Artificial Intelligence Evolution
http://ojs.wiserpub.com/index.php/AIE/

UNIVERSAL WISER PUBLISHER

Review Article

Software Test Case Generation Using Natural Language Processing (NLP): A Systematic Literature Review

Halima Ayenew[1], **Mekonnen Wagaw**[2*]

[1]Software Engineering, Wolo university, Ethiopia
[2]Software Engineering, Bahir Dar Institute of Technology, Bahir Dar University, Ethiopia
Email: monalitha@gmail.com

Received: 13 June 2023; **Revised:** 8 January 2024; **Accepted:** 9 January 2024

Abstract: Technologies for natural language processing (NLP) are employed to assist in the analysis and comprehension of human language. Researchers are increasingly focusing on NLP techniques to automate various software development tasks, such as software testing (test case generation). However, choosing the best NLP methods to create automated test cases is never simple. As a result, we look into using NLP techniques to create test cases. We identified 13 research articles published between 2015 and 2023 for this study. As a result, to generate automated test cases, 7 NLP techniques, 2 tools, and 1 framework have been suggested. In addition, 7 NLP algorithms have been discovered in the context of test case generation. Our evaluations indicate that the identified NLP techniques are very useful for automating the generation of test cases. The successful completion of software testing processes (test case generation) therefore requires the use of this approach/technique by software developers, testers, and software engineering teams in general. This paper will be beneficial for researchers engaged in the automation of software testing. Furthermore, it will also be helpful for academic researchers and software engineers (testers) seeking insights into the state of the art in test case generation automation. The paper discusses various tools and methods proposed for test case generation automation, aiding readers in evaluating and selecting the most suitable method for automated test case generation.

Keywords: NLP, systematic literature review (SLR), test cases, test case generation, software testing, software engineering

1. Introduction

For ensuring the quality of a software system, software testing is a vital, well-liked, but pricey activity [1]. Despite recent advancements in test automation, manual testing of software is still a common practice across the industry. In a scenario involving manual testing, the development team and the Quality Assurance (QA) engineers (testers) must invest even more time and effort in testing and test case design, raising the cost of testing for the business. Techniques for Natural Language Processing (NLP) offer hope for solving these issues. Due to the advanced and automated data processing capabilities that NLP techniques provide, they are now frequently used in software development to automatically generate requirements, design artifacts, and test cases from preliminary data [2-3].

Although NLP is frequently used to automate software development tasks, finding appropriate techniques and tools for developing automated test cases continues to be difficult. The difficulty of developing test cases is the primary

DOI: https://doi.org/10.37256/aie.5120243220

cause. Because the goal is to produce an output that can be put to use rather than simply classifying or identifying text or documents in data. To our knowledge, there hasn't been a recent review that summarizes the research on text case generation using NLP. This article investigates NLP techniques to generate automated test cases.

As a research methodology, a Systematic Literature Review (SLR) has been employed. To conduct SLR, the following research questions (RQs) have been developed:

RQ1: What role has been played by NLP in test case generation? This research question can help to illustrate the role and state of the art of NLP for the generation of test cases.

RQ2: How major NLP steps have been applied for test case generation? This research question can help to explore the steps in NLP test case generation processes.

RQ3: What are the techniques/tools proposed by the researchers, to generate test cases? This question aims at identifying the commonly used NLP test case generation mechanisms.

To provide answers to these questions, we tracked down all the work that had been done on test case generation. Our review is organized in the manner described above.

Section 2.1 describes the inclusion/exclusion criteria, Sections 2.2 and 2.3 describe the search process, Section 2.4 describes data extraction and synthesis, and Section 2.4 describes descriptions of chosen studies. It is described in 2.4. Section 4 will respond to Section 3.1 and the research queries. In Section 5, the goals and benefits of the study are covered, and the paper is concluded in Section 6.

2. Research methodology

A systematic literature review (SLR) was used to conduct this study [4]. Investigating and reviewing every existing research study that is relevant to the research questions is an appropriate and proper process for capturing and addressing the pertinent information precisely on the specific research area. The four main phases of this research study are as follows: 1) Inclusion and Exclusion criteria; 2) Search process; 3) Quality Assessment Criteria; and 4) Data Extraction and Synthesis.

