Large Language Models in Software Testing: A Scoping Review

Ruby Marshall
Universidade Federal de Lavras (UFLA)
São Sebastião do Paraíso, Minas Gerais, Brazil
ruby.marshall@estudante.ufla.br

Maurício Magri Ribas das Neves
Universidade Federal de Lavras (UFLA)
São Sebastião do Paraíso, Minas Gerais, Brazil
mauricio.neves@estudante.ufla.br

Bento Rafael Siqueira
Universidade Federal de Lavras (UFLA)
Lavras, Minas Gerais, Brazil
bento.siqueira@ufla.br

Samira Santos da Silva
Universidade Federal de Lavras (UFLA)
São Sebastião do Paraíso, Minas Gerais, Brazil
samirasilva@ufla.br

Abstract

Large Language Models (LLMs) have rapidly emerged as promising tools for supporting Software Engineering activities, particularly Software Testing, by assisting in tasks such as test generation, specification creation, and test data production. Despite the growing number of studies in this area, the current landscape of LLM applications, challenges, and research opportunities remains fragmented. This Scoping Review maps the use of LLMs across different Software Testing activities. A literature search was conducted in the IEEE Xplore and ACM Digital Library databases, covering publications from 2020 to 2025. The search retrieved 1,378 records; after removing 57 duplicates, 1,321 unique studies were screened, resulting in 12 primary studies included in the final synthesis. The findings indicate that LLMs are predominantly applied to the generation of testing artifacts, including unit tests, test inputs, formal specifications, and test oracles. The most frequently reported challenges involve reliability and hallucinations, variability across models and prompts, context and scalability limitations, and detection accuracy. The findings also indicate a trend toward hybrid testing workflows that integrate LLMs with traditional Verification and Validation (V&V) techniques. Overall, the evidence suggests that LLMs have potential to enhance software testing activities when employed as complementary components within reliable and reproducible testing processes.

Keywords

Large Language Models, Software Testing, Scoping Review

1 Introduction

Software testing is a fundamental activity in the software development lifecycle, playing a key role in ensuring the quality, reliability, and dependability of software systems [1, 15]. However, designing test cases, generating input data, defining test oracles, and maintaining testing artifacts remain labor-intensive tasks that require substantial effort and domain expertise. As software systems continue to increase in size and complexity, improving the efficiency and effectiveness of testing activities has become an important challenge for both academia and industry.

Recent advances in Large Language Models (LLMs) have significantly transformed Software Engineering (SE). Owing to their ability to understand and generate both natural language and source code, LLMs have demonstrated considerable potential for supporting software development activities, including code generation, debugging, documentation, program repair, and software testing [4, 7]. Within software testing, these models have been investigated for tasks such as unit test generation, test input generation, test oracle construction, formal specification generation, vulnerability repair, and test maintenance.

The increasing adoption of LLMs has stimulated a growing body of research investigating their application to software testing. Recent secondary studies have provided valuable overviews of this rapidly evolving field, highlighting both the opportunities and the challenges associated with integrating LLMs into software engineering practices [6–8, 18]. Nevertheless, existing reviews either adopt a broad Software Engineering perspective or focus on specific testing techniques, making it difficult to obtain a comprehensive understanding of how LLMs are being employed across different Software Testing activities.

Despite their promising capabilities, LLMs also introduce important challenges. Their probabilistic nature may lead to hallucinations, inconsistent outputs, limited reproducibility, and difficulties in assessing the correctness of generated artifacts, raising concerns regarding their reliability in software quality assurance [9]. These limitations have motivated increasing interest in combining LLMs with traditional Verification and Validation (V&V) techniques to improve the reliability of AI-assisted testing processes.

Motivated by the rapid evolution of this research area and the fragmented nature of the existing literature, this paper presents a Scoping Review on the use of LLMs in Software Testing. The review aims to systematically map current applications of LLMs across Software Testing activities, synthesize the challenges reported in the literature, and identify emerging research opportunities. Accordingly, the Research Questions (RQs) were designed to characterize the current applications (RQ1), reported challenges and limitations (RQ2), and emerging research opportunities (RQ3):

- **RQ1**: What are the main applications and techniques of LLMs across different Software Testing activities?
- **RQ2**: What are the main challenges and limitations associated with the adoption of LLMs for software testing?
- **RQ3**: What research opportunities and future directions have been identified in the literature on LLM-assisted software testing?

