myresearchgo Volume 2, Issue 6, June 2026, ISSN: 3107-3816 (Online)

Impact of Generative AI on Software Development Productivity

Satyavan Harinath Rajbhar
7506768043
Satyavan229@gmail.com
Shree Ram College of Commerce, Bhandup.
Artificial Intelligence and Software Engineering
2025–2026
Rambaug Upvan, thane west-400606

Abstract

Generative Artificial Intelligence (AI) has rapidly evolved into a transformative force within the software industry, reshaping how developers write, test, document, and maintain code. Tools such as GitHub Copilot, ChatGPT, Amazon CodeWhisperer (now Amazon Q Developer), and Google Gemini Code Assist leverage large language models trained on vast code repositories to generate functional code snippets, suggest bug fixes, and automate repetitive engineering tasks. This review paper examines the impact of Generative AI on software development productivity, with particular attention to coding speed, code quality, debugging efficiency, documentation, and overall developer experience. The study is based on secondary data drawn from peer-reviewed journals, industry reports, conference proceedings, and case studies published by technology organizations. The paper further investigates the benefits, limitations, and ethical considerations associated with the adoption of Generative AI in software engineering, along with its likely future trajectory in the global and Indian software industry.

Keywords: Generative AI, Software Development Productivity, Large Language Models, AI Pair Programming, Code Generation, Software Engineering, Developer Experience.

1. Introduction

Software development has historically been a labour-intensive discipline that depends heavily on human expertise for writing, testing, and maintaining code. Over the past decade, advancements in Artificial Intelligence (AI), and more specifically in Generative AI, have begun to fundamentally alter this landscape. Generative AI refers to a class of AI systems capable of producing new content—including text, images, and computer code—based on patterns learned from massive datasets during training.

In the context of software engineering, Generative AI tools such as GitHub Copilot, OpenAI's ChatGPT and Codex models, Amazon Q Developer, and Google Gemini Code Assist have emerged as "AI pair programmers" that assist developers in real time. These tools can generate entire functions from natural-

language prompts, suggest the next lines of code as a developer types, identify bugs, write unit tests, and even produce technical documentation.

The global software industry has witnessed a sharp rise in the adoption of such tools since 2021, driven by improvements in large language model (LLM) architecture, the availability of vast public code repositories for training, and growing pressure on organizations to accelerate software delivery cycles. Industry surveys conducted by GitHub, Stack Overflow, and McKinsey & Company suggest that a majority of professional developers now use some form of AI coding assistant in their daily workflow.

This paper focuses on examining the impact of Generative AI on software development productivity through a structured review of existing literature, industry reports, and case studies, with the aim of understanding both the opportunities and the risks associated with this technological shift.

2. Objectives of the Study

The objectives of this study are:

- To understand the role and underlying mechanisms of Generative AI tools in software development.
- To study the major applications of Generative AI across the software development lifecycle.
- To analyze the impact of Generative AI on developer productivity, code quality, and delivery speed.
- To identify the benefits and challenges associated with the adoption of Generative AI coding tools.
- To examine the future scope of Generative AI in the global and Indian software industry.

3. Research Methodology

This study adopts a descriptive and analytical research design and is based entirely on secondary data.

3.1 Sources of Data

- Peer-reviewed research papers and academic journals.
- Books and textbooks related to Artificial Intelligence and Software Engineering.
- Conference proceedings from IEEE, ACM, and similar bodies.
- Industry reports published by GitHub, Stack Overflow, McKinsey & Company, and Gartner.
- Published studies available through Google Scholar, IEEE Xplore, and ACM Digital Library.

The information gathered from these sources has been synthesized and analyzed to understand the influence of Generative AI tools on software development productivity.

4. Literature Review

Vaithilingam, Zhang, and Glassman (2022)

Their usability study on GitHub Copilot found that AI-generated code suggestions reduced the time developers spent searching for solutions online, although participants occasionally struggled to understand and debug AI-generated snippets.

Peng, Kalliamvakou, Cihon, and Demirer (2023)

Their randomized controlled experiment on GitHub Copilot reported that developers who used the AI assistant completed a programming task significantly faster than developers who did not, providing some of the earliest quantitative evidence of productivity gains.

Ziegler et al. (2024)

Their large-scale survey-based study at GitHub linked perceived productivity gains from AI coding tools to factors such as reduced cognitive load, faster task completion, and increased developer satisfaction.

Stack Overflow Developer Survey (2024)

This annual industry survey found that a large majority of professional developers had either adopted or planned to adopt AI tools in their development workflow, citing efficiency and learning support as the primary motivations.

McKinsey & Company (2023)

This industry report estimated that Generative AI tools could substantially reduce the time required for tasks such as code generation, code refactoring, and documentation writing, depending on the complexity of the task and the experience level of the developer.

