Journal of Computer Science and Information Technology
ISSN: 3080-3586
DOI: 10.61424/jcsit
Journal Homepage: www.bluemarkpublishers.com/index.php/JCSIT

JCSIT
BLUEMARK PUBLISHERS

| RESEARCH ARTICLE

Leveraging Artificial Intelligence for Software Development and System Design

Mikita Piastou
University of West Georgia, Carrollton, GA, United States
Corresponding Author: Mikita Piastou, **E-mail**: piastoumikita@gmail.com

| ABSTRACT

This paper constitutes a comprehensive review of the scientific literature pertaining to the application of artificial intelligence (AI) in the process of code writing and the development of software and systems. It analyses the influence of AI on the various stages of the software development life cycle. The potential of AI to automate code generation and enhance its quality is examined, with quantitative data on productivity improvements and error reduction providing a foundation for this analysis. The paper presents a synopsis of research conducted by prominent companies on the utilization of artificial intelligence within the domain of software development. The study concludes with the identification of future trends and potential research directions in this dynamic field.

| KEYWORDS

AI, coding, computer system, software, machine learning

| ARTICLE INFORMATION

ACCEPTED: 07 August 2025 **PUBLISHED:** 10 October 2025 **DOI:** 10.61424/jcsit.v2.i1.467

1. Introduction

These days, AI is really shaking up how software gets made. It's moved beyond just writing code. Now it's pitching in with testing and even helping manage projects. As software systems become more complex, it's clear that developers need to get comfortable with AI to not just to keep up, but to really take advantage of what it can do (Wang et al., 2024).

The requirements for software in the present age are subject to constant growth, thus pushing the limits of conventional development methodologies. The utilization of AI has emerged as a promising solution by automating and enhancing various facets of the process, thereby rendering its study highly pertinent. The central aim of this paper is to furnish a thorough review of scientific publications concerning the utilization of AI for code writing and software development in complex information systems.

The specific objectives of this study are as follows: firstly, to analyze the impact of AI on the various stages of the software development life cycle; secondly, to examine the use of AI for automated code generation and code quality improvement; thirdly, to evaluate quantitative and qualitative data on the effectiveness of AI in development; fourthly, to review the research of leading AI companies in this area; fifthly, to discuss the challenges, limitations, and ethical considerations associated with AI implementation; and finally, to identify future trends and potential research directions.

2. Materials and Methods

This review drew upon English-language research studies, analyses, and reports from major academic databases, including Scopus, Web of Science, PubMed, and MDPI, with a focus on software development and AI. To provide real-world context, information from the research teams of leading AI companies such as Google, Microsoft, and OpenAI, reports and analyses (e.g., McKinsey), and technology platforms (e.g., GitHub) were included. A review of the servicix.com website was conducted to ascertain the currency of the information it contains.

The selection of literature focused on studies published predominantly between 2019 and 2025 to cover the most recent advances. The search terms included "AI in software development," "AI for code writing," "generative AI in software engineering," "LLMs for code," and related terms. A balanced assessment was ensured by incorporating systematic literature reviews and empirical, quantitative studies (Alenezi & Akour, 2025; Pan et al., 2024; Kokol, 2024; Karlovs-Karlovskis, 2024).

The analysis included summarizing key findings, identifying common themes, and extracting quantitative and qualitative data about the impact and effectiveness of AI in software engineering.

3. Results and Discussion

3.1 Impact of AI on different phases of the software development life cycle

The implementation of artificial intelligence in the field of software development constitutes a discernible trade-off between substantial potential benefits and considerable challenges that necessitate meticulous consideration (Kiviniemi, 2024; Alenezi & Akour, 2025). The advantages of this approach are manifold, including increased developer productivity, faster development cycles, improved code quality and reliability, automation of repetitive tasks, better resource allocation, optimized project management, and reduced development and expertise costs (Pan et al., 2024; Kokol, 2024). The disadvantages of utilizing AI in software development include issues related to code accuracy and reliability, ethical implications, and the presence of security vulnerabilities in AI-generated code (Lulichac et al., 2025; Ashrafi et al., 2025).

