(Online) ISSN 2712-0562
Sustainable Engineering and Innovation *Original Research*
Vol. 7, No. 2, 2025, pp.463-476
https://doi.org/10.37868/sei.v7i2.id526

Using artificial intelligence in software development processes: Achievements and challenges

Vladyslav Kozub[1]*, Valentyn Druzhynin[2], Diana Trufanova[3], Pavlo Ihnatenko[4], Kateryna Kolos[5]

[1] Department of Information Technology and Systems, Educational and Research Institute of Mathematics and Information Technologies, Luhansk Taras Shevchenko National University, Ukraine
[2] Agile Engine LLC, USA
[3] Insurtech Software LLC, Ukraine
[4] Limited Liability Company ITEH, Ukraine
[5] Department of Cybernetics, Nanotechnology and Data Processing, Silesian University of Technology, Poland

*Corresponding author E-mail: kozubvuifmit@luguniv.edu.ua

Received Jun. 30, 2025
Revised Sep. 16, 2025
Accepted Sep. 26, 2025
Online Oct. 8, 2025

Abstract

This study consolidates contemporary methodologies for applying artificial intelligence in software engineering. Using the PRISMA protocol, an analysis of 60 peer-reviewed publications was conducted. Findings indicate that the use of generative tools (such as GitHub Copilot), AI-based testing platforms (like Testim.io and Diffblue), and DevOps automation systems (e.g., Harness.io) can lead to a 20–40% reduction in development time, while also enhancing code quality and minimizing errors. A key academic contribution of the research is the introduction of a three-tier classification of integration barriers – technical, organizational, and legal – that hinder the seamless adoption of AI technologies within the Software Development Life Cycle (SDLC), as well as the lack of standardized methodologies. The recommendations provided in this work are particularly relevant to software engineers, IT project leaders, and academic researchers, as they address crucial concerns related to model interpretability, system instability, the absence of unified standards, and regulatory ambiguity. The practical relevance of the study lies in presenting actionable strategies for the responsible, scalable, and ethically grounded deployment of AI-driven tools in industrial, academic, and research settings.

Keywords: Artificial intelligence, Code generation, Development automation, Ethical challenges, Software development

1. Introduction

Artificial intelligence (AI) has established itself as a pivotal technological advancement within the modern digital paradigm, exerting a profound influence that transcends routine automation and extends into critical domains such as education, scientific research, healthcare, industrial production, commerce, and public administration [1], [2], [3]. Thanks to its capacity for autonomous learning, rapid decision-making, processing

extensive datasets, and providing real-time support, AI is reshaping problem-solving approaches across multiple sectors. As global digitalization accelerates, the application of AI in software engineering – a foundational discipline that supports the information society – is garnering increasing attention [4], [5].

Intelligent systems are progressively playing a pivotal role in automating routine tasks and enhancing decision-making precision throughout these phases [6], [7], [8]. For instance, tools like ChatGPT and GitHub Copilot assist developers by generating code fragments and assessing algorithms [9], [10]. Moreover, machine learning techniques facilitate defect prediction, optimize resource distribution among teams, monitor productivity metrics, and streamline the entire software development lifecycle [11], [12], [13]. While the scientific and engineering communities acknowledge AI's transformative potential, practical challenges remain in effectively integrating these technologies into everyday workflows.

First and foremost, guaranteeing the transparency and interpretability of artificial intelligence models is crucial, particularly in safety-critical systems [14]. Many contemporary algorithms, such as deep neural networks, function as "black boxes," which complicates the process of validating their outputs. Additionally, challenges arise concerning the quality and representativeness of training datasets, as well as ethical dilemmas related to autonomous decision-making and ambiguous accountability frameworks for errors [15], [16], [17], [18]. The incorporation of AI into existing software architectures also presents challenges, as these architectures often lack the flexibility and adaptability required for the effective integration of machine learning techniques [19], [20]. Although numerous research articles address the use of AI in software development, most tend to focus on industry-specific applications. While there are in-depth investigations into specific phases of the software lifecycle, such as testing [7], code generation [6], and requirements management [12], a comprehensive, holistic analysis that encompasses all stages remains absent. Furthermore, there is a noticeable scarcity of systematic classifications regarding the challenges developers encounter when applying AI in practical IT projects [8], [21], [22].

This paper undertakes a critical examination of contemporary studies on the application of artificial intelligence in software engineering. It underscores significant progress made in the field, examines key technical and ethical dilemmas, and evaluates the current landscape of scholarly innovation. Through a structured and methodical framework, the review addresses knowledge gaps in the integration of AI across all phases of software development, setting the stage for future cross-disciplinary inquiry into intelligent software systems. The article focuses on the following research questions:

- What are the significant advancements achieved through the application of AI in automating and optimizing software development processes?
- What primary obstacles and limitations arise when incorporating AI into these development workflows?
- Based on the reviewed literature, what are the existing research gaps and potential directions for future study?

2. Research method

To achieve the goal of evaluating and consolidating contemporary methodologies for implementing artificial intelligence in software engineering, a structured review of existing scholarly literature was conducted. This methodology was chosen due to the relevance of the topic, its interdisciplinary nature, and the rapid advancement of AI technologies, all of which necessitate a well-structured and in-depth academic analysis. In contrast to traditional reviews, systematic reviews adhere to a predefined methodology for locating, evaluating, and synthesizing relevant research, thereby enhancing the objectivity, comprehensiveness, and reproducibility of their results. The literature search encompassed leading international academic databases, including IEEE Xplore, SpringerLink, Scopus, Google Scholar, and the ACM Digital Library. The search was conducted using targeted keywords such as "artificial intelligence in software development," "AI-driven programming," "machine learning in code generation," "AI in software testing," "automated debugging using AI," and "generative AI in software engineering." The timeframe for included publications ranged from 2019 to 2025,

focusing on recent works that capture the latest advancements related to generative models and deep learning techniques. Inclusion criteria were: (1) English-language scientific articles; (2) peer-reviewed conference or journal publications of high academic quality; (3) studies presenting clear methodologies or empirical data on AI applications within the software development lifecycle. Exclusion criteria ruled out review articles lacking transparent methodologies, publications from non-verified sources, and duplicates. From over 300 initially identified sources, 60 pertinent publications were selected. According to the PRISMA flowchart, only a portion of the identified sources passed the screening stages and were included in the final analysis, demonstrating the thoroughness and systematicity of the selection procedure. These encompassed practical implementations, theoretical foundations, methodological frameworks, and empirical research. Thematic analysis revealed three primary domains: first, progress in automating software lifecycle processes, including code generation, automated testing, defect identification, and continuous integration/continuous deployment (CI/CD); second, technical and ethical challenges in AI adoption, such as model interpretability, reliability, transparency in decision-making, and integration within existing development environments; third, research gaps concerning standardization, regulatory frameworks, and interdisciplinary collaboration across engineering, ethics, and legal perspectives in AI system development.

