AI-Centric Test Case Generation: A Systematic Review

Caio Gusmão
Federal University of Pernambuco
Recife, Brazil
cbg@cin.ufpe.br

Diego Calixto
Federal University of Pernambuco
Recife, Brazil
dhvc@cin.ufpe.br

José Nascimento
Federal University of Pernambuco
Recife, Brazil
jgsn2@cin.ufpe.br

Júlia Fragoso
Federal University of Pernambuco
Recife, Brazil
jamf@cin.ufpe.br

Abstract

Artificial Intelligence (AI) is transforming automated test generation, offering innovative solutions to enhance efficiency, coverage, and adaptability in software testing. However, a cohesive understanding of its strengths and limitations remains elusive. This systematic review analyzes 46 studies from IEEE Xplore and Scopus (2020–2024) to explore the role of AI in test generation, focusing on key techniques, benefits, challenges, and emerging trends. Our findings highlight the dominance of Large Language Models (LLMs), prized for their readability and flexibility, alongside hybrid approaches that merge evolutionary algorithms and machine learning to improve test coverage and performance. Despite these advancements, persistent challenges, such as model interpretability, data dependencies, and scalability, underscore the need for further refinement. Looking ahead, the field is moving toward tighter integration of AI methods, refined LLM optimization, and specialized applications for targeted domains. By synthesizing the latest empirical evidence, this review provides valuable insights for researchers and quality assurance teams navigating the dynamic landscape of AI-driven testing.

Keywords

Artificial Intelligence, Automated Test Generation, Large Language Models, Software Testing

1 Introduction

The increasing complexity of modern software systems, driven by agile development practices, distributed architectures, and evolving user expectations, has heightened the demand for efficient and scalable testing methodologies. Traditional test generation approaches, often relying on manual effort or rule-based automation, struggle to keep pace with rapid release cycles and the intricate interdependencies inherent in contemporary software engineering.[12] To address these challenges, the integration of Artificial Intelligence (AI) into test automation has emerged as a transformative paradigm, promising greater efficiency, broader coverage, and reduced human intervention. [8]

Recent advances in AI techniques, such as Machine Learning (ML), Deep Learning (DL), and Natural Language Processing (NLP), have demonstrated significant potential for automating test case generation, optimizing test suites, and identifying defects in complex systems. [26] [12] [37] For instance, generative AI models, including Large Language Models (LLMs), are being employed to synthesize test scenarios from natural language specifications, while reinforcement learning enables adaptive test prioritization. [26] Applications in critical domains, such as autonomous vehicles, highlight AI-driven test generation in the automotive sector, including emergency braking scenarios crucial for safety. [35] Likewise, studies underscore AI's effectiveness in unit test generation, defect detection, and regression testing, offering measurable improvements in accuracy and scalability compared to conventional methods.

Despite these advancements, AI adoption in test generation faces significant challenges. Issues such as model interpretability, data quality and availability, and computational costs remain unresolved. Moreover, the diversity of AI techniques, each with different trade-offs in applicability, efficiency, and coverage, necessitates a systematic analysis to guide practitioners and researchers. Existing literature often focuses on isolated case studies or specific techniques, leaving gaps in holistic assessments of benefits, limitations, and emerging trends.

This article presents a Systematic Literature Review (SLR) investigating how AI is transforming automated test generation in software development. Focusing on publications from 2020 to 2024, we analyze primary studies from the IEEE Xplore and Scopus databases to answer five research questions:

Q1: Which AI techniques are most commonly applied in automated test generation?
Q2: How do these approaches compare in terms of test coverage, efficiency, and practical
Q3: What are the benefits of implementing AI in test generation?
Q4: What are the key challenges and limitations in adopting AI for test generation?
Q5: What are the emerging trends and future research directions in this field?

Our review identifies key trends, including the growth of generative AI for scenario-based testing and the integration of ML for predictive test optimization. It also reveals persistent barriers, such as the need for robust benchmarks and tools to address the "black-box" nature of AI models. By synthesizing insights from 46 studies, this work provides a comprehensive mapping of the state of the art, practical implications for quality assurance teams, and a roadmap for researchers tackling challenges related to scalability, cost, and reliability in AI-driven testing.

The remainder of this article is structured as follows: Section 2 presents related work, highlighting relevant themes for the study's context. Section 3 details the SLR methodology, including search strategies and inclusion criteria. Sections 4 and 5 discuss the aggregated findings observed in the literature, followed by an analysis of the research questions. Finally, Section 6 concludes with recommendations for future research.

2 Background

Test generation is a critical phase in software development, playing a fundamental role in ensuring system quality and reliability. Well-designed tests enable early defect detection, reducing costs and minimizing risks associated with production failures.[39] Traditionally, test case creation has been conducted manually or with the support of automated techniques.

The main types of tests include unit testing, which evaluates individual components; integration testing, which verifies interactions between modules; and regression testing, which ensures that new changes do not introduce faults into existing functionalities.[17] With the advancement of agile methodologies and shorter development cycles, automated testing has become essential to keep pace with the rapid evolution of software systems.[17] However, the increasing complexity of modern systems has challenged these approaches, making them insufficient on their own to handle the scale and diversity of scenarios required for effective validation. [40] [34]

To enhance the agility of the quality assurance cycle, automated test case generation aims to reduce manual effort and increase the efficiency of software verification processes. [27] Conventional methods include model-based approaches, symbolic and concolic analysis, evolutionary approaches, and coverage-based techniques to identify relevant inputs and test scenarios. However, these methods often face challenges such as the combinatorial explosion of possible inputs, the need for adaptation to different domains, and the difficulty of capturing complex scenarios. [47] Additionally, traditional test generation may require significant human intervention to refine and validate the generated test cases, limiting its scalability in modern systems.

In this context, the application of Artificial Intelligence (AI) has emerged as a promising alternative to enhance automated test generation. Various AI-based approaches have been explored, including machine learning and deep learning, which leverage models trained on large datasets to identify patterns in source code or past software executions, enabling the generation of more representative test cases. [37] Moreover, evolutionary algorithms, such as genetic algorithms, have been applied to optimize test creation by employing strategies that maximize code coverage and efficiently identify potential defects. Another emerging approach involves the use of neural networks and large-scale language models, such as transformer-based models, which enable automated test generation by interpreting natural language descriptions and suggesting context-aware test scenarios. [47]

3 Method

This study follows the Systematic Literature Review (SLR) approach to investigate how Artificial Intelligence (AI) has been used for the automated generation of tests in software development. According to [21, 22], this type of review is structured to identify, evaluate, and interpret all relevant evidence available on a specific research question, employing a rigorous, transparent, and reproducible process. The methodology was divided into five main stages:

3.1 Definition of the Research Protocol

The prior definition of a protocol is essential to reduce bias and ensure the reproducibility of the SLR [22]. In this study, the protocol established the following points:

3.1.1 Research Objective. To investigate how the adoption of Artificial Intelligence (AI) is impacting the automatic generation of tests in software production, including the identification of the main techniques, benefits, challenges, and trends in the automated test generation process within the software production context.

3.1.2 Motivation. The growing complexity of software systems requires more efficient and automated strategies for test generation, reducing manual effort and increasing test coverage. Artificial Intelligence has been applied to enhance this process, using techniques such as machine learning, deep learning, and evolutionary algorithms. However, there are still challenges to be addressed, such as model interpretability, solution scalability, and validation of automatically generated test cases. Conducting an SLR allows the consolidation of existing knowledge and the identification of research gaps and opportunities [21].

3.1.3 Main Research Question. How has Artificial Intelligence been used for software test generation, and what are the challenges and benefits of these approaches?

3.1.4 Secondary Research Questions.

- Q1: Which AI techniques are most commonly applied in automated test generation?
- Q2: How do these approaches compare in terms of test coverage, efficiency, and practical applicability?
- Q3: What are the benefits of implementing AI in test generation?
- Q4: What are the main challenges and limitations faced in adopting AI for test generation?
- Q5: What are the emerging trends and future research directions in this area?

3.2 Search Strategy

Building an effective search strategy requires a clear definition of the main research terms and their synonyms to broaden the scope of the results. The choice of databases should consider those most representative of the study domain, prioritizing sources that publish research directly related to the investigated topic. This way, the selection of specialized academic databases provides a solid starting point for the identification, classification, and analysis of available studies [21]. Thus, the selected databases were IEEE Xplore and Scopus, widely used in Software Engineering and Artificial Intelligence.

The search string was developed based on the decomposition of the research question and synonym review:

("Generative AI" OR LLM OR AI OR "Artificial Intelligence" OR "Large Language Models") AND ("Software Testing" OR "Quality Assurance") AND ("Test Generation" OR "Automated Test Creation")

The search period was restricted to between 2020 and 2024, as recommended by [21], in order to capture updated and relevant studies. After executing the search string, 200 articles were retrieved from IEEE Xplore and 64 from Scopus. The results were exported to a spreadsheet and organized for the next selection stage.

