Empowering Sustainable Software Development with Generative AI

Murnawan[1], Muhamad Raihan Al Ghazali[2] and Viona Azzahra[3]

[1,2,3] Widyatama University, Bandung, Indonesia
murnawan@widyatama.ac.id

Abstract. This study investigates the impact of generative AI tools, such as GitHub Copilot and ChatGPT, on software development productivity and developer experience. Through an experiment involving 40 software developers across various tasks—such as code documentation, generation, and refactoring—the research assesses the effectiveness of AI in enhancing task completion rates and developer satisfaction. Results reveal that generative AI reduces task completion time by more than 50% for routine tasks, though it has a more modest effect on high-complexity tasks, with only a 10% reduction in time. Additionally, AI improves the success rate of high-complexity tasks by 28.57%, demonstrating its value in solving complex problems. Developers also reported significant increases in job satisfaction, focus, and entering a "flow" state, indicating a positive influence on well-being. While AI offers notable benefits for routine and moderately complex tasks, further research is required to assess its long-term impacts on more complex software development processes. This study supports the role of generative AI in advancing sustainable software development by fostering innovation, improving efficiency, and promoting developer well-being, aligning with the Sustainable Development Goals (SDG 8 and SDG 9).

Keywords: Generative AI in Software Development, Developer Productivity, Sustainable Development Goals (SDGs).

1 Introduction

Generative Artificial Intelligence (AI) has significantly transformed the software development landscape by fostering enhanced productivity, expediting task completion, and improving developer experiences. The integration of generative AI models, such as GitHub Copilot and ChatGPT, in software development environments presents considerable potential to streamline repetitive tasks, assist with complex problem-solving, and accelerate the entire software development lifecycle [1–3]. Despite these advancements, there remains a critical gap in understanding the long-term implications of AI on developer well-being and its capacity to address more complex, high-level tasks. This technological advancement aligns with the objectives of the Sustainable Development Goals (SDGs), particularly SDG 8 (Decent Work and Economic Growth) and

V. Mardiansyah and B. A. Prasetyo (eds.), *Proceedings of the Widyatama International Conference on Engineering 2024 (WICOENG 2024)*, Advances in Engineering Research 252,
https://doi.org/10.2991/978-94-6463-618-5_14

SDG 9 (Industry, Innovation, and Infrastructure), through its capacity to drive innovation and enhance both the efficiency and well-being of software developers [4].

Sustainable Development Goal 8 aims to promote decent work and economic growth by enhancing productivity and creating quality jobs. AI improves developer efficiency, enabling businesses to optimize their workforce and drive innovation, supporting economic growth [5]. Similarly, Sustainable Development Goal 9 focuses on industry, innovation, and infrastructure. Generative AI accelerates software development, fostering industrial growth and advancing digital infrastructure. AI-driven innovations also help create sustainable and scalable technological solutions, aligning with SDG 9 [6, 7].

A significant advantage of generative AI lies in its ability to automate routine tasks, such as code documentation, refactoring, and bug detection, thereby allowing developers to focus on higher-level, creative problem-solving [8]. Studies demonstrate that the use of generative AI tools can increase productivity substantially, reducing the time spent on fundamental coding tasks by as much as 55.8% [9]. Additionally, AI-powered systems enhance software quality by identifying potential errors or inefficiencies early, mitigating the risk of bugs and improving software reliability [10].

Beyond productivity, generative AI has also been shown to positively impact the developer experience by fostering a "state of flow," wherein developers work with greater efficiency and focus [11]. Developers using AI tools report increased job satisfaction and reduced cognitive load, which contributes to improved mental health and well-being [12, 13]. Furthermore, AI bridges the gap between novice and expert developers by offering real-time coding suggestions, thus accelerating the learning curve for less experienced engineers [14].

However, despite its potential, the integration of generative AI in software development is not without challenges. While AI tools can automate numerous coding tasks, developers must remain vigilant in monitoring the output for errors or inaccuracies, as AI systems may occasionally produce flawed or biased results [15]. Additionally, ethical considerations, data privacy concerns, and the potential for AI to perpetuate biases in software development necessitate careful evaluation and mitigation [16].

While there has been considerable research on the benefits of AI for automating routine coding tasks and improving developer productivity, limited research has focused on the long-term effects of generative AI in addressing complex problem-solving tasks and its broader impact on developer well-being. This paper aims to investigate how generative AI contributes to accelerating task completion, enhancing complex problem-solving abilities, and promoting developer well-being, while simultaneously addressing the associated challenges within the framework of sustainable software development.