The selection procedure included several phases. Initially, sources were assessed by title and annotation. A full-text review was then conducted to determine final inclusion in the sample. The procedure is visualized in the PRISMA flowchart (Figure 1), which details the stages of filtering and justification for the selection.

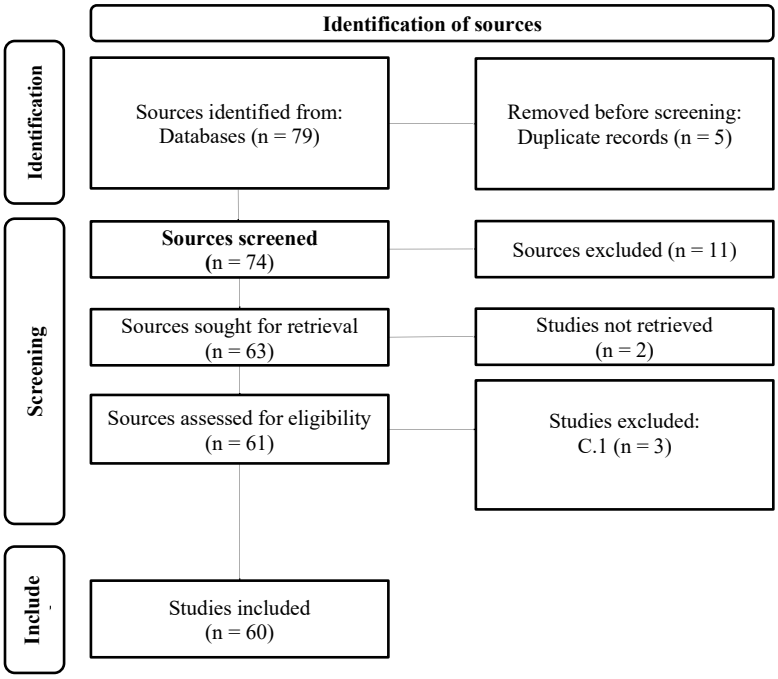

Figure 1. PRISMA flowchart

To deepen the examination of the selected materials, content analysis was employed, enabling the detection of recurrent patterns, thematic clusters, and areas lacking conceptual coherence. The analysis showed three dominant directions: (1) achievements in software life cycle automation (code generation, testing, CI/CD); (2) technical and ethical barriers (explainability, integration, responsibility); (3) interdisciplinary challenges (lack of regulatory framework, interaction of engineering and humanitarian components). Zotero was used to organize and manage bibliographic data, providing systematic storage, creation of tagged content, and automatic generation of bibliographies by publication standards. This helped to manage a wide range of sources effectively, cite cross-references, and quickly include scientific references in the text. The adopted research strategy enabled a well-founded consolidation of scholarly viewpoints regarding the use of artificial intelligence in software engineering, thereby providing a robust basis for analytical evaluation, integrative conclusions, and the delineation of promising pathways for future interdisciplinary exploration in the field.

3. Results

3.1. Results on AI achievements in software development

The review of existing literature highlights the significant impact of artificial intelligence on all essential stages of the software development process. Research findings demonstrate the effective integration of intelligent solutions in areas such as requirement elicitation, system architecture planning, automated coding, continuous integration and deployment, software validation, and post-deployment support. A particularly noteworthy development involves the use of natural language processing techniques to analyze user narratives and technical documentation, which helps reduce specification errors and improve the precision of business logic representation [8]. Tools like IBM DOORS NG facilitate the automated analysis of software requirements, thereby improving the consistency between documented specifications and their technical implementation. Significant strides have also been made in the code generation phase. A prominent example is GitHub Copilot, which leverages large language models to generate complete code snippets from comments or partial code inputs, effectively reducing development time and decreasing syntax and template errors. Additionally, other AI-powered tools such as Amazon CodeWhisperer and Tabnine are widely adopted in development environments, accelerating code writing in languages including Python, JavaScript, and Java.

In the testing industry, there is increasing pressure to automate tasks using machine learning. Tools like Testim.io and Diffblue Cover use NLP and ML to generate unit tests, perform regression testing, and identify error-prone code sections, reducing the burden on quality assurance and increasing software reliability [7].

Researchers are particularly focused on the role of AI in CI/CD processes. For example, services like Harness.io utilize AI to detect release defects, manage resources, accelerate deployment speed, and ensure service availability. Studies indicate a possible 15–25% reduction in release cycle times after applying these methods [11]. As shown in Table 1, the application of artificial intelligence tools at different stages of the software development life cycle (SDLC) allows achieving significant results: from increasing the accuracy of specifications and reducing errors in requirements to reducing code writing time by 20–40% and automating testing, which reduces the workload on QA by approximately 30%. As can be seen from Figure 2, the highest level of AI use in the software lifecycle is observed at the code generation stage (over 85%), while the lowest is at the architecture design stage (approximately 55%).

In addition, AI has demonstrated significant effectiveness in software support by predicting problems before they occur. These models analyze historical error data, code configurations, and change logs to predict the likelihood of problems occurring in specific modules. This approach enables the allocation of support resources to at-risk groups more effectively [13].

The summarized results are presented in Table 1, which shows the correlation between SDLC phases, AI-based tools, and recorded results.

Table 1. Application of AI in the phases of the software life cycle

SDLC phase	Tools/Approaches	Recorded results
Requirements Analysis	NLP Analysis of User Stories (IBM DOORS NG)	Increase specification accuracy, reduce requirements errors
Architecture Design	Evolutionary Algorithms for Design	Optimize architectural solutions, reduce human factors
Code Generation	GitHub Copilot, CodeWhisperer	Reduce code writing time by 20–40%
Testing	Testim.io, Diffblue Cover	Automate tests, reduce QA time by 30%
CI/CD	Harness.io, ML for DevOps	Accelerate releases by 15–25%, reduce failures.
Support	ML Models for Defect Prediction	Predict critical errors before release

According to the efficiency assessment based on the analyzed publications, the highest indicators were recorded at the code generation (85%) and testing (80%) stages, which is confirmed by the data visualized in Figure 2.