3.3 Study Collection and Selection

The study selection was conducted carefully to ensure that only the most relevant articles were included in the analysis. To achieve this, inclusion and exclusion criteria were defined, ensuring that the selected studies aligned with the research topic [22]. Only articles indexed in the IEEE Xplore and Scopus databases and written in English were considered. Furthermore, only publications from conferences and academic journals were included to ensure the scientific quality of the analyzed studies.

Articles that were outside the established period of 2020 to 2024, as well as those classified as secondary studies (such as systematic reviews or mapping studies), were excluded. Works that required payment for access, were duplicated, or were unavailable for download were also discarded. Additionally, articles that, upon analysis, showed no direct relation to the research questions were eliminated.

The application of the selection criteria occurred in two stages. First, each article was analyzed based on its title and abstract. In this phase, studies that clearly met the criteria were included in the next stage, while those that did not demonstrate adherence to the topic were excluded. Articles that raised doubts about their relevance were retained for more in-depth analysis.

For studies that remained in doubt after this first screening, a more detailed examination of the introduction and conclusion was conducted. Those that showed a clear alignment with the study's objectives were included in the final analysis base, while the others were excluded.

After this filtering, a significant reduction in the number of considered studies was observed. After the application of the first filter, 54 studies from IEEE Xplore and 21 from Scopus remained. All included studies were organized in a spreadsheet, consolidating the results for the subsequent stages of quality assessment and data synthesis.

3.4 Study Quality Assessment

It is essential to assess the methodological quality of the selected studies [22]. To qualify the selected studies, an evaluation was adopted based on five fundamental quality criteria. These criteria included: (i) the methodological clarity of the study; (ii) the proper definition of the research context; (iii) the explicit statement of the research objectives; (iv) the consistency and relevance in the discussion of the presented results; and (v) the identification of limitations, challenges, or threats to the validity of the findings.

Each article was evaluated based on these aspects, receiving a score according to its level of adherence: a score of 1.0 when the criterion was fully met; 0.5 for partial fulfillment; and 0.0 when the criterion was not met. To ensure the consistency and robustness of the final analysis base, only studies with a total score above 2.5 out of a maximum of 5 points were considered. After this filtering process, the final corpus consisted of 46 studies, with 33 from the IEEE Xplore database and 13 from Scopus.

3.5 Data Extraction and Analysis

The data extraction stage was conducted systematically and in a structured manner, based on the methodological guidelines of [22], which emphasize the importance of recording information in a standardized way to ensure consistency, traceability, and alignment with the objectives of the review. For organizational and traceability purposes, the articles were labeled with the codes PS01 to PS46, based on ascending order of publication year. Initially, each selected study was analyzed in terms of its responsiveness to the secondary research questions, and was classified according to the level of response identified: fully answers, partially answers, or does not answer.

After this initial evaluation, the studies that presented responses (complete or partial) were submitted to a thematic categorization, supported by the Notebook Lm tool, which assisted in organizing, classifying, and analyzing the extracted excerpts. The categories used were previously defined based on the secondary research questions and the trends observed in the literature, allowing for a more refined and theme-oriented analysis. It is important to note that each response from a primary study could be assigned to more than one category, depending on its content and relevance.

For Q1 (AI techniques applied to test generation), the articles were classified into categories such as: Large Language Models (LLMs), Reinforcement Learning (RL), Evolutionary Algorithms (EA), Deep Learning (DL), Natural Language Processing (NLP), traditional Machine Learning (ML), Generative AI (non-LLM), or No Response.

For Q2 (comparisons in coverage, efficiency, and applicability), the data were organized into: Superior coverage compared to traditional methods, Efficiency improvements, Practical deployment and usability, Comparison among AI techniques, and No Response.

For Q3 (benefits of implementing AI), the categories included: Increased test coverage and effectiveness, Improved efficiency and resource optimization, Automation and reduced manual effort, Adaptability to complex systems, Enhanced usability and collaboration, Real-world deployment and integration, Improved defect detection and accuracy, Learning and continuous improvement, and No Response.

For Q4 (main challenges and limitations), the studies were grouped into: Technical and model limitations, Data quality and dependency, Resource and infrastructure constraints, Scalability and generalization, Human dependency and usability, Reliability and consistency, Interpretability and transparency, Integration and tooling, Validation and ethical concerns, and No Response.

Finally, for Q5 (emerging trends and future directions), the categories considered were: Integration of AI techniques, Optimization of LLMs and prompt engineering, Automation and scalability, Domain-specific applications, Human-AI collaboration, Ethics, transparency and explainability, Advanced learning and adaptation, Validation and quality metrics, and No Response.

All extracted evidence was systematized into a spreadsheet to enable the organization, analysis, and categorization of the data. This process enabled a consistent thematic analysis aligned with the research questions, supporting the identification of patterns, recurring challenges, emerging trends, and remaining gaps in the literature, in accordance with methodological guidelines commonly applied in systematic reviews [22].

4 Results

This section details the systematic approach used to identify and select relevant studies for our review. From the defined search protocol, we initially identified 264 articles distributed across two major databases: 200 studies from IEEE Xplore and 64 from Scopus In the first phase, after applying the exclusion criteria, 54 studies from IEEE and 21 from Scopus were approved, totaling 75 selected articles. In the second phase, with a more refined selection, 34 articles from IEEE and 13 from Scopus were approved, resulting in 46 studies. A summary of the number of studies obtained during the selection phases can be seen in figure 1.

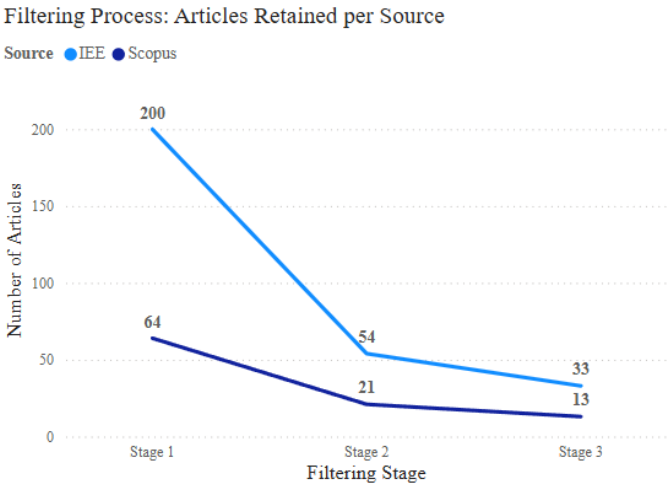

Figure 1: Articles retained per source

The studies were also analyzed regarding the distribution of their publications over time, aiming to illustrate the growing interest in the topic within the research's defined timeframe. The graph in Figure 2 shows the distribution of studies over the years. A similar number of articles was published in 2020, 2021, and 2022, indicating stable research output during this period. However, a significant surge occurred in 2023, followed by an even more pronounced peak in 2024, which recorded the highest number of studies on the subject. This exponential growth aligns with the rapid advancements and widespread adoption of artificial intelligence (AI) in recent years, particularly due to breakthroughs in generative AI models and their increasing applicability to diverse domains.

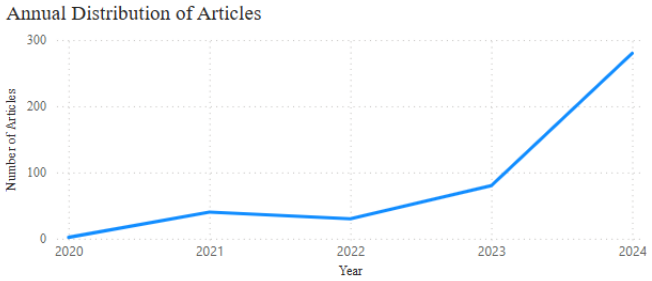

Figure 2: Annual distribution of articles

The figure 3 illustrates the distribution of contributing countries. The analysis of first authors across the 46 selected studies revealed a predominant contribution from China (28,26%) and India (17,39%), which collectively account for over half of the publications. This geographic concentration aligns with recent trends in global AI research investment, where both nations have significantly increased funding and institutional support for AI-based testing initiatives. The dominance of China and India may reflect: (1) the expansion of their AI talent pools in software engineering, (2) government policies prioritizing AI adoption in quality assurance, and (3) the availability of large-scale industrial datasets for testing research.

