Impact Of Generative AI On Software Development

Kale Aditi Anant[1], Dyaneshwar Balu Lokhande[2], Swaraj Sanjay Malve[3]
[1,2,3]*Jai Hind Institute of Research and Management*

Abstract*—Generative artificial intelligence (AI) tools, such as GitHub Copilot, ChatGPT, and other large language model (LLM)-based coding assistants, have rapidly become part of everyday software development. These tools can write code, suggest fixes, explain errors, and generate documentation, thereby changing how developers work at almost every stage of the software development life cycle. This paper studies the impact of generative AI on software development by reviewing existing research and industry data on developer productivity, code quality, and adoption trends. The study finds that generative AI tools can meaningfully speed up coding tasks and increase output, with less experienced developers often benefiting the most. At the same time, independent research shows a measurable rise in code churn and copy-pasted code, raising concerns about long-term code maintainability. The paper also finds a gap between how widely these tools are adopted and how much developers actually trust their output. Based on these findings, the paper proposes a simple framework for organizations to evaluate generative AI adoption by tracking productivity, quality, and trust together, rather than productivity alone.

I. INTRODUCTION

Software development has changed significantly in the last few years because of generative AI. Generative AI refers to systems that can create new content, including working code, based on a prompt or an existing piece of text. Tools built on large language models, such as GitHub Copilot, ChatGPT, Amazon CodeWhisperer, and similar assistants, can now suggest entire functions, fix bugs, write test cases, and explain unfamiliar code, all inside a developer's regular coding environment.

Until recently, developers relied mainly on documentation, search engines, and community forums, such as Stack Overflow, to solve problems while coding. Generative AI tools have added a new option: asking an AI assistant directly, inside the editor, and getting an instant, context-aware suggestion. This has changed daily developer habits, team workflows, and even how organizations think about training new developers.

This shift has attracted considerable research and industry attention. Some studies have reported strong productivity gains from using AI coding assistants. Other studies have raised concerns about code quality, over-reliance on AI suggestions, and reduced learning for beginners. Because the technology is new, there is no single, agreed-upon picture of its overall impact. This paper brings together evidence from both academic research and industry studies to build a clearer, balanced picture of how generative AI affects software development.

II. PROBLEM STATEMENT

Generative AI coding tools are being adopted quickly, but organizations and individual developers often lack a clear, evidence-based understanding of their real impact. Marketing claims from tool vendors tend to highlight speed and convenience, whereas some independent studies highlight risks such as rising code churn and lower code quality. Without a structured comparison of this evidence, teams may adopt these tools without understanding the trade-offs or avoid them without understanding the genuine benefits. There is a need to consolidate existing productivity, quality, and adoption data into one framework that developers and organizations can use to make informed decisions about generative AI adoption.

III. OBJECTIVES

- To understand how generative AI tools are currently being used across the software development life cycle.
- To review and compare existing research on the productivity impact of Gen AI coding tools.
- To examine the evidence of code quality and maintainability risks associated with AI-assisted coding.

- To study the current developer adoption, sentiment, and trust levels toward generative AI tools.
- We propose a simple framework that balances productivity gains against quality and trust considerations.

IV. LITERATURE REVIEW

Table 1: summarizes the key studies and industry reports referenced in this paper, drawn from academic research, controlled experiments, and independent industry analyses.

Table 1: Summary of reviewed studies on generative AI in software development

1	Kalliamvakou (2022)	GitHub-commissioned controlled study	Developers using GitHub Copilot completed coding tasks 55% faster than those who did not use it.	The task was narrow and standardized and may not reflect complex, real-world codebases.
2	Peng et al. (2023)	Randomized controlled trial (GitHub / Microsoft Research)	Confirmed significant speed gains from AI code suggestions in a controlled programming task.	It is limited to a single, short, well-defined task type.
3	Cui et al. (2024)	Field RCTs across 3 companies, 4,000+ developers	Copilot users completed approximately 26% more tasks per week, and less-experienced developers gained the most.	The effect size varies widely according to the developer's experience and company context.
4	Harness / SEI (2023)	Case study, enterprise engineering team	10.6% increase in pull requests and 3.5-hour reduction in average cycle time after Copilot adoption	Single-organization case study; not independently peer-reviewed.
5	Fahner & Gerth-Ramos (2023)	Industry case study, Mercedes-Benz	Estimated productivity gain of approximately 30 minutes per developer per day.	Self-reported estimate; total efficiency gain is difficult to isolate.
7	Stack Overflow (2024)	Global developer survey, ~65,000 respondents	76% of developers use or plan to use AI tools; only 43% trust their accuracy; and 45% say AI struggles with complex tasks.	Self-reported sentiment does not measure actual code outcomes.

