International Journal of Engineering and Applied Computing
Vol. 1, pp. 17-38, July 2026

REVIEW ARTICLE

A Systematic Literature Review on the Applications of Large Language Models in Software Requirements Engineering

Shafaque Aziz[1], Khushboo[2], Saim Akhtar[3], Sameera Jawed[4]

[1]Department of Computer Science and Engineering, Government Engineering College, Samastipur, Bihar, India
E-mail: shafaque.phd@gmail.com
[2]Amity Institute of Information Technology, Amity University, Patna, Bihar, India
E-mail: net.khushboo@gmail.com
[3]Department of Computer Engineering, J.C. Bose University of Science and Technology, YMCA, Faridabad, Haryana, India
E-mail: saimakhtarr@gmail.com
[4]Department of Computer Science and Technology, Maharaja Agrasen Institute of Technology, Affiliated with Guru Gobind Singh Indraprastha University, New Delhi, India
E-mail: sameerazoya3@gmail.com

Corresponding author: shafaque.phd@gmail.com

Received: 12/05/2026 | **Revised:** 16/06/2026 | **Accepted:** 27/06/2026

Abstract

The rapid advancement of large language models (LLMs) has significantly influenced various domains, including software engineering, where their potential to transform traditional practices is increasingly recognized. This systematic literature review investigates the role of LLMs in software requirements engineering, aiming to synthesize existing research, identify key trends, and highlight gaps in the current knowledge. We examine the multifaceted applications of LLMs across several dimensions, such as requirements elicitation, analysis, and specification, as well as their broader implications in code generation, testing, and security. The methodology involves a rigorous selection and analysis of relevant studies, followed by a thematic categorization to provide a comprehensive overview of the field. Our findings reveal that LLMs demonstrate considerable promise in automating and improving the accuracy of requirements-related tasks, yet challenges such as model interpretability, ethical concerns, and integration with existing workflows remain unresolved. The review also underscores the growing emphasis on prompt engineering and fine-tuning techniques to adapt LLMs for domain-specific needs. While the adoption of LLMs in software requirements engineering is still evolving, the evidence suggests a paradigm shift toward more intelligent and efficient practices. We conclude by discussing future research directions, emphasizing the need for empirical validation, interdisciplinary collaboration, and the development of robust frameworks to address the limitations and risks associated with LLM deployment in this critical area of software engineering.

Keywords

Software Engineering, Requirements Engineering, Large Language Model, Systematic Literature Review, Requirements Prioritization.

1. INTRODUCTION

Software Requirements Engineering (SRE) has always been an integral part of software development since it provides a basis for specifying system functionalities and user needs. Classic methods of requirements engineering require many manual actions and are mostly connected to interviewing stakeholders, analyzing documents, and performing validation tasks that might be resource-demanding and ambiguous (Elghariani & Kama, 2016). The growing complexity of software systems along with the necessity for shorter development cycles requires the exploration of innovative ways to automate and improve the process of requirements engineering. LLMs represent a significant revolution in artificial intelligence due to their unique abilities in

natural language processing tasks like understanding, generating, and reasoning about the information contained in texts (Minaee *et al.*, 2024). Due to the training on large text corpora, LLMs become capable of performing such tasks as text summarization, code generation, etc. that makes it possible to apply these models in the software engineering domain. Specifically, LLMs may be useful in the process of SRE for automating requirement elicitation, refining the vague requirements, and detecting inconsistencies in early-stage documentation (Arora et al., 2024; Rodriguez-Cardenas, 2025). However, the use of LLMs in SRE is not devoid of difficulties. First of all, it is important to mention some gaps revealed in current researches like the absence of benchmark datasets for evaluating LLMs in tasks related to SRE, the problem of maintaining the interpretability of the model in case of using it in decision-making processes, the issue of ethics and the need to take into account sensitive stakeholder environment, etc. Furthermore, although LLMs can successfully work with plain text, they face difficulties when trying to work with domain-specific jargon, cross-reference requirements, and other forms of input besides text (Doris *et al.*, 2025; Page *et al.*, 2021).

Our motivation for the current research stems from the rising popularity of LLMs and their potential to benefit software engineering practice. Our objective is to perform a state-of-the-art review of the available literature on the matter and provide an exhaustive overview of the use of LLMs in the context of SRE. Moreover, we would like to highlight the latest trends in the field as well as areas which require further investigation. Such a literature review will prove to be valuable for practitioners who want to implement LLM-based solutions and for researchers who can utilize the insights to focus on the most critical aspects. Finally, this paper contributes to the growing body of literature dedicated to the impact of Artificial Intelligence (AI) on software engineering in all its aspects, ranging from positive effects to possible drawbacks of the use of LLMs.

The rest of this paper is structured as follows: in Section 2, we explain the methodology of our literature review and the procedure of studies selection and analysis. The results of our study are reported in Section 3 and organized in several subsections related to different aspects of the use of LLMs in software engineering such as SRE, code generation, testing and security, and usage in education and the challenges associated with it. In Section 4, implications of our findings are discussed. Finally, in Section 5, we wrap up the paper by thinking about where this field might go in the future and what's next for LLMs.

2. METHODOLOGY

2.1 Review Protocol

In order to conduct an unbiased and comprehensive search and obtain all available publications, the review was conducted via six reputable academic databases and search engines which were chosen due to their popularity in computer science and software engineering research. IEEE Xplore was prioritized as it is the source of an enormous number of high-impact journals and conferences in software engineering. Also, ACM Digital Library was selected because of its concentration on computer sciences and emerging applications of artificial intelligence. Web of Science and Scopus was chosen as sources providing broad coverage of various disciplines as well as an efficient citation tracking. Finally, ScienceDirect and SpringerLink provided peer-reviewed journals with emphasis on empirical studies in software engineering. In addition, we employed Google Scholar to find more grey literature and preprints. The combination of those search engines and databases was meant to ensure the diversity of retrieved studies. The search keywords were selected in such a way as to provide for a targeted search. The phrases "Large language model" and "LLM", as well as "software requirements engineering" and "SRE", were used in order to receive as narrow as possible result while avoiding missing any useful literature. In order to exclude irrelevant papers such as review articles, surveys, meta-analyses, etc., the search terms were adjusted for every database as described in the search methods section.

2.2 Research Dimensions

The following areas are included in research on the application of LLMs in the field of SRE. The first area under consideration is the analysis of the role of LLMs in the accomplishment of the fundamental steps of SRE: elicitation, specification, and verification of requirements. Further, it is essential to investigate the use of LLMs in generation and refactoring of code by transforming natural-language requirements into code that can be run. Moreover, it is necessary to analyze ways of employing LLMs in testing activities, including generating test

cases and defect prediction. Another crucial issue is the investigation of the potential of LLMs in identification and mitigation of risks and vulnerabilities in software requirements and code. In addition, one needs to conduct studies on optimization of LLMs for particular SRE activities through the application of certain techniques. It is also important to explore the possibilities of implementing LLMs in education in software engineering. One more key point to take into account is the analysis of ethical and technical challenges associated with applying LLMs in the field of SRE.

2.3 Inclusion and Exclusion Criteria

In order to select relevant studies for analysis, we looked for studies that discussed the use of Large Language Models (LLMs) in SRE or related fields. These papers should also have been peer-reviewed and presented at scientific conferences or journals, and contain empirical results or novel contributions. It is crucial that these studies were published in English. To ensure that the selected studies present an evolving state-of-the-art, we did not restrict the selection based on publication year. At the same time, we excluded the studies which were not actually devoted to the topic of LLMs in SRE, as well as opinion pieces and theoretical considerations. We also decided to exclude studies that repeated conclusions of other selected studies, as well as the studies that we were unable to get hold of.

2.4 Study Selection Process

We started by searching for records and found 1,398 of them. But when we looked closer, we realized that 548 of those were duplicates, so we removed them. Then, the titles and abstracts of the remaining studies were assessed, and 545 records out of them were excluded due to irrelevance to the subject of the review. 300 articles remained after this step. The assessment of the whole text of the studies led to the exclusion of 180 studies as not corresponding to the inclusion criteria of the review, such as insufficient focus on LLMs or SRE. Thus, 86 studies qualified to be included in the review. For the illustration of the selection process of the records, one can refer to the flow chart presented in Fig 1. The biases that may influence the results of the selection process include exclusion of non-English publications and using of particular database search algorithms that might miss relevant studies due to unconventional indexing. To reduce these biases, snowballing was conducted in both directions for the important papers in order to find additional sources. However, fast development of LLM field requires taking into account that some new findings may not be covered by the existing literature.

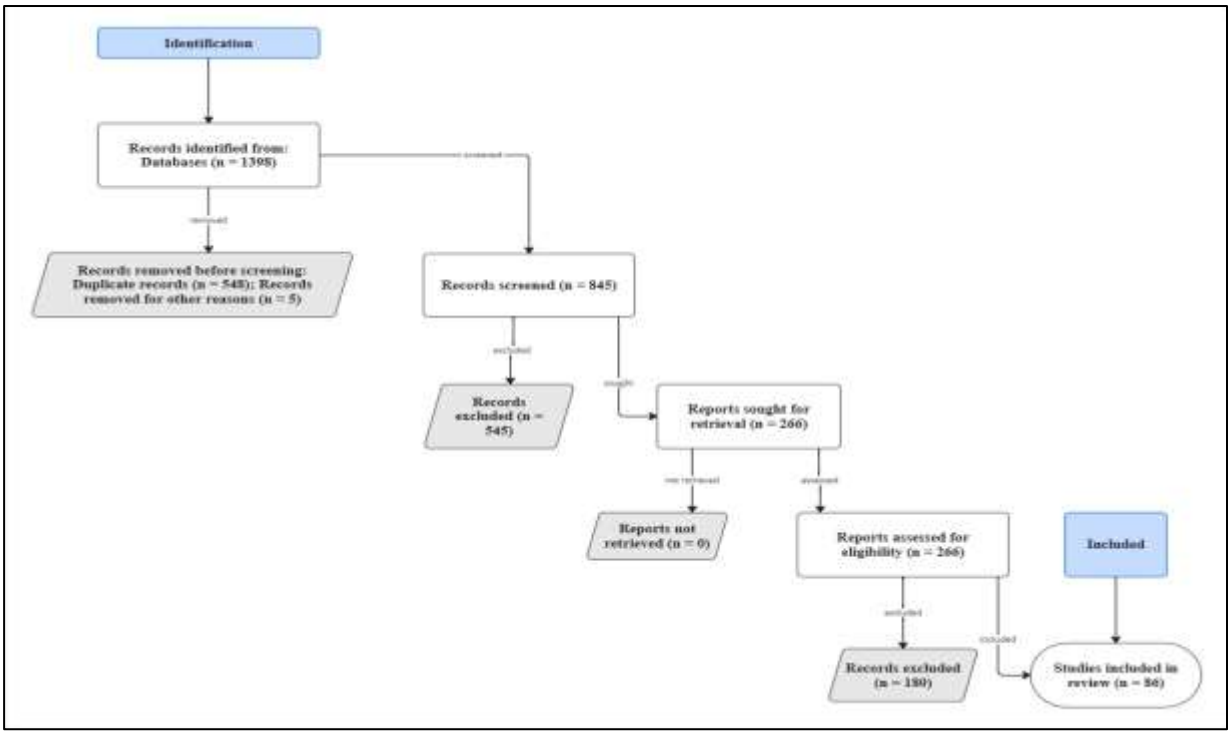

Fig. 1: Flowchart of the study selection process

3. RESULTS

3. 1 Research Trends

It is obvious that scientific activity in the field has grown significantly in recent years, see Figure 2. The analysis of the dates when the articles have been published allows to conclude that as much as 86% of the sources were published within 2023 and 2026. This timeframe coincides with the rise in popularity of transformer-based architectures and models such as GPT-4 and their incredible efficiency in natural language processing tasks. Since then, the field has gone far beyond the exploratory studies conducted in the late 2010s. It is currently growing very rapidly and its development is characterized by an exponential increase in the number of studies. Studies conducted from 2016 to 2022 offered many insights about the potential of LLMs in analyzing requirements, however, the interest has grown significantly only in 2023. Specifically, the volume of publications increased by thirteen times compared to the previous year. This abrupt increase demonstrates the growing awareness of LLMs among developers and their perception of the models as a disruptive innovation capable of solving some persistent issues in the requirements engineering process. Furthermore, the fact that the growth continued until 2024 and 2025 indicates that a revolution in requirement engineering, analysis, and management is taking place. Fig. 2 illustrates the rapid increase in publications on LLMs in Software Requirements Engineering, particularly after 2023, reflecting the growing research interest in this field. It is noteworthy that research on code generation and software testing accounts for almost 40% of the reviewed studies. The distribution of studies by application area is shown in Fig. 3.