2 Method

This study was designed to assess the impact of generative AI tools on software development productivity and developer experience, with a specific focus on tasks such as code generation, refactoring, and documentation. The research employed a mixed-method approach, combining quantitative data collection with qualitative feedback to

provide a comprehensive evaluation of AI's role in enhancing software development processes. Below are the key components of the method used in this study.

2.1 Participants

The study involved 40 software developers from multiple software companies in Bandung, Indonesia. The participants had varying levels of experience, ranging from junior to senior developers. To ensure representativeness, participants were drawn from different sectors of the software industry, including web development, mobile applications, and enterprise solutions.

Participants were categorized based on their experience with software development tools, including both novice and experienced users of generative AI. This stratification allowed the study to account for differences in familiarity with AI-assisted tools and measure its impact across skill levels

2.2 Study Design

The study utilized a within-subject design, where each participant performed software development tasks both with and without the assistance of generative AI tools. The generative AI tools utilized in this study include GitHub Copilot and OpenAI's GPT-based models, such as ChatGPT. These tools were integrated into the development environment to assist with code generation, refactoring, and documentation. For code generation tasks, the AI models provided real-time suggestions and auto-completion, leveraging pre-trained language models fine-tuned on extensive programming code datasets. In refactoring tasks, AI tools suggested code optimizations and structural improvements based on best practices in software engineering. For documentation tasks, AI generated contextual descriptions for code, aligning with industry documentation standards. The AI models used reinforcement learning from human feedback (RLHF) to adapt and refine outputs based on user input during task execution, ensuring a balance between automation and developer control. This design allowed for a direct comparison between AI-assisted and non-AI-assisted conditions for the same participants, minimizing variability due to individual differences in skill and experience.

Participants were required to complete the following tasks:

1. Code documentation: Writing clear and detailed documentation for existing codebases.
2. Code generation: Developing new code for specific functionalities or features.
3. Code refactoring: Modifying the structure of existing code to improve maintainability and performance without altering functionality.

Each participant completed half of the tasks with generative AI tools and the other half without AI tools. This approach ensured that all participants were exposed to both conditions, enabling a robust comparison.

2.3 Data Collection

A variety of data collection methods were employed to capture both objective and subjective outcomes:

1. Task Completion Time: Participants recorded the start and end times for each task, along with any breaks taken, to calculate the total time spent on each task. This data was analyzed to compare the time required to complete tasks with and without AI assistance.
2. Task Success Rate: For each task, success was defined as completing the task within a predefined time limit. The success rates were recorded for both AI-assisted and non-AI-assisted tasks.
3. Developer Experience Surveys: After each task, participants completed a survey to assess their experience. The survey measured:
   a. Happiness: "I felt happy while performing this task."
   b. Focus: "I was able to focus on satisfying and meaningful work."
   c. Flow state: "I experienced a 'flow' state during the task."
   d. Sense of accomplishment: "I felt a sense of accomplishment after completing the task."
4. Code Quality Evaluation: Automated tools, such as SonarQube, were used to evaluate the quality of the code produced. Metrics such as readability, maintainability, and the presence of bugs were assessed for both AI-assisted and non-AI-assisted code.

2.4 Task Complexity

To assess the role of generative AI across varying task difficulties, tasks were categorized into three levels of complexity:

1. Low complexity: Tasks that involved routine operations such as basic code documentation.
2. Medium complexity: Tasks that required moderate problem-solving and basic code refactoring.
3. High complexity: Tasks that required complex decision-making and advanced code generation or refactoring.

In high-complexity tasks, such as advanced code refactoring or generating complex algorithms, the AI models used a deeper understanding of programming patterns to suggest solutions that optimize performance and maintainability. For low and medium complexity tasks, simpler AI models were used to automate repetitive code generation or documentation. The AI's performance was measured not only by task completion speed but also by the accuracy of the code generated, using code quality metrics such as maintainability index and bug detection scores from tools like SonarQube. The level of complexity was recorded for each task, and this data was later used in the correlation analysis to evaluate the relationship between task complexity and the benefits of using AI tools.

2.5 Data Analysis

Several statistical methods were used to analyze the collected data:

1. Paired t-tests: Paired t-tests were employed to determine whether there were statistically significant differences in task completion time and developer experience between the AI-assisted and non-AI-assisted conditions. This method was appropriate given the within-subject design of the study, where each participant served as their own control.
2. Correlation Analysis: Pearson correlation coefficients were calculated to explore the relationship between task complexity and improvements in developer experience when using AI tools. This analysis aimed to assess whether the benefits of generative AI increased with task complexity.