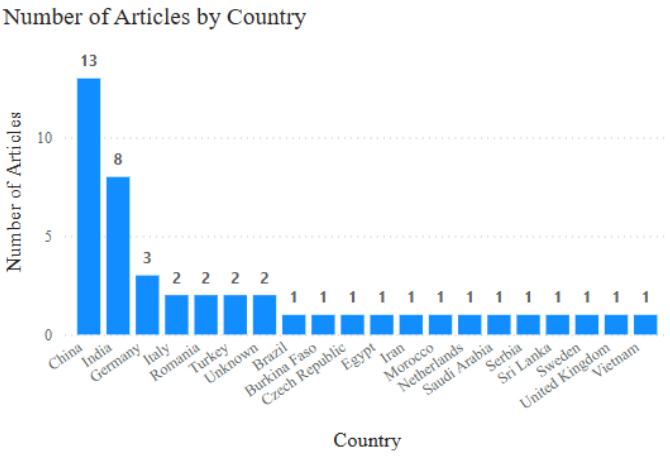

Figure 3: Distribution of the first author's country of origin

5 Discussion

This section organizes and analyzes the main findings of the review, addressing the five proposed research questions. We discuss: (Q1) the most commonly used AI techniques in test generation, (Q2) their comparisons in terms of coverage, efficiency, and applicability, (Q3) the identified benefits, (Q4) the main challenges faced, and (Q5) the emerging trends and future research directions. Each topic is addressed based on the evidence extracted from the 46 selected primary studies. [1]

[1]The full list of studies and their classifications is available in Table 1 in Appendix A.

5.1 Q1 – Which AI techniques are most commonly applied in automated test generation?

The analysis of the studies revealed that Large Language Models (LLMs) emerge as the most widely applied Artificial Intelligence technique in automated test generation, being mentioned in 29 studies (63%), as shown in Figure 4.

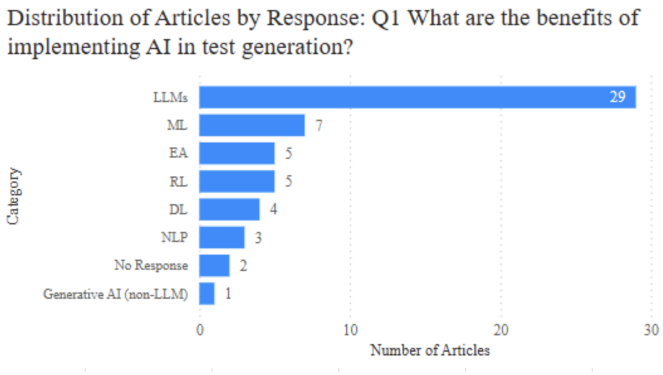

Figure 4: Distribution of studies by AI technique applied to automated test generation.

This predominance of LLMs reflects their ability to interpret natural language requirements, generate readable test code, and adapt to different application contexts, such as unit testing (PS21, PS40, PS46), API validation (PS38), and testing environments for complex systems like autonomous driving (PS32). For instance, PS16 highlights that ChatGPT generates tests with high readability and explanatory comments, while PS44 demonstrates that the use of TestGen-LLM increases code coverage in automated tests. Additionally, LLMs have often been integrated with complementary techniques such as prompt engineering (PS20, PS22, PS23, PS25, and PS30) and Retrieval-Augmented Generation (PS26, PS33), which further enhances their applicability and practical efficiency.

Traditional Machine Learning (ML) appears in 7 studies (15.2%), being applied in various automated testing contexts. In PS27, it is used for synthetic data generation and defect prediction; in PS28, for automatic traceability between requirements and tests; in PS08, for modeling web page interactions in system testing environments; and in PS36, through data clustering with k-means to organize and prioritize tests. PS07 proposes a hybrid approach combining ML with evolutionary algorithms to increase test case diversity and coverage. These applications demonstrate how traditional ML has been used to enhance efficiency, coverage, and automation across different stages of the software testing process.

Evolutionary Algorithms (EA) and Reinforcement Learning (RL) are tied, each appearing in 5 studies (10.9%). EAs have been applied to heuristic search of test cases (PS03, PS18) and combined with other techniques, such as ML (PS07) and LLMs (PS26), to simultaneously optimize multiple criteria, such as coverage and execution time. Moreover, PS43 combines EAs with non-LLM-based generative AI techniques—the only study in the sample categorized as Generative AI (non-LLM)—using Generative Adversarial Networks (GANs) for multi-objective test generation and selection. RL, in turn, is used to develop adaptive exploration strategies, particularly in contexts such as fuzzing (PS04, PS15) and simulation environments (PS06).

Deep Learning (DL) appears in 4 studies (8.7%), with applications focused on visual recognition of UI elements (PS05, PS09, PS13) and video analysis for visual test generation (PS10). Natural Language Processing (NLP) is identified in 3 studies (6.5%), mainly applied to semantic analysis of textual test case specifications (PS02) and automatic test generation from natural language requirements (PS27). It is worth noting that two studies in the sample did not provide a clear response to Q1 (PS01, PS14).

From this distribution, a consolidated trend of LLM adoption is evident, driven by the evolution of generative models and the ease of API integration. However, the variety of techniques found points to a multifaceted scenario, in which hybrid approaches are particularly promising. As previously noted, while LLMs dominate the current landscape, classical techniques and their combinations remain strategic resources in building robust, scalable, and adaptable solutions for the diverse demands of contemporary software development. [2]

5.2 Q2 - How do these approaches compare in terms of test coverage, efficiency, and practical applicability?

The second research question aimed to investigate how AI-based approaches compare in terms of test coverage, efficiency, and practical applicability. Based on the analysis of the 46 studies included in the review, the responses were classified into five main categories: Practical Deployment and Usability, Superior Coverage Compared to Traditional Methods, Efficiency Improvements, Comparison Among AI Techniques, and No Response. The distribution of these categories can be observed in Figure 5.

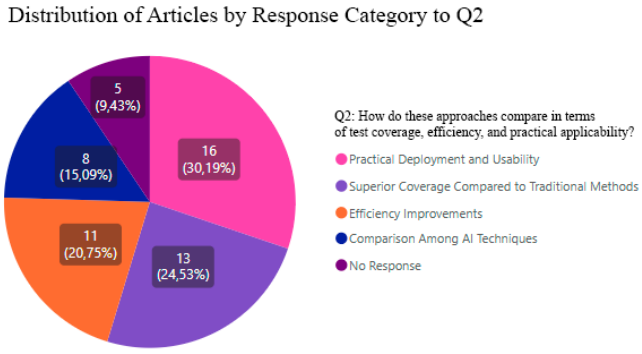

Figure 5: Distribution of articles responses by technique evaluation criteria.

The Practical Deployment and Usability category, present in 16 studies (30.19%), includes approaches that demonstrate concrete

[2]The distribution of studies by category related to Q1 is presented in Table 2 in Appendix A.

applicability in real-world software development and testing scenarios, either through ease of integration, automation of manual tasks, or compatibility with existing tools and processes. For example, study PS10 utilizes real-world videos to generate visual tests, facilitating automation for graphical user interfaces. Similarly, PS09 and PS13 apply Visual AI techniques to automate UI testing, focusing on replacing manual effort and automatically validating interface elements. PS25 highlights the use of generative AI to explore a broad range of scenarios during the testing process, enhancing coverage without relying on predefined scripts. PS27 proposes an integrated system combining NLP and ML that processes requirements and documentation to predict defects and generate tests automatically, aligning with real-world development workflows. Another example, PS33, reinforces this approach by applying evolutionary algorithms in automated testing environments, emphasizing continuous integration and multi-objective optimization.

The Superior Coverage Compared to Traditional Methods category appeared in 13 studies (24.53%), encompassing approaches that outperform traditional testing techniques, such as random testing or symbolic execution, in terms of metrics like line coverage, branch coverage, or scenario diversity. Study PS38 compares an LLM-based approach with a traditional heuristic and demonstrates superior results in terms of coverage and fault detection in RESTful services. PS07 reports significant improvements by combining evolutionary algorithms with machine learning, achieving up to a 50% increase in branch coverage. PS44, which also falls under the practical applicability category, shows that TestGen-LLM outperformed manual approaches in tests conducted at Meta. Other studies, such as PS03, PS22, PS24, PS31, and PS41, further highlight AI's potential to enhance test coverage—PS24, for instance, presents an AI-augmented software development approach that integrates LLM and ML techniques across different phases of the software lifecycle. Focusing on the testing phase, the study demonstrates that AI achieved 90% performance with only 10% human involvement, resulting in efficiency gains of up to 45%.

The Efficiency Improvements category, found in 11 studies (20.75%), includes approaches that demonstrate operational gains, such as reduced test generation time, lower computational resource usage, or automation of repetitive tasks. Study PS04 describes the application of Reinforcement Learning in fuzzing, significantly reducing the time required to achieve coverage. PS18 utilizes EvoMaster to generate automated tests across different languages and contexts, emphasizing the production of autonomous and reusable tests, thereby improving process efficiency by reducing the need for manual configuration and external integration. PS28 presents an ML-based automated approach for requirements traceability and testing, with empirical evidence showing improved accuracy and reduced manual effort, contributing to verification efficiency. PS08, in turn, proposes the use of machine learning to model web page behavior and predict interactions, enabling more efficient test generation in dynamic environments and reducing the reliance on manual scripting. These findings suggest that, in addition to test quality, operational efficiency is a key evaluation criterion for AI-based approaches.

