phrases and selected studies numbers at each phase. In phase 1, the search was performed on digital libraries mentioned in Section 2.6 by applying the search terms specified in Section 2.5. The search was on the base of titles, abstracts, and keywords of the research studies. We obtained 410 studies. Many of the studies were irrelevant and not precisely addressing our research questions. Therefore, in phase 2, we remove duplicate studies, and some studies were not in the English language. As a result, 352 studies were obtained. At phase 3, we categorized studies into four parts as shown in Fig. 4.

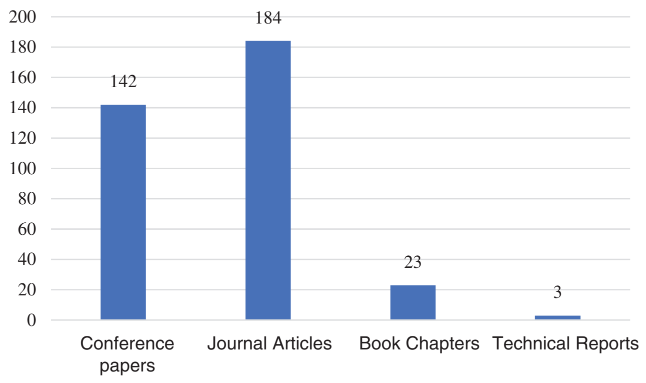

Figure 4: Number of articles categorized wise

Moreover, those studies were also removed because they are not fulfilling the following criteria:

In stage 4, we apply the quality assessment criteria mentioned in Section 2.8. After phase 4, we again verified the titles of research papers in the of 30 primary selected studies.