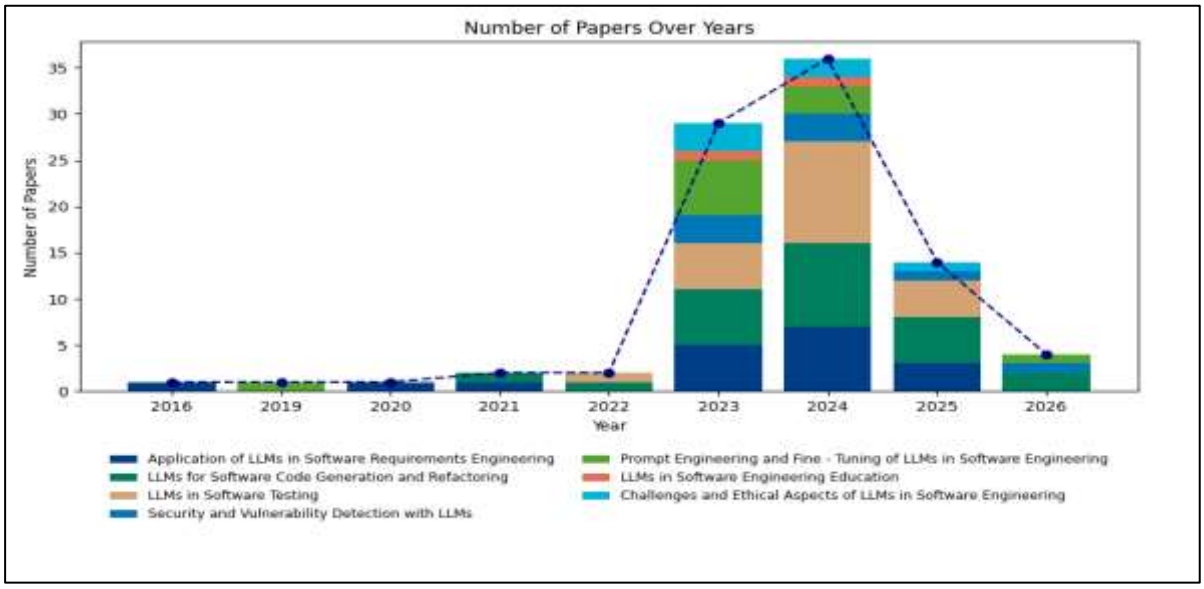

Fig. 2: Research trends in the domain of LLMs in SRE

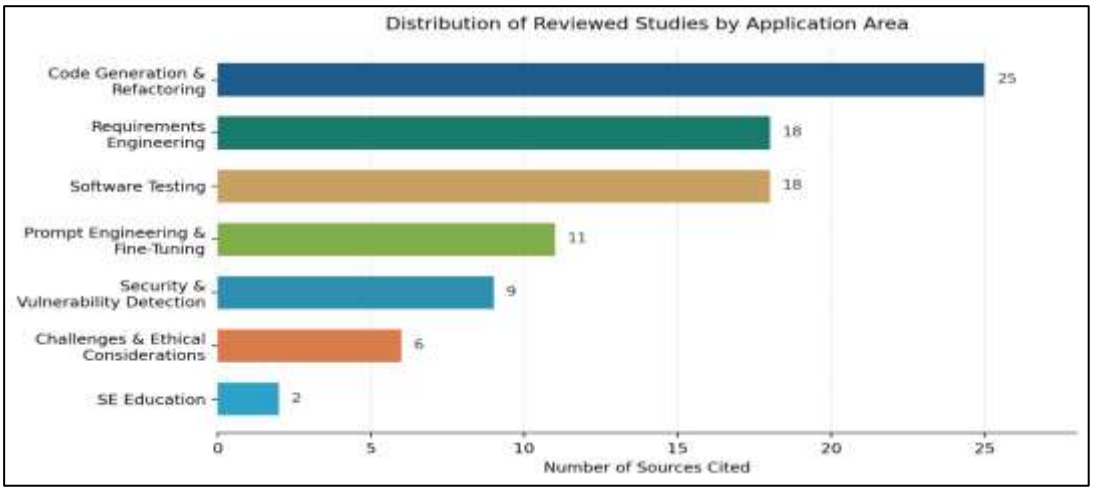

Fig. 3: Distribution of reviewed studies by application area.

As the development and execution of any plan follow each other naturally, such a high percentage is quite reasonable since LLMs may serve as tools of turning plans into actual code. Another vital issue that cannot be ignored is security. Despite the fact that there are fewer studies on this topic, they become more sophisticated – today researchers focus not only on finding bugs in the code but also on its analysis in the planning stage. One can also identify a new research area, which is known as prompt engineering and focuses on improving the performance of LLMs in various situations. Finally, despite the fact that education is not one of the priorities of research right now, one should admit that it is a positive tendency since training future engineers in this regard will play a great role in their professional activity.

3.2 Applications of LLMs in Software Requirements Engineering

The application of LLMs to SRE could turn out to be a revolution. LLMs could prove useful at various stages in the process of SRE. Our research examined numerous papers and classified them into six broad categories, which have been highlighted in Table 1 below. A breakdown of LLM Applications in SRE by specific task is shown in Figure 4. Each category explores particular difficulties and possibilities of SRE. All these categories revolve around exploring and implementing innovative solutions for improving SRE. By means of LLMs, we will be able to overcome several difficulties inherent in SRE.

Table 1: Taxonomy of LLM Applications in Software Requirements Engineering

Application Area	Specific Task	Technique/Focus	Sources
Requirements Elicitation and Analysis	Elicitation Support	AI-assisted questioning and refinement	(Ronanki *et al.*, 2023), (Ezzini *et al.*, 2023)
	Completeness Enhancement	Improving natural-language requirements	(Luitel *et al.*, 2023)
	Bias and Fairness Analysis	Gender bias in requirements tasks	(Treude & Hata, 2023)
Requirements Specification and Formalization	Natural-language to Formal Specs	LLM-based formalization	(Beg *et al.*, 2025)
	Specification Evaluation	Empirical assessment of LLM-generated specifications	(Krishna *et al.*, 2024)
Automated Requirements-to-Code	Code Generation from Requirements	End-to-end requirements-to-code pipelines	(Wei, 2024; Han *et al.*, 2024)
	Traceability and Prompt Engineering	Prompt strategies for traceability	(Rodriguez *et al.*, 2023)
Generative AI and Prompting in RE	Prompt Design and Patterns	Best practices for RE prompts	(Ronanki *et al.*, 2024)
	Generative AI for RE Tasks	General role of LLMs in RE	(Arora *et al.*, 2024; Dalpiaz & Niu, 2020)
Agile and Multi-Agent RE	User Story Improvement	LLM-based quality enhancement	(Zhang *et al.*, 2024)
	Multi-Agent Agile Development	LLM agents for agile RE and estimation	(Manish, 2024; Bui *et al.*, 2025)
Challenges and Case Studies	AI-Intense Systems RE Challenges	Unique challenges in AI-driven RE	(Heyn *et al.*, 2021)
	Agile RE Practices	Empirical case studies	(Bjarnason *et al.*, 2016)
General RE with AI	AI in Requirements Engineering	Historical and future perspectives	(Son, 2025; Dalpiaz & Niu, 2020)

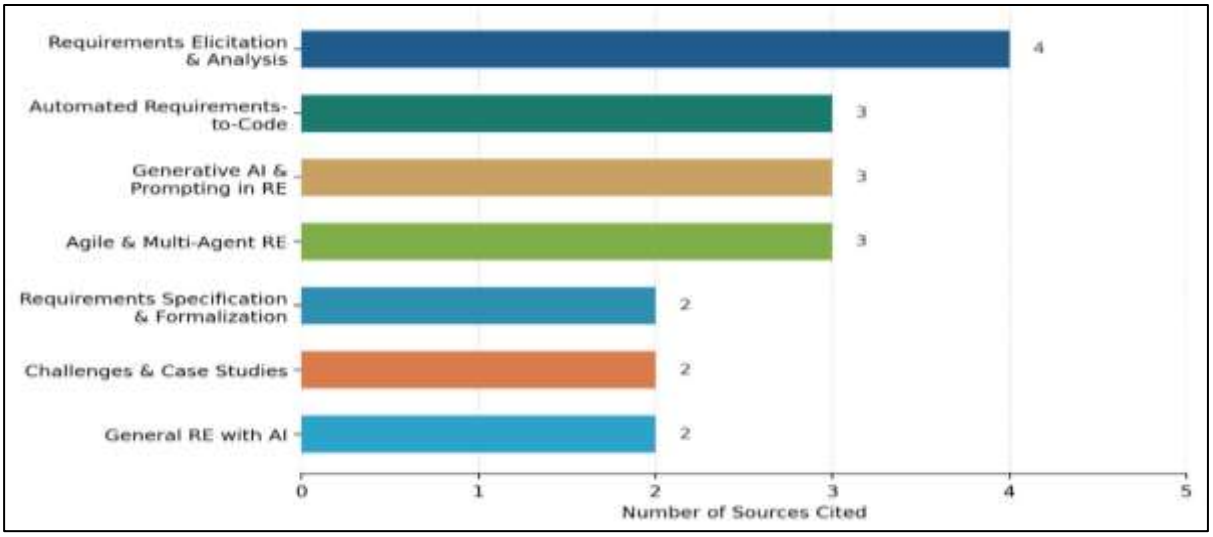

Fig. 4: Breakdown of LLM applications in SRE by specific task

In addition to helping to elicit the necessary information regarding what the user wants, LLMs can make ambiguous user requests more precise through asking questions and expanding on an inadequate description of the request. However, some studies indicate that LLMs are capable of generating biased outcomes when the task requires the involvement of gender-related issues. This should be kept in mind while testing the validity of the generated outcome in practice. LLMs can convert informal requirements into formal specifications. For instance, there is a study showing that LLMs are capable of doing this transformation rather effectively. Another study was concerned with the problem of verifying whether the output provided by the model is correct. In addition, there are several pieces of research indicating that LLMs can translate requirements into code. It allows connecting the high-level idea of what needs to be done and its practical implementation. The connection mentioned above is important due to the fact that it bridges the gap between user needs and their implementation.