In the Comparison Among AI Techniques category, eight studies (15.09%) directly compare different AI methods—for instance, LLMs versus evolutionary algorithms, or GPT-4 against other variants. Study PS26 compares approaches with and without prompt engineering and retrieval-augmented generation, demonstrating improvements in stability and reusability. PS40 and PS46 conduct cross-evaluations of different language models, considering criteria such as coverage, execution time, and clarity of generated tests. These comparisons contribute to understanding the relative strengths and limitations of each technique, supporting more informed choices depending on the application context.

Finally, five studies (9.43%) were classified under No Response, as they did not present empirical evaluations or explicit discussions regarding coverage, efficiency, or practical applicability. This methodological gap highlights the need for greater rigor and standardization in publications on AI-driven test generation, particularly in the assessment of obtained results.

Based on the analysis of RQ2, LLMs stand out for their practical applicability, ease of integration, readability, and real-world usability (PS38, PS44, PS16). Meanwhile, techniques such as EA, RL, and ML demonstrate superior performance in coverage and efficiency, particularly in complex domains (PS07, PS04, PS27). Studies like PS26 and PS43 reinforce a promising trend toward hybrid approaches that combine multiple techniques to enhance test robustness. These findings point to a multifaceted landscape where integrating different techniques could be a strategic approach to meeting the evolving demands of software engineering. [3]

5.3 Q3 - What are the benefits of implementing AI in test generation?

The implementation of Artificial Intelligence (AI) techniques in automated test generation offers significant benefits, as evidenced by the analysis of the studies. A quantitative analysis of these benefits is presented in the figure 6:

Distribution of Articles by Response: Q3 What are the benefits of implementing AI in test generation?

Category	Number of Articles
Improved Efficiency and Resource Optimization	16
Increased Test Coverage and Effectiveness	16
Enhanced Usability and Collaboration	12
Automation and Reduced Manual Effort	9
Improved Defect Detection and Accuracy	8
Real-World Deployment and Integration	8
Learning and Continuous Improvement	4
No Response	3
Adaptability to Complex Systems	2
Total	**78**

Figure 6: Distribution of articles by response to the question on AI benefits in test generation.

[3]The distribution of studies by category related to Q2 is presented in Table 3 in Appendix A.

The primary advantage observed is the increase in test coverage and effectiveness, identified in 34.8% of the studies (16/46). Evolutionary Algorithms (EA) and Large Language Models (LLMs) have proven effective in exploring complex search spaces. For instance, in PS03, genetic algorithm-based testing for automotive systems identified twice as many faults compared to random methods.

Furthermore, PS22 demonstrated improvements in code coverage using LLMs to generate processor tests, while PS07 explored the potential of combining multiple AI techniques to enhance test case generation. PS41 employed method decomposition with LLMs, surpassing traditional approaches by 20% in line and branch coverage.

Resource optimization and efficiency further complement these advancements, observed in 26.1% of the studies (12/46). Techniques such as Reinforcement Learning (RL) and evolutionary algorithms enabled substantial reductions in execution time and computational costs. PS04, for example, applied RL to accelerate fuzzing, leading to a 29

In PS24, AI achieved a 45% efficiency gain in the testing phase, significantly reducing manual effort. Additionally, AI automates test case development, contributing to process scalability. PS20 also highlights that AI-driven Natural Language Processing (NLP) improves test efficiency and accuracy, increasing automation levels and reducing the need for manual intervention.

The automation of repetitive tasks, present in 17.4% of the studies (8/46), reinforces the reduction in human intervention. For instance, PS08 automated web testing using Machine Learning (ML), dynamically adapting to interface changes. PS13 validated GUIs visually with Convolutional Neural Networks, eliminating the need for manual rewriting. Similarly, PS30 and PS35 generated test cases directly from natural language requirements using LLMs, simplifying complex workflows. This automation not only replaced labor-intensive manual steps, such as script coding, but also facilitated integration into dynamic environments.

Improvements in usability and collaboration, identified in 8.7% of the studies (4/46), stand out as a strategic differentiator. PS16 demonstrated that ChatGPT generates readable tests with clear annotations, enabling non-programmers to actively participate in the process. PS39 employs LLMs for automated documentation and scenario co-creation in educational settings, while PS42 integrated AI into agile practices (Behavior-Driven Development), standardizing communication between teams. These cases illustrate how AI democratizes test generation, making it more inclusive and transparent.

The practical application in real-world environments, validated in 6.5% of the studies (3/46), reinforces the maturity of AI-based solutions. PS18 integrated EvoMaster into CI/CD pipelines, generating self-executing tests for REST APIs. PS33 implemented large-scale UI testing in WeChat using LLMs, identifying critical bugs. Meanwhile, PS44 (Meta) incorporated LLMs into corporate processes, increasing unit test coverage by 25%. These examples demonstrate AI's feasibility in addressing industrial challenges, such as continuous integration and legacy systems.

Regarding precise defect detection, AI-based tools exhibit significant potential. Owl Eye (PS14), which utilizes computer vision, identifies visual inconsistencies in GUIs, reducing false positives through intelligent analysis of visual elements instead of pixel-by-pixel comparison. SoVAR (PS37) combines LLMs with constraint solvers to reconstruct automotive accident scenarios, showing notable improvements in information extraction accuracy compared to traditional methods.

While continuous learning has been explored in only a few studies, PS43 employed GANs to optimize hyperparameters in metamorphic testing, dynamically adapting to new inputs. This gap suggests future opportunities for techniques such as Reinforcement Learning and adaptive fine-tuning.

Finally, three studies did not provide conclusive insights into AI benefits. Nonetheless, LLMs and evolutionary algorithms stand out as the most versatile techniques, combining high coverage and efficiency. AI-driven automation has proven essential for complex systems (e.g., PS08 in web testing), while its integration into real-world environments (PS18, PS44) validates its industrial maturity. Despite the advancements, the scarcity of studies on continuous learning points to future challenges, such as incorporating dynamic feedback loops. Thus, AI is solidifying itself not only as a productivity tool but also as a strategic ally for innovation and quality in software testing. [4]

5.4 Q4 - What are the main challenges and limitations faced in the adoption of AI for test generation?

The analysis of 46 studies identified nine main categories of challenges in adopting AI for automated test generation, with technical and model limitations standing out as the most prevalent, appearing in 23 studies (31.5%), as illustrated in figure 7.

Distribution of Articles by Response: Q4 What are the main challenges and limitations in adopting AI for test generation?

Category	Number of Articles
Technical and Model Limitations	23
Reliability and Consistency	13
Human Dependency and Usability	8
Validation and Ethical Concerns	8
Integration and Tooling	6
Scalability and Generalization	6
Data Quality and Dependency	5
Resource and Infrastructure Constraints	5
Interpretability and Transparency	1
No Response	1
Total	**76**

Figure 7: Distribution of articles by response to the question on AI main challenges and limitation in test case generation.

[4]The distribution of studies by category related to Q3 is presented in Table 4 in Appendix A.

This predominance reflects the inherent difficulties of AI models' unpredictable behavior, such as hallucinations (PS22), nondeterminism (PS16), and overfitting (PS26). Specifically, PS03 and PS24 highlighted issues with non-reproducible failures in simulation environments and model decision opacity, respectively. These challenges are particularly critical in contexts such as autonomous vehicle testing (PS03, PS32) and microservices systems (PS29), where reliability is essential.

Reliability and consistency issues ranked second, appearing in 13 studies (17.8%), primarily manifesting as flaky tests and inconsistent results. PS08 identified challenges with variable loading times in web testing, while PS18 reported irreproducibility in evolutionary algorithms. PS36 further explored the effectiveness of metrics in ensuring test quality for autonomous vehicle systems.

Ethical and validation concerns, as well as challenges related to human dependency and usability, were tied, each appearing in 8 studies (9.6%). In the first group, PS27 warned of the risks of overlooking edge cases in overly automated approaches, while PS46 highlighted problems with nonsensical assertions generated by LLMs. In the second group, studies such as PS10 and PS39 underscored the ongoing need for human intervention for validation and correction, revealing limitations in the autonomy of current AI-based solutions.

Challenges related to scalability and generalization, as well as integration and tooling, also appeared equally, with 6 studies each (8.2%). PS07 and PS18 demonstrated difficulties in handling complex and evolving systems, while PS07 and PS17 identified compatibility issues with existing workflows. These findings suggest that the maturity of AI-driven solutions remains limited for complex enterprise environments.

Issues concerning data quality and dependency, along with resource and infrastructure constraints, appeared in 5 studies each (6.8%). PS05 and PS31 emphasized the critical importance of high-quality training data, while PS02 and PS33 highlighted challenges in computational cost and resource optimization. Lastly, the interpretability and transparency category was mentioned in only one study (1.4%), with PS06 raising concerns about understanding AI-generated test scenario specifications.

