Chapter 5

Artificial Intelligence as a Developer's Assistant: Reducing Overtime in Software Development

Ozlem Tek Mutlu[1]

Murat Nazli[2]

Abstract

Artificial Intelligence (AI) is transforming software development (SD) by streamlining repetitive tasks, boosting productivity, and cutting down overtime. This paper examines how AI tools support developers, analyzing their influence on SD processes and reducing overtime by automating tasks in SD. A semi-structured interview was conducted with 20 software professionals, including developers, team leads, managers, and architects. The findings reveal that despite many lack of accuracy concerns, participants widely use AI tools in various phases of SD, such as coding or testing. The majority of respondents agreed the use of AI instruments helped in reducing overtime in SD.

1. Introduction

The SD industry is known for its demanding schedules, tight deadlines and and project complexity. This situation is exacerbated by the repetitive and time-intensive nature of many tasks in the development lifecycle. This often leads to decreased efficiency, developer burnout, and a decline in code quality (Fraiwan & Khasawneh, 2024; Yang et al., 2022).

The advent of AI tools offers a way to mitigate these pressures. For instance, AI-enabled developer assistants such as GitHub Copilot and Tabnine are designed to automate repetitive tasks, support complex problem-solving, and provide instant feedback, thereby enhancing developer efficiency (Ajiga, 2024; Russo, 2024). AI instruments help developers focus on creative and strategic work, reducing the need for overtime by automating routine coding,

1 MA, Engineering Management Master Pr., Izmir Institute of Technology, ozlemtek7@gmail.com

2 Assoc. Prof., Izmir University of Economics, Vocational School, Izmir Institute of Technology nazli.murat@gmail.com, ORCID ID 0000-0003-0335-1706

https://doi.org/10.58830/ozgur.pub1255.c5108

debugging, and testing processes. Generative AI instruments such as ChatGPT/GitHub Copilot have presented to improve coding efficiency by up to 30%, freeing developers from repetitive workloads and helping projects stay on schedule (Nguyen-Duc et al., 2023; Sauvola et al., 2024). These tools not only increase productivity but also support better planning and execution, reducing stress and fostering a healthier work-life balance (Champa et al., 2024; Coutinho et al., 2024; Kytsak & Ovsiannykov, 2025).

Therefore, the unique study mainly explores the role of AI as a developer's assistant in reducing overtime and improving productivity. It evaluates the current capabilities of AI instruments, their association with development workflows, and their broader implications for developers and organizations.

2. Literature Review

AI has become a cornerstone in software engineering, transitioning traditional practices into more efficient, data-driven processes. The adoption of AI instruments has enhanced productivity by automating repetitive tasks and improving efficiency across the stages of Software Development Life Cycle (SDLC) (Alenezi & Akour, 2025).

2.1. The Impact of AI in Software Development Life Cycle

AI has significantly impacted SD. From requirements analysis to maintenance, AI tools are integrated throughout the SDLC. Gerosa et al. (2024) highlight AI's capacity to process large datasets, improving decision-making in requirements gathering. Coutinho et al. (2024) note the productivity enhancements offered by generative AI tools in coding and debugging, while Champa et al. (2024) emphasize the collaborative benefits of AI assistants in knowledge sharing and task management.

SDLC comprises several phases that guide the systematic development of software. AI tools have increasingly been integrated into each phase to improve efficiency and reduce developer workloads:

1. Requirement Analysis: AI tools analyze user feedback and historical data to extract and prioritize software requirements. By leveraging natural language processing, these tools ensure requirements align with user needs and business goals (Nguyen-Duc et al., 2023; Russo, 2024).

2. Design: During the design phase, AI-powered tools help generate optimal software architectures and design patterns. They automate the creation of models/diagrams, streamlining the documentation process and decreasing the errors (Ajiga, 2024; Mashkoor et al., 2022).

3. Implementation (Coding): Generative AI tools like GitHub Copilot support developers in writing code faster by supplying real-time suggestions and automating repetitive tasks. These tools reduce boilerplate coding and help maintain consistency across the codebase (Champa et al., 2024; Nguyen-Duc et al., 2023).