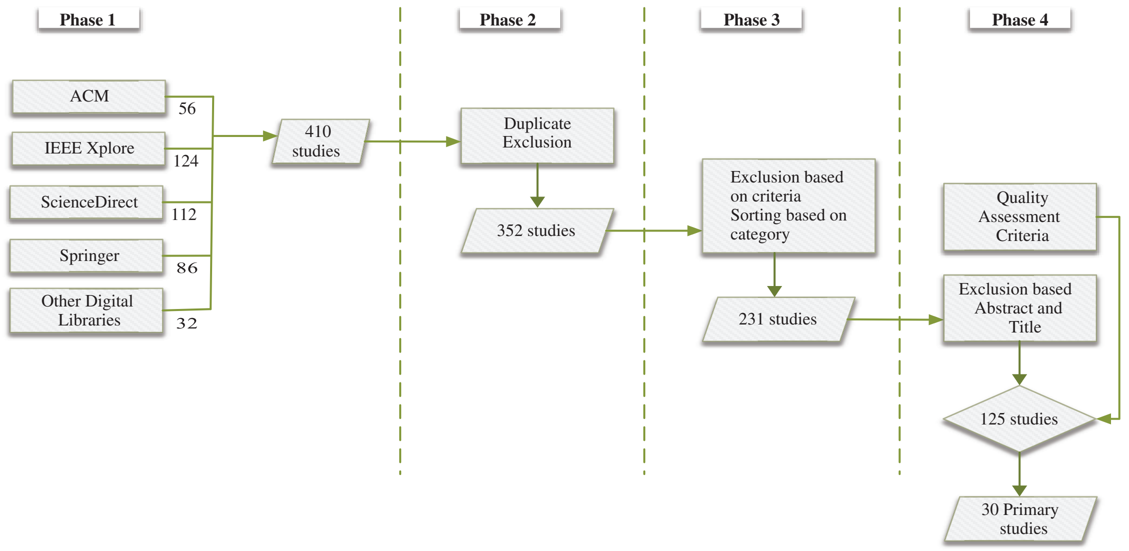

Figure 5: Phases of the search strategy

Table 1: Primary studies data extraction form

No	Data extraction attributes	Description	Address
General detail			
01	Study_id	Unique id of each primary study (PS)	
Studies description			
02	Title	Full Title of the primarily selected study	
03	Authors	Authors name of primary studies	
04	Year	Study publication year	
05	Type of study	Type of documents: conference paper, journal article, book chapter or technical report	
06	Publisher	Name of publisher	
Study content			
07	Objective	Main objectives of the study	RQ1, RQ2
08	What are test cases generation techniques which take input from the requirement	What is technique input to generate test cases Notation used	RQ1, RQ2
10	Coverage criteria	How effectively has the testing been performed?	RQ1
11	Challenges identified	What are the problems while generating test cases from requirements?	RQ2
12	Tools support	Is the approach is supported by tool and support is partial/fully automated?	RQ1
13	Validation	Which method is used for validation of the study?	RQ1, RQ2

2.7 *Quality of the Studies*

Kitchenham et al. [11] proposed criteria for assessing the primary studies' quality, reducing favoritism, and increasing validity when assessing the primary studies. Below mentioned questions were applied for the checking of the quality of primary studies.

- Are the aims and objectives of selected primary studies are defined and reported?
- Is the domain/context in which the research was conducted is defined efficiently?
- Are the test case generation techniques from the requirements (RQ1) in the study is mentioned clearly?
- Is there any reliable method for validation of the technique/approach?
- Are the outcomes of the study stated clearly and related to the object of study?
- Does the conclusion of the study is according to the defined goal of the study?

2.8 *Data Extraction*

For studies included in the automated test case generation review from requirements, we identified twelve elements mentioned in Tab. 1. In the data extraction stage, we collect the information related to the research questions from studies. For data extraction purposes, we develop a form with a test-retest process [11] for the reliability and correctness of the selected data. The data extraction form is shown in Tab. 1.

3 Results and Discussion

3.1 *Primary Studies*

After exploring for online resources of digital databases mentioned in Section 2.6, the first 410 studies were found. The study selection process is discussed in Section 2.7. As a result, we got 30 primary studies addressing requirement based test case generation: techniques, tool, and context of studies mentioned in Tab. 2.

3.2 *The Input to Generate Test Cases*

Fig. 6 shows a pie chart of the dissemination of input used to generate test cases. Most of the approaches take input from UML diagrams to generates functional test cases. 21% of approaches are taken input from natural language requirements. However, the focus of these approaches is the real-time embedded system. However, the researcher is intended to map the concept of enterprise application software (EAS), desktop application software. Studies that focus on the UML diagram as input are only focusing on functional test case generation. UML is unable to capture the non-functional requirements of the system.

3.3 *Notation Used*

The notation used is a fundamental aspect that is considered while defining requirements. Fig. 7 shows the analyzed approaches and notation used as input of requirements for the test case generation process. Most of the approaches are using 31% of approaches are using metamodel to capture information from requirements. Other notations used, such as UML diagrams, Behaviors trees, formal notation, and SCR notations.

3.4 *Transformation Techniques*

The transformation mechanism takes input from requirements using specific notations and finds the best set of test suits while covering requirements. Fig. 8. shows that 30% of studies are using particular generation algorithms. Other studies use standard techniques like active testing techniques model, BEAST Technique, bounded model checking, Class diagram, CPM, and Cyclomatic complexity. ECP, BVA & CRF, equivalency partition, NLP, Petri net, Round strip strategy and Extended Use cases, State transition diagram, symbolic model checker, text mining & symbolic execution methodology, TSL, and some techniques are unknown.