Sharma and Mehta (2024)

Their study on Indian IT firms observed that mid-sized and large software companies were increasingly integrating Generative AI tools into their development pipelines to remain competitive, while also raising concerns about data security and over-dependence.

The literature collectively suggests that Generative AI positively influences software development speed and developer satisfaction, while simultaneously introducing new challenges related to code reliability, security, and long-term skill development.

5. Generative AI in Software Development

Generative AI in the context of software engineering refers to AI systems capable of producing functional source code, test cases, documentation, and other engineering artefacts from natural-language instructions or partial code context.

Popular Generative AI tools used in software development include:

- GitHub Copilot
- OpenAI ChatGPT and Codex-based models
- Amazon Q Developer (formerly CodeWhisperer)
- Google Gemini Code Assist
- Tabnine and Replit Ghostwriter

Generative AI assists developers by:

- Generating boilerplate and repetitive code automatically.
- Suggesting complete functions based on comments or function names.
- Explaining unfamiliar code segments in plain language.
- Translating code between programming languages.
- Detecting potential bugs and security vulnerabilities.

Organizations benefit from Generative AI through faster development cycles, reduced time-to-market, and improved consistency across large codebases.

6. Core Techniques Underlying Generative AI Coding Tools

Generative AI coding assistants are built upon several underlying technical approaches.

6.1 Large Language Models (LLMs)

LLMs such as the GPT and Codex family are trained on enormous corpora of publicly available source code and natural-language text, enabling them to predict and generate syntactically and semantically appropriate code.

6.2 Code Completion and Autoregressive Generation

Most coding assistants generate output token-by-token, predicting the most probable next piece of code based on the preceding context, allowing real-time suggestions as a developer types.

6.3 Retrieval-Augmented Generation (RAG)

Advanced tools combine generative capability with retrieval mechanisms that search an organization's own codebase or documentation, allowing suggestions to remain consistent with internal coding standards and proprietary logic.

7. Applications of Generative AI in the Software Development Lifecycle

Generative AI is applied across multiple stages of the software development lifecycle:

7.1 Code Generation and Auto-Completion

AI tools generate code snippets, functions, and entire modules based on natural-language prompts or partial code, significantly speeding up the initial coding phase.

7.2 Automated Testing and Test-Case Generation

Generative AI can automatically draft unit tests and edge-case scenarios, improving test coverage while reducing the manual effort required from quality assurance teams.

7.3 Debugging and Error Detection

AI assistants help identify the probable cause of runtime errors and logical bugs, and frequently suggest corrected code directly within the development environment.

7.4 Documentation Generation

Generative AI tools can automatically generate inline comments, function-level documentation, and README files, helping maintain consistent and up-to-date project documentation.

7.5 Code Review and Refactoring

AI-assisted code review tools highlight code smells, suggest refactoring opportunities, and flag deviations from coding standards before code is merged into production branches.

8. Impact of Generative AI on Software Development Productivity

8.1 Increased Coding Speed

Generative AI substantially reduces the time required to write routine and boilerplate code, allowing developers to focus on complex problem-solving.

Benefits include:

- Faster completion of repetitive coding tasks.
- Reduced time spent searching documentation and forums.
- Quicker prototyping of new features.

8.2 Improved Code Quality and Consistency

AI-generated suggestions, when properly reviewed, can help enforce consistent coding patterns and reduce certain categories of syntax-level errors across a development team.

8.3 Reduced Cognitive Load on Developers

By automating repetitive and mechanical aspects of coding, Generative AI allows developers to conserve mental effort for architectural decisions and complex logic design.

8.4 Faster Onboarding of New Developers

Generative AI tools help new team members understand unfamiliar codebases more quickly by explaining code in natural language and suggesting context-appropriate completions.

8.5 Enhanced Collaboration and Knowledge Sharing

AI-generated documentation and code explanations make it easier for distributed and cross-functional teams to understand each other's work, improving overall collaboration.

9. Impact on Developer Experience and Skill Development

Generative AI significantly influences developer experience and the manner in which technical skills are developed within the industry:

Positive Impacts

- Reduced repetitive workload and improved job satisfaction.
- Faster learning of new programming languages and frameworks.
- Greater focus on creative and architectural problem-solving.
- Improved work-life balance due to faster task completion.

Negative Impacts

- Risk of reduced fundamental coding skills among junior developers.
- Potential over-reliance on AI-generated suggestions without sufficient understanding.
- Possible complacency in reviewing AI-suggested code thoroughly.

10. Advantages of Generative AI in Software Development

The table below summarizes the principal advantages of Generative AI adoption in software engineering, as reported across the reviewed literature and industry studies.

Advantage	Description
Faster Development Cycles	Reduces the time required to write, test, and deploy code, accelerating overall project timelines.

