Dhanushree K R, 2026, 14:4
ISSN (Online): 2348-4098
ISSN (Print): 2395-4752

International Journal of Science, Engineering and Technology
An Open Access Journal

Generative AI in Software Development

Dhanushree K R[1], Aniket Singh[2], Dr. V. Sathya, Associate Professor[3]
Department of MCA, T. John Institute of Technology, VTU[1,2]
Department of Computer Application T. John Institute of Technology, VTU[3]

Abstract- Generative AI is transforming software development through code generation, testing, debugging, documentation, and project support. This review examines the evolution, applications, benefits, challenges, and future scope of Generative AI in software engineering.Generative Artificial Intelligence (Generative AI) has emerged as a revolutionary technology that is transforming software development by automating coding tasks, improving productivity, and supporting developers throughout the Software Development Life Cycle (SDLC). Powered by Large Language Models (LLMs), Generative AI enables intelligent code generation, debugging, software testing, documentation, and code review, thereby reducing development time and minimizing human effort. This review paper presents a comprehensive analysis of the role of Generative AI in modern software engineering by examining recent research, industrial applications, and technological advancements. The paper discusses the adoption of AI-powered tools such as GitHub Copilot, ChatGPT, Amazon CodeWhisperer, and Google Gemini, highlighting their contributions to improving software quality and developer efficiency. Furthermore, it explores the key benefits of Generative AI, including enhanced productivity, faster development cycles, improved collaboration, and support for novice programmers. Despite these advantages, several challenges remain, such as data privacy concerns, security vulnerabilities, inaccurate code generation, copyright issues, and ethical considerations associated with AI-generated content. The paper also examines future research directions, including autonomous software engineering, AI-assisted DevOps, personalized coding assistants, and responsible AI governance. Based on an extensive review of recent literature, this study concludes that Generative AI has significant potential to reshape the future of software development when combined with human expertise and robust ethical practices. The findings provide valuable insights for researchers, educators, software professionals, and students interested in understanding the evolving impact of Generative AI on software engineering.

Keywords— Generative AI, Software Development, Large Language Models, Code Generation, Software Engineering, Artificial IntelligenceGenerative Artificial Intelligence (Generative AI), Artificial Intelligence (AI), Software Development, Software Engineering, Large Language Models (LLMs), Code Generation, AI-Assisted Programming, Intelligent Code Completion, Software Development Life Cycle (SDLC), Automated Software Testing, Code Review Automation, Software Debugging, Machine Learning, Deep Learning, Natural Language Processing (NLP), GitHub Copilot, ChatGPT, Google Gemini, Amazon CodeWhisperer, AI Ethics, Cybersecurity, Software Quality Assurance, DevOps, Cloud Computing, Developer Productivity, Intelligent Software Systems.

I. INTRODUCTION

Generative Artificial Intelligence has emerged as one of the most influential technologies in modern software engineering. Tools such as ChatGPT, GitHub Copilot, and Gemini assist developers in writing code, generating documentation, and improving productivity. Generative Artificial Intelligence (Generative AI) has emerged as one of the most significant technological advancements in

© 2026 Dhanushree K R This is an Open Access article distributed under the terms of the Creative Commons Attribution License (http://creativecommons.org/licenses/by/4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly credited.

the field of computer science and software engineering. Unlike traditional Artificial Intelligence (AI) systems that are designed to analyze data and make predictions, Generative AI has the ability to create new content such as text, source code, images, audio, videos, and software solutions by learning patterns from large datasets. The rapid evolution of Large Language Models (LLMs), including OpenAI's ChatGPT, GitHub Copilot, Google Gemini, Amazon CodeWhisperer, and Meta Code Llama, has transformed the way software is designed, developed, tested, and maintained. These intelligent systems assist developers by generating high-quality source code, explaining programming concepts, detecting errors, producing technical documentation, and automating repetitive software development tasks.

Software development has traditionally been a complex, time-consuming, and resource-intensive process requiring extensive programming knowledge, manual coding, debugging, testing, and maintenance. Developers often spend a significant amount of time writing repetitive code, fixing bugs, reviewing software quality, and preparing documentation. The increasing demand for faster software delivery, higher-quality applications, and reduced development costs has encouraged organizations to adopt AI-powered development tools. Generative AI addresses these challenges by enabling developers to automate routine programming activities, improve coding efficiency, reduce development time, and enhance overall software quality.

The integration of Generative AI into the Software Development Life Cycle (SDLC) has significantly changed modern software engineering practices. AI-powered coding assistants provide real-time code suggestions, automatically generate functions from natural language descriptions, identify programming errors, recommend optimized solutions, and assist in software testing and documentation. These capabilities improve developer productivity while allowing programmers to focus on solving complex business problems rather than repetitive coding tasks. Consequently, Generative AI is becoming an essential component of modern software engineering environments across industries such as healthcare, finance, education, manufacturing, e-commerce, and cloud computing.