The remainder of this paper is organized as follows. Section 2 reviews existing secondary studies and positions the present work within the current literature. Section 3 describes the adopted research methodology. Section 4 presents the findings of the review,

discusses their implications and the threats to the validity of this study. Finally, Section 5 concludes the paper and outlines directions for future work.

2 Related Work

The growing adoption of Large Language Models (LLMs) has motivated several secondary studies investigating their application in Software Engineering (SE) and, more specifically, in Software Testing. These reviews differ in scope, methodology, and research objectives, ranging from broad surveys on LLM-assisted software development to studies focused specifically on testing activities. This section discusses the most relevant contributions and positions the present review with respect to the current literature.

Hou et al. [7] present a comprehensive Systematic Literature Review (SLR) on the use of LLMs in Software Engineering. Their study analyzes applications of LLMs across multiple software engineering activities, including code generation, program comprehension, refactoring, code review, software maintenance, and software testing. Although software testing is included as one of the application domains, it represents only one of several Software Engineering activities investigated and is therefore not analyzed in depth.

Qi et al. [18] provide a survey dedicated to the use of LLMs in Software Testing. The authors discuss techniques such as automated test generation, test repair, and evaluation strategies, providing a broad overview of the field. However, the study does not adopt a systematic review protocol or emphasize reproducibility aspects.

Huang et al. [8] present an extensive survey of LLM applications in Software Testing, categorizing testing tasks supported by LLMs and discussing their challenges and future research directions. Their work provides a broad landscape of the field and serves as an important reference for understanding the range of testing tasks in which LLMs have been investigated.

Boukhlif et al. [3] conduct a comparative study of LLM applications in Software Testing based on 12 primary studies. Their analysis focuses on technical characteristics of LLM-assisted testing approaches, including the LLMs employed, input and output types, fine-tuning and prompt engineering practices, programming languages, testing frameworks, application domains, testing types, and solution availability. While their study provides a detailed characterization of the technical landscape of LLM-based testing, the present review adopts a complementary perspective by systematically synthesizing the applications, reported challenges and limitations, and emerging research opportunities in the field.

Complementarily, Görmez et al. [6] conduct a Systematic Mapping Study on the application of LLMs in Software Engineering. Their work identifies major research topics, application domains, and publication trends. Similar to the review by Hou et al. [7], software testing represents only one of several investigated areas and is therefore not analyzed in depth.

Building upon these studies, the present review provides a complementary perspective on LLM-assisted Software Testing. Specifically, this work:

- focuses exclusively on Software Testing and systematically characterizes the main applications of LLMs across different testing activities (RQ1);
- synthesizes the challenges and limitations reported in the primary studies concerning the adoption of LLMs in Software Testing (RQ2);
- identifies emerging research opportunities and future directions for LLM-assisted Software Testing (RQ3); and
- conducts this analysis through a reproducible Scoping Review protocol, with categories derived inductively from the selected primary studies.

Overall, the present review complements existing secondary studies by providing a focused and up-to-date synthesis of LLM-assisted Software Testing. While previous studies provide broad surveys of testing tasks or characterize technical aspects such as LLMs, prompting strategies, inputs, outputs, and testing types, this review jointly examines the applications, challenges, and research opportunities reported in the literature through a systematic Scoping Review protocol. Table 1 summarizes the main characteristics of the most relevant secondary studies and compares them with the present review.

3 Research Methodology

This study was conducted as a Scoping Review following the methodological framework proposed by Arksey and O'Malley [2], further refined by Peters et al. [17]. The reporting of the review was guided by the PRISMA Extension for Scoping Reviews (PRISMA-ScR) [21]. The review process was designed to ensure transparency and reproducibility by explicitly defining the research questions, search strategy, study selection criteria, and data extraction process. The description of each stage of the review is presented as follows.