Across these sources, a consistent pattern emerges: generative AI tools measurably speed up certain coding tasks, and the size of the benefit tends to be larger for less experienced developers. At the same time, independent, large-scale code analysis (GitClear) finds early warning signs around code quality, specifically rising code churn and a fall in refactoring activity that are not captured by studies that measure speed alone. Developer sentiment surveys add a third dimension: even as adoption grows, trust in AI-generated output remains limited, and many developers feel unsure how their own productivity gains, if any, are actually being measured

V. RESEARCH GAP

Most existing studies examine the impact of generative AI through a single lens, either productivity (speed, output volume) or quality (code churn, maintainability) or sentiment (adoption, trust), rather than considering all three together. This creates an incomplete picture: a tool that makes developers faster but also increases long-term maintenance costs may look purely positive in a productivity study, while the same tool may look purely negative in a code-quality study. There is a gap in the literature for a combined view that treats productivity, quality, and trust as connected outcomes that should be evaluated together when judging the real impact of generative AI on software development.

VI. PROPOSED METHODOLOGY

This study used a secondary research methodology, meaning that it was based on analyzing and comparing existing published studies, industry reports, and

developer surveys rather than running a new experiment. The steps followed are shown in Figure 1.

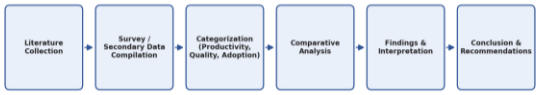


Figure 1: Methodology followed for this study

1. Literature Collection
Peer-reviewed papers, controlled experiments (RCTs), industry whitepapers, and large-scale developer surveys published between 2022 and 2024 were collected, focusing specifically on the effect of generative AI on software development.

2. Secondary Data Compilation
Quantitative findings, such as reported percentage productivity gains, code churn rates, and survey percentages, were extracted directly from the original sources, with each figure attributed to its source study.

3. Categorization
The findings were grouped into three categories: productivity (speed and output), quality (maintainability and code churn), and adoption/trust (usage rates and developer sentiment).

4. Comparative Analysis
Findings within each category were compared across studies to identify points of agreement, disagreement, and gaps in the current research.

5. Findings & Interpretation
Patterns across the three categories were interpreted together, rather than in isolation, to build the combined view discussed in Section VII.

VII. RESULTS AND DISCUSSION

Figure 2 shows reported productivity effects of generative AI coding tools across four independently conducted studies. While the exact figures differ because the studies measured different things (task completion time, tasks per week, pull requests, and time saved per day), all four points in the same direction: measurable productivity gains from using coding assistants.

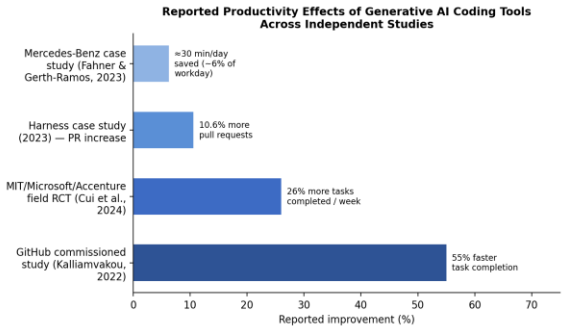


Figure 2: Reported productivity effects across independent studies.

Figure 3 shows results from the Stack Overflow 2024 Developer Survey, the largest publicly available survey of developer sentiment toward AI tools. While adoption is high, trust remains comparatively low — only 43% of developers trust the accuracy of AI-generated output, even though 81% agree that the main benefit of these tools is increased productivity.

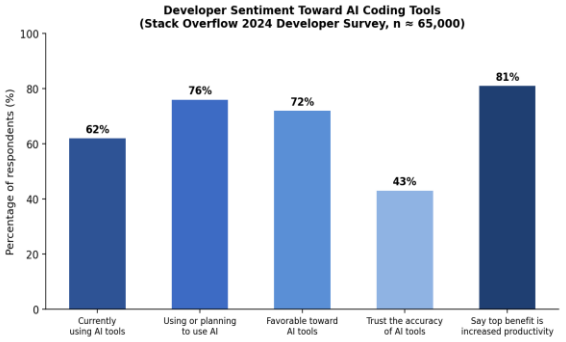


Figure 3: Developer adoption and sentiment toward AI coding tools (Stack Overflow, 2024).