In recent years, both academia and industry have shown increasing interest in the adoption of Generative AI for software development. Technology companies have introduced advanced AI-assisted programming platforms capable of supporting multiple programming languages and frameworks. These systems continuously learn from publicly available code repositories, software documentation, and programming best practices to provide intelligent recommendations. The widespread availability of these tools has made software development more accessible not only to experienced programmers but also to beginners and students who require guidance while learning programming concepts.

Despite its numerous advantages, the adoption of Generative AI also introduces several technical, ethical, and legal challenges. AI-generated code may contain security vulnerabilities, logical errors, outdated programming practices, or biased outputs. Since these models are trained using large-scale datasets collected from various sources, concerns regarding intellectual property rights, copyright infringement, data privacy, and responsible AI usage continue to grow. Furthermore, excessive dependence on AI-generated code may reduce developers' problem-solving abilities and programming skills if the generated outputs are accepted without proper verification. Therefore, human expertise remains essential for validating AI-generated solutions and ensuring software reliability, security, and maintainability.

This review paper aims to provide a comprehensive study of Generative AI in software development by examining its architecture, applications, benefits, challenges, ethical considerations, and future research directions. The paper reviews recent advancements in AI-assisted programming tools and discusses their impact on different phases of the Software Development Life Cycle. Additionally, it highlights current research trends, industrial adoption, and emerging opportunities that may

shape the future of intelligent software engineering. By summarizing existing literature and technological developments, this study offers valuable insights for researchers, software professionals, educators, and students seeking to understand the transformative role of Generative AI in modern software development.

Overall, Generative AI represents a paradigm shift in software engineering by enabling faster development, improved software quality, enhanced collaboration, and intelligent automation. As AI technologies continue to evolve, their integration with human creativity and domain expertise is expected to redefine the future of software development while promoting responsible, secure, and ethical use of artificial intelligence.

II. LITERATURE REVIEW

Recent studies indicate that Generative AI can significantly improve developer productivity and reduce development time. Researchers have explored AI-assisted coding, automated testing, bug detection, and software maintenance.

The rapid advancement of Generative Artificial Intelligence (Generative AI) has significantly transformed the field of software engineering by introducing intelligent systems capable of generating source code, debugging programs, creating documentation, and automating software development tasks. Recent studies indicate that Large Language Models (LLMs) have become valuable tools for improving developer productivity, reducing software development time, and enhancing code quality. This section reviews major research contributions related to the application of Generative AI in software development.

Chen et al. (2021) introduced OpenAI Codex, one of the earliest large language models specifically designed for code generation. The study demonstrated that Codex can convert natural language descriptions into executable source code across multiple programming languages. The researchers concluded that AI-assisted code generation significantly improves programming efficiency and reduces repetitive coding tasks. However, they also highlighted limitations such as incorrect code generation and dependency on high-quality training datasets.

Nijkamp et al. (2022) presented CodeGen, an open-source language model developed for program synthesis and intelligent code generation. The study compared CodeGen with existing AI coding assistants and reported that the model achieved competitive performance across several software engineering benchmarks. The authors emphasized that open-source code generation models could improve accessibility for researchers and educational institutions while encouraging further innovation in AI-assisted programming.

Li et al. (2022) proposed AlphaCode, an artificial intelligence system developed by DeepMind for solving competitive programming problems. The study demonstrated that AlphaCode successfully generated programming solutions that performed competitively with human programmers in coding competitions. The researchers concluded that Generative AI has the potential to solve complex programming challenges while supporting developers during algorithm design and implementation.

Objectives

The primary objective of this review paper is to analyze the role of Generative Artificial Intelligence (Generative AI) in transforming modern software development practices and to evaluate its impact on the Software Development Life Cycle (SDLC). The specific objectives of the study are:

- To study the concept and evolution of Generative Artificial Intelligence and its significance in modern software engineering.
- To analyze the applications of Generative AI in different phases of the Software Development Life Cycle, including code generation, debugging, software testing, documentation, and maintenance.
- To evaluate the effectiveness of AI-powered development tools such as ChatGPT, GitHub Copilot, Google Gemini, Amazon

CodeWhisperer, and other Large Language Model (LLM)-based coding assistants.
- To identify the benefits of Generative AI in improving developer productivity, software quality, development speed, and collaboration among software teams.
- To examine the challenges and limitations associated with AI-assisted software development, including security risks, privacy concerns, copyright issues, bias, and inaccurate code generation.

III. APPLICATIONS OF GENERATIVE AI

Code generation, debugging, documentation generation, test case generation, code review assistance, requirement analysis, and DevOps automation.