2.1 *Inclusion and exclusion criteria*

We defined 6 parameters for the inclusion and exclusion of research work as follows:

1. Studies were all written in English.

2. All included research must be from the years 2015 through 2023. Since only the most recent research is included, all other earlier studies ought to be disregarded.

3. The creation of test cases using Natural Language Processing (NLP) is mentioned in the study's title or abstract.

4. The study should only be used in a journal or conference paper.

5. Just the studies that dealt with the solutions to our research's questions should be included.

6. Include only those studies that are identical in the context of the research and leave out any that repeat in any context.

2.2 *Search process*

This thorough literature search was conducted using the ACM Digital Library, IEEE, Science Direct, Springer, and Google Scholar databases. These databases were picked because, in our opinion, they had the most complete listing of journals and proceedings for the papers that were chosen. To find articles that were not indexed in the aforementioned libraries, additional online digital libraries were also looked through, and other search databases were used. Together, they offer almost total coverage of all significant software engineering journals, conferences, and workshop papers.

We must use several search terms that are in line with our predetermined criteria, such as NLP-based test case generation, Test case generation using NLP, etc., to obtain relevant results and complete the search process. Table 1 provides a summary of the results based on search terms relevant to various databases. To find the published research from the years 2015-2023, we used a different filter called "2015-2023".

• When we enter various search terms in the designated databases, we receive 4,526 search results.

• Following our inclusion and exclusion criteria, we disregard 4,110 research studies based solely on their titles.

• Next, 343 research studies are disregarded based on their abstracts and our inclusion and exclusion criteria.

• Then, using the pertinent sections of the 73 research studies we had selected for the general study, we conducted a detailed analysis of the 73 papers, leaving out 60 research works.

• We include 13 research articles that meet our inclusion and exclusion criteria.

Table 1. Search terms and results

No	Term for search	Results		
		ACM	IEEE	Springer
1	Automation of test case generation using NLP	820	850	911
2	NLP-based test case generation	245	523	268
3	Test case generation using Natural Language Processing	1,200	234	361
4	Software testing using Natural Language Processing	420	156	1,260
5	Automatic test case generation using NLP approaches	360	264	2,010

2.3 *Quality assessment criteria*

To determine the significant outcomes of the research works, we established the standards of quality. Each research study that is included and its results are also defined by the criteria:

• The study's goals are stated in clear terms.

• The study outlines a primary research issue.

• The study adds to prior research or suggests a fresh approach.

• The study's methodology is spelled out.

• The results of the study were evaluated and discussed.

• The study uses appropriate evaluation techniques.

• Because it was our goal to incorporate the most recent research studies. To find studies that provide the most recent results, we used the filter of 2015 to 2023. Figure 1 displays the yearly distribution of the chosen papers.

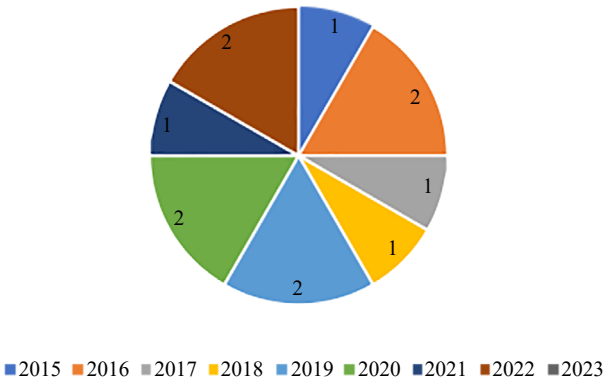

Figure 1. Year-wise distribution of selected studies

Here the following histogram in Figure 2 shows we have selected different research works from different publishers and databases. In the figure below, 4, 5, 3, and 1 papers selected from IEEE, ACM, Springer, and others, respectively.