Emerging areas include multi-agent LLM frameworks for agile development (Manish, 2024), in which autonomous agents collaborate in the process of requirements prioritization, as well as prompt engineering studies (Rodriguez *et al.*, 2023) optimizing traceability tasks. Notably, (Heyn et al., 2021) identifies the unique challenges for AI-intensive systems such as changing requirements and ethical concerns that require adaptive LLM solutions. Some research, including (Son, 2025) and (Dalpiaz & Niu, 2020) focuses on how AI technology was applied in the domain of SRE in the past and how it may be employed in the future. According to them, LLMs are the next step on the path to more intelligent RE tools. Categorization of such type indicates the variety of applications of LLMs but also points out the directions requiring further elaboration such as dealing with multi-modal requirements and longitudinal impacts of LLMs.

3.3 LLMs for Software Code Generation and Refactoring

The use of LLMs for generating and improving codes has proven to be one of the most vibrant fields of research, with research showing that they help to fill the gap between high-level requirements and their implementation. As shown in Table 2, we have categorized the included research studies into three main fields of applications, all of which tackle different issues. A breakdown of LLM applications in code generation and refactoring by task is depicted in Fig. 5.

Code generation is another field that relies heavily on the power of prompt engineering. Studies like (Li *et al.*, 2025) propose the use of structured chain-of-thought prompting to increase the logical consistency of generated code. Works like (Jain *et al.*, 2022; Austin *et al.*, 2021) investigate the possibility of large-scale program synthesis and demonstrate that LLMs are capable of successfully combining learned patterns and algorithmic reasoning skills. Test-driven approaches, like those described in (Rao *et al.*, 2023), make code generation consistent with code verification by training the model on aligned code-test pairs. Multi-agent systems in (Dong *et al.*, 2024) show how LLMs can be used in self-collaboration processes to iteratively improve the solution.

Table 2: Taxonomy of LLM Applications in Code Generation and Refactoring

Application Area	Task	Technique	Sources
Code Generation	General Code Synthesis	Prompt Engineering	(Subahi, 2026; Mu *et al.*, 2024; Jain *et al.*, 2022; Austin *et al.*, 2021; Li *et al.*, 2025; Li *et al.*, 2024)
		Test-Driven Generation	(Rao *et al.*, 2023; Mathews & Nagappan, 2024)
		Multi-Agent Collaboration	(Dong *et al.*, 2024; Pudari & Ernst, 2023)
	Domain-Specific Code Generation	DSL Creation	(Subahi, 2026)
		New Ecosystem Adaptation	(Aytekin *et al.*, 2026)
	Code Translation	Cross-Language Translation	(Yang *et al.*, 2024)
	Code Understanding	Summarization and Explanation	(Sun *et al.*, 2025; Nam *et al.*, 2024)
Code Improvement	Code Repair	Automated Bug Fixing	(Fan *et al.*, 2023; Jin *et al.*, 2023)
	Code Refactoring	Few-Shot Learning	(Fan *et al.*, 2023; Jin *et al.*, 2023)
		Autonomous Refactoring	(Fan *et al.*, 2023; Jin *et al.*, 2023)
Evaluation and Enhancement	Quality Assessment	Correctness Verification	(Liu *et al.*, 2023; Zhang *et al.*, 2025)
	Performance Optimization	Structured Prompting	(Li *et al.*, 2025)
	Real-World Deployment	Case Studies	(Zheng *et al.*, 2025; Ross *et al.*, 2023; Khojah *et al.*, 2024; Rajbhoj *et al.*, 2024; Bouzenia & Pradel, 2025)

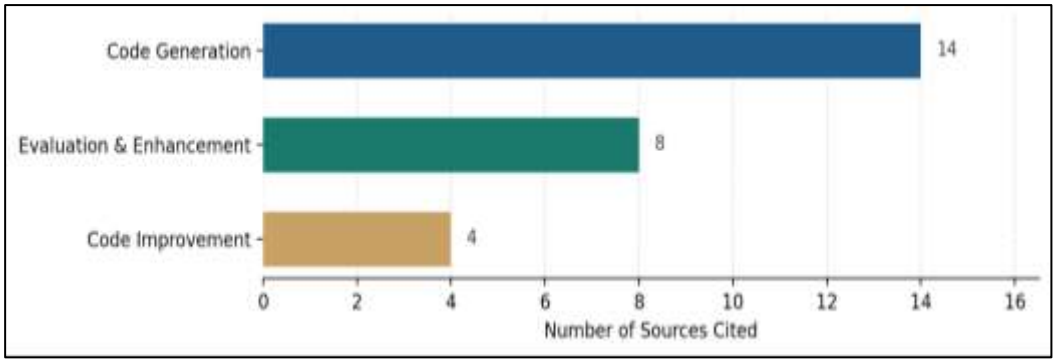

Fig. 5: Breakdown of LLM applications in Code Generation and Refactoring by task

There are also some domain-specific techniques applied to overcome the limitations of LLMs. For example, LLMs are employed for generating domain-specific languages (DSLs) by Subahi (2026). This way, it is possible to simplify the process of translating requirements into executable specifications. Study Aytekin et al. (2026) is devoted to adapting LLMs to work in an unusual programming ecosystem (e.g., ArkTS and HarmonyOS), and the authors emphasize the necessity of targeted fine-tuning in this case. Translation across programming languages is also an interesting question for research (Yang *et al.*, 2024). LLMs' abilities in code improvement are investigated in various works. Automated bug repair techniques in (Fan *et al.*, 2023; Jin *et al.*, 2023) use the power of LLMs to detect bugs in the code and fix them, achieving better results than static analysis techniques do. Few-shot learning approaches to refactoring are proposed in (Shirafuji *et al.*, 2023) and full autonomy is explored in (Zhang et al., 2024). It is important to assess the quality of code, which is accomplished in (Liu et al., 2023) through rigorous evaluations of the correctness of LLM-generated code and in (Zhang *et al.*, 2025) – through measuring its performance beyond the standard benchmarks.

Deployment studies shed light on problems encountered in real practice. Thus, Zheng *et al.* (2025) analyzes the course of conversations with LLMs during development processes, while Ross *et al.* (2023) observes how programmers use these models in practice. In addition, the case study by Rajbhoj *et al.* (2024) proves that ChatGPT usage allows developers to accelerate development cycles, although the inconsistency of code

requires sometimes human intervention. Study by Aytekin *et al.* (2026), which deals with ArkTS code generation, deserves special attention because it provides the baselines for the performance of LLMs in an emerging ecosystem. It shows once more the importance of the fine-tuning to achieve the best results in this case.

3.4 LLMs in Software Testing: Revolutionizing Test Automation and Quality Assurance

Research on the application of Large Language Models (LLMs) in software testing has expanded rapidly, demonstrating their potential to automate and improve various testing activities throughout the software development lifecycle. Recent studies have shown that LLMs can assist in generating test cases, automating test scripts, detecting vulnerabilities, localizing faults, and improving overall software quality. Based on the reviewed literature, the selected studies are categorized into several major application areas, as summarized in Table 3. A breakdown of LLM applications in software testing by specific task is illustrated in Figure 6.

Table 3: Taxonomy of LLM Applications in Software Testing

Application Area	Task	Technique	References
Test Case Generation	Unit Test Generation	LLM-based test synthesis	(Alagarsamy *et al.*, 2025; Alshahwan *et al.*, 2024; Lops *et al.*, 2025; Siddiq *et al.*, 2024; Wang *et al.*, 2024)
		Test suite assessment	(Lops *et al.*, 2025)
	Test Case from Bug Reports	Bug reproduction	(Kang *et al.*, 2023)
		Bug report to test case	(Plein *et al.*, 2024)
	Domain-Specific Test Generation	Fintech acceptance testing	(Xue *et al.*, 2024)
	Fuzzing	General-purpose fuzzing	(Xia *et al.*, 2024)
		Edge-case generation	(Deng *et al.*, 2024)
Test Script Automation	Test Script Generation	Language model-based synthesis	(Yu *et al.*, 2023)
	Test Migration	Language model adaptation	(Yu *et al.*, 2023)
Test Quality Improvement	Flaky Test Detection	Black-box prediction	(Fatima *et al.*, 2022)
	Test Oracle Automation	LLM-based oracle generation	(Molina *et al.*, 2025)
	Test Efficiency Enhancement	Generative AI optimization	(Neelapu, 2024)
Vulnerability Detection	Security Vulnerability Detection	LLM-based analysis	(Khare *et al.*, 2025; Guo *et al.*, 2024)
	Vulnerability Repair	LLM-based patching	(Fu *et al.*, 2023)
Fault Localization	Test-Free Fault Localization	LLM-based analysis	(Yang *et al.*, 2024)

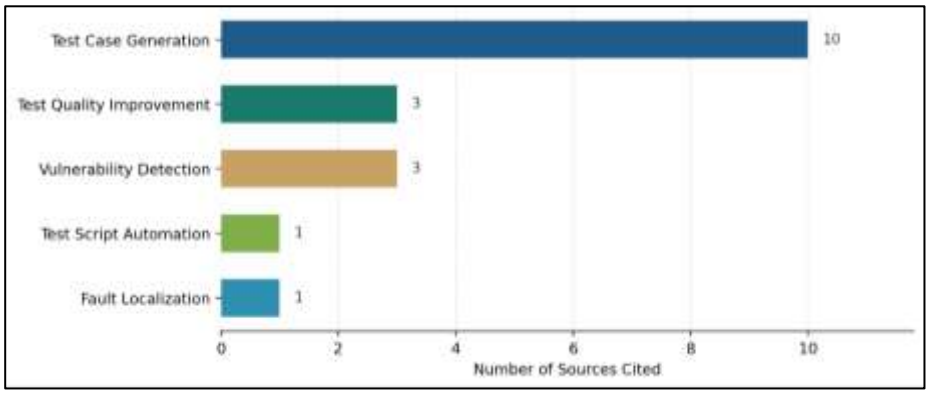

Fig. 6: Breakdown of LLM Applications in Software Testing

3.5 Test Case Generation and Automation

The literature review revealed that artificial intelligence models demonstrated high efficiency in automatically generating test cases and, particularly, test cases for unit testing. For example, some researches, including (Alshahwan *et al.*, 2024) and (Lops *et al.*, 2025), have indicated that such models were able to generate JUnit

tests covering a large number of functions. There was another research (Wang *et al.*, 2024) proposing a method of test generation precision increase by slicing methods. Furthermore, the study by (Kang *et al.*, 2023) made it possible to move one more step forward showing that models were able to reproduce failure scenarios on the basis of a limited amount of information. At the same time, there are some examples of successful use of artificial intelligence models to automatically generate test cases in the field, such as the work of (Xue *et al.*, 2024), which made complete automation of test case generation for acceptance testing in the financial field possible with the help of special prompts. Fuzzing techniques also became more advanced with the help of the integration of LLMs. For example, there is a recent study offering an innovative approach to fuzzing that makes it possible to generate inputs that cover a wide range of different programming languages on the basis of pre-trained knowledge. Another study pays attention to the identification of edge cases in deep learning libraries. The important thing is that both of them prove that LLMs can outperform traditional fuzzing techniques thanks to the use of their training data for the generation of unusual but valid test inputs.