This distribution highlights a complex landscape where technical challenges dominate, yet operational and ethical concerns also require attention. The prevalence of LLM-related issues (present in 36% of the analyzed studies) is particularly noteworthy, given their increasing adoption. However, as evidenced by PS07, PS26, and PS43, hybrid approaches that combine multiple AI techniques may offer promising pathways to mitigate these limitations, especially when complemented with XAI frameworks (PS36) and resource optimization strategies (PS02, PS33). [5]

5.5 Q5 - What are the emerging trends and future research directions in this area?

The analysis of emerging trends and future research directions in AI-driven automated test generation reveals a multifaceted landscape, structured into nine main categories, as presented in Figure 8. The percentage distribution of these categories highlights the research community's priority areas in this domain.

Distribution of Articles by Response: Q5 What are the emerging trends and future research directions in this area?

Category	Number of Articles
Advanced Learning and Adaptation	8
Automation and Scalability	10
Domain-Specific Applications	8
Ethics, Transparency, and Explainability	4
Human-AI Collaboration	6
Integration of AI Techniques	12
No Response	5
Optimization of LLMs and Prompt Engineering	12
Validation and Quality Metrics	8
Total	**73**

Figure 8: Distribution of articles by response to the question on emerging trends in test generation.

The Integration of AI Techniques category, representing 16.4% of the studies (12 papers), highlights the synergistic combination of multiple AI approaches. PS07 exemplifies this trend by hybridizing evolutionary algorithms with machine learning, achieving up to a 50% increase in branch coverage. Similarly, PS26 compares the effectiveness of LLMs against genetic algorithms, while PS43 employs adversarial networks (WGAN) to optimize metamorphic transformations.

The second most frequent category, Optimization of LLMs and Prompt Engineering (16.4%, 12 studies), reflects the growing interest in refining language models. PS22 introduces the LLM-TG framework, leveraging rule-based prompts for processor test generation. PS30 and PS35 explore advanced techniques such as few-shot prompting and chain-of-thought, demonstrating significant accuracy improvements. PS45 proposes Structural Chain of Thought (SCoT) for system test code generation.

The Automation and Scalability category (13.7%, 10 studies) emphasizes solutions that reduce manual effort and enable large-scale application. PS13 and PS14 present AI-based visual testing frameworks for graphical user interfaces, while PS29 (Kashef) automates testing in microservices. PS27 explores CI/CD pipeline integration with self-healing scripts.

The Domain-Specific Applications category (11%, 8 studies) highlights the specialization of AI techniques for particular contexts: PS32 and PS37 apply LLMs in autonomous vehicle testing (CARLA/LGSVL), PS11 focuses on ERP testing with generative AI, PS38 (KAT) tests RESTful APIs via dependency analysis, and PS33 employs LLMs with retrieval-augmented generation (RAG) for UI testing in WeChat. PS04 (RiverFuzzRL), while not exclusive to IoT, demonstrates applicability in fuzzing across multiple domains.

The Human-AI Collaboration category (8.2%, 6 studies) explores synergies between human expertise and intelligent systems. PS12 proposes conversational agents to assist testers, while PS39 engages

[5]The distribution of studies by category related to Q4 is presented in Table 5 in Appendix A.

students in validating AI-generated tests. PS44 (TestGen-LLM) combines recommendations from multiple LLMs with human review.

The Advanced Learning and Adaptation category (11%, 8 studies) includes techniques such as reinforcement learning (PS03, PS15) and self-correction mechanisms. PS04 employs RL for fuzzing, while PS41 (HITS) utilizes auto-debugging with LLMs to correct failing tests.

The categories Ethics, Transparency, and Explainability (5.5%, 4 studies) and Validation and Quality Metrics (11

Finally, 6.8% of the studies (5 papers) were classified as No Response, as they did not explicitly address future trends.

Key Development Axes The analysis identifies three primary future development axes:

Technological Convergence: The predominance of Integration of AI Techniques and Optimization of LLMs (16.4% each) indicates a clear trend toward hybrid approaches that combine the strengths of different AI paradigms. Studies such as PS07 and PS26 demonstrate that this strategy can overcome the limitations of isolated techniques.

Contextual Specialization: The significant interest in Domain-Specific Applications (11%) and Automation and Scalability (13.7%) reflects the field's maturation, with solutions increasingly tailored to complex domains (PS32, PS37) and integrated into modern development workflows (PS27, PS29).

Humanizing Automation: Although less frequent, the Human-AI Collaboration (8.2%) and Ethics/Explainability (5.5%) categories indicate a crucial evolution—the transition from autonomous to collaborative systems, where AI amplifies (rather than replaces) human expertise (PS12, PS44).

The findings suggest that the future of AI-driven test generation will be shaped by the strategic combination of (1) highly optimized LLMs via prompt engineering, (2) integration with classical AI techniques for robustness, and (3) domain-specialized frameworks with clear validation mechanisms and human collaboration. This triad appears promising in balancing technical innovation with practical applicability in complex industrial scenarios. [6]

6 Related work

The literature has explored various approaches to integrating Artificial Intelligence (AI) into automated test generation. An extensive grey literature review was conducted by [38], with over 1,200 sources, identifying solutions such as automatic test generation and self-healing mechanisms. The study highlights the growing adoption of these techniques by industry professionals but also underscores practical limitations, such as the low robustness of generated scripts and the difficulty in adapting to frequent changes in systems.

Complementarily, [15] reviewed 55 AI-based tools for test automation, analyzing their features and limitations. The most common solutions involve testing with NLP, visual automation, and test generation from requirements. However, the authors point out recurring challenges, such as the lack of context in the generated tests, an excess of irrelevant cases, and a shortage of robust validations for industrial environments.

[6]The distribution of studies by category related to Q5 is presented in Table 6 in Appendix A.

Our study distinguishes itself by conducting a Systematic Literature Review (SLR) focusing exclusively on empirical evidence published between 2020 and 2024 in the IEEE Xplore and Scopus databases. In contrast to previous works, we applied a methodological rigor aligned with Kitchenham's guidelines, assessing the quality of the studies and classifying 46 scientific papers according to the AI techniques used, observed benefits, reported limitations, and emerging trends.

Additionally, our study delves into the comparison between techniques—such as LLMs, Reinforcement Learning, and Evolutionary Algorithms—considering not only the frequency of use but also the practical implications of each approach in terms of test coverage, operational efficiency, and applicability in industrial contexts. In this way, we provide a comprehensive and up-to-date view of the state of the art, offering relevant insights for researchers and professionals interested in the strategic adoption of AI in software quality assurance.

7 Conclusion

This article presented a Systematic Literature Review (SLR) on the application of Artificial Intelligence (AI) in automated software test generation, analyzing 46 studies published between 2020 and 2024. The results highlighted the growing adoption of large language models (LLMs) and other AI techniques, such as evolutionary algorithms and reinforcement learning, to enhance test coverage, efficiency, and automation. Additionally, we identified persistent challenges, including technical limitations and reliability concerns, as well as emerging trends like domain-specific applications and human-AI collaboration.

In the conclusion that follows, we synthesize the key findings, discuss their practical implications for researchers and practitioners, and propose future directions to advance this rapidly evolving field. These recommendations aim not only to address

7.1 Limitations

While this study provides valuable insights through its analysis of English-language publications from IEEE Xplore and Scopus (2020-2024), several limitations should be noted. The exclusive focus on these databases and time period, though providing robust computer science coverage, may have omitted significant non-English contributions and earlier foundational research. These methodological decisions reflect a careful balance between research depth and practical considerations, ensuring analytical rigor while acknowledging potential impacts on the study's breadth. The resulting trade-offs, though necessary, may influence the generalizability of our findings.

Another limitation pertains to the inherent selection bias in the study filtering process. Despite the application of rigorous inclusion and exclusion criteria, the initial selection based on titles and abstracts may have led to the omission of works whose relevance would only become apparent upon deeper examination. Furthermore, the exclusion of secondary studies (such as systematic reviews or mapping studies) may have limited the perspective on well-established trends in the field.

Finally, the quality assessment of the studies was based on predefined methodological criteria, but the inherent subjectivity of this

process may have influenced the inclusion or exclusion of certain articles. Nevertheless, the transparency in describing the methods and the systematization of the data enhance the study's replicability and facilitate the reanalysis of results in other contexts.

While these limitations must be considered, it is believed that the findings of this work provide valuable contributions to researchers and practitioners, highlighting emerging trends, practical challenges, and future opportunities in the application of AI for automated test generation. The discussions presented can serve as a foundation for future research and the strategic adoption of these techniques in software projects.

7.2 Future Works

To the best of our knowledge, this systematic literature review represents one of the most comprehensive investigations into the application of Artificial Intelligence (AI) techniques for automated test generation in software engineering. Our analysis of 46 primary studies from IEEE Xplore and Scopus (2020-2024) reveals key patterns and gaps in this rapidly evolving field.