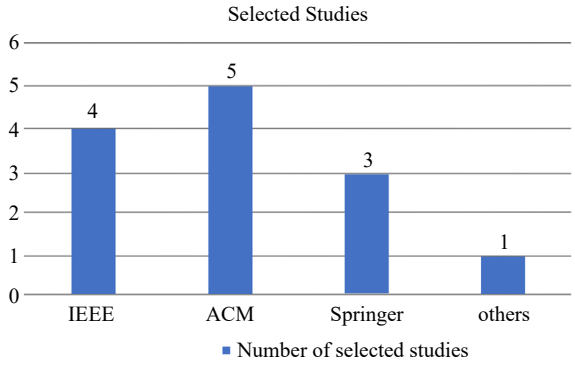

Figure 2. Name of publishers of selected studies

2.4 *Data extraction and synthesis*

A data extraction form that lists the information that was taken from candidate papers is included in the study. To find the solutions to our research questions, which are listed in Table 2 and Table 3, we identify specific data extraction components.

Table 2. Data extraction of selected research

No	Element	Description
1	Title	Title of the selected study
2	Authors	Authors of the selected study
3	Years	The year the selected study was published
4	Source	The collection in which the study was published
5	Publication Type	Type of publication, e.g., journal or conference paper
6	Keywords	Keywords of the study
7	Abstract	Abstract at the beginning of the study
8	NLP techniques	Test case generation techniques used in the study

Table 3. Data synthesis of selected research

No	Element	Description
1	Overview of Research	A brief overview of the research including the introduction of the proposed techniques, their validation method, and results, etc. (Section 3.1)
2	NLP Techniques	Basic NLP techniques used to get output (Table 4)
3	Tools and Techniques	Tools and techniques proposed in the study (NLP process) (Table 5)
4	Role of NLP for test case generation	The role of NLP in test case generation automation is briefly stated (Section 4-RQ1)

3. Results

3.1 *Overview of the selected research works*

Blasi et al. compiled a list of Javadoc methods that use natural language statements to convey the anticipated order of operations. [5] described a method for automatically locating time restrictions in Java classes. CallMeMaybe analyzes each sentence of the summary to determine time constraints. Then, it creates a temporary spec, a straightforward JSON structure that outlines the desired order of operations. Additional tools, like automated test case generators, can use the CallMeMaybe JSON structure because it is serializable. The translations made by CallMeMaybe are precise and generalize well to various document styles without relying on patterns or requiring training data. We discovered that using the authors' method on 73 subjects in 7 well-known Java systems, CallMeMaybe achieved 83 percent accuracy and 70 percent recall. The Randoop integration enriched 11,818 false positives flagged and 12,024 test cases that correctly failed due to time constraint violations for the two largest themes.

A semi-automated method for a machine-aided requirements formalization technique was put forth by Gröpler et al. [6]. The proposed approach aims to reduce the human effort required to generate test cases from textual requirements to validate the generated requirement models. It automates the process of model creation from requirements in natural language by using appropriate algorithms. They evaluated their strategy in the industrial use case of a battery charging approval system to show the results after first defining the used evaluation metrics. The outcome demonstrated that the algorithm is capable of producing complete, accurate, and consistent artifacts to a significant extent.

A scatter search approach was put forth by Liu et al. [7] to automatically generate test cases for NLP programs that cover all potential paths. In a limited number of test cases, this technique enables search-based algorithms to explore all input variables and cover the paths that demand particular input variables. The proposed scatter search strategy allows the compared state-of-the-art algorithms to cover all potential paths, according to three experiments conducted by the authors. The results of the experiment demonstrate that the proposed scatter search strategy can quickly cover the paths that call for particular input variables, and when search-based algorithms are combined with the scatter search strategy, many test cases and running times are saved.

Chunhui Wang and other people. Use Case Modeling for System-level, Acceptance Tests Generation (UMTG), an automated acceptance testing strategy with an emphasis on embedded systems, was introduced in [1]. To ensure that the requirements are being met, create executable test cases and system-level test data. Commonly known as acceptance testing, this activity. Use case specifications and domain models are both relied upon by the authors. Their goal was automated test case generation, and they mainly relied on accepted standards for describing requirements for sharing information among stakeholders in the embedded systems industry. They value the work he has done on the two embedded industrial systems. UMTG effectively creates acceptance test cases for automotive sensor systems, according to industry case studies. For test case generation, 96% of Object Constraint Language (OCL) constraints can be generated automatically and accurately by UMTG. The OCL generation process is extremely accurate 99 percent of the constraints that were generated are accurate.