***4. Testing*:** AI improves software testing by predicting defects and creating cases automatically. Tools like ChatGPT assist in creating test scripts, while machine learning models identify potential vulnerabilities and estimate system failures (Ajiga, 2024; Khaliq et al., 2022; Sauvola et al., 2024). In deployment, AI facilitates smoother software deployment by automating configuration management and monitoring. Predictive analytics ensure seamless rollouts by identifying and mitigating risks during deployment (Russo, 2024; Tatineni & Allam, 2024).

5. ***Maintenance:*** In the maintenance phase, AI tools proactively detect and resolve software issues. Predictive maintenance models suggest updates and patches before failures occur, ensuring long- term system stability (Nguyen-Duc et al., 2023; Russo, 2024).

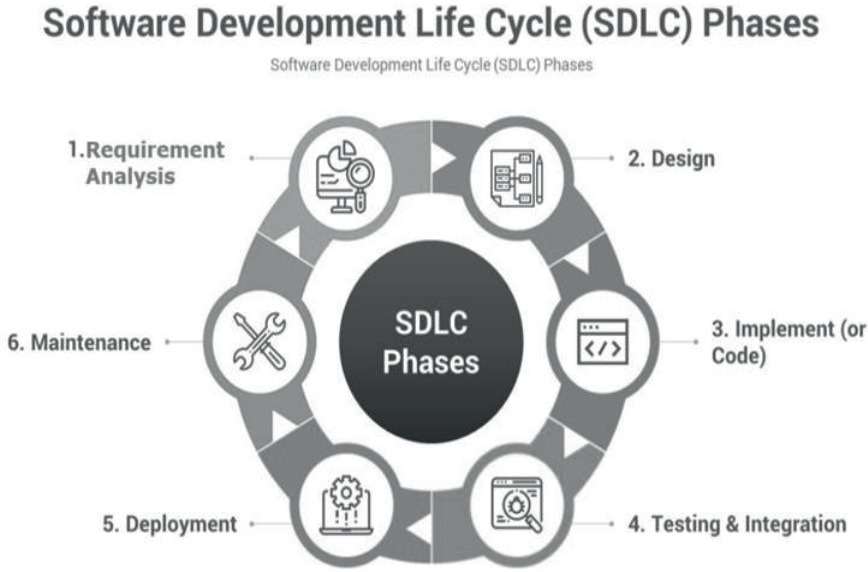

Figure 1 - SDLC Phases (Gladun, 2025)

2.2. Ethical Considerations and Challenges

While AI offers numerous benefits, its adoption in SD introduces several ethical challenges. Klemmer et al. (2024) identify concerns about data privacy and security, when sensitive data are processed by AI systems. Nguyen-Duc et al. (2023) highlight algorithmic bias as a significant issue, where outputs may inadvertently reflect biases in the training data.

Accountability and transparency are crucial for building trust in AI systems. Sauvola et al. (2024) advocate for explainable AI frameworks to ensure stakeholders can understand and validate AI- driven decisions. Marar (2024) points out intellectual property ambiguities associated with generative AI, where ownership of AI-generated code remains unclear. Pointing out these ethical concerns is critical for responsible AI integration in SD.

3. Methodology

A semi-structured interview method was conducted with 20 software professionals from different backgrounds, including developers, team leads, managers, and architects to understand their experiences with AI tools and their influence on reducing overtime (Karyotakis et al., 2026) through sending an e-mail to the professionals. The key questions were asked between November 15, 2024 and December 10, 2024.

The following nine questions formed by the researchers are asked to the participants. The first three questions are about their demographic characteristics, such as their role in the organization, their work experience in the field, and the region they work in. The other six questions (question 4- question 9) address AI usage habits and its impact on reducing overtime.

1. Your primary role in SD?
2. Work experience in SD?
3. Which region do you work in?
4. What AI instruments do you currently use in your SD workflow?
5. How often do you use these instruments in daily tasks?
6. Which areas of SD do you think AI impacts the most?
7. To what extent has AI reduced the time you spend on repetitive tasks?
8. Has the use of AI tools helped in reducing overtime hours?
9. Struggles you came across when using AI tools in SD?

4. Findings & Discussions

Participants included 9 developers, 2 technical leads, 7 managers, 1 project manager, and 1 architect, with experience ranging from 2 to more than 15 years. Respondents represented regions including Europe, Asia, and South America. These findings align with Yang et al. (2022), which highlights the global adoption of AI tools across diverse professional roles in software engineering.