Table 2: Tool support and validation of approaches

ID	Tools	Context	ID	Tools	Context
PS1	Junit	Case study the car rental system	PS16	TestGen-IF	A case study on intelligent transportation systems
PS2	GenTCase	Experiment study	PS17	jUCMNav	Application message sequence charts
PS3	LEIRIOS	Overview study	PS18	Unknown	Library system, web publishing system
PS4	No	Case study embedded system	PS19	EDT-Test	Experiment study
PS5	No	Experiment study	PS20	NDT-suite	NA
PS6	Visual Paradigm	Case study enterprise system	PS21	BT Analyser	Automated teller machine example
PS7	RT-Tester, T-VEC tool	Aerospace and automotive industry	PS22	APTGEN	Digital applications
PS8	STATEST 1.1.0.	Case study bank system	PS23	Eclipse base Tool ReDSeT	Case study
PS9	DODT tool MoMuT:: REQ MATLAB Simulink	Embedded system	PS24	Unknown	Automotive domain
PS10	Junit	Agile testing in the network	PS25	Junit	Experiment study web-based e-commerce system
PS11	C-Set	Automobile control system	PS26	aToucan4Test	Case networking domain
PS12	Unknown	Model driven testing	PS27	Spreadsheet	No
PS13	C&L tool	Web application	PS28	T-VEC tool	Vending machine, nuclear power plant control
PS14	NDT- suite	Experiment study web application	PS29	REBATE	Case study industrial cyber-physical systems
PS15	aToucan4Test	Enterprise and autopilot application	PS30	known	Experiment aerospace

3.5 Requirements Coverage Criteria

This section briefly summarizes the requirements coverage criteria results. Coverage criteria indicate how efficiently the testing has been performed. There are two main categories of coverage criteria: structural coverage and requirement coverage [13]. Fig. 9. Indicates that 12.37% of studies use requirements coverage, 20% of studies cover path coverage, and 17% study basic coverage. Similarly, other used coverage criteria include code coverage, edge and flow weight coverage, events coverage, fault coverage, and functional and MC/DC coverage.

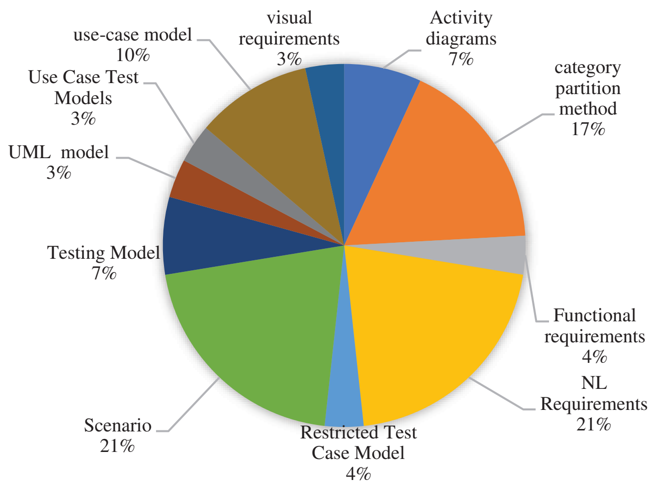

Figure 6: Input technique to generate test cases

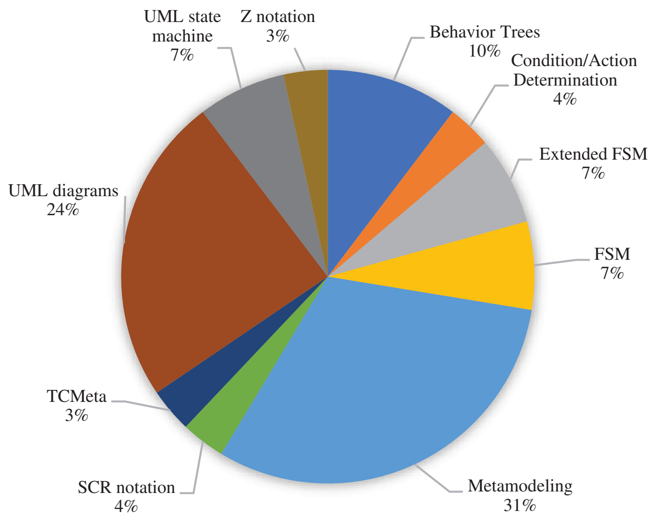

Figure 7: Notation used in test cases generation from requirements

Mainly three are three coverage criteria (functional, structural chain, tolerance range) to establish a test suite's adequacy. The functional criterion is based on a user-specified requirement for the verification of each functional requirement. This criterion validates that functional requirements cover the given set of test cases.