A tool-supported methodology called CiRA (Conditionals in Requirements Artifacts) is presented by Fischbach et al. [8] and is capable of generating the bare minimum number of test cases from conditional statements in informal requirements. They carry out a case study with three businesses to assess CiRA. The research shows that of the 578 manually created test cases, 71.8% can be generated automatically by CiRA.

An approach to automatically convert user requirements to test cases using model-driven engineering (MDE) and natural language processing (NLP) is presented by Allala et al. in their article [2]. This method takes an XMI (XML Metadata Interchange) source model for the user requirement as input, validates it against a metamodel, processes the text in the various user requirement components, and then transforms the source model into a target model (test case) using the information gathered during NLP processing. The user requirements for the experiments came from the projects of a Florida International University (FIU) undergraduate software engineering class and Ultimate Software. They validated their work using the experimental validation method.

A system that builds test cases based on keywords in context from the functional requirement of the Software Requirement Specification document was proposed by Ansari et al. [3]. To extract test cases for testing, the proposed system automatically analyzes the functional requirements from software requirement specifications. The author wants to cut down on the time and effort software testers spend testing a product.

A framework was suggested by Viggiato et al. [9] for automatically analyzing test cases that are written in natural language and offering useful suggestions for how to improve testing cases. Their framework is made up of reconfigurable elements and modules for analysis that can suggest changes to the terminology of a new test case through language modeling, test steps that might be missing from a new test case through frequent item and association rule mining, and similar test cases that are already present in the test suite through text embedding and clustering. With test cases created to evaluate the Prodigy Math game, the three modules were thoroughly assessed using data from the industry. Their evaluation findings demonstrate that they can use statistical and neural language models to recommend terminology improvements with high accuracy (up to 88 percent). Their association rules can also, on average, 98 percent of the time per test case, correctly recommend missing test steps. Finally, they discovered comparable test cases with excellent performance (an F-score of roughly 83 percent).

An approach with numerous novel techniques was put forth by Li et al. [10] to group similar NL test steps. The method can accurately cluster the test steps and decrease the number of clusters, which greatly reduces the manual work required later. They tested the approach's efficacy using test cases from WeChat, a sizable business app, and integrated its implementation into the app's testing system. Additionally, compared to the baseline approach, their method improves cluster quality by 79.8% while cutting the number of clusters by 65.9%.

By extending Restricted Use Case Modeling (RUCM), Yue et al. [11] proposed a Tata Consultancy Service (TCS) language Restricted Test Case Modeling (RTCM) was developed. Additionally, they suggested his aToucan4Test tool. The tool accepts the RTCM's TCS as input and generates either manual test cases or automatically executable test cases based on various coverage criteria defined in the RTCM. To validate the RTCM, they inspected 30 auto-generated TCSs and manually modeled two industry case studies. AToucan4Test was also used to create automatically executable test cases after modeling the three components of a video conferencing system developed by Cisco Systems in Norway. Two different versions of commercial software were used to successfully run these test cases.

Yakusu is a method that combines program analysis and natural language processing to extract from a bug report a test case that replicates the problem detailed in the report, according to Fazzini et al. [12], Yakusu was implemented, and its empirical evaluation was done by subjecting it to a set of 62 actual bug reports. For 59.7% of the reports, Yakusu was able to automatically generate tests.

A method for automatically generating test oracles for anomalous behavior from Javadoc comments was proposed by Goff et al. [13]. This method combines runtime instrumentation and natural language processing. The test input generation tool can be used in conjunction with its implementation, Toradocu. They conduct experiments to validate their method. According to their experimental analysis, Toradocu reduced false positives in EvoSuite by 33% and improved debugging efficiency in Randoop and EvoSuite test suites by 8% and 16%, respectively.