In addition to generation, LLMs also contribute to test quality. For example, the Flakify framework applies language models to predict whether tests are prone to be flaky (unstable), which is a critical issue in regression testing. Another use case, in which LLMs are successfully applied is test oracle automation. In particular, the test oracle automation system described in (Molina *et al.*, 2025) allows the verification of expected test output without any manual specifications. The use of LLMs may allow achieving even more efficient testing as it is demonstrated in the research (Neelapu, 2024). The authors of (Neelapu, 2024) claim that the integration of generative AI into testing procedures leads to shortened testing timelines. In short, LLMs provide better results when it comes to test quality. As for the research conducted by (Santos et al., 2024), it provides us with a broader perspective regarding the use of LLMs in real testing scenarios. This work helps us to question what is really happening: are we testing these models or are they testing us? This study reveals two-way relationships between LLMs and the testing process, which affects each other.

Security testing also benefits from developments in LLMs field. Vulnerability detection researches (Khare et al., 2025) and (Guo et al., 2024) analyze performance of models in detecting vulnerabilities in the code but indicate some limitations when it comes to the detection of novel attack patterns. The research by (Fu et al., 2023) studies LLM-based vulnerability repair and illustrates the ability of LLMs to suggest patches for detected vulnerabilities. Fault detection is yet another area where advances are made. For instance, one research (Yang *et al.*, 2024) allowed fault detection without the use of any tests. It seems like the discovery could revolutionize debugging because it does not depend on any existing tests. Combining these technologies, LLMs might become autonomous testing agents (Feldt *et al.*, 2023). This means that the testing processes could be managed almost entirely by LLMs.

3.6 Security and Vulnerability Detection with LLMs

The use of LLMs for security and vulnerability identification is an example of the new paradigm that is used to protect software from potential risks and vulnerabilities. Based on the information provided in Table 4, we are able to distinguish three main fields of applications based on the different areas of security analysis and vulnerability management. A breakdown of LLM applications in security and vulnerability detection is depicted in Fig. 7.

Table 4: Taxonomy of LLM Applications in Security and Vulnerability Detection

Application Area	Task	Approach	Sources
Vulnerability Detection	General Detection	Empirical Evaluation	(Guo *et al.*, 2024; Noever, 2023; Zhou *et al.*, 2024)
		Chain-of-Thought Prompting	(Nong *et al.*, 2024)
	Supply Chain Analysis	Failure Analysis	(Singla *et al.*, 2023)
Vulnerability Repair	Automated Fixing	LLM-based Repair	(Noever, 2023; Fu *et al.*, 2023)
	Secure Code Generation	Fine-Tuning LLMs	(Li *et al.*, 2026)
Formal Verification	Self-Healing Software	LLMs + Formal Methods	(Tihanyi *et al.*, 2025)
	AI-Generated Code Validation	Formal Verification Integration	(Tihanyi *et al.*, 2023)

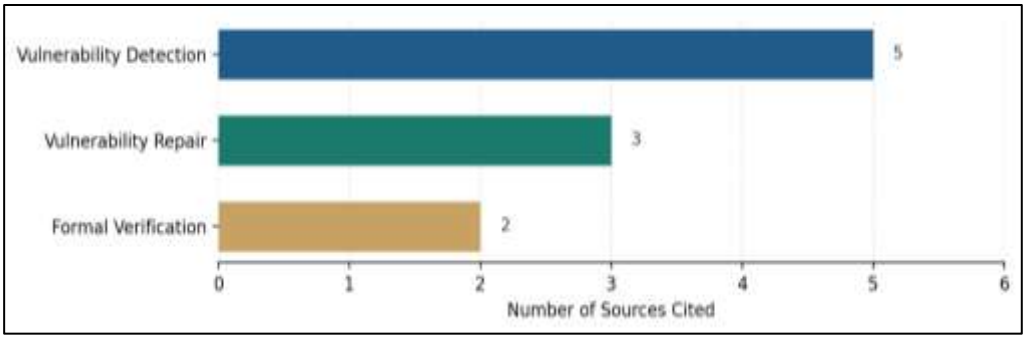

Fig. 7: Breakdown of LLM applications in Security and Vulnerability Detection

3.7 Vulnerability Detection and Analysis

The utilization of LLMs in security and vulnerabilities detection is a breakthrough in terms of ensuring software safety against newly emerging threats. As illustrated in Table 4, the selected literature can be classified into three main applications of LLMs, each of which relates to a different aspect of security analysis and vulnerabilities handling. LLMs demonstrate considerable potential for identifying various vulnerabilities in software. In particular, the research conducted by (Guo et al., 2024) explores their ability beyond the context of typical test cases and concludes that LLMs demonstrate high accuracy in pattern identification but struggle to adapt to unknown vulnerabilities. The research by (Zhou et al., 2024) presents promising findings regarding the use of language models for vulnerabilities detection and stresses the importance of the design of the language models and the dataset for training. This point is significant from the perspective of enhancing the security of software. With an appropriate use of LLMs, it is possible to identify and eliminate security vulnerabilities.

The analysis of supply chains for security purposes is another critical application area of LLMs. Specifically, the research conducted by (Singla *et al.*, 2023) analyses the effectiveness of language models in processing the history of software supply chain failures. The presented research illustrates how language models can analyze the complex dependency graph and discover the vulnerabilities but highlights the issues with the interpretation of subtle transitive vulnerabilities. The chain-of-thought prompting technique developed by (Nong *et al.*, 2024) constitutes a methodological advance allowing models to elaborate on the process of detecting vulnerabilities step by step.

3.8 Vulnerability Repair and Secure Code Generation

LLMs are not only effective in identifying software vulnerabilities but also show considerable potential in repairing them automatically. For instance, one study examined the ability of LLMs to recognize and patch software vulnerabilities and outlined certain criteria for assessing the model's ability to repair certain types of vulnerabilities. Additionally, one research was conducted on the ability of ChatGPT to classify and patch software vulnerabilities. Thus, it provides an understanding of the practical application of commercial LLMs in security. It is essential since it can provide insights into possible ways of utilizing LLMs in vulnerability fixing. This will ensure the safety of applications and make it less likely that they will face attacks. In general, LLMs can be considered a promising technology for solving issues with software vulnerabilities.

Creating secure code is a valuable approach that allows preventing possible problems. One study (Li *et al.*, 2026) analyzes the process of fine-tuning of LLMs for code generation, which ensures higher reliability of such generated code. It is clear why this happens since some other researches indicate that task-specific training of models increases their performance in security-related tasks.

3.9 Formal Verification and Validation

Application of AI for improving security systems is also a brilliant idea. Some experts have offered an innovative concept of creating self-healing software. Such a software is developed by implementing LLMs and formal verification techniques for analyzing code. As a result, such an approach allows solving security issues during the operation of software and is extremely helpful in case of newly emerged security vulnerabilities since it enables addressing them in real-time. For instance, in case of attacks, self-healing software will be able

to detect the issue and resolve the vulnerability in real-time. Thus, it can be considered one of the promising approaches to ensure safety of a system and prevent potential attacks in the future. FormAI dataset proposed by researchers is the breakthrough for checking the security and safety of AI-generated code. This work proposes using formal verification techniques for testing the security properties of the code generated with the help of LLMs. This work is quite valuable as it combines the generation of code through AI models and formal methods in the sphere of verification.

Studies conducted by (Fu *et al.*, 2023) provide additional information regarding LLMs in finding, classifying, and fixing various weaknesses in security. It is important to note that such research gives an additional idea of the progress of technologies discussed. As a result, taking into consideration all above-mentioned works, it can be said that LLMs have made quite significant progress, but there are some issues that need to be solved. In particular, LLMs have the following limitations: difficulties in identifying new types of attacks, lack of generalizability, difficulty to use LLMs alongside existing tools, lack of support of specific languages. Fortunately, this direction develops very fast and people try to solve these issues through improvements of models, training techniques, and result verification.

3.10 Prompt Engineering and Fine-Tuning of LLMs in Software Engineering

The proper use of LLMs in software engineering work highly relies on two major methods that are prompt engineering and fine-tuning of the model. As can be seen in Table 5 and Figure 8 below, we have systematically classified the reviewed studies based on their specific approaches towards LLM optimization in software engineering scenarios.

Table 5: Taxonomy of Prompt Engineering and Fine-Tuning Techniques

Technique Category	Specific Approach	Application Domain	Sources
Prompt Engineering	General Strategies	Requirements Elicitation	(Ronanki *et al.*, 2024; Rodriguez *et al.*, 2023)
		Code Generation	(Li *et al.*, 2024; White *et al.*, 2024)
	Domain-Specific Patterns	DSL Creation	(Subahi, 2026)
		Software Traceability	(Rodriguez *et al.*, 2023)
	Adversarial Robustness	Prompt Robustness Evaluation	(Zhu *et al.*, 2023)
Fine-Tuning	Human Preference Alignment	General SE Tasks	(Ziegler *et al.*, 2019)
	Self-Generated Instructions	Autonomous Improvement	(Wang *et al.*, 2023)
	Declarative Pipelines	Self-Improving Systems	(Khattab *et al.*, 2023)
Comparative Studies	Prompting vs. Fine-Tuning	Automated SE Tasks	(Shin *et al.*, 2023)
Evaluation Benchmarks	Real-World Task Performance	GitHub Issue Resolution	(Jimenez *et al.*, 2023)

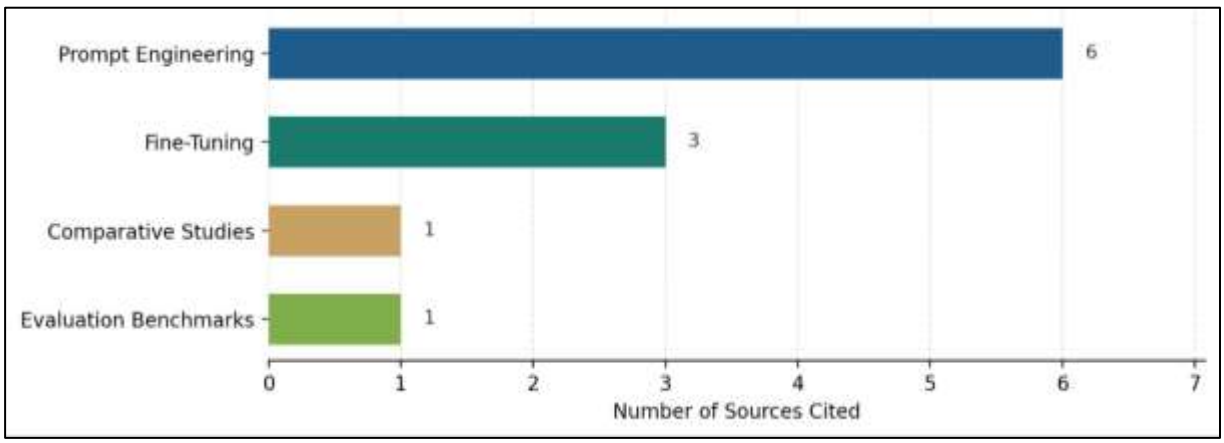

Fig. 8: Breakdown of Prompt Engineering and Fine-Tuning Techniques by specific approach