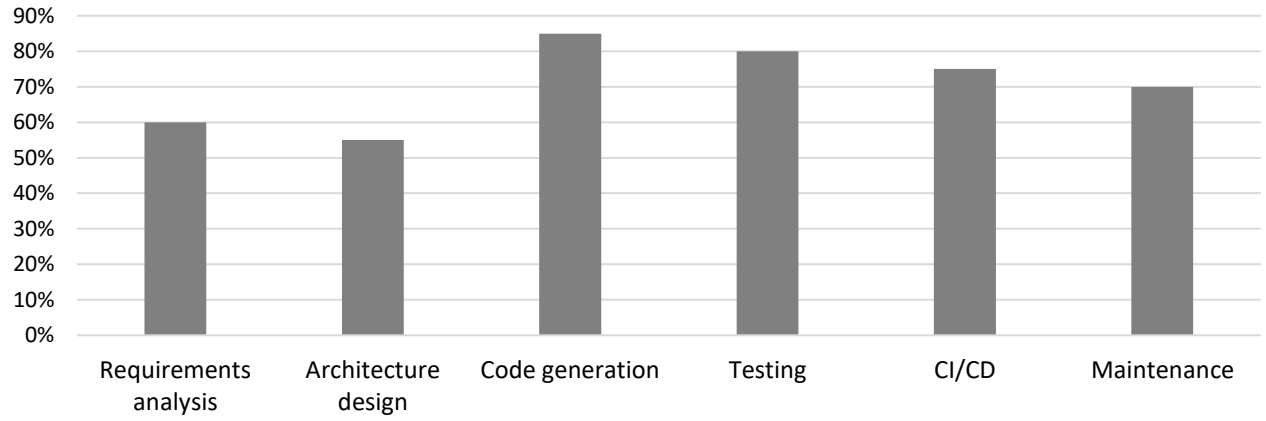

Figure 2. AI effectiveness in different phases of software development, %

During the support and CI/CD phases, the effectiveness indicators are somewhat lower but show consistent improvement. Currently, the least compelling aspect of artificial intelligence is noticeable at the architecture design stage due to the lack of universal integration models for complex systems. Specific publications present divergent views regarding the practical efficacy of AI tools in real-world settings. While some empirical studies report enhanced team productivity resulting from AI integration, others highlight challenges such as diminished system transparency, increased complexity in debugging processes, and a heightened reliance on comprehensive technical documentation [8], [12]. Overall, the proper integration of AI into developer support workflows leads to increased productivity, improved code quality, and enhanced user satisfaction.

3.2. Findings on challenges and limitations in AI implementation

The study identified that integrating AI into software development processes faces multiple challenges spanning technical, ethical, methodological, and legal domains. The complexity and interrelation of these issues make the adoption of AI in industrial software engineering particularly challenging. Technical difficulties are predominantly associated with the interpretability of outputs produced by neural networks. Due to the intricate operational logic of these models, it is often unclear how specific decisions or solutions are derived, complicating the validation process in mission-critical contexts [14]. Another major issue concerns the variability of results when executing identical queries, especially within generative models, where minor parameter changes can lead to significantly different outcomes. This unpredictability undermines confidence in the reliability and consistency of AI-driven systems over time [23].

Ethical obstacles arise from the lack of trust in autonomous decisions, especially in environments where AI performs technical and management roles. In cases where AI delegates tasks, modifies code, or suggests solutions without proper context, the question arises: Who should be held accountable for the results? Studies have documented cases of ambiguous procedures for determining responsibility for errors made by AI, leading to possible legal conflicts over damages [10], [18]. Methodological problems include integration difficulties – for example, conflicts between DevOps logic and automated changes made by AI in the pipeline [17]. Developers often note that AI, working autonomously, can generate changes that contradict team agreements or security policies and are not easily auditable. Moreover, the success of local implementation does not guarantee effective scalability of solutions at the level of large distributed teams or corporations, which is especially noticeable in long-term projects [24].

The legal aspects of the issue remain one of the least formalized, although they may determine the boundaries of AI implementation in the future. In particular, there are no established mechanisms for fixing copyright on code created using generative models, such as Copilot or CodeWhisperer. This complicates the legal use of such

fragments in commercial projects [18]. The issue of liability for potential damages caused by autonomous AI actions in production systems is also not regulated [25]. As can be seen from Table 2, the main challenges of implementing artificial intelligence in software development cover technical, ethical, methodological, and legal aspects: from the opacity of models and the instability of their results to the problems of liability for errors and the uncertainty of copyright on the generated code.

A systematic classification of the main challenges is presented in Table 2 below.

Table 2. Main challenges of implementing AI in software development

Call type	Description	Example	Reference
Technical	Unexplainability of models (black box)	Deep neural networks do not allow us to explain the logic of code generation.	[14]
Technical	Instability of model results with the same input data	The same queries to generative models give different answers	[23]
Ethical	Problem of trust in autonomous solutions	AI solutions are applied without sufficient justification	[10]
Ethical	Lack of responsibility for AI solutions	It is impossible to establish who is responsible for a bug in the generated code.	[18]
Methodological	Conflicts with DevOps/CI processes	AI can modify the code without considering the context of changes within the project.	[17]
Methodological	Problems of scaling AI solutions in large teams	The tools work well locally, but do not scale to the organization	[24]
Legal	Uncertainty of copyright on generated code	It is not known who owns the rights to the code created by Copilot	[18]
Legal	Lack of standards of responsibility for AI errors	There are no legal mechanisms for holding someone liable for damages	[25]

It is worth highlighting the contradictions recorded in scientific publications. Some studies show such benefits as increased efficiency and code quality [12], [26], while others indicate such risks as reduced transparency and complexity of verification [10], [27], [28]. This highlights the need for further research to align AI practices with industry requirements.

3.3. Results on identified research gaps and prospects for further research

Analyzing recent academic work reveals both the advancements made and the hurdles faced in incorporating AI into software engineering. Additionally, it highlights structural shortcomings that hinder progress in this field. To classify these problems, a three-part typology of research gaps has been proposed, taking into account technical (micro), organizational (meso), and regulatory-institutional (macro) aspects. This framework helps to classify challenges and understand their interaction in practical software development. At the micro level, focusing on technical implementation, fundamental obstacles have been observed, particularly the lack of clarity in results when using opaque deep neural networks that function as a "black box" [14], [18].