Jiansong Zhang and associates. A brand-new, unified ACC system was suggested by [14]. There are restrictions on the automated compliance checking (ACC) systems currently in use. The extraction of regulatory information from regulatory text documents and its encoding into rule format requires a significant amount of manual work when relying on proprietary, hard-coded rules to express regulatory requirements. increase. The updated ACC system integrates EXPRESS technology, (1) semantic logic-based representation of information, inference can be fully automated, and (2) semantic natural language processing techniques and databases that automatically extract and transform both regulatory information (in regulatory documents) and design information [in building information models (BIMs)] for automatic compliance justification. BIM test cases were examined for compliance with Chapter 19 of the 2009 International Building Code to validate the system. In comparison to the manually created gold standard, it achieved a recall rate of 98.7% and a nonconformance detection accuracy of 87.6%.

Table 4. List of selected researches

No	Authors	Title	Publication year	Publication type
1	Arianna Blasi et al.	Call me maybe: Using NLP to automatically generate unit test cases respecting temporal constraints	2022	Journal
2	Ahlam Ansari et al.	Constructing test cases using natural language processing	2017	Conference
3	Robin Gröpler et al.	NLP-based requirements formalization for automatic test case generation	2021	Journal
4	Fangqing Liu et al.	Search-based algorithm with scatter search strategy for automated test case generation of NLP toolkit	2019	Journal
5	Chunhui Wang et al.	Automatic generation of acceptance test cases from use case specifications: An NLP-Based approach	2022	Journal
6	Jannik Fischbach et al.	Automatic creation of acceptance tests by extracting conditionals from requirements: NLP approach and case study	2023	Journal
7	Sai Chaithra et al.	Towards transforming user requirements to test cases using MDE and NLP	2019	Conference
8	Markos Viggiato et al.	Using natural language processing techniques to improve manual test case descriptions	2022	Journal
9	Linyi Li et al.	Clustering test steps in natural language toward automating test automation	2020	Conference
10	Tao Yue et al.	RTCM: A natural language-based, automated, and practical test case generation framework	2015	Journal
11	Mattia Fazzini et al.	Automatically translating bug reports into test cases for mobile apps	2018	Journal
12	Alberto Goff et al.	Automatic generation of oracles for exceptional behaviors	2016	Journal
13	Jiansong Zhang et al.	Integrating semantic NLP and logic reasoning into a unified system for fully automated code-checking	2017	Journal

4. Answer to research questions

RQ1: What role has been played by NLP in test case generation?

The technology called Natural Language Processing (NLP) is used to decipher and comprehend human language. To ascertain the meaning of symbolic strings of natural language following the laws of formal grammar, this includes lexical and syntactic analysis. The authors are Marcos Vijat and others. This article [9] uses NLP technology to stem words and turn sentences into word lists. The group of Goff et al. [13] identified distinct sentences and produced a semantic map for each one using natural language processing (Stanford Parser).

A parse tree and additional data, like grammatical relations, make up the semantic graph. NLP was used by Anasri et al. [3] to process, analyze, and extract the necessary data for the creation of test cases from the functional requirement document. The CallMeMaybe technique was developed by Blasi et al. [5] and includes a CallMeMaybe Constraint Finder that analyzes the sentences it receives from the CallMeMaybe Summary Extractor to find propositions that describe temporal constraints. Using the Stanford Parser, it creates a semantic graph for the English text, navigates the graph to find propositions pertinent to temporal dependencies, and then takes advantage of temporal dependencies to create temporal proposition series. To identify pertinent syntactic entities that can then be mapped to semantic entities, Gröpler et al. [6] used an NLP parser to gather basic syntactic information about the words and their relationships to one another. Wang et al. [10] Use case specifications expressed with RUCM were used to generate test models (UCTMs), and use case specifications were used to generate OCL constraints for test input data generation. In this study by Zhang et al. [14], NLP techniques are used for preprocessing to get the building codes' raw natural language text ready for further processing. Dehyphenation, tokenization, sentence-splitting, morphological analysis, Part-of-speech (POS) tagging, phrase structure analysis, and gazetteer list analysis are some of the syntactic and semantic features of the text that are generated and described by the feature generator.