3 Results and Discussion

In this section, we present and analyze the impact of generative AI tools on software developers' productivity, task completion time, and overall developer experience. Our study involved over 40 developers from software companies across Bandung, Indonesia, who were tasked with performing software development activities with and without generative AI tools. These activities were categorized into three main areas: code generation, refactoring, and documentation. The data was collected from multiple sources, including time-tracking records, task surveys, and code quality evaluations.

3.1 Task Completion Time

The primary focus of this study was to assess how generative AI affects the time required to complete various software development tasks. Developers were asked to perform typical tasks both with and without the use of generative AI tools. The results clearly indicate that the use of generative AI significantly reduces task completion time, especially for routine and repetitive tasks. As shown in Table 1, the use of generative AI reduced task completion time by 50% for code documentation and 55% for code generation.

However, for high-complexity tasks, AI provides only a 10% reduction in time. These results demonstrate that while generative AI significantly reduces time for routine tasks, its impact on more complex tasks is limited, aligning with existing research on AI's role in automating repetitive coding activities.

Table 1. Task Completion Time Comparison

Task	Without Generative AI (%)	With Generative AI (%)	
		Low Estimate	High Estimate
Code documentation	100	45	50
Code generation	100	35	45
Code refactoring	100	20	30
High-complexity tasks	100	10	10

However, for high-complexity tasks, AI provides only a 10% reduction in time. These results demonstrate that while generative AI significantly reduces time for routine tasks, its impact on more complex tasks is limited, aligning with existing research on AI's role in automating repetitive coding activities.

The most significant improvements, as shown in Figure 1, are observed in routine tasks such as code documentation and generation. For more complex tasks, the time saved is relatively small, indicating that generative AI is more effective for simpler tasks.

However, for more complex tasks such as code refactoring and high-complexity problem-solving, the improvements were less pronounced. As seen in Figure 1, the impact of AI on high-complexity tasks was minimal, with only a 10% reduction in completion time. This suggests that while generative AI excels in automating simpler tasks, its effectiveness diminishes when applied to tasks that require logical reasoning, creativity, and contextual understanding—areas where human expertise remains critical.

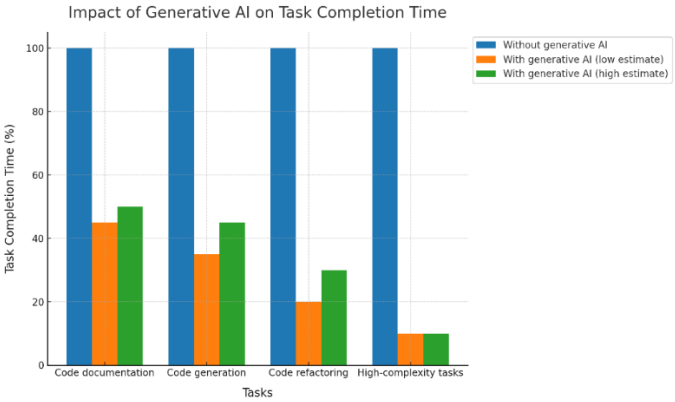

Fig. 1. Impact of Generative AI on Task Completion Time

3.2 Task Complexity and Success Rate

Generative AI had the greatest impact on high-complexity tasks, improving the success rate by 28.57%. As presented in Table 2, the success rate improvement is most pronounced in high-complexity tasks, with minimal gains for low-complexity tasks.

Table 2. Success Rate of Task Completion by Complexity Level

Task Complexity	Without Gen-AI (%)	With Gen-AI (%)	Improvement (%)
High complexity	42	54	28.57
Medium complexity	73	83	13.70
Low complexity	92	94	2.17

This demonstrates the value of AI in assisting with more complex tasks, reducing cognitive load, and speeding up problem-solving, while offering limited benefits for

simpler tasks. As visualized in Figure 2, AI proves most beneficial for high-complexity tasks, while its impact on simpler tasks is limited.