The structural coverage criteria check how the given test suite covers many behaviors of the requirements specification. Finally, the resistance goes criteria to determine what kind of qualities distinguished resilience scope of each information variable must be chosen for definitive tests, such as an average worth, two limits esteem, and one out-of-run esteem [14]. Almeida et al. [15] proposed an approach for requirement coverage for aerospace safety-critical software development. Verification and validation process, for example, the use of MC/DC (Modified

Condition/Decision Coverage). The proposed approach automatically generates test cases from the requirements written in XML format.

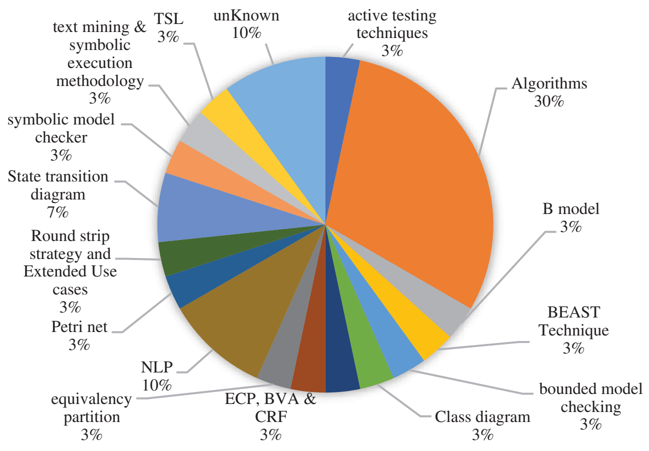

Figure 8: Transformation techniques

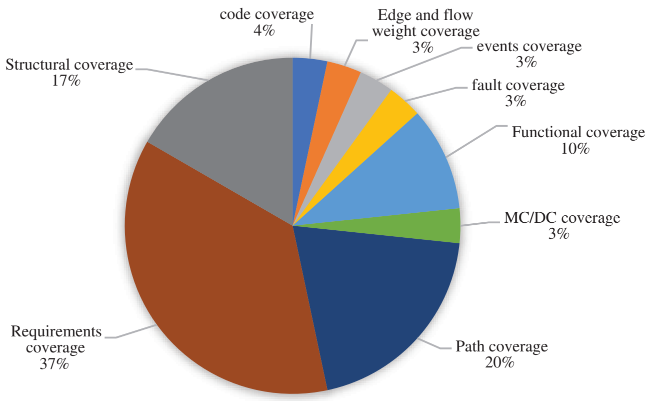

Figure 9: Coverage criteria used in primary studies

The coverage criteria specify the requirements to be covered by the system test cases. However, in reality, due to the infeasibility problem, such criteria are limited. Infeasibility problem dealt with system specifications that cannot be covered by any test case. To resolve this issue, authors revisited and tried to improve existing techniques such as the resistance go criteria to determine each information variable's qualities distinguished resilience scope must be chosen for explicit tests, such as an ordinary worth, two limits, and one out-of-run esteem. Researchers proposed a new technique lightweight gray box scheme, by combining these two techniques [16].

3.6 Tools Used for Test Cases Automation

The tool field indicates that the degree of tool development to support each approach. The details of these tools and their context are described in Tab. 2.

3.7 Relationship Between Requirement and Test Case Generation

The software industry's goal is to develop the error-free software that fulfills the needs of system stakeholders. The longer the gap between error introduction and error discovery, the higher the error cost. Testing is an essential phase during the development cycle of software, and generating test cases is an essential activity during the testing phase. Software Requirements are often stated in Natural Language (NL), and in NL requirements, it is hard to detect the ambiguity and incompleteness. The analyst manually extraction the features from the NL requirement documents. During the requirements phase, testing activity participates in the software requirement verification process and helps identify ambiguity, inconsistency, and missing/incomplete requirements. The analyst uses NL for communication because it is easy to understand, and no specialized training is required to understand NL. It is universal, and everyone can explain and apply NL to any problem domain. Fig. 9 is a summary of requirements issues that maybe affect the TCG process. Since during systems testing, 56% of errors are followed back to mistakes in the requirements. Ineffectively sorted out requirements, regularly in NL, are among the significant reasons for software projects' disappointments. Ambiguity is one of the significant issues in NL requirements. Any statement is considered ambiguous if it has multiple meanings. In NL requirements, it is hard to detect the ambiguity and incompleteness. How can unambiguous, consistent, and complete Natural Language Requirements improve automated test case generation?