RQ2: How major NLP steps have been applied for test case generation?

Depending on the needs for test case generation, the authors of the papers we have reviewed applied NLP steps either collectively or individually. The three NLP steps of tokenization, POS tagging, and parsing are typically used. Speech component (POS) Parsing is a method for determining the semantic relationships between words in a sentence, and tagging assigns the parts of speech to each word in a text (for example, noun, verb, pronoun, and adjective). Based on a predetermined set of rules (such as the identification of whitespace and punctuation), tokenization divides a sentence into tokens. [1, 10].

Table 5. Identification of NLP techniques

NO	NLP steps	List of researches	Total
1	Parsing alone	[3, 7, 13]	3
2	Tokenizing + POS tagging + Parsing	[1, 5, 6, 14]	4
3	Tokenizing alone	[8-11]	4
4	Tokenizing + POS tagging	[2, 12]	2

RQ3: What are the techniques/tools proposed by the researchers, to generate test cases?

Due to the severity of the issue, the researchers proposed various test case generation mechanisms based on their analyses of the chosen studies. In the Table 6 below, we listed the suggested methods, instruments, validation strategies, and algorithms applied by the researchers during their investigation.

Table 6. Proposed tools and techniques

No	Study	Techniques/tools proposed	Validation method	Algorithm used
1	Arianna Blasi et al.	CallMeMaybe technique	Evaluated on a benchmark of 73 classes randomly selected from seven popular Java systems	Test case generators algorithm
2	Ahlam Ansari et al.	N/A	N/A	Test case generators algorithm
3	Robin Gröpler et al.	Ifak's prototypical tool ModGen	Evaluated on industrial (e-mobility domain) use case of a battery charging approval system	Rule-based algorithm
4	Fangqing Liu et al.	Scatter search strategy	Experimental validation	Search-based algorithm with scatter search strategy (SA-SS)
5	Chunhui Wang et al.	UMTG approaches	Case study-twindustrial embedded systems	Building path condition algorithm
6	Jannik Fischbach et al.	Tool-supported approach CiRA (Conditionals in Requirements Artifacts)	Case study-with three company	N/A
7	Sai Chaithra et al.	N/A	Experimental validation	N/A
8	Markos Viggiato et al.	Framework	Evaluated on the data from the industry with the test cases designed to test the Prodigy Math game	N/A
9	Linyi Li et al.	N/A	Evaluated their approach on a large industrial mobile app, WeChat	N/A
10	Tao Yue et al.	A test cases generation tool (aToucan4Test)	Case study (manually modeling two industrial systems)	Test case generator algorithm
11	Mattia Fazzini et al.	Yakusu-test case generation technique	Experimental validation	Bug report analysis algorithm
12	Alberto Goff et al.	Toradocu-test case generation technique	Experimental validation	Condition Translator algorithm
13	Jiansong Zhang et al.	SNACC system	Validation by a BIM test case with Chapter 19 of the IBC 2009	Information Extraction and transformation algorithms and logic-based automated reasoning algorithm

5. Benefits and aims

This Systematic Literature Review (SLR) is significant because it sheds light on the application of NLP to test case generation. The variety of NLP techniques used for test case generation, including tokenization, POS tagging, and parsing, is another important aspect of this review. The outcome suggests that these techniques can offer many advantages, including the elimination of classification errors, particularly when dealing with large and complex data, the reduction of testing costs, and the assurance that testing is systematic and adequately addresses all requirements.

This review aims to look into the most recent studies where automated test cases have been created using NLP techniques.

6. Conclusion

To create automated test cases, this paper investigates Natural Language Processing (NLP) techniques. To complete this study, a Systematic Literature Review (SLR) was conducted. Based on the review protocol, 13 research articles published between 2015 and 2023 have been chosen. To generate automated test cases, 7 NLP techniques, 2 tools, and 1 framework have been suggested, and 7 NLP algorithms have been discovered in the context of test case generation. Even though this study provides a thorough overview of NLP methods for creating automated test cases.