As shown in Table 3, the problems of implementing artificial intelligence in software development manifest themselves at different levels: from technical (micro level), related to the opacity of models and instability of results, to organizational (meso level), related to the impact on team dynamics and methodologies, and institutional and legal (macro level), where the lack of legal norms regarding authorship and liability for AI errors is particularly relevant.

This feature limited the choice of audit and created risks to the system's crucial reliability. In addition, the lack of flexible mechanisms for integrating AI tools into object-oriented programming languages or frameworks hinders scalability [12], [26]. In addition, generative models have shown performance fluctuations, producing significantly different results from identical inputs for no apparent reason [27], [29]. The mesolevel demonstrates how AI affects the organizational dynamics of work teams. An area where research lacks understanding is the impact of AI on internal team relationships, such as role allocation, trust, and decision-making [8], [30]. There is a lack of research examining how AI interacts with agile development approaches, such as agile or scrum [31], [32]. The impact on collective responsibility, adaptation to automated tools, and communication transformation in critical development teams often remains unexplored, despite its strategic importance [33], [34].

The identified gaps at the macro level relate to regulatory and legal aspects. First, there is the lack of an agreed-upon legal method for identifying the creators of AI-generated code [10], [22]. Currently, there is uncertainty about who bears legal responsibility for the operation of an autonomous system, which hinders the integration of AI into commercial products [17], [35]. Additionally, there is a lack of regulatory oversight to ensure the transparency of data during model training, especially in corporate settings where confidentiality is crucial [19], [36].

A summary typology of the problems above is presented in Table 3, which classifies them based on the level of occurrence and illustrates how technical, organizational, and legal aspects interact with one another.

Table 3. Typology of unresolved issues by system levels

Level	Issue
Micro level (technical implementation)	Unexplainability of the models (black box problem)
Micro level (technical implementation)	Insufficient adaptation of AI tools to programming languages and frameworks
Micro level (technical implementation)	Instability of the generative model results
Meso level (organizational context)	Lack of research on the impact of AI on the distribution of roles and team productivity
Meso level (organizational context)	Lack of research on the interaction of AI with agile methodology
Meso level (organizational context)	Lack of analysis of changes in team dynamics when implementing AI
Macro level (institutional and legal context)	Lack of legal norms regarding the authorship of AI code
Macro level (institutional and legal context)	Uncertainty in the sphere of responsibility for AI errors in software

Table 3 demonstrates that none of the gaps is isolated – they all form a complex system of constraints and contradictions that require not point-based but systemic solutions. Further research needs to consider the various obstacles and adopt interdisciplinary strategies that address both the technical and socio-humanitarian aspects of IIT implementation. Attention should be paid to the relationship between technical analysis (to improve models, algorithms, and integration solutions), legal measures (to ensure compliance and ethics in innovation), and teamwork research, with a focus on how AI impacts interaction, task delegation, and decision-making processes in development settings. A complete integration of these areas can increase the efficiency and reliability of software while facilitating its seamless integration into the organizational environment, thereby reducing the risks associated with technology resistance, ethical issues, or legal ambiguities. Given the rapid spread of generative AI models and their growing impact on professional practice, such methodologies should evolve from exceptions to become the standard for understanding digital transformation in science.

4. Discussion

Applying artificial intelligence to software development demands a comprehensive analysis that balances practical findings with the systemic constraints consistently noted in academic discourse. This study aims to consolidate the current scientific understanding in this domain critically, emphasizing significant achievements, prevailing challenges, and research gaps encountered by industry professionals. Overall, the findings highlight notable advancements in utilizing AI to optimize software performance evaluation [20] and enhance customer service processes, reflecting substantial potential for interdisciplinary knowledge exchange [37]. Addressing the first research question concerning principal accomplishments, there is clear evidence of improved automation of repetitive tasks, including code generation [26], testing procedures [25], and quality assurance within complex distributed environments [31]. These results align with the literature that characterizes machine learning as a pivotal stage in the evolution of software development methodologies [35]. However, the adoption of novel techniques, such as integrating large language models into programming workflows, introduces new risks and challenges, particularly related to security, result consistency, and model governance [33].

Regarding limitations, as noted in [30], a critical barrier remains the absence of standardized frameworks to support the sustainable integration of AI throughout the software development lifecycle (SDLC). This observation is corroborated by review studies [32], which underscore the fragmented nature of current approaches and the limited transferability of solutions across diverse industries. In domains such as energy efficiency, AI also demonstrates promising capabilities, although model tuning for specific system contexts continues to pose challenges [38]. Furthermore, an analysis of deep learning methods in software engineering, presented in [39], reveals varying levels of model effectiveness depending on the task type, underscoring the necessity for tool adaptation.

Concerning research gaps, emerging studies on generative testing [40] and project-based learning [41] are beginning to address some aspects; however, many experts, including reference [27], highlight a significant lack of generalizable insights regarding the deployment of large language models in real-world business pipelines. Additionally, there is limited research focusing on AI's influence on software testing methodologies [42], where conventional techniques still predominate and AI-driven components frequently function as "black boxes" [43]. The interpretability issue, discussed in [44], undermines trust in AI-generated outcomes and restricts their practical implementation. Similarly, study [45] illustrates a diverse range of AI applications but also points to output instability caused by parameter adjustments. This concern is reinforced by review [46], which notes that evaluating AI result quality remains a contentious subject.

The synthesis of these findings suggests that while AI demonstrates successes in specific areas of software development, overall progress is uneven and marked by uncertainty. In advanced fields such as large language model testing, results have yet to achieve widespread dissemination [47]. Consequently, engineering debt is accumulating due to inconsistent solutions [36] and the inherent challenges of integrating AI within legacy systems [48].

As the analysis of current research progresses, it is essential to focus on the real-world impact of employing generative AI technologies within educational contexts and in the training of future software engineering professionals. As noted in [34], incorporating systems like ChatGPT and GitHub Copilot into the learning environment helps students grasp program structure and enhances their critical thinking capabilities. This highlights the value of generative AI not only as a technical asset but also as a pedagogical instrument. However, as pointed out in [23], a noticeable gap remains between the theoretical capabilities of large language models (LLMs) and their actual effectiveness in software development, particularly during testing phases.