4 Challenges in Requirements Based-Testing

We identified the following challenges in test case generation from requirements.

4.1 Traceability

The first identified challenge in the requirements-based testing approaches is lack of traceability. The traceability from requirements to executable test cases is one of the keys factors of success in any software project [15]. Traceability is mandatory in many engineering standards, such as (e.g., IEEE-Standard 830-1998). Traceability is the linkage with the source; a document is considered as traceable if it is clear and linked with reference. Traceability has two types: (1) backward traceability, connection with the previous stage of development, and (2) forward traceability connection with all subsequent steps from existing state [17].

Dealing with a Traceability Matrix from requirements to test cases is the actual test in keen card programming approval. It gives numerous advantages in the overall programming lifecycle: Verifies that all requirements were applied and tried; Smooths change sway examination, by perceiving application components influenced by a prerequisite change; Supports in the venture the executives by following the test inclusion of every necessity; Helps to recognize the related requirements when a test falls flat [18].

In the literature, many references focus and address the traceability issue in software development and specifically in requirements and testing level (extra due space required cannot define all); for example, state-of-the-art Model-based Testing is addressed the tractability but with missing support of non-functional requirements testing. Most of the Model-based testing approaches focus on functional testing as the detail is mention in Section 3.5 coverage criteria, selection algorithms, and the like.

4.2 Requirements and Testing Alignment

Alignment of requirements and tests is an essential factor that may help during the development process coordination between organizational units is very important to develop a good quality software product on time and within budget.

Sabaliauskaite, et al. [19] conducted an interview study with the research question of what challenges in aligning requirements and testing are. As a result of the study, the researcher categorized the challenges as under organization and processes, people, tools, requirements process, change management, traceability, and measurement [20].

4.3 Change Management

Change inevitably happens, so there must be a plan for changes to requirements throughout a development project and after the product is released to the field. Thus, we must establish processes and tools to manage changing requirements on how to evaluate that the functional and non-functional requirements are effectively comprehended in software implementation. The main challenging to handle the change of how to map each source's requirements onto the individual test cases. In an industrial context, the suitability of mapping a requirement is usually not formally established. This gap is that most of the time is filled with test review and inspection. Therefore, there is a need to develop a mechanism that can validate each requirement successfully implemented.

4.4 Natural Language Requirements Issues

Another critical issue recognized that natural language requirements such as ambiguity, inconsistency, and incompleteness of natural language statements significantly impact software testing quality. Furthermore, 56% of errors are traced back to errors in requirements [21]. Inadequately sorted out requirements, regularly in everyday language, are among the significant reasons for programming ventures' disappointments. Ambiguity is one of the most crucial issues in natural language requirements. Any statement is considered as Ambiguous if it has multiple meanings.

5 Threat to Validity

A systematic literature review is a method of Evidence-Based Software Engineering (EBSE). Validity is the primary concern in EBSE studies while filtering the studies. In this study, we take care of threats to construct, internal, and external validities [22].

Construct validity refers to this systematic literature review to fulfill its primary purpose or not. To address the construct validity, we first define review phases, searching strategy as the primary concerns. The search keywords used in this study were extracted from two research questions. We tried our best to match these keywords against studies from digital libraries mentioned in Section 2.6. Though, and we cannot guarantee the completeness and the comprehensiveness of the searching strategy. We looked through each chosen essential examination's references rundown to distinguish extra important essential investigations to decrease this risk. Furthermore, the pursuit uncovered three articles in the Chinese language [23–25], which we excluded.