The second key technique in obtaining expected results from LLMs while keeping the base settings unchanged is prompt engineering. There are multiple studies dedicated to showing that well-designed prompts may dramatically enhance performance in tasks such as requirements elicitation and software traceability. For instance, some papers found that proper prompts helped language models understand the meaning of a task and gave correct answers to questions that were asked of them. In particular, this was relevant to the domain of software development since communication here had to be particularly precise. With help of specialized prompt patterns, it is possible to create domain-specific languages which would facilitate the process of translation of natural language requirements into formal specification. This technique can be particularly useful for tasks related to software development, in which language models can be applied in order to produce technical documentation. Hence, the research on the topic of prompt engineering is vital for extracting greater value from LLMs and enhancing their performance. One of the drawbacks of prompts is the high possibility of manipulation. Some research demonstrated that slight changes of vocabulary in prompts lead to drastic change of output of LLMs. Therefore, there is a need to come up with a solution to make the prompts more reliable. Also, there was a paper that considered how to use prompts in order to improve quality and effectiveness of code. As a result, it generated certain patterns that could be applied in general situations, which can be very useful for developers.

Finally, fine-tuning approaches are another possible way of adaptation of LLMs to software engineering tasks. The paper (Ziegler *et al.*, 2019) considers how reinforcement learning can help in making language models align with human preferences in code generation and documentation tasks. Autonomous improvement approaches as suggested in (Wang *et al.*, 2023; Khattab *et al.*, 2023) allow models to self-refine their abilities using self-generated instructions and declarative pipeline instead of huge human-labeled dataset. Comparing different approaches, one may obtain valuable information about their advantages and disadvantages. For instance, there was a paper (Shin *et al.*, 2023) which compared the performance of prompting and fine-tuning approaches in automated software engineering tasks. According to its results, the effectiveness of the approach largely depended on the context and required certain trade-off between effort and performance enhancement. To evaluate these approaches, one can use a benchmark such as SWE-bench (Jimenez *et al.*, 2023) to check how well the language models can solve real-world tasks from GitHub repository. The paper (Li *et al.*, 2024) has developed AceCoder – a special mechanism for programming code production that is superior to other existing methods. This happens due to the usage of knowledge from the domain. Thus, with correct prompts one can obtain considerable results without performing complex fine-tuning procedure. Having evaluated all of these studies, one obtains an overview of current situation in terms of prompt engineering and fine-tuning for software engineering purposes. They show potential to make a significant change, however, there is a need to continue research in order to make them more reliable, efficient and flexible.

3.11 Transforming Software Engineering Education through LLMs

Incorporation of LLMs within the education of software engineering constitutes a new paradigm shift in terms of teaching methodologies and poses both benefits and challenges to educational organizations. As seen in Table 6, we have conducted a systematic classification of the studies included based on the scope and contribution methods of the field.

Table 6: Taxonomy of LLM Applications in Software Engineering Education

Application Focus	Educational Impact	Key Findings	Sources
Practical AI Integration	Hands-on curriculum enhancement	LLMs as interactive teaching assistants	(Kozov *et al.*, 2024)
Generative AI Transformation	Automated learning support	Revolutionizing requirements and process model education	(Daun & Brings, 2023)

Kozov *et al.* (2024) focuses on applying AI and LLMs in practice for the purposes of teaching software engineering. This research demonstrates that such tools can serve as useful virtual teaching assistants providing instant feedback on students' performance. It is important to acknowledge the significance of the research

conducted by the authors since it proves that LLMs can play an important role in connecting theoretical knowledge acquired in classes with practical applications in such complex topics as software design and patterns. The most notable feature of LLMs' use in teaching software engineering is their ability to ensure personalized learning experience. Each student will receive an explanation and advice that is customized according to his or her performance. Daun & Brings (2023) provides an overview of how generative AI technologies including ChatGPT are transforming software engineering education. The study focuses on the analysis of the influence of LLMs on teaching software process models and requirements engineering in which LLMs prove to be highly proficient in generating and adapting educational content. It becomes obvious that there are many benefits in using LLMs in terms of automated question generation, adaptive learning materials creation, and provision of instant feedback. These capabilities make it possible to increase the efficiency and engagement of the learning process in this case. The researchers also note the necessity to develop new methods of assessing students since they may use LLMs during the learning process. The use of LLMs in software engineering education allows for a personalized learning experience since each student receives relevant feedback based on his or her performance.

LLMs in software engineering education not only allow for delivering the content but also reshape the process of learning. With the help of LLMs, it is possible to create scenarios when students are able to work with simulated requirements of the projects and get analysis of their solutions. The use of LLMs proves to be valuable in requirements engineering since students usually find this topic rather abstract. All these studies demonstrate that although LLMs cannot substitute the role of human instructors in education, they can become powerful tools in this process since they allow for personalizing instruction, scaling of educational resources, and providing constant learning support even outside traditional classroom. Current research indicates that the most effective use of LLMs in educational processes is associated with combining the generation capabilities of the latter with pedagogical frameworks. Such approach helps in making sure that AI-generated content meets learning goals while keeping academic standards. The introduction of LLMs into software engineering education also implies certain problems related to issues of academic integrity, critical thinking development, and students' problem-solving skills.

3.12 Challenges and Ethical Considerations in LLM Deployment

However, when you use LLMs in software engineering, it becomes highly complex. Various challenges as well as ethical questions have to be addressed. We carefully analyzed the key points made in the articles reviewed in relation to software engineering, such as those indicated in Table 7, and classified them. Such analysis is helpful in understanding the implications for research as well as practice. A breakdown of challenges and ethical considerations in LLM applications is depicted in Fig. 9.

Table 7: Taxonomy of Challenges and Ethical Considerations in LLM Applications

Category	Specific Challenge	Impact Area	Sources
Technical Limitations	Hallucination and Factual Errors	Requirements Specification	(Ozkaya, 2023; Kaddour *et al.*, 2023)
	Context Window Constraints	Large-Scale System Analysis	(Chen *et al.*, 2025)
	Domain Adaptation Difficulties	Specialized Software Domains	(Sallou *et al.*, 2024)
Security Risks	Prompt Injection Vulnerabilities	System Integrity	(Lu *et al.*, 2024)
	Data Leakage Concerns	Proprietary Information Protection	(Akbar *et al.*, 2023)
Ethical Concerns	Bias Propagation	Requirements Elicitation	(Ozkaya, 2023)
	Accountability Gaps	Critical System Development	(Sallou *et al.*, 2024)
	Intellectual Property Issues	Code Generation	(Akbar *et al.*, 2023)

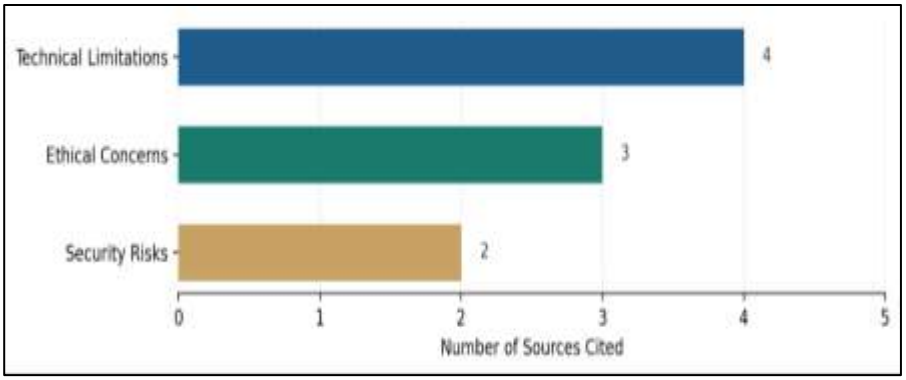

Fig. 9: Breakdown of Challenges and Ethical Considerations in LLM Applications

One of the key challenges for LLMs is the tendency to create outputs which are inconsistent with facts due to their hallucination capability, as noted in (Ozkaya, 2023) and (Kaddour et al., 2023). These studies point out that application of LLMs to requirement specification purposes leads to the generation of plausible yet impossible system behaviors, potentially causing costly design errors unless identified. Limitations in terms of context window size, highlighted in (Chen et al., 2025), are another challenge that makes it difficult to analyze large software systems due to difficulty in retaining coherent understanding over long documents. The work by (Sallou *et al.*, 2024) identifies the problem of domain adaptation, revealing the inability of LLMs to cope with specific software engineering terminology and contexts without additional fine-tuning. Deployment of LLMs raises security threats as well. The work of (Lu *et al.*, 2024) reveals the possibility of prompt injection attacks, leading to potential manipulation of the outputs of the models due to malicious inputs. The work by (Akbar *et al.*, 2023) complements these findings highlighting the risks of data leakage related to use of commercial LLMs APIs which might save and leak proprietary information in the form of requirements document or code snippets.

Ethical issues raised by the use of LLMs are complicated by themselves. Ozkaya (2023) highlights how biases from the training data of the model can be transferred to requirements elicitation and cause exclusion and unfairness in the generated system design. Problems of accountability related to LLM outputs containing errors affecting system functionality or safety are analyzed in (Sallou *et al.*, 2024). Intellectual property concerns related to LLM-generated code are discussed in (Akbar *et al.*, 2023). The research conducted by (Chen *et al.*, 2025) provides further insights into the challenges faced by practitioners in terms of error identification and correction in LLM outputs. The findings of this study confirm the key challenges outlined in our taxonomy and provide additional practical evidence to the problems related to LLM use. Overall, analysis of these studies allows us to conclude that although the potential of LLMs in software engineering is considerable, their use introduces a variety of challenges requiring systematic solutions. In order to ensure the responsible application of LLM technologies in software engineering field, we need to develop appropriate technical, policy, and ethical measures.

4. DISCUSSION

Once you review the existing evidence base as a whole, you can detect several recurring themes that reframe the perception of applying LLMs to SRE. In particular, what becomes apparent is the fact that while LLMs represent a valuable instrument capable of boosting the automation and efficiency of certain stages of the process, the application of this technology brings up a series of complex technical and ethical challenges. According to the evidence, LLMs have the potential to greatly enhance the process of translating natural language requirements into formal specifications, yet the reviewed works draw attention to certain unresolved issues related to the reliability, interpretability, and domain-specificity of these tools.