3.1 Search Strategy

The literature search was conducted in two major digital libraries widely used in Software Engineering research: IEEE Xplore[1] and the ACM Digital Library[2]. To capture recent advances in LLM-based approaches, the search was restricted to studies published between 2020 and 2025. All searches were performed in November 2025.

The search string was designed by combining terms related to Large Language Models and Software Testing using the Boolean operators `AND` and `OR`. Minor syntactic adaptations were applied to accommodate the requirements of each digital library while preserving the same search semantics. The search string was iteratively refined through pilot searches to ensure that representative studies were retrieved while minimizing irrelevant results.

The final search string was:

```
("Large Language Model" OR LLM OR "Generative AI"
OR ChatGPT OR "Foundation Model") AND ("Software
Testing" OR "Test Generation" OR "Test Case" OR
"Test Automation" OR "Bug Repair" OR "AI-assisted
Testing")
```

The search was applied to the title, abstract, and keywords of each publication.

3.2 Inclusion and Exclusion Criteria

To ensure the selection of relevant primary studies, explicit Inclusion Criteria (IC) and Exclusion Criteria (EC) were defined. A study

[1] https://ieeexplore.ieee.org/
[2] https://dl.acm.org/

Table 1: Comparison between existing secondary studies and the present review.

Characteristic	Hou et al. [7]	Qi et al. [18]	Huang et al. [8]	Boukhlif et al. [3]	Görmez et al. [6]	Present Review
Primary scope	LLMs in SE	LLMs for Testing	LLMs for Testing	LLMs for Testing	LLMs in SE	LLMs in Software Testing
Study type	SLR	Survey	Survey	Comparative Study	Systematic Mapping	Scoping Review
Exclusive focus on Software Testing	No	Yes	Yes	Yes	No	Yes
Systematic review protocol	Yes	No	Partial	No	Yes	Yes
Synthesis of reported challenges	General	Yes	Yes	Partial	General	Yes
Identification of future research opportunities	General	Yes	Yes	Yes	General	Yes
Main analytical focus	SE applications	Testing techniques	Testing tasks	Technical characteristics	SE research topics	Applications, challenges, and opportunities

was included only if it satisfied all inclusion criteria and none of the exclusion criteria. The inclusion and exclusion criteria were applied sequentially during both the screening and eligibility phases of the review. Table 2 summarizes the adopted criteria.

Table 2: Inclusion (IC) and Exclusion (EC) Criteria.

Type	ID	Description
Inclusion	IC1	Primary peer-reviewed study (conference or journal).
Inclusion	IC2	Published between 2020 and 2025 (inclusive).
Inclusion	IC3	Full text available in English, Spanish, or Portuguese.
Inclusion	IC4	Addresses the use of LLMs in Software Testing or in artifacts that directly support testing.
Inclusion	IC5	Proposes, evaluates, or empirically discusses a technique, tool, or framework.
Exclusion	EC1	Secondary studies (e.g., systematic literature reviews, surveys, or systematic mapping studies).
Exclusion	EC2	Studies unrelated to Software Testing or testing-support artifacts (e.g., only code generation).
Exclusion	EC3	Full text not available.
Exclusion	EC4	Studies that do not employ LLMs (e.g., only traditional AI techniques).
Exclusion	EC5	Duplicate records across databases or extended versions of already included studies.

3.3 Study Selection

The study selection process followed four sequential phases adapted from the PRISMA-ScR workflow. The complete workflow is illustrated in Figure 1.

(1) **Identification**: The search strategy retrieved a total of **1,378 records** from IEEE Xplore and the ACM Digital Library.
(2) **Deduplication**: After removing **57 duplicate records**, **1,321 unique studies** remained for screening.
(3) **Screening**: Titles, abstracts, and keywords were evaluated according to the predefined inclusion and exclusion criteria, resulting in **47 candidate studies**.
(4) **Eligibility**: The full texts of the 47 candidate studies were assessed to determine their relevance with respect to the proposed research questions. After this stage, **12 primary studies** were selected for the final qualitative synthesis.