Internal validity refers to how system literature review primary studies are extracted from 410 studies found from digital libraries. It focuses on errors in the design and conduct of the study [12]. To address internal validity, we emphasized data extraction.

We used the data extraction form and quality assessment criteria proposed by Kitchenham et al. [11]. Though, in a few studies, some necessary information used to summarize primary studies in Appendix A Tab. 2 was missing. It may be considered as a threat to internal validity.

External validity refers to how we might be wrong to generalize the study outcomes. [12]. External validity is all about the generalizability of the study results. The following issues constrain our systematic review: (1) it is focused on a very specific problem; (2) the problem under focus is relatively new, and (3) it covers a predefined period (2001–2018).

6 Conclusion

This study has presented a review study on Requirements based test case generation from 2000 to 2018. The study carried out to acquire knowledge areas requirements-based test case generation and identify future research. The emphasis was on finding answers for identified research questions (Section 2.4). For this propose, we identified 30 studies based on quality assessment criteria (Section 2.8). Our results show that it is possible to generate test cases from requirements, based on testing approaches identified in primary studies (RQ1). For the formalism of requirements, most approaches use UML-based models such as use case, activity, and class diagrams to generate test cases. For the transformation from natural language requirements to formal model following approaches are applied, such as Petri net, scenario, model checking, bounded model checking, and state transition diagram. These models used different knowledge representation Z notation, finite state machines, behavior trees, metamodeling, and UML state machines. Likewise, for coverage criteria in the path, multiple-condition, edge and flow weight, flow weight, event, and requirement coverage.

As for RQ2, most studies discuss challenges (Section 4) associated with requirements-based test generation. These are trackability, alignment of requirements and testing, requirement coverage, and natural language requirements issues, i.e., ambiguity, incompleteness, and inconsistency.

In the future, we will develop a model of test case generation from requirements using Natural Processing Language. This model will have capabilities to fit in the context of the DevOps software development process. It takes user stories' input in the form of natural language requirements, classifies them, refines them, automatically translates natural language requirements to a logical format to be validated, and finally generates test cases from these requirements. Furthermore, this dynamic model will help the practitioner in successful software project development while maximizing the quality and minimizing the project's cost and delivery in a short time. This model will also prove a new paradigm for the research community to focus on natural language requirements rather than the unified modeling.

Funding Statement: This work was supported by Tecnologico de Monterrey, Mexico.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] IEEE, "ISO/IEC/IEEE international standard—systems and software engineering," ISO/IEC/IEEE 29148:2011(E), pp. 1–94, 2011.

[2] P. C. Jorgensen, *Software Testing: A Craftsman's Approach*, 4th ed. London: CRC Press, 2016.

[3] IEEE Standards Committee, "IEEE standard for software and system test documentation," IEEE Std 829-2008 (Revision of IEEE Std 829-1998)—Redline, pp. 1–161, 2008.

[4] H. Shah, M. J. Harrold and S. Sinha, "Global software testing under deadline pressure: Vendor-side experiences," *Information and Software Technology*, vol. 56, no. 1, pp. 6–19, 2014.

[5] M. Felderer and R. Ramler, "Integrating risk-based testing in industrial test processes," *Software Quality Journal*, vol. 22, no. 3, pp. 543–575, 2014.

[6] M. Nazir and R. A. Khan, "Testability estimation model (TEM OOD)," in *Proc. of Int. Conf. on Computer Science and Information Technology*, Berlin, Heidelberg: Springer, pp. 178–187, 2012.

[7] D. E. Semos, J. Bozic, B. Garn, M. Leithner, F. Duan *et al.,* "Testing TLS using planning-based combinatorial methods and execution framework," *Software Quality Journal*, vol. 27, no. 2, pp. 703–729, 2019.

[8] B. Aysolmaz, H. Leopold, H. A. Reijers and O. Demirörs, "A semi-automated approach for generating natural language requirements documents based on business process models," *Information and Software Technology*, vol. 93, pp. 14–29, 2018.