Advantage	Description
Lower Routine Workload	Automates boilerplate and repetitive coding tasks, freeing developers for higher-value work.
Improved Documentation	Generates clear and consistent documentation alongside code, reducing maintenance overhead.
Better Learning Support	Helps developers learn new languages, frameworks, and best practices through contextual explanations.
Cost Efficiency	Reduces the overall person-hours required per feature, lowering software development costs over time.

11. Challenges and Limitations

Despite its numerous advantages, the adoption of Generative AI in software development presents several notable challenges.

11.1 Code Accuracy and Hallucination Issues

Generative AI models can produce code that appears syntactically correct but is logically flawed, insecure, or based on non-existent library functions, a phenomenon often referred to as hallucination.

11.2 Security and Intellectual Property Risks

AI-generated code may inadvertently reproduce patterns from copyrighted or vulnerable source code present in training data, raising legal and security concerns for organizations.

11.3 Over-Reliance and Skill Erosion

Excessive dependence on AI-generated code may gradually weaken developers' fundamental programming and debugging skills, particularly among those early in their careers.

11.4 Integration and Implementation Cost

Enterprise-grade licensing, infrastructure adaptation, and employee training required to effectively integrate Generative AI tools can involve significant upfront investment.

11.5 Bias and Lack of Contextual Understanding

AI models may lack deep understanding of an organization's specific business logic or architectural constraints, occasionally generating suggestions that are technically valid but contextually inappropriate.

12. Future Scope

The future of Generative AI in software development appears highly promising, with continuous improvements expected in the coming years.

Future developments may include:

- Fully autonomous AI agents capable of completing multi-step engineering tasks.
- Deeper integration of AI assistants with enterprise codebases through retrieval-augmented systems.
- Improved AI-driven security scanning and vulnerability remediation.
- AI-assisted low-code and no-code development platforms.
- Personalized AI assistants trained on individual or organizational coding styles.
- Greater use of AI in automated software testing and quality assurance.

These developments are expected to further transform software engineering practices and continue reshaping productivity benchmarks across the global technology industry, including in India's rapidly expanding IT and software services sector.

13. Findings of the Study

The study reveals that:

- Generative AI has meaningfully improved coding speed and reduced time spent on repetitive tasks.
- AI coding assistants positively influence developer satisfaction and learning, particularly among early-career developers.
- Code quality benefits depend heavily on the rigor of human review of AI-generated suggestions.
- Security, accuracy, and intellectual-property concerns remain significant barriers to unrestricted adoption.
- Organizations that combine Generative AI tools with structured review processes report the strongest productivity outcomes.
- The future trajectory of Generative AI in software engineering points toward increasingly autonomous and context-aware development assistance.

14. Conclusion

Generative AI is playing an increasingly significant role in transforming software development productivity across the global technology industry. By automating repetitive coding tasks, assisting with debugging, generating documentation, and supporting test-case creation, Generative AI tools enable developers to focus their efforts on higher-value design and problem-solving activities.

Although challenges such as code accuracy, security risks, and the possibility of skill erosion remain important concerns, the overall body of evidence reviewed in this study suggests that the productivity benefits of

Generative AI substantially outweigh its limitations when the technology is adopted responsibly and accompanied by adequate human oversight.

Therefore, Generative AI is emerging as a key driver of efficiency and innovation in modern software engineering, and its continued evolution is likely to further reshape development practices, team structures, and skill requirements within the software industry in the years ahead.

References

GitHub, Inc. (2023). The impact of AI on developer productivity. GitHub Blog.

McKinsey & Company. (2023). Unleashing developer productivity with generative AI. McKinsey Digital Insights.

Peng, S., Kalliamvakou, E., Cihon, P., & Demirer, M. (2023). The impact of AI on developer productivity: Evidence from GitHub Copilot. arXiv preprint.

Sharma, R., & Mehta, K. (2024). Adoption of generative AI tools in Indian software organizations: A study of opportunities and challenges. International Journal of Computer Applications.

Stack Overflow. (2024). Stack Overflow developer survey 2024. Stack Overflow Insights.

Vaithilingam, P., Zhang, T., & Glassman, E. L. (2022). Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. CHI Conference on Human Factors in Computing Systems Extended Abstracts.

Ziegler, A., Kalliamvakou, E., Li, X. A., Rice, A., Rifkin, D., Simister, S., Sittampalam, G., & Aftandilian, E. (2024). Measuring GitHub Copilot's impact on productivity. Communications of the ACM.

Sommerville, I. (2021). Software Engineering. Pearson Education.

Pressman, R. S., & Maxim, B. R. (2020). Software Engineering: A Practitioner's Approach. McGraw-Hill Education.

Various research articles and white papers from IEEE Xplore, ACM Digital Library, Springer, and Google Scholar databases.

Industry reports and technical articles related to Generative AI and software engineering trends, 2023–2026.