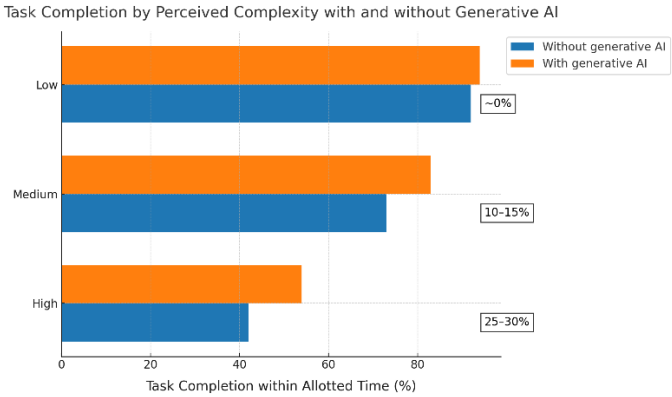

Fig. 2. Task Completion by Perceived Complexity

Figure 2 shows that generative AI significantly improves task completion rates for high-complexity tasks, while offering minimal benefits for low-complexity tasks. This indicates that AI is most effective for challenging tasks that demand greater cognitive effort, whereas its impact on simpler tasks is limited, as these are easily managed without AI assistance.

3.3 Developer Experience

We investigated the impact of generative AI on developers' subjective experience during software development tasks. Post-task survey data indicated improvements in happiness, focus, flow state, and sense of accomplishment when using AI tools. These results suggest that generative AI not only enhances productivity but also positively affects overall developer experience. As shown in Table 3, there is a significant improvement in developer happiness and flow state when using AI tools. Additionally, the increased sense of accomplishment underscores the role of AI in reducing cognitive load and enhancing job satisfaction.

Table 3. Comparison of Developer Experience

Developer Experience	Without Gen-AI	With Gen-AI	Improvement
I felt happy	42 %	54 %	28.57 %
Satisfying and meaningful work	73 %	83 %	13.70 %
I was in a "flow" state	25 %	45 %	80.00 %
I had a sense of accomplishment	35 %	55 %	57.14 %

As illustrated in Figure 3, the improvements in developer experience—especially in achieving a 'flow' state—are clearly visualized. The ability of AI tools to reduce the

cognitive load of routine tasks directly supports the developers' enhanced focus and job satisfaction.

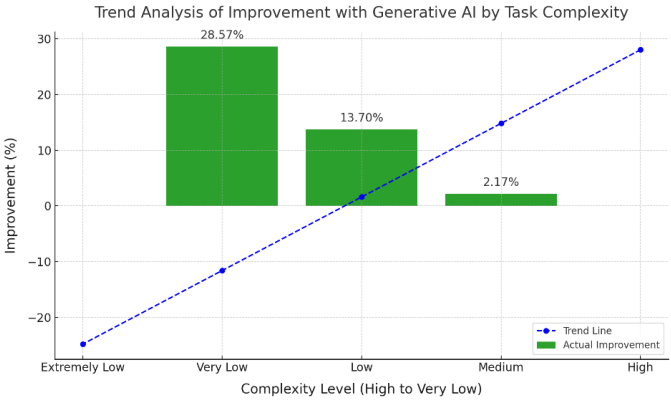

Fig. 3. Trend Analysis of Improvement with Generative AI

Generative AI enhances both productivity and developer experience. Table 3 demonstrates significant improvements in well-being metrics, including a 28.57% increase in happiness and an 80% improvement in achieving a "flow" state. This "flow" is critical for sustaining high productivity and job satisfaction. As illustrated in Figure 3, the improvements in developer experience are particularly significant in achieving a 'flow' state and reducing mental strain. These findings suggest that AI not only boosts productivity but also contributes to the overall well-being of developers by reducing mental strain and increasing task satisfaction.

3.4 Correlation Analysis

To evaluate the relationship between task complexity and the improvement in user experience after the implementation of Generative AI, a Pearson correlation analysis was conducted. The complexity levels were assigned numerical values (3 = High, 2 = Medium, 1 = Low) and correlated with the percentage of improvement observed across four key user experience categories: “I felt happy,” “Satisfying Work,” “Flow State,” and “Sense of Accomplishment.”

Table 4. Correlation between Task Complexity and Improvement in User Experience

Category	Correlation Coefficient	p-Value
I felt happy	0.90	0.285
Satisfying Work	0.91	0.275
Flow State	0.94	0.214
Sense of Accomplishment	0.82	0.386

The results in Table 4 show strong positive correlations across all categories, with coefficients ranging from 0.82 to 0.94. This suggests that as task complexity increases,

the benefits of using generative AI also increase. However, the p-values for all categories exceed 0.05, meaning the correlations, while strong, are not statistically significant at the 95% confidence level. This could be due to the limited sample size or data variability. Further studies with larger datasets are recommended to validate these findings.