The necessity for human oversight and validation, potential skills shortages, resistance to change, and the lack of holistic frameworks that integrate AI throughout must also be considered. In the realm of artificial intelligence, ethical considerations play a pivotal role in ensuring the integrity and fairness of the algorithms employed. A key concern is the potential for bias, which can result in analyses that are not representative of the population or produce inequitable outcomes. This bias can stem from over-reliance on automation, which may hinder the development of human skills and compromise code quality. Effective utilization of AI tools requires a workforce proficient in both AI methodologies and software development (Alenezi & Akour, 2025; Kiviniemi, 2024).

New software development training programs will be necessary to equip developers with these skills. The objective of forthcoming research endeavors is to enhance AI capabilities to address current limitations in software modeling and improve the reliability of these tools. There is a clear need for specialized AI tools tailored to areas such as safety-critical systems (Ashrafi et al., 2025; Kiviniemi, 2024).

Ongoing research in multi-agent interaction and runtime debugging holds considerable promise for improving AI-assisted code generation. AI's potential in this field is wide-ranging: it can suggest optimal software architectures, user interface and user experience layouts, and system designs based on given constraints. It can also generate layouts and specifications and assist in defining and reusing solution architectures (Ashrafi et al., 2025; Wang et al., 2024).

The use of AI algorithms can improve the accuracy of cost and scope estimates, as well as project timelines. AI can analyze vast amounts of data on successful projects and patterns, providing developers with informed recommendations that might not be immediately apparent (Pan et al., 2024; Karlovs-Karlovskis, 2024).

AI-based tools such as GitHub Copilot, DeepMind's AlphaCode, and others have demonstrated the ability to generate code snippets, complete functions, and solve complex programming problems. Additionally, AI enhances user experience through intelligent autocompletion and code suggestions. Numerous tools and research efforts dedicated to code generation underscore its central role in integrating AI into software development (Microsoft, 2018; OpenAI, 2021; Nikolov et al., 2025).

The utilization of AI within software testing can significantly improve effectiveness. AI integration automates testing processes, predicts code segments likely to contain defects, and identifies vulnerabilities. AI-driven testing enables automatic generation of test cases, streamlining the process and improving defect detection. Furthermore, AI can facilitate mutation testing, which is traditionally labor-intensive and often does not cover all scenarios. AI broadens the scope and efficacy of testing, helping identify latent defects (Finio & Downie, 2024; Ashrafi et al., 2025).

The integration of AI can also enhance continuous integration and continuous delivery (CI/CD) pipelines, facilitating application deployment management across diverse environments. AI aids in identifying areas for code refactoring and optimization post-deployment, continuous performance monitoring, anomaly detection, and problem prediction (Kiviniemi, 2024; Finio & Downie, 2024).

Additionally, AI models can produce natural language documentation, automatically generate documentation alongside code development, and create API guides and code explanations (OpenAI, 2021).

3.2 Quantitative data on the effectiveness of AI

Researchers have found that developers using GitHub Copilot were able to complete coding tasks approximately 55% faster than their non-Copilot counterparts. According to a report by McKinsey, the use of generative AI tools by development teams has been found to accelerate coding task completion by up to 100%. The use of Copilot in code generation was also associated with a higher success rate in passing contemporary benchmarks on the initial attempt (Microsoft, 2018).

The implementation of Microsoft IntelliCode resulted in a 30% increase in coding speed and a 25% reduction in errors. Users of Snyk Code reported a 40% increase in early-stage error detection and a 35% reduction in security vulnerabilities (Finio & Downie, 2024).

Google DeepMind's AlphaCode demonstrated a 65% success rate in competitive programming tasks and enhanced developer productivity by 25% (Nikolov et al., 2025).

Figure 1. Quantitative concerns about AI adoption

A survey revealed that 74% of respondents agreed that AI reduced the time required for routine coding tasks, while 62% observed a decline in post-release defects in projects employing AI assistance. Teams that received formal AI training exhibited a 30% increase in efficiency. A study of IBM's AI-based defect prediction tool showed a 20% reduction in post-release defects and a 15% increase in defect detection rates during development (Finio & Downie, 2024).