Furthermore, [24] identifies several obstacles that hinder the real-world adoption of AI-based software testing solutions, including issues of reliability, inconsistent outcomes, insufficient training datasets, and integration challenges within DevOps workflows. These concerns are supported by a systematic mapping study in [49], which underlines the scarcity of empirical data on the actual impact of machine learning on software quality. In

light of this, reference [50] proposes a revised approach to validating AI-driven solutions in software testing, advocating for a hybrid model that combines human oversight with autonomous system capabilities.

Concerns have also been raised regarding overdependence on AI technologies. For example, [51] examines the educational use of ChatGPT and warns of a growing reliance on generative models, which alter the traditional cognitive interaction between developers and algorithms. This phenomenon calls for deeper investigation. Moreover, reference [28] illustrates the heterogeneity of standards in AI testing practices, ranging from open-source frameworks to proprietary enterprise solutions. This lack of uniformity complicates cross-study comparisons and hinders the development of consistent, standardized methodologies.

A comprehensive evaluation of AI integration into high-risk systems is presented in [52], where researchers emphasize the need for unified testing strategies for AI modules, particularly in critical domains such as autonomous vehicles [53]. They argue that conventional testing approaches fail to account for the distinctive characteristics of AI behavior and suggest the use of adaptive methods, such as real-time simulation and rare-event scenario modeling, to address this limitation. Reference [54] goes further, calling for the development of an entirely new testing paradigm tailored to the stochastic nature of AI systems. The authors contend that traditional test case construction, based on deterministic software expectations, is incompatible with the probabilistic outputs of modern AI models. Addressing this discrepancy requires both technical innovations and a rethinking of software engineering and validation methodologies.

Reference [55] emphasizes the unreliability of AI planning in real-world conditions. Recent studies have shown that this problem remains relevant, highlighting that even small changes in input data can significantly alter the initial decision, a crucial factor for real-time systems, such as aviation or medical software. This finding becomes even more critical with the advent of autonomous decision-making systems, underscoring the need to test models under complex, dynamic operating conditions.

Finally, reference [56] suggests the use of search-augmented generation (RAG) for context-aware code generation as a promising approach that combines AI Copilot capabilities with flexible knowledge structures. This signals a shift from specialized models to tailored solutions applicable to all stages of the SDLC. In comparing our findings with prior research, the relevance of the identified challenges and research gaps is further validated. Nonetheless, the analysis reveals that scholarly understanding of the long-term consequences of AI integration in software development remains at a formative stage.

Artificial intelligence can be integrated into software and hardware modules to enhance testing technologies in mining engineering [57]. Moreover, big data has emerged as a pivotal asset, empowering businesses to extract actionable insights, streamline operations, and make informed decisions [58]. In addition, artificial intelligence transforms marketing strategies by enabling automation, personalization, and predictive analytics [59]. Furthermore, fuzzy logic methods are widely applied in artificial intelligence to support decision-making under uncertainty [60]. The software development process encompasses multiple stages, including planning, design, coding, testing, deployment, maintenance, and updates, and in this context, VR testing has made an impact on human–system interactions through its realism, which has helped to increase the accuracy of usability testing [61]. Finally, integrations of innovative technologies like IoT, big data, AI, 5G, and robotics enable smart aquaculture to mechanize and optimize real-time surveillance, supporting data-driven decision-making and efficiency [62]. Consequently, these examples collectively illustrate that the integration of AI and other advanced technologies significantly enhances efficiency, decision-making, and innovation across diverse domains.

This review systematizes existing approaches, identifies critical areas of underexplored research, and proposes a multi-layered typological framework to support future interdisciplinary inquiry. On a practical level, the implications are multifaceted. For software practitioners, tools such as GitHub Copilot and automated testing environments present opportunities to enhance productivity, reduce coding errors, and allocate greater focus to complex development tasks. However, their practical use necessitates a robust comprehension of AI

mechanisms and awareness of associated risks, including potential vulnerabilities in software security. Project managers should evaluate their development management strategies to adapt pipelines to handle autonomous modules, ensure clear segregation of duties, and address the need for transparency and ethical decisions under the influence of AI. Researchers can use the review's results to identify areas with potential for future research, with a particular emphasis on interdisciplinary research that incorporates software development across various industries. However, it is essential to note some limitations of the review: the analysis focused primarily on English-language peer-reviewed sources, which may not fully reflect regional or non-traditional approaches to incorporating AI into software development. Furthermore, due to rapid technological advances, some findings may quickly become outdated and will require updating in subsequent reviews.

5. Conclusions

The exploration of artificial intelligence applications in software development has aimed to outline current progress, identify significant challenges, and suggest directions for future inquiry. AI technologies are now influencing nearly every stage of the software development lifecycle—from initial requirements analysis to long-term system maintenance. The most substantial benefits have been observed in areas such as code generation and automated testing, where AI contributes to reduced development timelines, increased efficiency, and a decline in error rates. Tools based on machine learning and advanced language models have demonstrated their effectiveness in generating code structures, automating test scripts, detecting bugs, and predicting risks.

However, several obstacles persist in the seamless adoption of AI within real-world software engineering workflows. Key technical challenges include the opacity of AI model decision-making, the complexity of adapting intelligent solutions across diverse application domains, and the difficulty of harmonizing traditional development frameworks with autonomous AI-driven processes. In parallel, ethical and legal concerns emerge, such as trust in machine-generated code, accountability for defects, and ownership of AI-produced content.

Despite the growing volume of scholarly literature in this field, notable research gaps remain. These include understanding AI's influence on agile team collaboration, developing standardized metrics for assessing AI quality, enhancing model transparency, and addressing the legal implications of using generative technologies in professional settings.

This review synthesizes contemporary research findings, highlights prevailing patterns and inconsistencies, and provides a comprehensive overview of AI's evolving role in software engineering. It not only consolidates the current body of knowledge on AI integration but also identifies priority areas for further interdisciplinary research. Notably, comparative empirical studies evaluating the effectiveness of AI tools in various collaborative environments show encouraging outcomes. In addition, ongoing investigations are focusing on explainable AI methods tailored for software development, ethical frameworks for generative models, and the formulation of regulatory standards to guide the use of intelligent systems in software production.