Automated Software Testing
Generative AI can automatically generate unit tests, integration tests, and test cases based on application requirements. AI systems identify different execution paths and create test scenarios to improve software quality. Automated testing helps developers detect defects early in the development process, reducing maintenance costs and improving application reliability.

Code Review and Quality Assurance
Code review is essential for maintaining software quality. Generative AI assists by analyzing source code for coding standard violations, security vulnerabilities, duplicate code, and inefficient programming practices. AI-generated suggestions help developers improve code readability, maintainability, and overall software quality while reducing manual review effort.

Software Documentation
Preparing software documentation is often considered a repetitive task by developers. Generative AI automatically generates technical documentation, API documentation, user manuals, installation guides, and code comments. Well-structured documentation improves software maintainability, facilitates knowledge sharing among team members, and supports future software enhancements.

Requirement Analysis
Generative AI supports software requirement analysis by converting user requirements written in natural language into structured software specifications. AI assists business analysts in identifying functional and non-functional requirements, reducing ambiguity, and improving communication between clients and development teams. This contributes to more accurate software planning and design.

Learning and Developer Assistance
Generative AI acts as an intelligent programming tutor by explaining programming concepts, algorithms, and coding techniques. Students and beginner developers can receive instant guidance, understand programming errors, and learn best coding practices through conversational AI systems. This makes AI a valuable educational tool in computer science and software engineering.

Challenges
Security vulnerabilities, hallucinated code, privacy concerns, copyright issues, model bias, and overreliance on AI-generated content. Although Generative Artificial Intelligence (Generative AI) has transformed software development by improving productivity and automating coding tasks, it also introduces several technical, ethical, legal, and security challenges. Organizations and developers must address these issues to ensure the responsible and effective use of AI-powered development tools.

Future Scope
Future systems may support autonomous software development, intelligent DevOps, personalized coding assistants, and advanced human-AI collaboration. Generative Artificial Intelligence (Generative AI) is rapidly evolving and is expected to play a transformative role in the future of software engineering. As Large Language Models (LLMs) become more accurate, efficient, and context-aware, their applications will extend beyond code generation to support intelligent decision-making, autonomous software development, and advanced

software maintenance. The future of Generative AI presents significant opportunities for researchers, developers, industries, and educational institutions.

Autonomous Software Development

One of the most promising future directions is the development of autonomous software engineering systems capable of designing, developing, testing, and maintaining software applications with minimal human intervention. AI systems may automate the entire Software Development Life Cycle (SDLC), enabling faster project completion while reducing development costs and human effort.

Advanced AI Coding Assistants

Future AI-powered programming assistants will provide more accurate, context-aware, and personalized coding support. These systems will understand project requirements, software architecture, and organizational coding standards to generate reliable, secure, and optimized code. They will also assist developers by explaining complex algorithms, suggesting best practices, and improving software quality.

IV. CONCLUSION

Generative AI is reshaping software engineering by improving efficiency and reducing development effort. Responsible adoption, governance, and validation remain essential Generative Artificial Intelligence (Generative AI) has emerged as a transformative technology in modern software development, significantly improving the efficiency, quality, and speed of software engineering processes. By leveraging Large Language Models (LLMs) and advanced machine learning techniques, Generative AI supports developers throughout the Software Development Life Cycle (SDLC), including requirement analysis, code generation, debugging, software testing, documentation, code review, and maintenance. AI-powered tools such as ChatGPT, GitHub Copilot, Google Gemini, and Amazon CodeWhisperer have demonstrated their ability to automate repetitive programming tasks, reduce development time, and enhance developer productivity.

This review highlights that Generative AI offers numerous benefits, including intelligent code assistance, automated documentation, improved software quality, and faster application development. It also provides valuable learning support for students and novice programmers by explaining programming concepts and generating code examples. Furthermore, organizations adopting AI-assisted development can improve collaboration, reduce operational costs, and accelerate software delivery while maintaining high development standards.

REFERENCES

1. M. Chen et al., "Evaluating Large Language Models Trained on Code," arXiv preprint arXiv:2107.03374, 2021.
2. E. Nijkamp et al., "CodeGen: An Open Large Language Model for Code Generation," arXiv preprint arXiv:2203.13474, 2022.
3. Y. Li et al., "Competition-Level Code Generation with AlphaCode," Science, vol. 378, no. 6624, pp. 1092–1097, 2022.
4. H. Pearce et al., "Asleep at the Keyboard? Assessing the Security of GitHub Copilot's Code Contributions," IEEE Symposium on Security and Privacy Workshops, 2022.
5. P. Vaithilingam, T. Zhang, and E. Glassman, "Expectation vs. Experience: Evaluating the Usability of AI Coding Assistants," CHI Conference on Human Factors in Computing Systems, 2022.