The study makes three main contributions: First, it provides empirical evidence of the dominant role of Large Language Models (LLMs) in test generation, being employed in 63% of the analyzed studies (29/46), while also mapping the complementary use of other AI techniques such as Traditional Machine Learning (15.2%), Evolutionary Algorithms (10.9%), Reinforcement Learning (10.9%) and Deep Learning (8,7%). Second, it offers a comparative analysis of these approaches, demonstrating that while LLMs excel in practical deployment and usability (34.8% of studies), hybrid methods combining multiple AI techniques often achieve superior test coverage and efficiency. Third, the review identifies critical challenges, particularly technical limitations of AI models (50% of studies) and reliability concerns (28,3%), while also highlighting emerging trends such as the integration of AI techniques (26.1%), optimization of LLMs and prompt engineering (26.1%), and automation and scalability (21.7%).

These findings underscore the need for future research focused on: (1) developing robust validation frameworks for AI-generated tests, (2) creating guidelines for effective technique selection based on testing context, and (3) establishing best practices for integrating AI testing tools into industrial workflows while addressing ethical and reliability concerns. The study serves as both a reference point for researchers and a practical guide for practitioners adopting AI in test automation.

8 References

[15] Vahid Garousi et al. "AI-powered test automation tools: A systematic review and empirical evaluation". In: *arXiv preprint arXiv:2409.00411* (2024). Version 2, last revised 26 Feb 2025. DOI: 10.48550/arXiv.2409.00411. arXiv: 2409.00411 [cs.SE].

[17] Muhammad Abid Jamil et al. "Software testing techniques: A literature review". In: *2016 6th international conference on information and communication technology for the Muslim world (ICT4M)*. IEEE. 2016, pp. 177–182.

[21] Barbara Kitchenham. *Procedures for Performing Systematic Reviews*. Tech. rep. TR/SE-0401 and NICTA Technical Report 0400011T.1. Keele, UK and Sydney, Australia: Joint Technical Report, Keele University and National ICT Australia Ltd., July 2004. URL: https://www.inf.ufsc.br/~aldo.vw/kitchenham.pdf.

[22] Barbara Kitchenham, David Budgen, and Pearl Brereton. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*. Tech. rep. EBSE-2007-01. Version 2.3. EBSE Technical Report, Keele University and Durham University, July 2007. URL: https://www.cs.ubc.ca/~murphy/SE-Reading-Group/papers/Guidelines-SLR.pdf.

[38] Filippo Ricca, Alessandro Marchetto, and Andrea Stocco. "AI-based Test Automation: A Grey Literature Analysis". In: *2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*. 2021, pp. 263–270. DOI: 10.1109/ICSTW52544.2021.00051.

9 Primary Studies

[1] M. Almutawa, Q. Ghabrah, and M. Canini. "Towards LLM-Assisted System Testing for Microservices". In: *Proceedings of the 2024 IEEE 44th International Conference on Distributed Computing Systems Workshops (ICDCSW '24)*. IEEE, Sept. 4, 2024, pp. 29–34. DOI: 10.1109/ICDCSW63686.2024.00011. URL: https://doi.org/10.1109/ICDCSW63686.2024.00011.

[2] Nadia Alshahwan et al. "Automated Unit Test Improvement using Large Language Models at Meta". In: *Companion Proceedings of the 32nd ACM International Conference on the Foundations of Software Engineering (FSE '24 Companion)*. ACM, 2024, pp. 185–196. DOI: 10.1145/3663529.3663839. URL: https://doi.org/10.1145/3663529.3663839.

[3] Anonymous. "Comparing the Adaptability of a Genetic Algorithm and an LLM-Based Framework for Automated Software Test Data Generation: In the Context of Web Applications". In: *Proceedings of the 2024 IEEE 3rd International Conference on Data, Decision and Systems (ICDDS '24)*. IEEE, Mar. 12, 2025, pp. 1–6. DOI: 10.1109/ICDDS62937.2024.10910287. URL: https://doi.org/10.1109/ICDDS62937.2024.10910287.

[4] A. Arcuri et al. "Building an Open-Source System Test Generation Tool: Lessons Learned and Empirical Analyses with EvoMaster". In: *Software Quality Journal* 31.3 (2023), pp. 947–990. DOI: 10.1007/s11219-023-09620-w. URL: https://doi.org/10.1007/s11219-023-09620-w.

[5] S. Bhatia et al. "Unit Test Generation using Generative AI: A Comparative Performance Analysis of Autogeneration Tools". In: *Proceedings of the 2024 International Workshop on Large Language Models for Code (LLM4Code '24)*. ACM, 2024, pp. 54–61. DOI: 10.1145/3643795.3648396. URL: https://doi.org/10.1145/3643795.3648396.

[6] A. Chaudhuri et al. "C-Testing of AI Accelerators". In: *Proceedings of the 2020 IEEE 29th Asian Test Symposium (ATS '20)*. IEEE, Dec. 2020, pp. 1–6. DOI: 10.1109/ATS49688.2020.9301581. URL: https://doi.org/10.1109/ATS49688.2020.9301581.

[7] M. Dahiya et al. "Tracing Feature Tests to Textual Requirements". In: *Proceedings of the 2024 IEEE International Conference on Information Reuse and Integration for Data Science (IRI '24)*. IEEE, Oct. 8, 2024, pp. 120–125. DOI: 10.1109/IRI62200.2024.00035. URL: https://doi.org/10.1109/IRI62200.2024.00035.

[8] G. De Vito et al. "AGORA: An Approach for Generating Acceptance Test Cases from Use Cases". In: *Proceedings of the 2024 50th Euromicro Conference on Software Engineering and Advanced Applications (SEAA '24)*. IEEE, Dec. 27, 2024, pp. 126–133. DOI: 10.1109/SEAA64295.2024.00027. URL: https://doi.org/10.1109/SEAA64295.2024.00027.

[9] Y. Deng et al. "LLM-TG: Towards Automated Test Case Generation for Processors Using Large Language Models". In: *Proceedings of the 2024 IEEE 42nd International Conference on Computer Design (ICCD '24)*. IEEE, Jan. 2, 2025, pp. 389–396. DOI: 10.1109/ICCD63220.2024.00066. URL: https://doi.org/10.1109/ICCD63220.2024.00066.

[10] H. Ebadi et al. "Efficient and Effective Generation of Test Cases for Pedestrian Detection - Search-based Software Testing of Baidu Apollo in SVL". In: *Proceedings of the 2021 IEEE International Conference on Artificial Intelligence Testing (AITest '21)*. IEEE, Oct. 2021, pp. 103–110. DOI: 10.1109/AITEST52744.2021.00030. URL: https://doi.org/10.1109/AITEST52744.2021.00030.

[11] R. Feldt et al. "Towards Autonomous Testing Agents via Conversational Large Language Models". In: *Proceedings of the 2023 38th IEEE/ACM International Conference on Automated Software Engineering (ASE '23)*. IEEE, Nov. 8, 2023, pp. 1688–1693. DOI: 10.1109/ASE56229.2023.00148. URL: https://doi.org/10.1109/ASE56229.2023.00148.

[12] S. Feng et al. "Enabling Cost-Effective UI Automation Testing with Retrieval-Based LLMs: A Case Study in WeChat". In: *Proceedings of the 2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*. IEEE, Nov. 29, 2024, pp. 1973–1978. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10765004.

[13] A. Gamal et al. "Owl Eye: An AI-Driven Visual Testing Tool". In: *Proceedings of the 2023 5th Novel Intelligent and Leading Emerging Sciences Conference (NILES '23)*. IEEE, Nov. 1, 2023, pp. 312–315. DOI: 10.1109/NILES59815.2023.10296575. URL: https://doi.org/10.1109/NILES59815.2023.10296575.

[14] A. Garg and D. Sharma. "Generative AI for Software Test Modelling with a Focus on ERP Software". In: *Proceedings of the 2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT '23)*. IEEE, Mar. 20, 2024, pp. 187–193. DOI: 10.1109/ICAICCIT60255.2023.10466102. URL: https://doi.org/10.1109/ICAICCIT60255.2023.10466102.

[16] A. Guo et al. "SoVAR: Building Generalizable Scenarios from Accident Reports for Autonomous Driving Testing". In: *Proceedings of the 2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*. IEEE,

Nov. 29, 2024, pp. 268–280. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10764850.

[18] Z. Jiang et al. "Towards Understanding the Effectiveness of Large Language Models on Directed Test Input Generation". In: *Proceedings of the 2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*. IEEE, Nov. 29, 2024, pp. 1408–1420. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10765050.

[19] M. Jiri, B. Emese, and P. Medlen. "Leveraging Large Language Models for Python Unit Test". In: *Proceedings of the 2024 IEEE International Conference on Artificial Intelligence Testing (AITest '24)*. IEEE, Sept. 25, 2024, pp. 95–100. DOI: 10.1109/AITest62860.2024.00020. URL: https://doi.org/10.1109/AITest62860.2024.00020.