Declaration of competing interest

The authors declare that they have no known financial or non-financial competing interests in any material discussed in this paper.

Funding information

No funding organization received funding for this study.

Author contribution

The contribution to the paper is as follows: V. Kozub, V. Druzhynin, D. Trufanova, P. Ihnatenko, K. Kolos: study conception and design; V. Kozub, V. Druzhynin, D. Trufanova, P. Ihnatenko, K. Kolos: data collection; V. Kozub, V. Druzhynin, D. Trufanova, P. Ihnatenko, K. Kolos: analysis and interpretation of results;

V. Kozub, V. Druzhynin, D. Trufanova, P. Ihnatenko, K. Kolos: draft preparation. All authors approved the final version of the manuscript.

References

[1] S. Sharov et al., “Using MOOC to Learn the Python Programming Language,” International Journal of Emerging Technologies in Learning, vol. 18, no. 2, pp. 17–23, 2023, https://doi.org/10.3991/ijet.v18i02.36431

[2] M. Storozhyk, “Philosophy of future: analytical overview of interaction between education, science, and artificial intelligence in the context of contemporary challenges,” Futurity Philosophy, vol. 3, no. 1, pp. 23–47, 2024, https://doi.org/10.57125/FP.2024.03.30.02

[3] Z. S. Mukhametzhanova, A. N. Daurenbekova, G. K. Zhanibekova, K. S. Syzdykova, and G. Kaliakparova, “Evaluation of influence of innovation on enterprise productivity,” *Space and Culture, India*, vol. 7, no. 1, pp. 186–193, 2019, https://doi.org/10.20896/saci.v7i1.527

[4] N. Smailov et al., “Streamlining digital correlation-interferometric direction finding with spatial analytical signal,” Informatyka, Automatyka, Pomiary w Gospodarce i Ochronie Środowiska, vol. 14, no. 3, pp. 43–48, 2024, https://doi.org/10.35784/iapgos.6177

[5] T. Wang et al., “Exploring the potential impact of artificial intelligence (AI) on international students in higher education: Generative AI, chatbots, analytics, and international student success,” Applied Sciences, vol. 13, no. 11, p. 6716, 2023, https://doi.org/10.3390/app13116716

[6] Ł. Korzeniowski and K. Goczyła, “Artificial intelligence for software development: the present and the challenges for the future,” Biuletyn Wojskowej Akademii Technicznej, vol. 68, no. 1, 2019, https://doi.org/10.5604/01.3001.0013.1464

[7] Z. Khaliq, S. U. Farooq, and D. A. Khan, “Artificial intelligence in software testing: Impact, problems, challenges and prospect,” *arXiv preprint*, arXiv:2201.05371, 2022, https://doi.org/10.48550/arXiv.2201.05371

[8] A. Deshmukh, D. S. Patil, J. S. Mohan, G. Balamurugan, and A. K. Tyagi, “Transforming next generation-based artificial intelligence for software development: current status, issues, challenges, and future opportunities,” in Emerging Technologies and Digital Transformation in the Manufacturing Industry, IGI Global, 2023, pp. 30–66, https://doi.org/10.4018/978-1-6684-8088-5.ch003

[9] L. Banh and G. Strobel, “Generative artificial intelligence,” Electronic Markets, vol. 33, p. 63, 2023, https://doi.org/10.1007/s12525-023-00680-1

[10] C. Ebert and P. Louridas, "Generative AI for Software Practitioners," in IEEE Software, vol. 40, no. 4, pp. 30-38, July-Aug. 2023, https://doi.org/10.1109/MS.2023.3265877

[11] U. K. Durrani, M. Akpinar, M. Fatih Adak, A. Talha Kabakus, M. Maruf Öztürk and M. Saleh, "A Decade of Progress: A Systematic Literature Review on the Integration of AI in Software Engineering Phases and Activities (2013-2023)," in IEEE Access, vol. 12, pp. 171185-171204, 2024, https://doi.org/10.1109/ACCESS.2024.3488904

[12] V. Kulkarni, A. Kolhe, and J. Kulkarni, “Intelligent Software Engineering: The Significance of Artificial Intelligence Techniques in Enhancing Software Development Lifecycle Processes,” in Intelligent Systems Design and Applications (ISDA 2021), A. Abraham et al., Eds., vol. 418, Cham: Springer, 2022, https://doi.org/10.1007/978-3-030-96308-8_7

[13] J. Pachouly, S. Ahirrao, K. Kotecha, G. Selvachandran, and A. Abraham, “A systematic literature review on software defect prediction using artificial intelligence: Datasets, Data Validation Methods, Approaches, and Tools,” Engineering Applications of Artificial Intelligence, vol. 111, p. 104773, 2022, https://doi.org/10.1016/j.engappai.2022.104773

[14] L. Longo et al., “Explainable Artificial Intelligence (XAI) 2.0: A manifesto of open challenges and interdisciplinary research directions,” Information Fusion, vol. 106, p. 102301, 2024, https://doi.org/10.1016/j.inffus.2024.102301

[15] C. Selvaraj, I. Chandra, and S. K. Singh, “Artificial intelligence and machine learning approaches for drug design: challenges and opportunities for the pharmaceutical industries,” Molecular Diversity, vol. 26, pp. 1893–1913, 2022, https://doi.org/10.1007/s11030-021-10326-z

[16] Y. M. Al-Worafi, “Artificial Intelligence and Machine Learning for Drug Safety,” in Technology for Drug Safety, Cham: Springer, 2023, https://doi.org/10.1007/978-3-031-34268-4_7

[17] B. Hutchinson et al., “Towards accountability for machine learning datasets: Practices from software engineering and infrastructure,” in Proc. 2021 ACM Conf. on Fairness, Accountability, and Transparency, Mar. 2021, pp. 560–575, https://doi.org/10.1145/3442188.3445918

[18] R. I. Mukhamediev et al., “Review of artificial intelligence and machine learning technologies: Classification, restrictions, opportunities and challenges,” Mathematics, vol. 10, no. 15, p. 2552, 2022, https://doi.org/10.3390/math10152552

[19] A. Aldoseri, K. N. Al-Khalifa, and A. M. Hamouda, “Re-thinking data strategy and integration for artificial intelligence: concepts, opportunities, and challenges,” Applied Sciences, vol. 13, no. 12, p. 7082, 2023, https://doi.org/10.3390/app13127082