Table 1 - The primary role of software professionals in the organization (n = 20)

Answers	Percent	Number
Developer	45	9
Tester	0	0
Architect	5	1
Technical Lead	10	2
Manager	35	7
Other (specify): Project manager	5	1

Table 2 - The work experience of software professionals in the field (n = 20)

Answers	Percent	Number
Less than 1 year	0	0
1-3 years	10	2
4-7 years	20	4
8-15 years	20	4
More than 15 years	50	10

Table 3 - The region of software professionals they work in (n = 20)

Answers	Percent	Number
North America	0	0
Europe	50	10
Asia	40	8
South America	5	1
Africa	0	0
Australia/Oceania	0	0
Other	5	1

AI Tool Usage: Widely used tools included ChatGPT, GitHub Copilot and custom AI models, with all participants use these tools in their daily tasks. Nguyen-Duc et al. (2023) stated that the consistent use of these tools reflects their significant role in automating repetitive tasks and improving efficiency.

Table 4 - AI tools that software professionals currently use in their SD workflow (n=20)

Answers	Percent	Number
GitHub Copilot	60	12
ChatGPT	90	18
Amazon CodeWhisperer	5	1
Tabnine	0	0
Other (specify): OpenAI Codex	0	0
Amazon Q	30	6

Table 5 - The frequency of use of AI instruments in their daily tasks (n= 20)

Answers	Percent	Number
Never	0	0
Rarely	15	3
Sometimes	40	8
Often	35	7
Always	10	2

Impact on Repetitive Tasks: All respondents use AI tools for repetitive tasks such as debugging, documentation, and code reviews. About 25% of respondents reported significant reductions in time spent on repetitive tasks, confirming the findings of Fraiwan and Khasawneh (2024), which stated that AI decreases manual effort and lets developers focus on high-value activities.

Table 6 - Areas where AI tools have the most impact (n= 20)

Answers	Percent	Number
Coding	95	19
Testing	45	9
Debugging	35	7
Documentation	45	9
Project Management	15	3
Other (specify): Exception handling, best practices and standards	10	2

Table 7 - Reductions in time spent on repetitive tasks (n= 20)

Answers	Percent	Number
Not at all	10	2
Slightly	30	6
Moderately	35	7
Significantly	25	5

Reduction in Overtime: 70% noted a marked reduction in overtime hours, crediting AI's ability to automate mundane tasks and optimize workflows. This aligns with Russo (2024), which reported a 30% increase in productivity when AI tools were integrated into development processes.

Table 8 - Has the use of AI tools helped in reducing overtime hours? (n= 20)

Answers	Percent	Number
Yes, they helped a lot	70	14
No, they did not	30	6

Challenges: Key obstacles included lack of accuracy and integration difficulties. These challenges are consistent with Klemmer et al. (2024), which highlights the technical and ethical hurdles faced during AI adoption.

Table 9 - Challenges when using AI tools in SD

#	Key responses
1	Lack of accuracy and incomplete information most of the time
2	Integration issues, lack of accuracy
3	Wrong suggestions
4	When I asked questions about some nuget packages, II realized that it did not bring up-to-date information, so I was lost a short time.
5	Sometimes, it provides too generic information. Sometimes, it does not link follow-up questions with original questions.
6	Lack of accuracy for tests. Until today any AI could not create tests that were executable. AI code helped a lot but it always need improvements and fixes

These key findings highlight AI's transformative position in SD while underscoring the need to address technical barriers for broader adoption.

5. Conclusion

AI has become a vital tool in transforming SD by automating repetitive work, optimizing workflows, and reducing the burden on developers. This study demonstrates that AI tools are widely adopted across several phases of the SDLC, significantly improving efficiency and reducing overtime.

Despite their advantages, challenges such as lack of accuracy, integration issues, and ethical considerations must be addressed to maximize their potential. Future studies should pay attention to improving the transparency, accuracy, and adaptability of AI systems to foster greater trust and integration in software engineering practices. By stressing these struggles, organizations can harness AI's potential to streamline development processes, enhance productivity, and support developers in achieving a better work- life balance.

The study has a few constraints. 1) The number of respondents from a single organization, 2) The study specifically focuses on how AI tools support developers, analyzing their impact on SD processes and reducing overtime by automating tasks in SD other than the general influence of AI on organizational processes. Future research can increase the sample size and include other Information Technology organization's standpoint on the effect of AI on how they do their work, create a certain level of performance, and happy about it, including job satisfaction, job absorption, job dedication, and job vigor.