From a theoretical standpoint, the discussed works provide insights into the framework of intelligent requirements engineering, where LLMs act as mediators between human stakeholders and technical artifacts. Specifically, the papers by (Ronanki *et al.*, 2023) and (Beg *et al.*, 2025) provide insight into the ability of language models to structure natural language requirements, while the works by (Wei, 2024) and (Han *et al.*, 2024) reveal

how these models can help translate requirements into executable code. Thus, it becomes clear that rather than simply automating the existing processes, LLMs completely reshape the nature of the task, allowing to continuously refine and adjust the process through interactive communication in natural language. The implications for conceptual models of SRE are significant since LLMs allow to substitute the linear workflows with dynamic processes of AI mediation and negotiation of requirements.

Currently, the application of LLMs in SRE workflows proves to be quite promising and useful on the industrial level. For instance, the papers such as (Zhang *et al.*, 2024) and (Manish, 2024) suggest that the use of LLMs to refine user stories and facilitate agile project planning improves the quality of SRE and streamlines the process of development. However, to be able to apply the findings on the practical level, we should consider the operational challenges discussed in (Ozkaya, 2023) and (Lu *et al.*, 2024), such as the validation and security of output artifacts. Therefore, in addition to LLMs' capability to assist with certain aspects of SRE, we should develop certain practices to validate the result and ensure its quality and safety. Specifically, developing specialized prompting techniques, as described in (Ronanki *et al.*, 2024) and (Rodriguez *et al.*, 2023), emerges as an essential skill of the software engineer's profile.

Methodological limitations of this review deserve careful consideration. The fast-paced nature of LLM research implies that our analysis may lack the latest advancements in LLM-related publications that emerged after our literature search period. The fact that the majority of the studies considered in this paper examine English language-based requirements and Western development context raises questions related to cultural and linguistic biases introduced into our review. Moreover, a significant variability in the quality of the empirical validation in included studies renders difficult drawing firm conclusions on relative effectiveness of various LLM techniques in SRE. These limitations imply that our conclusions are supposed to reflect the state of knowledge in this domain within the specified period of time rather than provide objective facts regarding capabilities of LLM technologies. Some gaps in existing research are revealed. First of all, we have to conduct some longitudinal studies in order to reveal an effect of incorporating LLMs into SRE on the quality of software developed and cost of software maintenance over the course of time. One more important issue that requires investigation is a possibility to use LLMs in multimodal requirement processing, that is, when LLMs have to analyze both diagrams, tables, and textual elements. In addition to that, we should conduct more research related to the combination of LLMs with more formal methods in order to eliminate a problem of LLM hallucination through some advanced techniques, such as neurosymbolic integration. Moreover, an ethical dimension of the use of LLMs in software engineering needs to be further explored. In particular, the ethical problems associated with the use of LLMs in software development relate to the necessity to ensure that AI-generated requirements are bias-free. To address this problem, one can develop frameworks for auditing AI-generated requirements in order to identify any ethical biases. Some directions for future research may include the following: Conducting longitudinal studies to investigate LLM effects on software quality and maintenance costs; Investigating multimodal requirements processing through LLMs; Combining LLMs with formal methods to solve the problem of LLM hallucination; Investigating ethical dimensions of LLM adoption in software engineering; Developing audit frameworks for identifying biases in AI-generated requirements.

The influence of technological innovations on education appears to be ambivalent and includes both strengths and weaknesses that have to be taken into consideration. In particular, certain studies such as (Kozov *et al.*, 2024) and (Daun & Brings, 2023) suggest that advanced language models may become useful aids in educating people, although further research should be conducted in order to define how to update software engineering courses in accordance with the requirements of AI-driven professional environment. It means that besides teaching practical skills such as prompt handling and model fine-tuning, it is necessary to teach students how to critically assess results obtained from AI models and understand their limitations. While the use of these models may make software development more accessible due to the facilitation of requirement specification, there exists a danger that people may start relying on automated systems excessively and neglect basic engineering skills. There are certain issues related to the security of using LLMs in the process of SRE. Researchers as well as representatives of the industry should pay much attention to these problems. Several studies, such as (Guo *et al.*, 2024) and (Noever, 2023) prove that LLMs may be applied for finding vulnerabilities, although at the same time they introduce new vulnerabilities that can be used by attackers. It is vital to find

ways to implement LLMs in development processes safely and avoid additional security risks. One of the possible solutions for that is the implementation of standardized testing of LLMs' ability to handle security-related problems in order to allow a systematic comparison of various approaches and models. The collective evidence suggests that it is possible to observe the beginnings of the paradigm shift in SRE, in which AI becomes an active participant instead of being a mere tool. In order to successfully implement AI, interdisciplinary collaboration between software engineers, AI researchers, cognitive scientists, and ethicists is required. The studies described above serve as the basis for this work; however, many things still have to be done in order to establish proper theoretical framework, empirically prove its effectiveness and develop practical guidelines for LLM integration in SRE.

5. CONCLUSION

Through this systematic literature review, the multiple roles that have been played by the LLMs in the SRE process have been investigated. In addition to this, different applications of LLMs to SRE processes and the associated trends have been highlighted. The results show the great ability of LLMs in automating and improving the traditional SRE processes, from elicitation to validation and traceability. Nevertheless, some challenges, such as model interpretability, domain adaptation, and ethical considerations, remain to be resolved to facilitate the responsible use of LLMs. The practical significance of these findings is evident through the recommendations that can be provided for the relevant stakeholders. With the use of LLMs in the process of software development, it becomes essential to provide effective ways to validate the results and to perform prompt engineering for reducing risks such as hallucination and bias. At the same time, due to the changing abilities of LLMs, it is important to ensure the corresponding adaptation in research and practice. Longitudinal studies and interdisciplinary cooperation are needed to establish the appropriate benchmarks and ethical frameworks.

CREDIT AUTHOR STATEMENT

1. **Shafaque Aziz:** Conceptualization, Methodology, Project administration, Writing – Original Draft
2. **Khushboo:** Supervision, Validation, Writing – Review & Editing
3. **Saim Akhtar:** Visualization, Formal analysis, Validation, Writing – Review & Editing
4. **Sameera Jawed:** Investigation, Data Curation, Validation, Writing – Review & Editing

References

Akbar, M. A., Khan, A. A., & Liang, P. (2023). Ethical aspects of ChatGPT in software engineering research. *IEEE Transactions on Artificial Intelligence*, Vol. 6, Issue 2, pp. 254-267. https://doi.org/10.1109/TAI.2023.3318183

Alagarsamy, S., Tantithamthavorn, C., Takerngsaksiri, W., Arora, C., & Aleti, A. (2025). Enhancing large language models for text-to-testcase generation. *Journal of Systems and Software*, Vol. *230*, 112531. https://doi.org/10.1016/j.jss.2025.112531

Alshahwan, N., Chheda, J., Finogenova, A., Gokkaya, B., Harman, M., Harper, I., ... & Wang, E. (2024, July). Automated unit test improvement using large language models at meta. *32nd ACM International Conference on the Foundations of Software Engineering*, pp. 185-196. https://doi.org/10.1145/3663529.3663839

Arora, C., Grundy, J., & Abdelrazek, M. (2024). Advancing requirements engineering through generative ai: Assessing the role of LLMs. *Generative AI for Effective Software Development*, pp. 129-148. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-55642-5_6

Aytekin, M. C., Yılmaz, F. G., & Demirezen, M. U. (2026). Automating code generation for a new ecosystem: establishing baselines with large language model-based code generation for ArkTS and HarmonyOS. *Automated Software Engineering*, Vol. *33*, Issue 2. https://doi.org/10.1007/s10515-026-00599-9

Beg, A., O'Donoghue, D., & Monahan, R. (2025). Formalising Software Requirements with Large Language Models. *ADAPT Annual Conference*.

Bjarnason, E., Unterkalmsteiner, M., Borg, M., & Engström, E. (2016). A multi-case study of agile requirements engineering and the use of test cases as requirements. *Information and Software Technology*, Vol. *77*, pp. 61-79. https://doi.org/10.1016/j.infsof.2016.03.008

Bouzenia, I., & Pradel, M. (2025). You name it, i run it: An LLM agent to execute tests of arbitrary projects. *ACM on Software Engineering*, Vol. *2*, pp. 1054-1076. https://doi.org/10.1145/3728922

Bui, T. L., Dam, H. K., & Hoda, R. (2025, November). An LLM-based multi-agent framework for agile effort estimation. *40th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1032-1043. https://doi.org/10.1109/ASE63991.2025.00090

Chen, X., Gao, C., Chen, C., Zhang, G., & Liu, Y. (2025). An empirical study on challenges for LLM application developers. *ACM Transactions on Software Engineering and Methodology*, Vol. *34*, Issue 7, pp. 1-37. *https://doi.org/10.1145/3715007*

Dalpiaz, F., & Niu, N. (2020). Requirements engineering in the days of artificial intelligence. *IEEE Software*, Vol. 37, Issue 4, pp. 7-10. https://doi.org/10.1109/MS.2020.2986047

Daun, M., & Brings, J. (2023, June). How ChatGPT will change software engineering education. In *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education*, Volume 1, pp. 110-116. *https://doi.org/10.1145/3587102.3588815*

Deng, Y., Xia, C. S., Yang, C., Zhang, S. D., Yang, S., & Zhang, L. (2024). Large language models are edge-case generators: Crafting unusual programs for fuzzing deep learning libraries. *46th IEEE/ACM International Conference on Software Engineering*, pp. 1-13. https://doi.org/10.1145/3597503.3623343

Dong, Y., Jiang, X., Jin, Z., & Li, G. (2024). Self-collaboration code generation via ChatGpt. *ACM Transactions on Software Engineering and Methodology*, Vol. *33, Issue* 7, pp. 1-38. https://doi.org/10.1145/3672459

Doris, A. C., Grandi, D., Tomich, R., Alam, M. F., Ataei, M., Cheong, H., & Ahmed, F. (2025). Designqa: A multimodal benchmark for evaluating large language models' understanding of engineering documentation. *Journal of Computing and Information Science in Engineering*, Vol. 25, Issue 2, 021009. https://doi.org/10.1115/1.4067333

Elghariani, K., & Kama, N. (2016). Review on Agile requirements engineering challenges. *3rd International Conference on Computer and Information Sciences*, pp. 507-512. https://doi.org/10.1109/ICCOINS.2016.7783267

Ezzini, S., Abualhaija, S., Arora, C., & Sabetzadeh, M. (2023). Ai-based question answering assistance for analyzing natural-language requirements. *IEEE / ACM 45th International Conference on Software Engineering*, pp. 1277-1289. https://doi.org/10.1109/ICSE48619.2023.00113

Fan, Z., Gao, X., Mirchev, M., Roychoudhury, A., & Tan, S. H. (2023). Automated repair of programs from large language models. *IEEE/ACM 45th International Conference on Software Engineering*, pp. 1469-1481. https://doi.org/10.1109/ICSE48619.2023.00128

Fatima, S., Ghaleb, T. A., & Briand, L. (2022). Flakify: A black-box, language model-based predictor for flaky tests. *IEEE Transactions on Software Engineering*, Vol. 49, Issue 4, pp. 1912-1927. https://doi.org/10.1109/TSE.2022.3201209