The screening and full-text assessment were independently conducted by two reviewers. Whenever uncertainties or disagreements regarding the eligibility of a study arose, the inclusion and exclusion criteria were revisited and discussed by both reviewers until consensus was reached, ensuring consistency throughout the selection process.

3.4 Data Extraction

For each included study, the following information was extracted: publication year, publication venue, software testing activity, evaluation methodology, reported challenges, and future research directions. The extracted information was organized in a structured spreadsheet and subsequently synthesized qualitatively to answer the proposed Research Questions.

3.5 Data Synthesis

Since the objective of this study is exploratory, a qualitative synthesis was performed. For RQ1–RQ3, categories were derived inductively by comparing the extracted findings across studies and iteratively grouping conceptually similar findings, enabling the

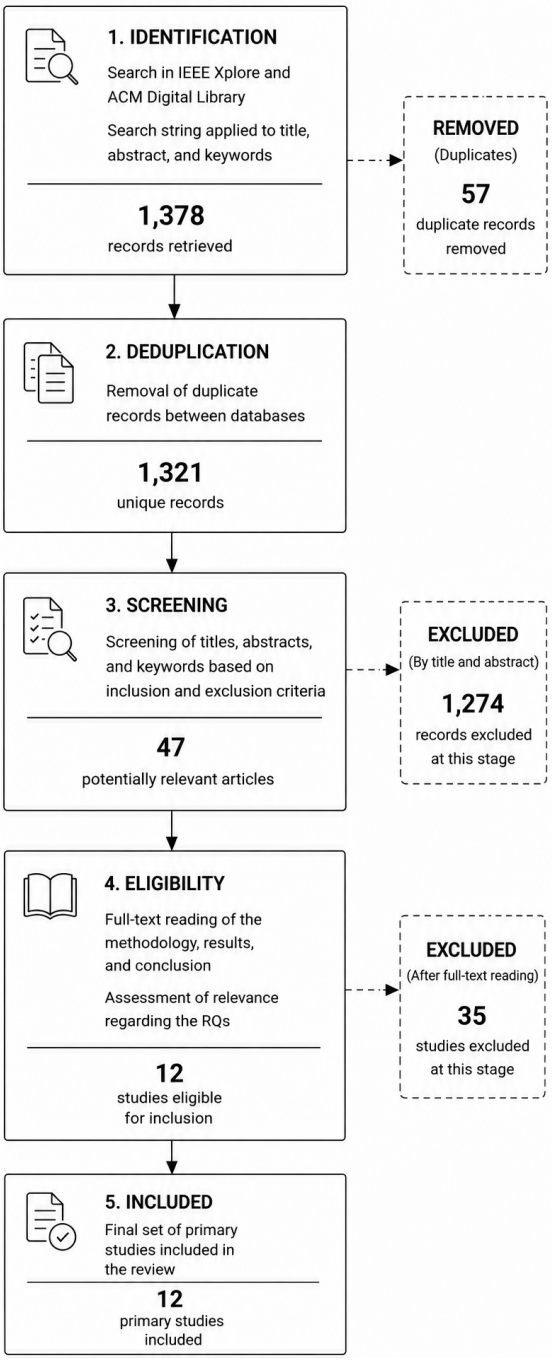

Figure 1: Study selection workflow.

identification of common applications, recurring challenges, and emerging research trends in LLM-based Software Testing.

4 Results and Analysis

This section presents the findings of the scoping review organized according to the three proposed Research Questions (RQs). The results synthesize evidence from the 12 primary studies included in the review, highlighting the main application domains of Large Language Models (LLMs) in Software Testing, the challenges reported in the literature, and emerging research opportunities.

4.1 Applications of LLMs in Testing Activities (RQ1)

The analysis of the 12 primary studies indicates that LLMs have been applied to a wide range of Software Testing activities, including unit test generation, API testing, vulnerability analysis and repair, generation of testing-support artifacts, and methodological and educational support. Despite this diversity, approaches aimed at generating testing artifacts remain the predominant application. Although the application domains are diverse, many studies combine LLMs with traditional verification and validation techniques, suggesting a trend toward hybrid testing workflows. Table 3 summarizes the main categories of applications identified in the primary studies, together with representative examples and their respective contributions.