Table 2 consolidates the key findings from this review across the three categories identified in the methodology: productivity, quality, and adoption/trust.

Table 2: Consolidated findings by dimension.

Coding speed	Up to 55% faster completion on standardized tasks; ~26% more tasks completed per week in field settings.	Kalliamvakou (2022); Cui et al. (2024)
Pull requests / throughput	10.6% increase in pull requests; ~3.5-hour reduction in cycle time.	Harness (2023)

Daily time saved	Approximately 30 minutes saved per developer per day.	Fahner & Gerth-Ramos (2023)
Code churn (quality risk)	Projected to double vs. 2021 baseline; more copy-pasted code, less refactoring.	Harding & Kloster, GitClear (2024)
Adoption	76% of developers use or plan to use AI tools (up from 70% in 2023).	Stack Overflow (2024)
Trust	Only 43% of developers trusted the accuracy of AI-generated code/output.	Stack Overflow (2024)
Skill-gap effect	Less-experienced developers show larger productivity gains than senior developers.	Cui et al. (2024); Noy & Zhang (2023)

Discussion

The findings suggest that generative AI's impact on software development cannot be summarized with a single number or a single verdict. On one hand, the evidence for productivity gains is fairly consistent: across four independently run studies using different methods, all report a measurable improvement in speed or output when developers use generative AI tools, and less experienced developers appear to benefit the most. This is a meaningful finding for organizations that are training junior developers, since it suggests generative AI may help narrow the early-career skill gap.

On the other hand, the GitClear analysis of 153 million lines of code is a rare example of quality-focused research at scale, and its findings are harder to dismiss: rising code churn and falling refactoring activity are signs that some of the code being produced faster is also being produced less carefully. This does not cancel out the productivity gains, but it does mean that raw speed should not be the only metric organizations track when adopting these tools.

Finally, the trust gap identified in the Stack Overflow survey is worth highlighting on its own. A situation where adoption (76%) is much higher than trust (43%) suggests that many developers are using these tools out of necessity, habit, or organizational pressure, rather than full confidence in the output. This points to a continued need for human code review, testing, and oversight even as AI-assisted coding becomes standard practice.

VIII. CONCLUSION

Generative AI is changing software development in a way that is genuinely significant but not simple. It reliably makes many coding tasks faster and appears to particularly help less experienced developers close the gap with more senior colleagues. At the same time, independent, code-level research shows real risks to long-term code quality and maintainability that are easy to miss if an organization only tracks speed. Developer trust in these tools also remains lower than their usage rate would suggest. Organizations adopting generative AI for software development should therefore track productivity, code quality, and developer trust together, rather than treating fast output as the only sign of success. Future research would benefit from long-term, code-quality-focused studies that follow specific teams over multiple years, as most current evidence covers only the first two to three years of mainstream generative AI adoption.

REFERENCES

[1] Kalliamvakou, E., “Research: Quantifying GitHub Copilot's impact on developer productivity and happiness,” GitHub Blog, 2022.

[2] Peng, S., E. Kalliamvakou, P. Cihon, and M. Demirer, “The impact of AI on developer productivity: Evidence from GitHub Copilot,” arXiv preprint arXiv:2302.06590, 2023.

[3] Cui, Z., M. Demirer, S. Jaffe, L. Musolff, S. Peng, and T. Salz, “The effects of generative AI on high-skilled work: Evidence from three field experiments with software developers,” MIT Working Paper, 2024.

[4] Harness, “The impact of GitHub Copilot on developer productivity: A case study,” Harness SEI Blog, 2023.

[5] Fahner, M., and R. Gerth-Ramos, “Measuring the impact of GitHub Copilot at Mercedes-Benz,” Industry Case Study, 2023.

[6] Harding, W., and M. Kloster, “Coding on Copilot: 2023 data shows downward pressure on code quality,” GitClear Research (Alloy.dev Research), 2024.

[7] Stack Overflow, “2024 Developer Survey: Artificial Intelligence,” Stack Overflow, 2024.
[8] Noy, S., and W. Zhang, “Experimental evidence on the productivity effects of generative artificial intelligence,” Science, vol. 381, no. 6654, pp. 187–192, 2023.
[9] InfoQ, “Study shows AI coding assistant improves developer productivity,” InfoQ News, 2024.
[10] Communications of the ACM, “Measuring GitHub Copilot's impact on productivity,” Communications of the ACM, 2024.