Feldt, R., Kang, S., Yoon, J., & Yoo, S. (2023). Towards autonomous testing agents via conversational large language models. *38th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1688-1693. IEEE. https://doi.org/10.1109/ASE56229.2023.00148

Fu, M., Tantithamthavorn, C. K., Nguyen, V., & Le, T. (2023). ChatGpt for vulnerability detection, classification, and repair: How far are we? *30th Asia-Pacific Software Engineering Conference*, pp. 632-636. https://doi.org/10.1109/APSEC60848.2023.00085

Guo, Y., Patsakis, C., Hu, Q., Tang, Q., & Casino, F. (2024). Outside the comfort zone: Analysing llm capabilities in software vulnerability detection. *European Symposium on Research in Computer Security*, pp. 271-289. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-70879-4_14

Han, H., Kim, J., Yoo, J., Lee, Y., & Hwang, S. W. (2024). Archcode: Incorporating software requirements in code generation with large language models. *Annual Meeting of the Association for Computational Linguistics*, Vol. 1, pp. 13520-13552. https://doi.org/10.48550/arXiv.2408.00994

Heyn, H. M., Knauss, E., Muhammad, A. P., Eriksson, O., Linder, J., Subbiah, P., ... & Tungal, S. (2021, May). Requirement engineering challenges for ai-intense systems development. *IEEE/ACM 1st Workshop on AI Engineering-Software Engineering for AI*, pp. 89-96. IEEE. https://doi.org/10.1109/WAIN52551.2021.00020

Jain, N., Vaidyanath, S., Iyer, A., Natarajan, N., Parthasarathy, S., Rajamani, S., & Sharma, R. (2022). Jigsaw: Large language models meet program synthesis. *44th International Conference on Software Engineering*, pp. 1219-1231. https://doi.org/10.1145/3510003.3510203

Jain, N., Vaidyanath, S., Iyer, A., Natarajan, N., Parthasarathy, S., Rajamani, S., & Sharma, R. (2022). Jigsaw: Large language models meet program synthesis. *44th International Conference on Software Engineering*, pp. 1219-1231. https://doi.org/10.1145/3510003.3510203

Jimenez, C. E., Yang, J., Wettig, A., Yao, S., Pei, K., Press, O., & Narasimhan, K. (2024, May). Swe-bench: Can language models resolve real-world github issues? *International Conference on Learning Representations*, Vol. 2024, pp. 54107-54157.

Jin, M., Shahriar, S., Tufano, M., Shi, X., Lu, S., Sundaresan, N., & Svyatkovskiy, A. (2023). Inferfix: End-to-end program repair with LLMs. *31st ACM Joint European Software Engineering Conference and Symposium on The Foundations of Software Engineering*, pp. 1646-1656. https://doi.org/10.1145/3611643.3613892

Kaddour, J., Harris, J., Mozes, M., Bradley, H., Raileanu, R., & McHardy, R. (2023). Challenges and applications of large language models. *arXiv preprint arXiv:2307.10169*. *https://doi.org/10.48550/arXiv.2307.10169*

Kang, S., Yoon, J., & Yoo, S. (2023, May). Large language models are few-shot testers: Exploring LLM-based general bug reproduction. *IEEE/ACM 45th International Conference on Software Engineering,* pp. 2312-2323. https://doi.org/10.1109/ICSE48619.2023.00194

Khare, A., Dutta, S., Li, Z., Solko-Breslin, A., Alur, R., & Naik, M. (2025). Understanding the effectiveness of large language models in detecting security vulnerabilities. *IEEE Conference on Software Testing, Verification and Validation,* pp. 103-114. https://doi.org/10.1109/ICST62969.2025.10988968

Khattab, O., Singhvi, A., Maheshwari, P., Zhang, Z., Santhanam, K., Vardhamanan, S., ... & Potts, C. (2023). Dspy: Compiling declarative language model calls into self-improving pipelines. *arXiv preprint arXiv:2310.03714*. *https://doi.org/10.48550/arXiv.2310.03714*

Khojah, R., Mohamad, M., Leitner, P., & de Oliveira Neto, F. G. (2024). Beyond code generation: An observational study of chatgpt usage in software engineering practice. *ACM on Software Engineering*, Vol. 1(FSE), pp. 1819-1840. https://doi.org/10.1145/3660788

Kozov, V., Ivanova, G., & Atanasova, D. (2024). Practical Application of AI and Large Language Models in Software Engineering Education. *International Journal of Advanced Computer Science & Applications*, Vol. *15, Issue* 1. https://doi.org/10.14569/ijacsa.2024.0150168

Krishna, M., Gaur, B., Verma, A., & Jalote, P. (2024, June). Using LLMs in software requirements specifications: An empirical evaluation. *32nd International Requirements Engineering Conference,* pp. 475-483. https://doi.org/10.1109/RE59067.2024.00056

Li, J., Li, G., Li, Y., & Jin, Z. (2025). Structured chain-of-thought prompting for code generation. *ACM Transactions on Software Engineering and Methodology*, Vol. 34, Issue 2, pp. 1-23. https://doi.org/10.1145/3690635

Li, J., Rabbi, F., Cheng, C., Sangalay, A., Tian, Y., & Yang, J. (2026). An exploratory study on fine-tuning large language models for secure code generation. *Empirical Software Engineering*, Vol. *31*, Issue 4. https://doi.org/10.1007/s10664-026-10803-9

Li, J., Zhao, Y., Li, Y., Li, G., & Jin, Z. (2024). Acecoder: An effective prompting technique specialized in code generation. *ACM Transactions on Software Engineering and Methodology*, Vol. 33, Issue 8, pp. 1-26. https://doi.org/10.1145/3675395

Liu, J., Xia, C. S., Wang, Y., & Zhang, L. (2023). Is your code generated by ChatGpt really, correct? rigorous evaluation of large language models for code generation. *Advances in Neural Information Processing Systems*, Vol. *36*, pp. 21558-21572. https://ui.adsabs.harvard.edu/link_gateway/2023arXiv230501210L/doi:10.48550/arXiv.2305.01210

Lops, A., Narducci, F., Ragone, A., Trizio, M., & Bartolini, C. (2025, March). A system for automated unit test generation using large language models and assessment of generated test suites. *IEEE International Conference on Software Testing, Verification and Validation Workshops,* pp. 29-36. https://doi.org/10.1109/ICSTW64639.2025.10962454

Lu, Q., Zhu, L., Xu, X., Xing, Z., Harrer, S., & Whittle, J. (2024). Towards responsible generative ai: A reference architecture for designing foundation model-based agents. *IEEE 21st International Conference on Software Architecture Companion,* pp. 119-126. *https://doi.org/10.1109/ICSA-C63560.2024.00028*

Luitel, D., Hassani, S., & Sabetzadeh, M. (2023). Using language models for enhancing the completeness of natural-language requirements. In *International Working Conference on Requirements Engineering: Foundation for Software Quality*, pp. 87-104. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-29786-1_7

Manish, S. (2024). An autonomous multi-agent LLM framework for agile software development. *International Journal of Trend in Scientific Research and Development*, Vol. 8, Issue 5, pp. 892-898.

Mathews, N. S., & Nagappan, M. (2024, October). Test-driven development and llm-based code generation. *39th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1583-1594. https://doi.org/10.1145/3691620.3695527

Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., & Gao, J. (2024). Large language models: A survey. *arXiv preprint arXiv:2402.06196*. https://doi.org/10.48550/arXiv.2402.06196

Molina, F., Gorla, A., & d'Amorim, M. (2025). Test Oracle Automation in the era of LLMs. *ACM Transactions on Software Engineering and Methodology*, Vol. *34*, Issue 5, pp. 1-24. https://doi.org/10.1145/3715107

Mu, F., Shi, L., Wang, S., Yu, Z., Zhang, B., Wang, C., ... & Wang, Q. (2024). Clarifygpt: A framework for enhancing llm-based code generation via requirements clarification. *ACM on Software Engineering*, *1*, pp. 2332-2354. https://doi.org/10.1145/3660810

Nam, D., Macvean, A., Hellendoorn, V., Vasilescu, B., & Myers, B. (2024, April). Using an LLM to help with code understanding. *IEEE/ACM 46th International Conference on Software Engineering*, pp. 1-13. https://doi.org/10.1145/3597503.3639187

Neelapu, M. Enhancing Software Testing Efficiency with Generative AI and Large Language Models. *IJLRP-International Journal of Leading Research Publication*, Vol. *5*, Issue 12.

Noever, D. (2023). Can large language models find and fix vulnerable software? *arXiv preprint arXiv:2308.10345*. https://doi.org/10.48550/arXiv.2308.10345

Nong, Y., Aldeen, M., Cheng, L., Hu, H., Chen, F., & Cai, H. (2024). Chain-of-thought prompting of large language models for discovering and fixing software vulnerabilities. *arXiv preprint arXiv:2402.17230*. https://doi.org/10.48550/arXiv.2402.17230

Ozkaya, I. (2023). Application of large language models to software engineering tasks: Opportunities, risks, and implications. *IEEE software*, Vol. 40, Issue 3, pp. 4-8. *https://doi.org/10.1109/MS.2023.3248401*

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., ... & Moher, D. (2021). The PRISMA 2020 statement: an updated guideline for reporting systematic reviews. *bmj*, Vol. *372*. https://doi.org/10.1136/bmj.n71

Plein, L., Ouédraogo, W. C., Klein, J., & Bissyandé, T. F. (2024). Automatic generation of test cases based on bug reports: a feasibility study with large language models. *IEEE/ACM 46th International Conference on Software Engineering: Companion Proceedings*, pp. 360-361. https://doi.org/10.1145/3639478.3643119

Pudari, R., & Ernst, N. A. (2023). From copilot to pilot: Towards AI supported software development. *arXiv preprint arXiv:2303.04142*. https://doi.org/10.48550/arXiv.2303.04142

Rajbhoj, A., Somase, A., Kulkarni, P., & Kulkarni, V. (2024). Accelerating software development using generative AI: ChatGPT case study. *17th Innovations in Software Engineering Conference*, pp. 1-11. https://doi.org/10.1145/3641399.3641403

Rao, N., Jain, K., Alon, U., Le Goues, C., & Hellendoorn, V. J. (2023). CAT-LM training language models on aligned code and tests. *38th IEEE/ACM International Conference on Automated Software Engineering*, pp. 409-420. https://doi.org/10.1109/ASE56229.2023.00193

Rodriguez, A. D., Dearstyne, K. R., & Cleland-Huang, J. (2023). Prompts matter: Insights and strategies for prompt engineering in automated software traceability. *31st International Requirements Engineering Conference Workshops*, pp. 455-464. https://doi.org/10.48550/arXiv.2308.00229

Rodriguez-Cardenas, D. (2025, June). Towards More Interpretable Large Language Models for Code. *33rd ACM International Conference on the Foundations of Software Engineering*, pp. 1270-1272. https://doi.org/10.1145/3696630.3731464

Ronanki, K., Berger, C., & Horkoff, J. (2023). Investigating chatgpt's potential to assist in requirements elicitation processes. *49th IEEE Euromicro Conference on Software Engineering and Advanced Applications*, pp. 354-361. https://doi.org/10.1109/SEAA60479.2023.00061