Overall, LLMs are predominantly used to generate testing artifacts, including unit tests, test inputs, formal specifications, and test oracles. Most approaches integrate LLMs with established validation mechanisms, such as static analysis, compilation checks, execution-based validation, coverage measurement, or differential testing, indicating a trend toward hybrid testing workflows

4.2 Challenges and Limitations (RQ2)

The challenges reported in the reviewed studies can be grouped into four recurring themes: reliability and hallucinations, variability across models and prompts, context and scalability limitations, and detection accuracy.

- **Reliability and hallucinations**: Several studies [11, 16] report that LLMs frequently generate tests or patches that appear plausible but contain logical errors or fail during execution. Kulsum et al. [11], for example, demonstrate that incorporating *patch validation feedback* substantially improves repair effectiveness, reinforcing that LLM-generated outputs require independent verification. Similarly, Nan et al. [16] show that additional guidance is often necessary to improve the correctness and reliability of generated tests.
- **Variability across models and prompts**: Comparative evaluations [20] reveal considerable variation in performance across different LLMs and prompt formulations. This sensitivity introduces non-deterministic behavior, making experimental results more difficult to reproduce and compare across studies.
- **Context and scalability limitations**: Studies on API testing [10, 12] report difficulties when applying LLMs to large input spaces, multi-file projects, and systems with complex execution contexts. These scenarios often require additional techniques, such as semantic guidance, multi-agent collaboration, or input space partitioning, to maintain effectiveness.
- **Detection accuracy**: Yu et al. [23] report that multilingual vulnerability detection remains challenging, with reduced precision across different programming languages and heterogeneous codebases. Their findings indicate that further

Table 3: Categories of LLM applications in Software Testing and associated studies (RQ1).

Category	Studies	Description and contribution
Unit test generation	[16, 20, 22]	The most mature application identified. LLMs are combined with static analysis [22] and with explicit test intention information [16] to generate unit tests with higher coverage and diversity, compared to traditional automated generation tools. Comparative studies across different LLMs [20] show significant variability in success rates.
API testing and multi-agent systems	[10, 12]	LLMs are used for input space partitioning in libraries and APIs [12], as well as in multi-agent approaches that leverage semantic graphs and guided input generation [10]. These techniques aim to produce semantically coherent inputs, exposing vulnerabilities and undesired behaviors in REST interfaces.
Vulnerability repair and analysis	[11, 23]	Studies such as Kulsum et al. [11] evaluate LLMs in automated vulnerability repair tasks, incorporating *patch validation feedback* to close the correction loop. Yu et al. [23] investigate vulnerability detection in multilingual scenarios, highlighting the potential of LLMs in heterogeneous code environments, although still facing accuracy challenges.
Generation of support artifacts (oracles, specifications, acceptance criteria)	[13, 19, 24]	LLMs are used to generate formal specifications [13], acceptance criteria for *user stories* [19], and artifacts for differential testing of WebAssembly runtimes [24]. These artifacts act as test oracles or reference specifications, increasing the level of automation and rigor in the testing process.
Methodological and educational support	[5, 14]	Margabandu et al. [14] propose an "LLM oracle" to support the design of experiments in combinatorial testing, making complex methodologies more accessible. Fang et al. [5] compare the use of ChatGPT with traditional checklists in teaching unit testing, analyzing the impact on learning and test quality.

Table 4: Challenges and limitations reported in LLM-assisted Software Testing studies (RQ2).

Challenge	Main findings
Reliability and hallucinations	Plausible but incorrect outputs; validation mechanisms (e.g., patch validation, explicit testing intentions, independent verification) remain necessary [11, 16].
Variability across models and prompts	Performance differs substantially across LLMs and prompts, limiting reproducibility [20].
Context and scalability limitations	Large codebases, stateful APIs, and complex systems require semantic guidance, input partitioning, or multi-agent collaboration [10, 12].
Detection accuracy	Reduced accuracy in multilingual vulnerability detection across heterogeneous codebases [23].

advances are required before LLMs can be reliably applied to multilingual vulnerability analysis.