4 Conclusions

Generative AI has shown significant potential to enhance software development productivity, aligning with the goals of SDG 8 and SDG 9. This study demonstrates that AI tools can reduce task completion time by over 50% for routine tasks like code documentation and generation, thus optimizing workforce efficiency and promoting decent work and economic growth. For complex tasks, AI improved success rates by 28.57%, supporting industrial innovation and infrastructure growth as outlined in SDG 9.

Moreover, generative AI positively influences the developer experience, enhancing focus, flow, and job satisfaction. Although the correlation between task complexity and developer experience is strong, further research with larger datasets is needed to confirm these long-term benefits, especially for high-complexity tasks.

In summary, generative AI supports sustainable software development by improving productivity, fostering innovation, and enhancing developer well-being, contributing to both economic growth and technological advancement.

References

1. Noy, S., Zhang, W.: Experimental evidence on the productivity effects of generative artificial intelligence. Science (1979). 381, (2023).
2. Ebert, C., Louridas, P.: Generative AI for Software Practitioners. IEEE Softw. 40, (2023). https://doi.org/10.1109/MS.2023.3265877.
3. Brynjolfsson, E., Li, D., Raymond, L.: Generative Ai at Work. SSRN Electronic Journal. (2023). https://doi.org/10.2139/ssrn.4426942.
4. Russo, D.: Navigating the Complexity of Generative AI Adoption in Software Engineering. ACM Transactions on Software Engineering and Methodology. 33, (2024). https://doi.org/10.1145/3652154.
5. Khlaisamniang, P., Khomduean, P., Saetan, K., Wonglapsuwan, S.: Generative AI for Self-Healing Systems. In: 18th International Conference on Artificial Intelligence and Natural Language Processing and International Conference on Artificial Intelligence and Internet of Things, iSAI-NLP 2023 (2023).
6. Bajaj, Y., Samal, M.K.: Accelerating Software Quality: Unleashing the Power of Generative AI for Automated Test-Case Generation and Bug Identification. Int J Res Appl Sci Eng Technol. 11, (2023).
7. Eilam, T., Bello-Maldonado, P.D., Bhattacharjee, B., Costa, C., Lee, E.K., Tantawi, A.: Towards a Methodology and Framework for AI Sustainability Metrics. In: 2nd Workshop on Sustainable Computer Systems, HotCarbon 2023 (2023)
8. Kulkarni, V., Reddy, S., Barat, S., Dutta, J.: Toward a Symbiotic Approach Leveraging Generative AI for Model Driven Engineering. In: Proceedings - ACM/IEEE

26th International Conference on Model Driven Engineering Languages and Systems, MODELS 2023 (2023).
9. Peng, S., Kalliamvakou, E., Cihon, P., Demirer, M.: The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. ArXiv. 1–19 (2023).
10. Megahed, F.M., Chen, Y.J., Ferris, J.A., Knoth, S., Jones-Farmer, L.A.: How generative AI models such as ChatGPT can be (mis)used in SPC practice, education, and research? An exploratory study. Qual Eng. 36, (2024).
11. Bilgram, V., Laarmann, F.: Accelerating Innovation With Generative AI: AI-Augmented Digital Prototyping and Innovation Methods. IEEE Engineering Management Review. 51, (2023). https://doi.org/10.1109/EMR.2023.3272799.
12. Bull, C., Kharrufa, A.: Generative Artificial Intelligence Assistants in Software Development Education: A Vision for Integrating Generative Artificial Intelligence into Educational Practice, Not Instinctively Defending Against It. IEEE Softw. 41, (2024). https://doi.org/10.1109/MS.2023.3300574.
13. Bull, C., Kharrufa, A.: Generative AI Assistants in Software Development Education. ArXiv. (2023).
14. Parikh, N.A.: Empowering business transformation: The positive impact and ethical considerations of generative AI in software product management - A systematic literature review. In: Transformational Interventions for Business, Technology, and Healthcare (2023). https://doi.org/10.4018/979-8-3693-1634-4.ch016.
15. Wach, K., Duong, C.D., Ejdys, J., Kazlauskaitė, R., Korzynski, P., Mazurek, G., Paliszkiewicz, J., Ziemba, E.: The dark side of generative artificial intelligence: A critical analysis of controversies and risks of ChatGPT. Entrepreneurial Business and Economics Review. 11, (2023). https://doi.org/10.15678/EBER.2023.110201.
16. Kenthapadi, K., Lakkaraju, H., Rajani, N.: Generative AI meets Responsible AI: Practical Challenges and Opportunities. In: Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2023).

.