[20] Shanthi Karpurapu et al. "Comprehensive Evaluation and Insights Into the Use of Large Language Models in the Automation of Behavior-Driven Development Acceptance Test Formulation". In: *IEEE Access* 12 (2024), pp. 58715–58721. DOI: 10.1109/ACCESS.2024.3391815. URL: https://doi.org/10.1109/ACCESS.2024.3391815.

[23] C. Landing et al. "Cluster-Based Parallel Testing Using Semantic Analysis". In: *Proceedings of the 2020 IEEE International Conference on Artificial Intelligence Testing (AITest '20)*. IEEE, Aug. 2020, pp. 99–106. DOI: 10.1109/AITEST49225.2020.00022. URL: https://doi.org/10.1109/AITEST49225.2020.00022.

[24] T. Le et al. "KAT: Dependency-Aware Automated API Testing with Large Language Models". In: *Proceedings of the 2024 IEEE Conference on Software Testing, Verification and Validation (ICST '24)*. IEEE, Aug. 27, 2024, pp. 82–92. DOI: 10.1109/ICST60714.2024.00017. URL: https://doi.org/10.1109/ICST60714.2024.00017.

[25] Y. Liu. "Natural Language Processing Technology Based on Artificial Intelligence in Software Testing". In: *Proceedings of the 2024 3rd International Conference on Artificial Intelligence and Computer Information Technology (AICIT '24)*. IEEE, Oct. 30, 2024, pp. 1–6. DOI: 10.1109/AICIT62434.2024.10730603. URL: https://doi.org/10.1109/AICIT62434.2024.10730603.

[26] O. Loubiri and S. Maag. "Automated Web Testing Using Machine Learning and Containerization". In: *Proceedings of the 2022 26th International Conference on Circuits, Systems, Communications and Computers (CSCC '22)*. IEEE, Jan. 2023, pp. 113–121. DOI: 10.1109/CSCC55931.2022.00029. URL: https://doi.org/10.1109/CSCC55931.2022.00029.

[27] L. Naimi et al. "A New Approach for Automatic Test Case Generation from Use Case Diagram Using LLMs and Prompt Engineering". In: *Proceedings of the 2024 International Conference on Circuit, Systems and Communication (ICCSC '24)*. IEEE, Aug. 6, 2024, pp. 1–5. DOI: 10.1109/ICCSC62074.2024.10616548. URL: https://doi.org/10.1109/ICCSC62074.2024.10616548.

[28] N. Neelofar and A. Aleti. "Towards Reliable AI: Adequacy Metrics for Ensuring the Quality of System-Level Testing of Autonomous Vehicles". In: *Proceedings of the 2024 IEEE/ACM 46th International Conference on Software Engineering (ICSE '24)*. IEEE, June 14, 2024, pp. 816–827. DOI: 10.1145/3597503.3623314. URL: https://doi.org/10.1145/3597503.3623314.

[29] M. M. Olmez and E. Gehringer. "Automation of Test Skeletons Within Test-Driven Development Projects". In: *Proceedings of the 2024 IEEE/ACM Conference on Software Engineering Education and Training (CSEET '24)*. IEEE, 2024, pp. 1–10. DOI: 10.1109/CSEET62301.2024.10663016. URL: https://doi.org/10.1109/CSEET62301.2024.10663016.

[30] M. Olsthoorn. "More Effective Test Case Generation with Multiple Tribes of AI". In: *Proceedings of the 2022 IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion '22)*. ACM, June 2022, pp. 286–290. DOI: 10.1145/3510454.3517066. URL: https://doi.org/10.1145/3510454.3517066.

[31] W. C. Ouédraogo et al. "LLMs and Prompting for Unit Test Generation: A Large-Scale Evaluation". In: *Proceedings of the 2024 39th IEEE/ACM International Conference on Automated Software Engineering (ASE '24)*. IEEE, Nov. 29, 2024, pp. 2464–2465. URL: https://ieeexplore.ieee.org/stamp/stamp.jsp?arnumber=10764990.

[32] C. Paduraru, M. Paduraru, and A. Stefanescu. "RiverFuzzRL: An Open-Source Tool to Experiment with Reinforcement Learning for Fuzzing". In: *Proceedings of the 2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST '21)*. IEEE, May 2021, pp. 430–435. DOI: 10.1109/ICST49551.2021.00055. URL: https://doi.org/10.1109/ICST49551.2021.00055.

[33] C. Paduraru, M. Paduraru, and A. Stefanescu. "RiverGame: A Game Testing Tool Using Artificial Intelligence". In: *Proceedings of the 2022 IEEE Conference on Software Testing, Verification and Validation (ICST '22)*. IEEE, June 2022, pp. 422–432. DOI: 10.1109/ICST53961.2022.00048. URL: https://doi.org/10.1109/ICST53961.2022.00048.

[34] S. Pangavhane et al. "AI-Augmented Software Development: Boosting Efficiency and Quality". In: *Proceedings of the 2024 International Conference on Decision Aid Sciences and Applications (DASA '24)*. IEEE, Jan. 17, 2025, pp. 1–5. DOI: 10.1109/DASA63652.2024.10836523. URL: https://doi.org/10.1109/DASA63652.2024.10836523.

[35] N. Petrovic et al. "LLM-Driven Testing for Autonomous Driving Scenarios". In: *Proceedings of the 2024 2nd International Conference on Foundation and Large Language Models (FLLM '24)*. IEEE, Jan. 28, 2025, pp. 173–178. DOI: 10.1109/FLLM63129.2024.10852505. URL: https://doi.org/10.1109/FLLM63129.2024.10852505.

[36] R. K. C. Ragel and F. F. Balahadia. "Visual Test Framework: Enhancing Software Test Automation with Visual Artificial Intelligence and Behavioral Driven Development". In: *Proceedings of the 2023 IEEE 15th International Conference on Humanoid, Nanotechnology, Information Technology, Communication and Control, Environment, and Management (HNICEM '23)*. IEEE, July 15, 2024, pp. 1–5. DOI: 10.1109/HNICEM60674.2023.10589222. URL: https://doi.org/10.1109/HNICEM60674.2023.10589222.

[37] N. Retzlaff. "AI Integrated ST Modern System for Designing Automated Standard Affirmation System". In: *Proceedings of the 2024 4th International Conference on Advance Computing and Innovative Technologies in Engineering (ICACITE '24)*. IEEE, Aug. 9, 2024, pp. 1386–1391. DOI: 10.1109/ICACITE60783.2024.10616416. URL: https://doi.org/10.1109/ICACITE60783.2024.10616416.

[39] R. Santos et al. "Are We Testing or Being Tested? Exploring the Practical Applications of Large Language Models in Software Testing". In: *Proceedings of the 2024 IEEE Conference on Software Testing, Verification and Validation (ICST '24)*. IEEE, Aug. 27, 2024, pp. 353–360. DOI: 10.1109/ICST60714.2024.00039. URL: https://doi.org/10.1109/ICST60714.2024.00039.

[40] Max Schafer et al. "An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation". In: *IEEE Transactions on Software Engineering* 50.1 (2024), pp. 85–105. DOI: 10.1109/TSE.2023.3334955. URL: https://doi.org/10.1109/TSE.2023.3334955.

[41] Denis Schwachhofer et al. "Large Language Model-Based Optimization for System-Level Test Program Generation". In: *Proceedings of the 2024 IEEE International Symposium on Defect and Fault Tolerance in VLSI and Nanotechnology Systems (DFT '24)*. IEEE, 2024. DOI: 10.1109/DFT63277.2024.10753556. URL: https://doi.org/10.1109/DFT63277.2024.10753556.

[42] X. Shen et al. "EcoDialTest: Adaptive Mutation Schedule for Automated Dialogue Systems Testing". In: *Proceedings of the 2023 IEEE International Conference on Software Analysis, Evolution and Reengineering (SANER '23)*. IEEE, May 15, 2023, pp. 933–939. DOI: 10.1109/SANER56733.2023.00113. URL: https://doi.org/10.1109/SANER56733.2023.00113.

[43] Y. Shi et al. "Restricted Natural Language and Model-Based Adaptive Test Generation for Autonomous Driving". In: *Proceedings of the 24th International Conference on Model-Driven Engineering Languages and Systems (MODELS '21)*. IEEE, 2021, pp. 101–111. DOI: 10.1109/MODELS50736.2021.00019. URL: https://doi.org/10.1109/MODELS50736.2021.00019.

[44] Gaadha Sudheerbabu et al. "Iterative Optimization of Hyperparameter-based Metamorphic Transformations". In: *Proceedings of the 2024 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW '24)*. IEEE, 2024, pp. 13–20. DOI: 10.1109/ICSTW60967.2024.00016. URL: https://doi.org/10.1109/ICSTW60967.2024.00016.