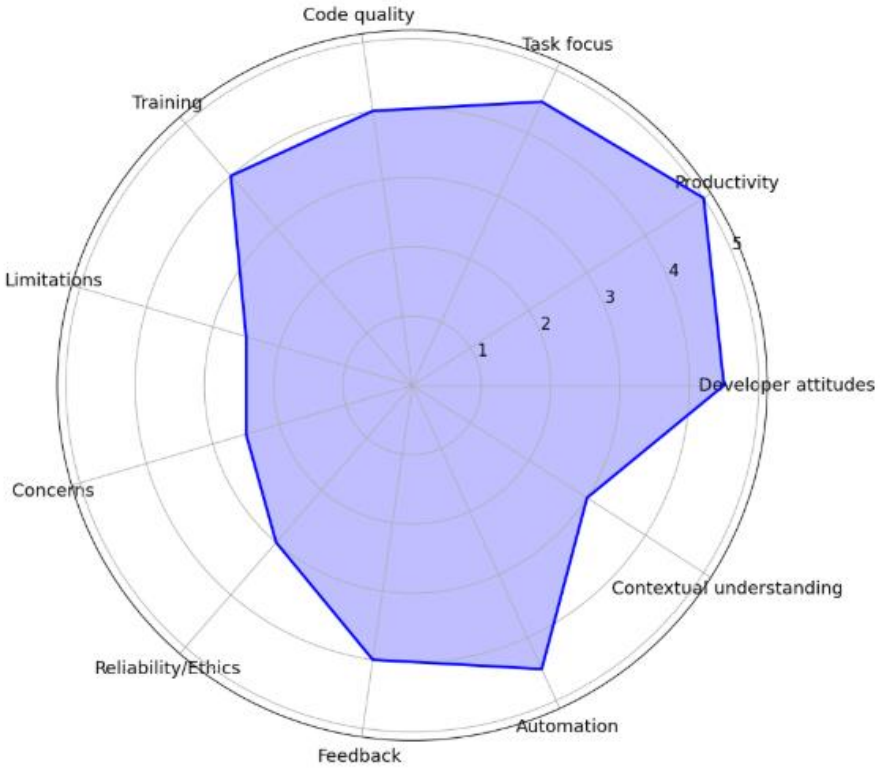

Figure 2. Qualitative assessment of AI in development work

Quantitative data strongly supports the claim that AI tools can enhance developer productivity, accelerate development time, and improve code quality by reducing bugs and vulnerabilities. The consistent positive results across various tools and studies provide substantial evidence of the practical benefits of AI in complex software development (Pan et al., 2024; Kiviniemi, 2024).

Table 1. Qualitative assessments of the impact of AI on software development

Aspect	**Qualitative assessment**
Developer attitudes	Generally positive, seen as an enabler, improving efficiency
Productivity	Significant improvement
Task focus	Frees up for more complex tasks
Code quality	Improves
Training	Can serve as an instructor
Limitations	Difficulty with complex tasks
Concerns	Job cuts, code proficiency
Reliability, ethical considerations	Requires human oversight
Feedback	Offers real-time feedback
Automation of routine tasks	Helps automate repetitive and boring tasks
Contextual understanding	Sometimes may not meet complex project requirements
Trust in AI	Only 40% of respondents expressed complete trust in AI-generated solutions
Lack of training	58% of respondents felt that insufficient AI training among team members hinders full implementation
Cost of implementation	High upfront costs are a significant barrier to AI adoption for 52% of respondents
Resistance to change	Adoption of AI technologies may be met with resistance from practitioners accustomed to traditional methodologies

3.3 AI research for software development from leading companies

Google's research encompasses methodologies for the automated synthesis of specialized hash functions that demonstrate superior performance compared to conventional libraries. Google is leveraging generative AI to facilitate internal code migration in substantial codebases, such as Google Ads, in conjunction with abstract syntax tree–based methodologies. Google is also developing multi-agent AI frameworks, such as Paper2Code, with the objective of automating the generation of code from scientific articles (Research. Google; Nikolov et al., 2025).

It is evident from reports from 2024 that a substantial proportion of Google's code is authored by AI. Microsoft offers tools such as GitHub Copilot (developed with OpenAI) and IntelliCode, which have been shown to result in significant improvements in productivity and quality (Microsoft, 2018; OpenAI, 2021).

The primary focus of research endeavors is to enhance the efficacy of AI-assisted code generation through the utilization of multi-agent interaction and runtime debugging methodologies. Microsoft is also developing Language Agents to automate Windows OS, indicating a broader application of AI that extends beyond coding. The primary focus of OpenAI's research is the evaluation and enhancement of AI's capacity to adhere meticulously to instructions when generating code in multiple languages. Furthermore, uncertainty in code generation models and methods for highlighting potential errors are also being explored. OpenAI is developing models such as o3, o4, and GPT-5, with enhanced reasoning capabilities to improve problem-solving and code generation efficiency (OpenAI, 2021; Ashrafi et al., 2025).