4.3 Research Opportunities and Future Directions (RQ3)

The reviewed studies identify several opportunities for advancing the use of LLMs in Software Testing.

- **Hybrid testing workflows**: Several studies advocate integrating LLMs with established Verification and Validation (V&V) techniques, including static analysis [22], differential testing [24], automated patch validation [11], and execution-based verification. Such hybrid workflows aim to improve both the productivity and the reliability of AI-assisted testing.
- **Knowledge extraction and specification generation**: Multiple approaches employ LLMs to derive formal specifications, acceptance criteria, semantic relations, and input partitions from source code or documentation [10, 12, 13, 19]. These capabilities suggest broader opportunities for integrating LLMs with model-based, specification-driven, and semantic-guided testing techniques.
- **Human-centered testing**: Educational and methodological studies [5, 14, 19] indicate that software testers increasingly assume supervisory roles, including prompt engineering, validation of generated artifacts, experimental design, and critical assessment of AI-assisted outputs. Rather than replacing testers, LLMs are expected to augment human decision-making.
- **Improved test generation strategies**: Recent work suggests that incorporating richer contextual information, such as explicit testing intentions [16], represents a promising direction for improving the coverage, diversity, and effectiveness of automatically generated test cases.

4.4 Discussion

The findings of this review suggest that the adoption of Large Language Models (LLMs) in Software Testing is evolving from exploratory applications toward a structured integration with established testing practices. Rather than replacing traditional Verification and Validation (V&V) techniques, the reviewed studies increasingly position LLMs as complementary components that automate labor-intensive activities while relying on validation mechanisms

such as static analysis, differential testing, or execution-based verification. This trend indicates that hybrid testing workflows represent a prominent direction for AI-assisted Software Testing.

Another important observation concerns the evolving role of software test engineers. Instead of reducing the need for human expertise, LLMs shift engineers' responsibilities toward supervising AI-generated artifacts, designing effective prompts, interpreting results, and validating outputs. This reinforces the view that human expertise remains essential for trustworthy AI-assisted testing.

Table 5 summarizes the main findings obtained for each research question.

Table 5: Summary of the key findings for each RQ.

RQ	Key Findings
RQ1	LLMs are predominantly applied to the generation of testing artifacts, including unit tests, test inputs, formal specifications, test oracles, and supporting testing activities.
RQ2	The main reported challenges include reliability and hallucinations, variability across models and prompts, context and scalability limitations, and detection accuracy.
RQ3	The main research opportunities involve hybrid testing workflows, knowledge extraction and specification generation, human-centered AI-assisted testing, and improved test generation strategies.

4.5 Threats to Validity

Like any secondary study, this review is subject to limitations concerning the search strategy, study selection, evidence synthesis, and temporal validity. First, the search was limited to IEEE Xplore and the ACM Digital Library. Although these databases provide strong coverage of Software Engineering and directly index major venues in the field, they do not encompass the entire relevant literature. Thus, the identified primary studies should not be considered an exhaustive representation of research on LLMs in software testing. In addition, the search relied on predefined terms such as "Software Testing", "Test Generation", "Test Case", and "Bug Repair". Relevant studies using alternative terminology may therefore not have been retrieved.

Second, restricting the review to publications in English, Spanish, or Portuguese between 2020 and 2025 may have excluded relevant evidence. To mitigate selection bias, predefined inclusion and exclusion criteria were consistently applied. Nevertheless, study selection and classification involve researcher judgment, and some degree of subjectivity cannot be completely eliminated.

Third, the synthesis does not formally weight primary studies according to their level of evidence or publication maturity. Consequently, studies with different levels of empirical support may receive similar visibility. The findings should therefore be interpreted as a characterization of reported approaches, trends, and research directions rather than as an assessment of the strength of evidence supporting individual techniques.