Ronanki, K., Cabrero-Daniel, B., Horkoff, J., & Berger, C. (2024). Requirements engineering using generative ai: Prompts and prompting patterns. *Generative AI for Effective Software Development*, pp. 109-127. Cham: Springer Nature Switzerland. https://doi.org/10.48550/arXiv.2311.03832

Ross, S. I., Martinez, F., Houde, S., Muller, M., & Weisz, J. D. (2023, March). The programmer's assistant: Conversational interaction with a large language model for software development. *28th International Conference on Intelligent User Interfaces*, pp. 491-514. https://doi.org/10.1145/3581641.3584037

Sallou, J., Durieux, T., & Panichella, A. (2024, April). Breaking the silence: the threats of usingLLMs in software engineering. *ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, pp. 102-106. *https://doi.org/10.1145/3639476.3639764*

Santos, R., Santos, I., Magalhaes, C., & de Souza Santos, R. (2024). Are we testing or being tested? exploring the practical applications of large language models in software testing. *IEEE Conference on Software Testing, Verification and Validation*, pp. 353-360. https://doi.org/10.1109/ICST60714.2024.00039

Shin, J., Tang, C., Mohati, T., Nayebi, M., Wang, S., & Hemmati, H. (2023). Prompt engineering or fine tuning: An empirical assessment of large language models in automated software engineering tasks. *arXiv preprint arXiv:2310.10508*. https://doi.org/10.48550/arXiv.2310.10508

Shirafuji, A., Oda, Y., Suzuki, J., Morishita, M., & Watanobe, Y. (2023, December). Refactoring programs using large language models with few-shot examples. *30th Asia-Pacific Software Engineering Conference*, pp. 151-160. https://doi.org/10.1109/APSEC60848.2023.00025

Siddiq, M. L., Da Silva Santos, J. C., Tanvir, R. H., Ulfat, N., Al Rifat, F., & Carvalho Lopes, V. (2024, June). Using large language models to generate junit tests: An empirical study. *28th International Conference on Evaluation and Assessment in Software Engineering*, pp. 313-322. https://dl.acm.org/doi/proceedings/10.1145/3661167

Singla, T., Anandayuvaraj, D., Kalu, K. G., Schorlemmer, T. R., & Davis, J. C. (2023). An empirical study on using large language models to analyze software supply chain security failures. *Workshop on Software Supply Chain Offensive Research and Ecosystem Defenses*, pp. 5-15. https://doi.org/10.1145/3605770.3625214

Son, H. S. (2025). Using large language models in software requirements analysis. *Edelweiss Appl. Sci. Technol*, Vol. 9, Issue 3, pp. 856-863.

Subahi, A. F. (2026). Prompt Engineering for Advancing Software Engineering: Using Generative Language Models in the Development of Domain-Specific Languages. *IEEE Access*, Vol. *14*, pp. 22834-22850. https://doi.org/10.1109/ACCESS.2026.3663331

Sun, W., Miao, Y., Li, Y., Zhang, H., Fang, C., Liu, Y., ... & Chen, Z. (2025). Source code summarization in the era of large language models. *IEEE/ACM 47th International Conference on Software Engineering*, pp. 1882-1894. https://doi.org/10.1109/ICSE55347.2025.00034

Tihanyi, N., Bisztray, T., Jain, R., Ferrag, M. A., Cordeiro, L. C., & Mavroeidis, V. (2023). The formai dataset: Generative ai in software security through the lens of formal verification. *19th International Conference on Predictive Models and Data Analytics in Software Engineering*, pp. 33-43. https://doi.org/10.1145/3617555.3617874

Tihanyi, N., Charalambous, Y., Jain, R., Ferrag, M. A., & Cordeiro, L. C. (2025). A new era in software security: Towards self-healing software via large language models and formal verification. *IEEE/ACM International Conference on Automation of Software Test*, pp. 136-147. IEEE. https://doi.org/10.1109/AST66626.2025.00020

Treude, C., & Hata, H. (2023). She elicits requirements and he tests: Software engineering gender bias in large language models. *IEEE/ACM 20th International Conference on Mining Software Repositories*, pp. 624-629. https://doi.org/10.1109/MSR59073.2023.00088

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., & Hajishirzi, H. (2023). Self-instruct: Aligning language models with self-generated instructions. *61st Annual Meeting of the Association for Computational Linguistics*, Vol. 1, pp. 13484-13508. https://doi.org/10.18653/v1/2023.acl-long.754

Wang, Z., Liu, K., Li, G., & Jin, Z. (2024). Hits: High-coverage llm-based unit test generation via method slicing. *39th IEEE/ACM International Conference on Automated Software Engineering*, pp. 1258-1268. https://doi.org/10.1145/3691620.3695501

Wei, B. (2024). Requirements are all you need: From requirements to code with LLM. In *2024 IEEE 32nd International Requirements Engineering Conference*, pp. 416-422. https://doi.org/10.1109/RE59067.2024.00049

White, J., Hays, S., Fu, Q., Spencer-Smith, J., & Schmidt, D. C. (2024). Chatgpt prompt patterns for improving code quality, refactoring, requirements elicitation, and software design. In *Generative ai for effective software development*, pp. 71-108. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-55642-5_4

Xia, C. S., Paltenghi, M., Le Tian, J., Pradel, M., & Zhang, L. (2024). Fuzz4all: Universal fuzzing with large language models. *IEEE/ACM 46th International Conference on Software Engineering*, pp. 1-13. https://doi.org/10.1145/3597503.3639121

Xue, Z., Li, L., Tian, S., Chen, X., Li, P., Chen, L., ... & Zhang, M. (2024). Llm4fin: Fully automating LLM-powered test case generation for fintech software acceptance testing. *33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 1643-1655. https://doi.org/10.1145/3650212.3680388

Yang, A. Z., Le Goues, C., Martins, R., & Hellendoorn, V. (2024). Large language models for test-free fault localization. *46th IEEE/ACM International Conference on Software Engineering*, pp. 1-12. https://doi.org/10.1145/3597503.3623342

Yang, Z., Liu, F., Yu, Z., Keung, J. W., Li, J., Liu, S., ... & Li, G. (2024). Exploring and unleashing the power of large language models in automated code translation. *ACM on Software Engineering*, Vol. *1*(FSE), pp. 1585-1608. https://doi.org/10.1145/3660778

Yu, S., Fang, C., Ling, Y., Wu, C., & Chen, Z. (2023). LLM for test script generation and migration: Challenges, capabilities, and opportunities. *IEEE 23rd International Conference on Software Quality, Reliability, and Security*, pp. 206-217. https://doi.org/10.1109/QRS60937.2023.00029

Zhang, Y., Ruan, H., Fan, Z., & Roychoudhury, A. (2024, September). Autocoderover: Autonomous program improvement. *33rd ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 1592-1604. https://doi.org/10.1145/3650212.3680384

Zhang, Y., Xie, Y., Lit, S., Liu, K., Wang, C., Jia, Z., ... & Liao, X. (2025, April). Unseen horizons: Unveiling the real capability of LLM code generation beyond the familiar. *IEEE/ACM 47th International Conference on Software Engineering*, pp. 604-615. https://doi.org/10.1109/ICSE55347.2025.00082

Zhang, Z., Rayhan, M., Herda, T., Goisauf, M., & Abrahamsson, P. (2024). Llm-based agents for automating the enhancement of user story quality: An early report. *International Conference on Agile Software Development*, pp. 117-126. Cham: Springer Nature Switzerland. https://doi.org/10.1007/978-3-031-61154-4_8

Zheng, Z., Ning, K., Zhong, Q., Chen, J., Chen, W., Guo, L., ... & Wang, Y. (2025). Towards an understanding of large language models in software engineering tasks. *Empirical Software Engineering*, Vol. 30, Issue 2. https://doi.org/10.1007/s10664-024-10602-0

Zhou, X., Zhang, T., & Lo, D. (2024). Large language model for vulnerability detection: Emerging results and future directions. *ACM/IEEE 44th International Conference on Software Engineering: New Ideas and Emerging Results*, pp. 47-51. https://doi.org/10.1145/3639476.3639762

Zhu, K., Wang, J., Zhou, J., Wang, Z., Chen, H., Wang, Y., & Xie, X. (2023). Promptrobust: Towards evaluating the robustness of large language models on adversarial prompts. *ACM Workshop on Large AI Systems and Models with Privacy and Safety Analysis*, pp. 57-68. https://doi.org/10.1145/3689217.3690621

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., ... & Irving, G. (2019). Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*. https://doi.org/10.48550/arXiv.1909.08593

AUTHOR'S BIOGRAPHY

Shafaque Aziz is serving as an Assistant Professor in the Department of Computer Science and Engineering, Government Engineering College, Samastipur, Bihar, India. She has completed her Diploma, B.E. and M. Tech. in Computer Engineering from Jamia Millia Islamia, New Delhi, India and currently pursuing Ph.D. from Jamia Millia Islamia, New Delhi. Her research interest includes artificial intelligence, computer vision, soft computing with focus on intelligent decision-making information system for sign language recognition. She has also worked on AI-driven prediction model and aims to improve human-computer interaction through intelligent decision making.

Khushboo received the Bachelor's degree in Computer Applications and the Master's degree in Computer Applications from BIT Mesra, India. She is currently serving as an Assistant Professor in the Department of Computer Applications in Amity University Patna. She is pursuing research in the field of Artificial Intelligence and Machine Learning. Her research interests include Federated Learning, Multimodal, Data Analytics, and Explainable AI. She has authored research papers in reputed journals and conferences and is actively involved in teaching undergraduate and postgraduate courses in Computer Science

Saim Akhtar is currently pursuing the B.Tech. degree in Computer Engineering from J.C. Bose University of Science and Technology, YMCA, Faridabad, Haryana, India, where he is in the final year (Semester VII) of his undergraduate studies (2024–2027). He received the Diploma in Computer Engineering from University Polytechnic, Jamia Millia Islamia, in 2024. He is an active member of the Google Developers Group (GDG) JCBUST, where he regularly participates in technical events, workshops, and community-driven learning initiatives. His project focuses on Software Engineering, particularly the application of fuzzy multi-criteria decision-making techniques for software requirements prioritization and decision-support systems. His academic interests include Requirements Engineering, Artificial Intelligence, Soft Computing, Multi-Criteria Decision Making (MCDM), Fuzzy TOPSIS, Decision Support Systems, and Machine Learning.

Sameera Jawed is currently pursuing the B.Tech. degree in Computer Science and Technology from Maharaja Agrasen Institute of Technology, Guru Gobind Singh Indraprastha University, New Delhi, India, where she is in the final year of her undergraduate studies. She received the Diploma in Computer Engineering with First Division and Distinction from Jamia Millia Islamia, New Delhi, India, in 2024. During her diploma, she served as the Students' Placement Coordinator and actively participated in academic and technical activities. Her academic interests include Artificial Intelligence, Machine Learning, Deep Learning, Natural Language Processing, Data Science, Software Engineering, Requirements Engineering, and Intelligent Decision Support Systems. She has also completed IBM SkillsBuild Project-Based Learning Programs in Agentic AI and Front-End Web Development, strengthening her knowledge of modern AI applications and web technologies.