References

Ajiga, D. (2024). Enhancing software development practices with AI insights in high-tech companies. *International Journal of Artificial Intelligence and Applications*, 12(4), 23-34. https://doi.org/10.51594/csitrj.v5i8.1450

Alenezi, M., & Akour, M. (2025). AI-driven innovations in software engineering: a review of current practices and future directions. *Applied Sciences*, 15(3), 1344. https://doi.org/10.3390/app15031344

Champa, A. I., Rabbi, F., Nachuma, C., & Zibran, M. (2024). ChatGPT in action: Analyzing its use in software development. In Proceedings of the 21st International Conference on Mining Software Repositories (MSR '24). https://doi.org/10.1145/3643991.3645077

Coutinho, M., Marques, L., Santos, A., Dahia, M., Franca, C., & Santos, R. (2024). The role of generative AI in software development productivity. ACM International Conference on AI-Powered Software. https://doi.org/10.1145/3664646.3664773

Fraiwan, M., & Khasawneh, N. (2024). A review of ChatGPT applications in education, marketing, software engineering, and healthcare. *Journal of Emerging Technologies*, 16(3), 101- 115.https://arxiv.org/pdf/2305.00237

Gerosa, M., Trinkenreich, B., Steinmacher, I., & Sarma, A. (2024). Can AI serve as a substitute for human subjects in software engineering research? *Automated Software Engineering*, 31(13), 200-215. https://doi.org/10.1007/s10515-023-00409-6

Gladun, S. (2025). What Is the Software Development Life Cycle (SDLC) and How Does It Work? https://agilie.com/blog/what-is-the-software-development-life-cycle-sdlc-and-how-does-it-work

Karyotakis, I., Talos, E., & Spinellis, D. (2026). TGIF: the evolution of developer commit times. *Empirical Software Engineering*, 31(3), 52. https://doi.org/10.1007/s10664-025-10767-2

Khaliq, Z., Farooq, S. U., & Khan, D. A. (2022). Artificial intelligence in software testing: Impact, problems, challenges, and prospect. *Journal of Software Testing Practices*, 11(5), 56-72. https://arxiv.org/pdf/2201.05371

Klemmer, J. H., Horstmann, S. A., Patnaik, N., et al. (2024). Using AI assistants in software development: A qualitative study on security practices and concerns. *ACM SIGSAC Conference on Computer and Communications Security*. https://doi.org/10.1145/3658644.3690283

Kytsak, T., & Ovsiannykov, D. (2025). Working hours of the future: AI technologies, collective synergy, and biorhythms as the foundation for productive work. *Social and labour relations: theory and practice*, 15(1), 1-10. doi:10.21511/slrtp.15(1).2025.01

Marar, H. W. (2024). Advancements in software engineering using AI. *Computer Software and Media Applications*, 6(1), 3906. https://doi.org/10.24294/csma.v6i1.3906

Mashkoor, A., Menzies, T., Egyed, A., & Ramler, R. (2022). Artificial intelligence and software engineering: Are we ready? *Computer,* 55(3), 24-28. https://doi.org/10.1109/MC.2022.3144805

Nguyen-Duc, A., Daniel, B., Przybylek, A., et al. (2023). Generative artificial intelligence for software engineering. *Journal of Software Innovations,* 19(3), 78-92. https://arxiv.org/pdf/2310.18648

Russo, D. (2024). Navigating the complexity of generative AI adoption in software engineering. *ACM Transactions on Software Engineering and Methodology*, 33(5), 135-145. https://doi.org/10.1145/3652154

Sauvola, J., Tarkoma, S., Klemettinen, M., Riekki, J., & Doermann, D. (2024). Future of software development with generative AI. *Automated Software Engineering,* 31(26), 1-15. https://doi.org/10.1007/s10515-024-00426-z

Tatineni, S., & Allam, K. (2024). AI-driven continuous feedback mechanisms in DevOps for proactive performance optimization and user experience enhancement. *Journal of AI Innovations*, 4(1), 114-120. https://healthsciencepub.com/index.php/jaihm/article/view/75

Yang, Y., Xia, X., Lo, D., & Grundy, J. (2022). A survey on deep learning for software engineering. *ACM Computing Surveys*, 54(10s), 1-3. https://doi.org/10.1145/35052