Finally, the rapid evolution of LLM technologies represents a temporal threat to external validity. As new models, tools, and empirical studies continue to emerge, the findings represent a snapshot of the literature within the period covered by the review. Despite these limitations, the review provides a structured account of the studies identified through the defined protocol and a complementary perspective by organizing the evidence across different Software Testing activities.

5 Conclusion

This paper presented a Scoping Review on the use of Large Language Models (LLMs) in Software Testing. Following a structured review protocol, 12 primary studies published between 2020 and 2025 were identified and analyzed. The review synthesizes the current state of research by mapping the main application domains of LLMs, the challenges reported in the literature, and emerging research opportunities.

The findings indicate that LLMs are predominantly employed to support the generation of testing artifacts, including unit tests, test inputs, formal specifications, and test oracles. Despite these promising applications, the reviewed studies consistently report challenges related to reliability, hallucinations, variability across models and prompts, and scalability. Consequently, the literature increasingly advocates integrating LLMs with traditional Verification and Validation (V&V) techniques, suggesting that hybrid testing workflows represent a prominent direction for the practical adoption of AI-assisted Software Testing.

The review also identifies several opportunities for future research, including hybrid testing workflows, improved knowledge extraction and specification generation, human-centered AI-assisted testing, and improved test generation strategies incorporating richer contextual information.

Like any secondary study, this review has some limitations. The search was restricted to the IEEE Xplore and ACM Digital Library databases and to publications written in English, Spanish, or Portuguese, which may have excluded relevant studies indexed in other digital libraries or published in other languages. Furthermore, given the rapid evolution of LLMs, new evidence may have emerged after the search period considered in this review.

Overall, the evidence synthesized in this review suggests that the future of LLM-assisted Software Testing lies not in replacing established testing methodologies, but in integrating the reasoning capabilities of LLMs with the rigor of conventional Software Testing practices. Such hybrid approaches have the potential to improve productivity while preserving the reliability, transparency, and trustworthiness required for modern software systems.

Artifact Availability

The replication package for this study is publicly available in a GitHub Repository[3]. It includes the search strings, the list of included studies, the extracted data, and the materials used to conduct the review.

[3]https://github.com/mauris81/replication-package-LLM-testing-

References

[1] Paul Ammann and Jeff Offutt. 2017. *Introduction to Software Testing* (2 ed.). Cambridge University Press.

[2] Hilary Arksey and Lisa O'Malley. 2005. Scoping Studies: Towards a Methodological Framework. *International Journal of Social Research Methodology* 8, 1 (2005), 19–32. doi:10.1080/1364557032000119616

[3] Mohamed Boukhlif, Nassim Kharmoum, and Mohamed Hanine. 2024. Llms for intelligent software testing: a comparative study. In *Proceedings of the 7th International Conference on Networking, Intelligent Systems and Security*. 1–8.

[4] Angela Fan et al. 2024. A Survey of Large Language Models for Software Engineering. *arXiv preprint arXiv:2402.06196* (2024).

[5] Z. Fang, J. Li, A. Liang, G. R. Bai, and Y. Huang. 2025. A Comparative Study on ChatGPT and Checklist as Support Tools for Unit Testing Education. In *Proceedings of the 33rd ACM International Conference on the Foundations of Software Engineering (FSE Companion)*. ACM, 871–882.

[6] M. Görmez, B. Turhan, A. Bener, and A. Camkiran. 2024. Large Language Models for Software Engineering: A Systematic Mapping Study. In *EuroSPI 2024 (Communications in Computer and Information Science)*. Springer.

[7] X. et al. Hou. 2024. Large Language Models for Software Engineering: A Systematic Literature Review. *ACM Transactions on Software Engineering and Methodology* 33, 8 (2024).

[8] Y. Huang, J. Wang, C. Chen, Z. Liu, S. Wang, and Q. Wang. 2024. Software Testing with Large Language Models: Survey, Landscape, and Vision. *IEEE Transactions on Software Engineering* 50, 4 (2024), 911–936.

[9] Ziwei Ji et al. 2023. Survey of Hallucination in Natural Language Generation. *Comput. Surveys* 55, 12 (2023), 1–38.