[20] A. Tariq et al., “Software measurement by using artificial intelligence,” Journal of Nanomaterials, vol. 2022, no. 1, p. 7283171, 2022, https://doi.org/10.1155/2022/7283171

[21] H. Sofian, N. A. M. Yunus and R. Ahmad, "Systematic Mapping: Artificial Intelligence Techniques in Software Engineering," in IEEE Access, vol. 10, pp. 51021-51040, 2022, https://doi.org/10.1109/ACCESS.2022.3174115

[22] N. A. Perifanis and F. Kitsios, “Investigating the influence of artificial intelligence on business value in the digital era of strategy: A literature review,” Information, vol. 14, no. 2, p. 85, 2023, https://doi.org/10.3390/info14020085

[23] J. Wang, Y. Huang, C. Chen, Z. Liu, S. Wang and Q. Wang, "Software Testing With Large Language Models: Survey, Landscape, and Vision," in IEEE Transactions on Software Engineering, vol. 50, no. 4, pp. 911-936, April 2024, https://doi.org/10.1109/TSE.2024.3368208

[24] N. Alshahwan, M. Harman and A. Marginean, "Software Testing Research Challenges: An Industrial Perspective," 2023 IEEE Conference on Software Testing, Verification and Validation (ICST), Dublin, Ireland, 2023, pp. 1-10, https://doi.org/10.1109/ICST57152.2023.00008

[25] M. Krichen, “How Artificial Intelligence Can Revolutionize Software Testing Techniques,” in Innovations in Bio-Inspired Computing and Applications. IBICA 2022, A. Abraham, A. Bajaj, N. Gandhi, A. M. Madureira, and C. Kahraman, Eds., vol. 649, Lecture Notes in Networks and Systems, Cham: Springer, 2023, https://doi.org/10.1007/978-3-031-27499-2_18

[26] A. Ramalingam, J. A. Milan, and M. A. Mani, “Artificial Intelligence and Machine Learning in Software Development”, Quing: Int. J. Innov. Res. Sci. Eng., vol. 2, no. 2, pp. 78–86, Aug. 2024.

[27] A. Fan et al., "Large Language Models for Software Engineering: Survey and Open Problems," 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE), Melbourne, Australia, 2023, pp. 31-53, https://doi.org/10.1109/ICSE-FoSE59343.2023.00008

[28] F. Ricca, A. Marchetto and A. Stocco, "AI-based Test Automation: A Grey Literature Analysis," 2021 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW), Porto de Galinhas, Brazil, 2021, pp. 263-270, https://doi.org/10.1109/ICSTW52544.2021.00051

[29] D. K. Kanbach et al., “The GenAI is out of the bottle: generative artificial intelligence from a business model innovation perspective,” Review of Managerial Science, vol. 18, pp. 1189–1220, 2024, https://doi.org/10.1007/s11846-023-00696-z

[30] A. Atadoga, U. J. Umoga, O. A. Lottu, and E. O. Sodiy, “Tools, techniques, and trends in sustainable software engineering: A critical review of current practices and future directions,” World Journal of Advanced Engineering Technology and Sciences, vol. 11, no. 1, pp. 231–239, 2024, https://doi.org/10.30574/wjaets.2024.11.1.0051

[31] S. Martínez-Fernández et al., “Software engineering for AI-based systems: a survey,” ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 31, no. 2, pp. 1–59, 2022, https://doi.org/10.1145/3487043

[32] J. M. Ali, “AI-driven software engineering,” Advances in Engineering Innovation, vol. 3, pp. 0–0, 2023, https://10.54254/2977-3903/3/2023030

[33] I. Ozkaya, "Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications," in IEEE Software, vol. 40, no. 3, pp. 4-8, May-June 2023, https://doi.org/10.1109/MS.2023.3248401

[34] S. Haldar, M. Pierce and L. Fernando Capretz, "Exploring the Integration of Generative AI Tools in Software Testing Education: A Case Study on ChatGPT and Copilot for Preparatory Testing Artifacts in Postgraduate Learning," in IEEE Access, vol. 13, pp. 46070-46090, 2025, https://doi.org/10.1109/ACCESS.2025.3545882

[35] G. Giray, “A software engineering perspective on engineering machine learning systems: State of the art and challenges,” Journal of Systems and Software, vol. 180, p. 111031, 2021, https://doi.org/10.1016/j.jss.2021.111031

[36] A. Aleti, "Software Testing of Generative AI Systems: Challenges and Opportunities," *2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE)*, Melbourne, Australia, 2023, pp. 4-14, https://doi.org/10.1109/ICSE-FoSE59343.2023.00009

[37] S. Gupta, S. S. Gupta, K. McMath, and S. Sugandh, “Enhancing complex wound care by leveraging artificial intelligence: an artificial intelligence chatbot software study,” Wounds, vol. 35, no. 8, pp. E265–E267, Aug. 2023, https://doi.org/10.25270/wnds/23073. PMID: 37643453

[38] N. Siegmund, J. Dorn, M. Weber, C. Kaltenecker and S. Apel, "Green Configuration: Can Artificial Intelligence Help Reduce Energy Consumption of Configurable Software Systems?," in Computer, vol. 55, no. 3, pp. 74-81, March 2022, https://doi.org/10.1109/MC.2021.3120048

[39] Y. Yang, X. Xia, D. Lo, and J. Grundy, “A survey on deep learning for software engineering,” ACM Computing Surveys (CSUR), vol. 54, no. 10s, pp. 1–73, 2022, https://doi.org/10.1145/3505243

[40] L. Layman and R. Vetter, "Generative Artificial Intelligence and the Future of Software Testing," in Computer, vol. 57, no. 1, pp. 27-32, Jan. 2024, https://doi.org/10.1109/MC.2023.3306998

[41] P. Morais, M. J. Ferreira and B. Veloso, "Improving Student Engagement With Project-Based Learning: A Case Study in Software Engineering," in IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, vol. 16, no. 1, pp. 21-28, Feb. 2021, https://doi.org/10.1109/RITA.2021.3052677

[42] H. Hourani, A. Hammad and M. Lafi, "The Impact of Artificial Intelligence on Software Testing," 2019 IEEE Jordan International Joint Conference on Electrical Engineering and Information Technology (JEEIT), Amman, Jordan, 2019, pp. 565-570, https://doi.org/10.1109/JEEIT.2019.8717439