4. Conclusion

The article has brought out the manner in which AI impacts code writing and software development. The use of AI tools and techniques has been discovered to have tremendous benefits in terms of productivity, quality and efficiency of code in accordance with quantitative and qualitative information.

It must be understood that prominent figures in the art of artificial intelligence are leading R&D for this specific industry, thus advancing innovation in the field of code generation. Nevertheless, it must be known that there are risks involved in AI adoption in the form of ethics, safety, reliability, and the development of appropriate skills.

The author is sure that to keep up-to-date on the influence of AI in software development, articles must cover recent scientific research on the problem of the creation of specialized AI tools developed in connection with sophisticated information systems, forming reliable ethical rules and legal regulation, and multi-agent collaboration.

Funding: This research received no external funding.
Conflicts of Interest: The authors declare no conflict of interest.
Publisher's Note: All claims expressed in this article are solely those of the authors and do not necessarily represent those of their affiliated organizations, or those of the publisher, the editors and the reviewers.

References

[1] Alenezi, M., Akour, M. (2025). AI-Driven Innovations in Software Engineering: A Review of Current Practices and Future Directions. Applied Sciences 15, 1344. https://doi.org/10.3390/app15031344
[2] Antti K. (2024). Automation and AI in Software Development: Research Trends, Advantages, Disadvantages, Threats, and Opportunities. Master's Thesis, 60 pages. https://erepo.uef.fi/server/api/core/bitstreams/1dbdc2fe-50d2-40c6-9f1d-9d53d4209b03/content
[3] Ashrafi, N., Bouktif, S., Mediani, M. (2025). Enhancing LLM Code Generation: A Systematic Evaluation of Multi-Agent Collaboration and Runtime Debugging for Improved Accuracy, Reliability, and Latency. https://doi.org/10.48550/ARXIV.2505.02133
[4] Finio, M. and Downie, A. (2024). AI in software development. [online] Ibm.com. Available at: https://www.ibm.com/think/topics/ai-in-software-development
[5] Karlovs-Karlovskis, U. (2024). Generative Artificial Intelligence Use in Optimising Software Engineering Process: A Systematic Literature Review. Applied Computer Systems 29, 68–77. https://doi.org/10.2478/acss-2024-0009
[6] Kokol, P. (2024). The Use of AI in Software Engineering: A Synthetic Knowledge Synthesis of Recent Research Literature. Information 15, 354. https://doi.org/10.3390/info15060354
[7] Lulichac Ramos, G., Pantoja Payajo, F.O., Torres Villanueva, M. (2025). La IA Generativa en el Desarrollo de Software: Impacto en Diversas Industrias. Innov. softw. 6, 76–100. https://doi.org/10.48168/innosoft.s23.a259
[8] Microsoft (2018). Microsoft Research. [online] Microsoft Research. Available at: https://www.microsoft.com/en-us/research/
[9] Nikolov, S., Codecasa, D., Sjovall, A., Tabachnyk, M., Chandra, S., Taneja, S., Ziftci, C. (2025). How is Google using AI for internal code migrations? https://doi.org/10.48550/ARXIV.2501.06972
[10] OpenAI (2021). OpenAI Research. [online] OpenAI. Available at: https://openai.com/research/
[11] Pan, S., Wang, L., Zhang, T., Xing, Z., Zhao, Y., Lu, Q., Sun, X. (2024). "I Don't Use AI for Everything": Exploring Utility, Attitude, and Responsibility of AI-empowered Tools in Software Development. https://doi.org/10.48550/ARXIV.2409.13343
[12] Research.google. (n.d.). The latest research from Google [online] Available at: https://research.google/blog/

[13] Wang, L., Yang, F., Zhang, C., Lu, J., Qian, J., He, S., Zhao, P., Qiao, B., Huang, R., Qin, S., Su, Q., Ye, J., Zhang, Y., Lou, J.-G., Lin, Q., Rajmohan, S., Zhang, D., Zhang, Q. (2024). Large Action Models: From Inception to Implementation. https://doi.org/10.48550/ARXIV.2412.10047