[10] M. Kim, T. Stennett, S. Sinha, and A. Orso. 2025. A Multi-Agent Approach for REST API Testing with Semantic Graphs and LLM-Driven Inputs. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE, 1409–1421.

[11] U. Kulsum, H. Zhu, B. Xu, and M. d'Amorim. 2024. A Case Study of LLM for Automated Vulnerability Repair: Assessing Impact of Reasoning and Patch Validation Feedback. In *Proceedings of the 1st ACM International Conference on AI-Powered Software (AIware)*. ACM, 103–111.

[12] J. et al. Li. 2025. LLM Based Input Space Partitioning Testing for Library APIs. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE, 1436–1448.

[13] L. Ma, S. Liu, Y. Li, X. Xie, and L. Bu. 2025. SpecGen: Automated Generation of Formal Program Specifications via Large Language Models. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE, 16–28.

[14] L. Margabandu, Z. Chen, W. E. Wong, and C.-W. Hsu. 2025. A Design of Experiments Oracle LLM for Research-Grounded Combinatorial Testing. In *2025 25th International Conference on Software Quality, Reliability, and Security Companion (QRS-C)*. IEEE, 82–90.

[15] Glenford J. Myers, Corey Sandler, and Tom Badgett. 2011. *The Art of Software Testing* (3 ed.). John Wiley & Sons.

[16] Z. Nan, Z. Guo, K. Liu, and X. Xia. 2025. Test Intention Guided LLM-Based Unit Test Generation. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE, 1026–1038.

[17] Michaela D. B. Peters, Christina M. Godfrey, Hanan Khalil, Patricia McInerney, Deborah Parker, and Cassia Baldini Soares. 2020. Guidance for Conducting Systematic Scoping Reviews. *JBI Evidence Synthesis* 18, 10 (2020), 2119–2126. doi:10.11124/JBIES-20-00167

[18] F. Qi, Y. Hou, N. Lin, S. Bao, and N. Xu. 2024. A Survey of Testing Techniques Based on Large Language Models. In *Proceedings of the 2024 International Conference on Computer and Multimedia Technology (ICCMT)*. ACM, 280–284.

[19] J. Rawson and S. Reddivari. 2025. A ChatGPT-Powered Tool for Automating Context-Aware Acceptance Criteria Generation for User Stories. In *2025 IEEE International Conference on Information Reuse and Integration and Data Science (IRI)*. IEEE, 325–330.

[20] Marlen Treviño-Villalobos, C. Quesada-López, E. Jiménez-Delgado, R. Quirós-Oviedo, and I. Díaz-Oreiro. 2025. A Comparative Evaluation of ChatGPT, DeepSeek and Gemini in Automatic Unit Test Generation: a Success Rate Analysis. In *2025 IEEE VIII Congreso Internacional en Inteligencia Ambiental, Ingenieria de Software y Salud Electronica y Movil (AmITIC)*. IEEE, 1–8.

[21] Andrea C. Tricco, Erin Lillie, Wasifa Zarin, Kelly K. O'Brien, Heather Colquhoun, Danielle Levac, David Moher, et al. 2018. PRISMA Extension for Scoping Reviews (PRISMA-ScR): Checklist and Explanation. *Annals of Internal Medicine* 169, 7 (2018), 467–473. doi:10.7326/M18-0850

[22] W. Wei. 2025. Static Analysis and LLM for Comprehensive Java Unit Test Generation. In *2025 8th International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE)*. IEEE, 87–92.

[23] J. et al. Yu. 2025. A Preliminary Study of Large Language Models for Multilingual Vulnerability Detection. In *Proceedings of the 34th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA Companion)*. ACM, 161–168.

[24] S. Zhou, J. Wang, H. Ye, H. Zhou, C. L. Goues, and X. Luo. 2025. LWDIFF: an LLM-Assisted Differential Testing Framework for Webassembly Runtimes. In *Proceedings of the IEEE/ACM 47th International Conference on Software Engineering (ICSE)*. IEEE, 153–164.