[45] A. Tuzmen. "Use of Generative Artificial Intelligence in the Education of Software Verification and Validation". In: *Proceedings of the 2024 IEEE Frontiers in Education Conference (FIE '24)*. IEEE, Feb. 26, 2025, pp. 1–6. DOI: 10.1109/FIE61694.2024.10893333. URL: https://doi.org/10.1109/FIE61694.2024.10893333.

[46] Zejun Wang et al. "HITS: High-coverage LLM-based Unit Test Generation via Method Slicing". In: *Proceedings of the 2024 39th ACM/IEEE International Conference on Automated Software Engineering (ASE '24)*. ACM, 2024, pp. 1258–1268. DOI: 10.1145/3691620.3695501. URL: https://doi.org/10.1145/3691620.3695501.

[47] G. Yi et al. "Exploring the Capability of ChatGPT in Test Generation". In: *Proceedings of the 2023 IEEE 23rd International Conference on Software Quality, Reliability, and Security Companion (QRS-C '23)*. IEEE, July 8, 2023, pp. 72–80. DOI: 10.1109/QRS-C60940.2023.00013. URL: https://doi.org/10.1109/QRS-C60940.2023.00013.

[48] H. Yin, H. Mohammed, and S. Boyapati. "Leveraging Pre-Trained Large Language Models (LLMs) for On-Premises Comprehensive Automated Test Case Generation: An Empirical Study". In: *Proceedings of the 2024 9th International Conference on Intelligent Informatics and Biomedical Sciences (ICIIBMS '24)*. IEEE, Dec. 17, 2024, pp. 597–607. DOI: 10.1109/ICIIBMS62405.2024.10792720. URL: https://doi.org/10.1109/ICIIBMS62405.2024.10792720.

[49] S. Yu et al. "LLM for Test Script Generation and Migration: Challenges, Capabilities, and Opportunities". In: *Proceedings of the 2023 IEEE International Conference on Software Quality, Reliability and Security (QRS '23)*. IEEE, Nov. 15, 2023, pp. 206–217. DOI: 10.1109/QRS60937.2023.00029. URL: https://doi.org/10.1109/QRS60937.2023.00029.

[50] C. Zhang et al. "Construction of GUI Elements Recognition Model for AI Testing Based on Deep Learning". In: *Proceedings of the 2021 8th International Conference on Dependable Systems and Their Applications (DSA '21)*. IEEE, Dec. 2021, pp. 508–515. DOI: 10.1109/DSA52907.2021.00075. URL: https://doi.org/10.1109/DSA52907.2021.00075.

[51] Y. Zhao et al. "Avgust: Automating Usage-Based Test Generation from Videos of App Executions". In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*

(ESEC/FSE '22). ACM, Nov. 14, 2022, pp. 421–433. DOI: 10.1145/3540250.3549134. URL: https://doi.org/10.1145/3540250.3549134.

Appendix A Complementary tables

This appendix aims to support the article with information about the primary studies and research questions.

Table 1: Primary Studies Articles

Primary Study	Article Title
PS01	[6]
PS02	[23]
PS03	[10]
PS04	[32]
PS05	[50]
PS06	[43]
PS07	[30]
PS08	[26]
PS09	[33]
PS10	[51]
PS11	[14]
PS12	[11]
PS13	[36]
PS14	[13]
PS15	[42]
PS16	[47]
PS17	[49]
PS18	[4]
PS19	[39]
PS20	[25]
PS21	[19]
PS22	[9]
PS23	[31]
PS24	[34]
PS25	[27]
PS26	[3]
PS27	[37]
PS28	[7]
PS29	[1]
PS30	[48]
PS31	[45]
PS32	[35]
PS33	[12]
PS34	[18]
PS35	[8]
PS36	[28]
PS37	[16]
PS38	[24]
PS39	[29]
PS40	[5]
PS41	[46]
PS42	[20]
PS43	[44]
PS44	[2]
PS45	[41]
PS46	[40]

Table 2: Distribution of Studies by AI Technique Applied to Test (Q1)

AI Technique	Studies (PS)
Large Language Models (LLMs)	PS11, PS12, PS16, PS17, PS19, PS20, PS21, PS22, PS23, PS24, PS25, PS26, PS28, PS29, PS30, PS31, PS32, PS33, PS34, PS35, PS37, PS38, PS39, PS40, PS41, PS42, PS44, PS45, PS46
Traditional Machine Learning (ML)	PS07, PS08, PS24, PS27, PS28, PS33, PS36
Evolutionary Algorithms (EA)	PS03, PS07, PS18, PS26, PS43
Reinforcement Learning (RL)	PS04, PS06, PS09, PS15, PS27
Deep Learning (DL)	PS05, PS09, PS10, PS13
Natural Language Processing (NLP)	PS02, PS16, PS27
Generative AI (non-LLM)	PS43
No Response	PS01, PS14

Table 3: Distribution of Studies by Categories of Approaches in Automated Test Generation (Q2)

AI Evaluation Criterion	Studies (PS)
Practical Deployment and Usability	PS09, PS10, PS11, PS13, PS14, PS16, PS17, PS19, PS21, PS25, PS27, PS33, PS35, PS39, PS42, PS44
Superior Coverage Compared to Traditional Methods	PS03, PS07, PS15, PS16, PS20, PS21 PS22, PS24, PS31, PS37, PS38, PS41, PS44
Efficiency Improvements	PS01, PS02, PS04, PS07, PS08, PS15, PS18, PS24, PS27, PS28, PS36
Comparison Among AI Techniques	PS23, PS26, PS29, PS30, PS32, PS34, PS40, PS46
No Response	PS05, PS06, PS12, PS43, PS45

Table 4: Distribution of Studies by Benefit Categories in AI-Based Test Generation (Q3)

AI Benefit Category	Studies (PS)
Increased Test Coverage and Effectiveness	PS03, PS04, PS06, PS07, PS11, PS22, PS23, PS24, PS26, PS30, PS34, PS38, PS40, PS41, PS44, PS46
Improved Efficiency and Resource Optimization	PS02, PS04, PS05, PS07, PS13, PS15, PS21, PS22, PS24, PS25, PS26, PS30, PS32, PS35, PS42, PS45
Automation and Reduced Manual Effort	PS08, PS09, PS10, PS11, PS13, PS17, PS20, PS27, PS39
Adaptability to Complex Systems	PS04, PS08
Enhanced Usability and Collaboration	PS09, PS12, PS16, PS17, PS19, PS23, PS29, PS31, PS39, PS40, PS44, PS46
Real-World Deployment and Integration	PS18, PS19, PS29, PS32, PS33, PS35, PS37, PS42
Improved Defect Detection and Accuracy	PS10, PS14, PS15, PS27, PS33, PS37, PS38, PS43
Learning and Continuous Improvement	PS12, PS41, PS43, PS45
No Response	PS01, PS28, PS36

Table 5: Distribution of Studies by Challenge Categories in AI-Based Test Generation (Q4)

AI Challenge Category	Studies (PS)
Technical and Model Limitations	PS04, PS05, PS11, PS12, PS14, PS16, PS17, PS20, PS22, PS23, PS24, PS25, PS26, PS32, PS34, PS35, PS36, PS40, PS41, PS43, PS44, PS45, PS46
Reliability and Consistency	PS03, PS08, PS09, PS16, PS18, PS21, PS28, PS31, PS38, PS40, PS41, PS44, PS45
Validation and Ethical Concerns	PS24, PS27, PS30, PS36, PS38, PS42, PS43, PS46
Human Dependency and Usability	PS10, PS13, PS14, PS19, PS21, PS22, PS29, PS39
Scalability and Generalization	PS07, PS08, PS15, PS25, PS35, PS37
Integration and Tooling	PS07, PS17, PS18, PS27, PS29, PS33
Data Quality and Dependency	PS05, PS10, PS31, PS37, PS42
Resource and Infrastructure Constraints	PS02, PS12, PS20, PS33, PS39
Interpretability and Transparency	PS06
No Response	PS01

Table 6: Distribution of Studies by Categories of Emerging Trends and Future Directions in Automated Test Generation

Emerging Trend Category	Studies (PS)
Advanced Learning and Adaptation	PS03, PS08, PS15, PS22, PS34, PS41, PS43, PS45
Automation and Scalability	PS06, PS07, PS08, PS12, PS13, PS25, PS27, PS28, PS29, PS35
Domain-Specific Applications	PS04, PS09, PS11, PS14, PS17, PS20, PS32, PS33
Ethics, Transparency, and Explainability	PS19, PS24, PS27, PS44
Human-AI Collaboration	PS12, PS17, PS19, PS29, PS31, PS42
Integration of AI Techniques	PS02, PS03, PS04, PS07, PS09, PS11, PS23, PS26, PS39, PS40, PS45, PS46
Optimization of LLMs and Prompt Engineering	PS10, PS14, PS20, PS21, PS22, PS23, PS25, PS30, PS32, PS34, PS35, PS41
Validation and Quality Metrics	PS16, PS26, PS31, PS36, PS39, PS42, PS44, PS46
No Response	PS01, PS05, PS18, PS37, PS38