In general, software engineering teams should adopt NLP-based test case generation tools and methods because they constitute a valuable approach that enhances the overall effectiveness, reliability, and efficiency of the software testing process. This plays a crucial role in delivering high-quality software products.

7. Limitation and future work

In this systematic literature review, the main limitation was the challenge of acquiring a large number of publications on the selected topic. Although this study provides a comprehensive overview of NLP techniques and tools for automated test case generation, there is still a need to conduct a comparative analysis of the identified tools and algorithms to highlight their respective strengths and weaknesses. Consequently, in the forthcoming article, we intend to carry out a comparative assessment of the identified tools and methodologies.

Conflict of interest

The authors declare no competing financial interest.

Reference

[1] Wang C, Pastore F, Goknil A, Briand LC. Automatic generation of acceptance test cases from use case specifications: An NLP-based approach. *IEEE Transactions on Software Engineering*. 2022; 48(2): 585-616.

[2] Allala SC, Sotomayor JP, Santiago D, King TM, Clarke PJ. Towards transforming user requirements to test cases using MDE and NLP. *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)*. Milwaukee, WI, USA: IEEE; 2019. p.350-355.

[3] Ansari A, Shagufta MB, Sadaf Fatima A, Tehreem S. Constructing test cases using natural language processing. *2017 Third International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB)*. Chennai, India: IEEE; 2017. p.95-99.

[4] Kitchenham B. Procedures for performing systematic reviews. *Keele University Technical Report*. 2004; 33(2004): 1-26.

[5] Blasi A, Gorla A, Ernst MD, Pezzè M. Call me maybe: Using NLP to automatically generate unit test cases respecting temporal constraints. *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*. Rochester, MI, USA: Association for Computing Machinery; 2022. p.1-11.

[6] Gröpler R, Sudhi V, García EJC, Bergmann A. NLP-based requirements formalization for automatic test case generation. *Proceedings of the 29th International Workshop on Concurrency, Specification and Programming (CS&P 2021)*. Berlin, Germany: CEUR-WSCEUR-WS; 2021. p.18-30.

[7] Liu F, Huang H, Yang Z, Hao Z, Wang J. Search-based algorithm with scatter search strategy for automated test case generation of NLP toolkit. *IEEE Transactions on Emerging Topics in Computational Intelligence*. 2019; 5(3): 491-503.

[8] Fischbach J, Frattini J, Vogelsang A, Mendez D, Unterkalmsteiner M, Wehrle A, et al. Automatic creation of acceptance tests by extracting conditionals from requirements: NLP approach and case study. *Journal of Systems and Software*. 2023; 197: 111549.

[9] Viggiato M, Paas D, Buzon C, Bezemer CP. Using natural language processing techniques to improve manual test case descriptions. *2022 IEEE/ACM 44th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP)*. Pittsburgh, PA, USA: IEEE; 2022. p.311-320.

[10] Li LY, Li ZW, Zhang WJ, Zhou J, Wang PC, Wu J, et al. Clustering test steps in natural language toward automating test automation. *Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. New York, NY, United States: Association for Computing Machinery; 2020. p.1285-1295.

[11] Yue T, Ali S, Zhang M. RTCM: A natural language based, automated, and practical test case generation framework. *Proceedings of the 2015 International Symposium on Software Testing and Analysis*. New York, NY, United States: Association for Computing Machinery; 2015. p.397-408.

[12] Fazzini M, Prammer M, D'Amorim M, Orso A. Automatically translating bug reports into test cases for mobile apps. *Proceedings of the 27th ACM SIGSOFT International Symposium on Software Testing and Analysis*. New York, NY, United States: Association for Computing Machinery; 2018. p.141-152.

[13] Goffi A, Gorla A, Ernst MD, Pezzè M. Automatic generation of oracles for exceptional behaviors. *Proceedings of the 25Th International Symposium on Software Testing and Analysis*. New York, NY, United States: Association for Computing Machinery; 2016. p.213-224.

[14] Zhang J, El-Gohary NM. Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking. *Automation in Construction*. 2017; 73: 45-57.