[43] R. Lima, A. M. R. da Cruz and J. Ribeiro, "Artificial Intelligence Applied to Software Testing: A Literature Review," 2020 15th Iberian Conference on Information Systems and Technologies (CISTI), Seville, Spain, 2020, pp. 1-6, https://doi.org/10.23919/CISTI49556.2020.9141124

[44] J. Gao, C. Tao, D. Jie and S. Lu, "Invited Paper: What is AI Software Testing? and Why," 2019 IEEE International Conference on Service-Oriented System Engineering (SOSE), San Francisco, CA, USA, 2019, pp. 27-2709, https://doi.org/10.1109/SOSE.2019.00015

[45] M. Islam, F. Khan, S. Alam and M. Hasan, "Artificial Intelligence in Software Testing: A Systematic Review," *TENCON 2023 - 2023 IEEE Region 10 Conference (TENCON)*, Chiang Mai, Thailand, 2023, pp. 524-529, https://doi.org/10.1109/TENCON58879.2023.10322349

[46] C. Tao, J. Gao and T. Wang, "Testing and Quality Validation for AI Software–Perspectives, Issues, and Practices," in IEEE Access, vol. 7, pp. 120164-120175, 2019, https://doi.org/10.1109/ACCESS.2019.2937107

[47] V. Bayrı and E. Demirel, "AI-Powered Software Testing: The Impact of Large Language Models on Testing Methodologies," 2023 4th International Informatics and Software Engineering Conference (IISEC), Ankara, Turkiye, 2023, pp. 1-4, https://doi.org/10.1109/IISEC59749.2023.10391027

[48] Y. Liang, "Application of Artificial Intelligence Algorithm and Deductive Database in Special Scene Recognition," 2022 4th International Conference on Smart Systems and Inventive Technology (ICSSIT), Tirunelveli, India, 2022, pp. 1012-1015, https://doi.org/10.1109/ICSSIT53264.2022.9716323

[49] V. H. S. Durelli et al., "Machine Learning Applied to Software Testing: A Systematic Mapping Study," in IEEE Transactions on Reliability, vol. 68, no. 3, pp. 1189-1212, Sept. 2019, https://doi.org/10.1109/TR.2019.2892517

[50] D. Lo, "Trustworthy and Synergistic Artificial Intelligence for Software Engineering: Vision and Roadmaps," 2023 IEEE/ACM International Conference on Software Engineering: Future of Software Engineering (ICSE-FoSE), Melbourne, Australia, 2023, pp. 69-85, https://doi.org/10.1109/ICSE-FoSE59343.2023.00010

[51] S. Jalil, S. Rafi, T. D. LaToza, K. Moran and W. Lam, "ChatGPT and Software Testing Education: Promises & Perils," *2023 IEEE International Conference on Software Testing, Verification and Validation Workshops (ICSTW)*, Dublin, Ireland, 2023, pp. 4130-4137, https://doi.org/10.1109/ICSTW58534.2023.00078

[52] M. Alcon, H. Tabani, J. Abella and F. J. Cazorla, "Enabling Unit Testing of Already-Integrated AI Software Systems: The Case of Apollo for Autonomous Driving," 2021 24th Euromicro Conference on Digital System Design (DSD), Palermo, Italy, 2021, pp. 426-433, https://doi.org/10.1109/DSD53832.2021.00071

[53] B. Orazbayev et al., "System of models for simulation and optimization of operating modes of a delayed coking unit in a fuzzy environment," *Sci Rep*, vol. 13, no. 1, p. 14317, Aug. 2023, https://doi.org/10.1038/s41598-023-41455-0

[54] C. Ebert, D. Bajaj and M. Weyrich, "Testing Software Systems," in IEEE Software, vol. 39, no. 4, pp. 8-17, July-Aug. 2022, https://doi.org/10.1109/MS.2022.3166755

[55] W. Villegas-Ch, J. García-Ortiz and S. Sánchez-Viteri, "Toward Intelligent Monitoring in IoT: AI Applications for Real-Time Analysis and Prediction," in IEEE Access, vol. 12, pp. 40368-40386, 2024, https://doi.org/10.1109/ACCESS.2024.3376707

[56] Y. Wang, S. Guo and C. W. Tan, "From Code Generation to Software Testing: AI Copilot with Context-Based RAG," in IEEE Software, https://doi.org/10.1109/MS.2025.3549628

[57] A. Kopesbayeva, A. Auezova, M. Adambaev, and A. Kuttybayev, "Research and development of software and hardware modules for testing technologies of rock mass blasting preparation," in *New Developments in Mining Engineering 2015: Theoretical and Practical Solutions of Mineral Resources Mining*. CRC Press, 2015, pp. 185–192. https://doi.org/10.1201/b19901-33

[58] D. Kobets, O. Vorkunova, L. Yaremenko, V. Krasnoshchok, and O. Zhurba, "Using big data to increase the efficiency of business processes in the digital economy of Ukraine," *Periodicals of Engineering and Natural Sciences*, vol. 13, no. 1, pp. 97–110, 2025. https://doi.org/10.21533/pen.v13.i1.279

[59] M. Potwora, O. Vdovichena, D. Semchuk, L. Lipych, and V. Saienko, "The use of artificial intelligence in marketing strategies: Automation, personalization and forecasting," *Journal of Management World*, vol. 2, pp. 41–49, 2024. https://doi.org/10.53935/jomw.v2024i2.275

[60] B. Orazbayev, E. Ospanov, N. Kissikova, N. Mukataev, and K. Orazbayeva, "Decision-making in the fuzzy environment on the basis of various compromise schemes," *Procedia Computer Science*, vol. 120, pp. 945–952, 2017. https://doi.org/10.1016/j.procs.2017.11.330

[61] I. Hunko, O. Muliarevych, R. Trishchuk, S. Zybin, and P. Halachev, "The role of virtual reality in improving software testing methods and tools," *Journal of Theoretical and Applied Information Technology*, vol. 102, no. 11, pp. 4723–4734, 2024. [Online]. Available: https://www.jatit.org/volumes/Vol102No11/6Vol102No11.pdf

[62] D. Roy, M. Padhiary, P. Roy, and J. Akhtar Barbhuiya, "Artificial intelligence-driven smart aquaculture: Revolutionizing sustainability through automation and machine learning," *EthAIca*, vol. 4, p. 398, Jul. 2025. https://doi.org/10.56294/ai2025398