[9] O. Olajubu, S. Ajit, M. Johnson, S. Turner, S. Thomson *et al.,* "Automated test case generation from domain specific models of high-level requirements," in *Proc. of the 2015 Conf. on research in adaptive and convergent systems, ACM*, Prague Czech Republic, pp. 505–508, 2015.

[10] R. Gao, J. S. Eo, W. E. Wong, X. Gao and S.-Y. Lee, "An empirical study of requirements-based test generation on an automobile control system," in *Proc. of the 29th Annual ACM Symp. on Applied Computing—SAC*, New York, USA: ACM Press, pp. 1094–1099, 2014.

[11] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," Technical report, EBSE, 2007.

[12] B. Kitchenham and P. Brereton, "A systematic review of systematic review process research in software engineering," *Information and Software Technology*, vol. 55, no. 12, pp. 2049–2075, 2013.

[13] M. Utting and B. Legeard, *Practical model-based testing: A tools approach*. Morgan Kaufmann, Elsevier, 2010.

[14] I. Ober and I. Ober, "SDL 2011: Integrating system and software modeling," in *Proc. 15th Int. SDL Forum Toulouse*, France: Springer, 2011.

[15] M. A. Almeida, J. de Melo Bezerra and C. M. Hirata, "Automatic generation of test cases for critical systems based on MC/DC criteria," in *Proc. of 2013 IEEE/AIAA 32nd Digital Avionics Systems Conf.*, IEEE, East Syracuse, NY, USA, pp. 7C5-1–7C5-10, 2013.

[16] B. Sébastien, M. Delahaye, R. David, N. Kosmatov, M. Papadakis *et al.,* "Sound and quasi-complete detection of infeasible test requirements," in *2015 IEEE 8th Int. Conf. on Software Testing, Verification and Validation*, IEEE, Graz, Austria, pp. 1–10, 2015.

[17] B. Vogel-Heuser, A. Fay, I. Schaefer and M. Tichy, "Evolution of software in automated production systems: Challenges and research directions," *Journal of Systems and Software*, vol. 110, pp. 54–84, 2015.

[18] F. Bouquet, E. Jaffuel, B. Legeard, F. Peureux and M. Utting, "Requirements traceability in automated test generation," *ACM SIGSOFT Software Engineering Notes*, vol. 30, no. 4, pp. 1, 2005.

[19] G. Sabaliauskaite, A. Loconsole, E. Engström, M. Unterkalmsteiner, B. Regnell *et al.,* "Challenges in aligning requirements engineering and verification in a large-scale industrial context," in *LNCS Department of Computer Science, Lund University*, vol. 6182. Sweden: Springer, pp. 128–142, 2010.

[20] J. Kukkanen, K. Väkeväinen, M. Kauppinen and E. Uusitalo, "Applying a systematic approach to link requirements and testing: A case study," in *Proc. 16th Asia-Pacific Software Engineering Conf.*, IEEE, Penang, Malaysia, pp. 482–488, 2009.

[21] K. Ellis, *Business analysis benchmark: The impact of business requirements on the success of technology projects*. New Castle: IAG Consulting, 2008.

[22] C. Wohlin, P. Runeson, M. Höst, M. Ohlsson, B. Regnell *et al.,* "Experimentation in software engineering: An introduction—kluwer academic publishers," *Doedrecht the Netherlands*, 2000.

[23] Z. Xiaoyan, H. Ning and Y. Ying, “OWL-S based test case generation,” *Journal-Beijing University of Aeronautics and Astronautics*, vol. 34, no. 3, pp. 327, 2008.

[24] Y. Yu, N. Huang and Q. Luo, “OWL-S based interaction testing of web service-based system,” in *Proc. 3rd Int. Conf. on Next Generation Web Services Practices*, IEEE, Seoul, South Korea, pp. 31–34, 2007.

[25] Y. Ying, J. Maozhong and H. Ning, “Testing control flow of composite service,” *Journal of Beijing University of Aeronautics and Astronautics*, vol. 35, pp. 117–121, 2009.