Balancing Innovation and Dependency: The Role of AI Tools in Modern Coding Practices

Yiğithan Döler
Department of Information Systems and Technologies
University of Bilkent
Çankaya, Ankara
yigithan.doler@ug.bilkent.edu.tr

Salih Elcicek
Department of Information Systems and Technologies
University of Bilkent
Çankaya, Ankara
salih.elcicek@ug.bilkent.edu.tr

Abstract
The adoption of artificial intelligence (AI) tools in software development has generated discussions regarding their influence on both code quality and the skills of developers. AIpowered platforms such as GitHub Copilot and Amazon CodeWhisperer significantly boost efficiency by handling routine coding tasks. However, there are rising concerns about the potential negative impact these tools might have on developers' coding expertise over time. This study investigates whether reliance on AI tools leads to a decrease in code quality and a weakening of developers' critical problem-solving abilities. By reviewing existing research and analyzing data, this paper evaluates the trade-offs between the productivity benefits of AI and the risks of over-dependence on automated solutions. Additionally, it explores the concept of "AI-induced technical debt," where excessive reliance on AI-generated code could lead to long-term challenges, such as reduced code maintainability. Ultimately, this research contributes to the ongoing debate by examining the benefits and potential downsides of AI's role in the future of software development.

Introduction

The advent of artificial intelligence (AI) tools has brought transformative changes to software development, revolutionizing how developers approach coding tasks. Platforms such as GitHub Copilot and Amazon CodeWhisperer are becoming integral in automating repetitive coding, enhancing productivity, and enabling developers to focus on complex challenges. However, this newfound convenience comes with potential drawbacks, particularly regarding over-reliance on AI and its implications on developer expertise and code quality. This study delves into the double-edged nature of AI tools in coding practices. While they enhance efficiency and collaboration, concerns about their long-term impact on critical problem-solving skills, especially among junior developers, are growing. Furthermore, the concept of "AI-induced technical debt" raises questions about the maintainability of AI-generated code and its implications for the future of software systems. By combining data analysis and survey findings, this research aims to explore the delicate balance between leveraging AI for innovation and avoiding excessive dependency that could hinder skill development and code integrity.

II. Hypotheses

This research is guided by the following hypotheses, which aim to evaluate the impact of AI tools on software developers' skills, productivity, and the quality of code:

H1: The use of AI tools improves the productivity of developers by automating repetitive tasks.

H2: Heavy reliance on AI tools negatively impacts the critical problem-solving skills of junior developers.

H3: AI tools contribute to the accumulation of technical debt in software projects by encouraging short-term fixes over long-term solutions.

H4: AI tools enhance collaboration and teamwork among developers by providing shared coding suggestions.

H5: Senior developers perceive AI tools as beneficial for focusing on high-level problem-solving tasks rather than routine coding.

III. Methodology

This study employs a mixed-method approach to analyze the effects of AI tools on developer productivity, skills, and code quality. The methodology was designed to test the hypotheses outlined above and provide comprehensive insights into how AI tools influence software development practices. The primary dataset was obtained through an online survey distributed to 50 software developers, encompassing various experience levels such as junior, mid-level, and senior developers. The survey consisted of structured questions designed to measure participants' perceptions of AI tools in five key areas: productivity, problem-solving skills, technical debt, collaboration, and task prioritization. Participants rated their agreement on these topics using a 5-point Likert scale, ranging from "Strongly Disagree" to "Strongly Agree."

The survey questions were grouped into the following categories to align with the research hypotheses:

- **Productivity:** Questions evaluating the efficiency gains from AI tools (H1).

- **Problem-Solving Skills:** Questions focused on how AI tools affect critical thinking, especially for junior developers (H2).
- **Technical Debt:** Questions assessing the long-term impact of AI-generated code on maintainability (H3).
- **Collaboration:** Questions exploring AI's role in enhancing teamwork and shared coding practices (H4).
- **Task Prioritization:** Questions addressing how senior developers leverage AI tools for high-level tasks (H5).
-

The participants included 30% junior developers, 50% mid-level developers, and 20% senior developers. This diverse sample ensured that the study captures insights across different career stages.

Incomplete responses and outliers were removed during data preprocessing. The cleaned dataset was analyzed using statistical software (R) to generate descriptive and inferential statistics.

- **Descriptive Analysis:** Metrics such as means, standard deviations, and frequency distributions were calculated to summarize the data.
- **Inferential Analysis:** Correlation and regression analyses were conducted to test the hypotheses and identify relationships between variables.

The results were visualized using bar charts, scatter plots, and correlation heatmaps to illustrate key findings effectively. Tools such as ggplot2 in R were employed for this purpose.

This study has certain limitations. Firstly, it relies on self-reported data, which may introduce bias. Secondly, the sample size, while diverse, is relatively small, limiting the generalizability of the findings. Lastly, the study does not account for industry-specific variations in AI tool adoption.

IV. Descriptive Analysis

Role Distribution Table

Role Distribution Summary

Role	Count
Junior Developer	9
Mid-level Developer	17
Other	5
Senior Developer	6
Tech Lead/CTO	3

Survey Questions Table

Question Number	Question Text
Q1	1. How many years of experience do you have in software development?
Q2	2. What is your current role?
Q3	3. I frequently use AI Tools for software development
Q4	4. Which AI-powered coding assistant(s) do you use?
Q5	5. AI tools have decreased the deep knowledge I have of a specific programming language.
Q6	6. AI-generated code often requires significant modification or rework.
Q7	7. AI tools have improved my problem-solving skills.
Q8	8. AI tools negatively impact junior developers by limiting their learning opportunities.
Q9	9. AI tools help senior developers focus on high-level coding tasks rather than repetitive work.
Q10	10. AI tools improve collaboration and teamwork within software development teams.

The ultimate goal of this survey was to uncover the varying impacts of AI tools on developers' workflows, skills, and team dynamics. By analyzing responses across different roles, we sought to identify patterns and trends that would inform best practices for integrating AI tools into software development processes.

Table 1 summarizes the distribution of roles among the survey participants. These roles included Junior Developers, Mid-Level Developers, Senior Developers, and Tech Leads/CTOs, ensuring a diverse set of perspectives. The role distribution was intentionally broad to capture the varying impacts of AI tools across different levels of expertise and responsibility.

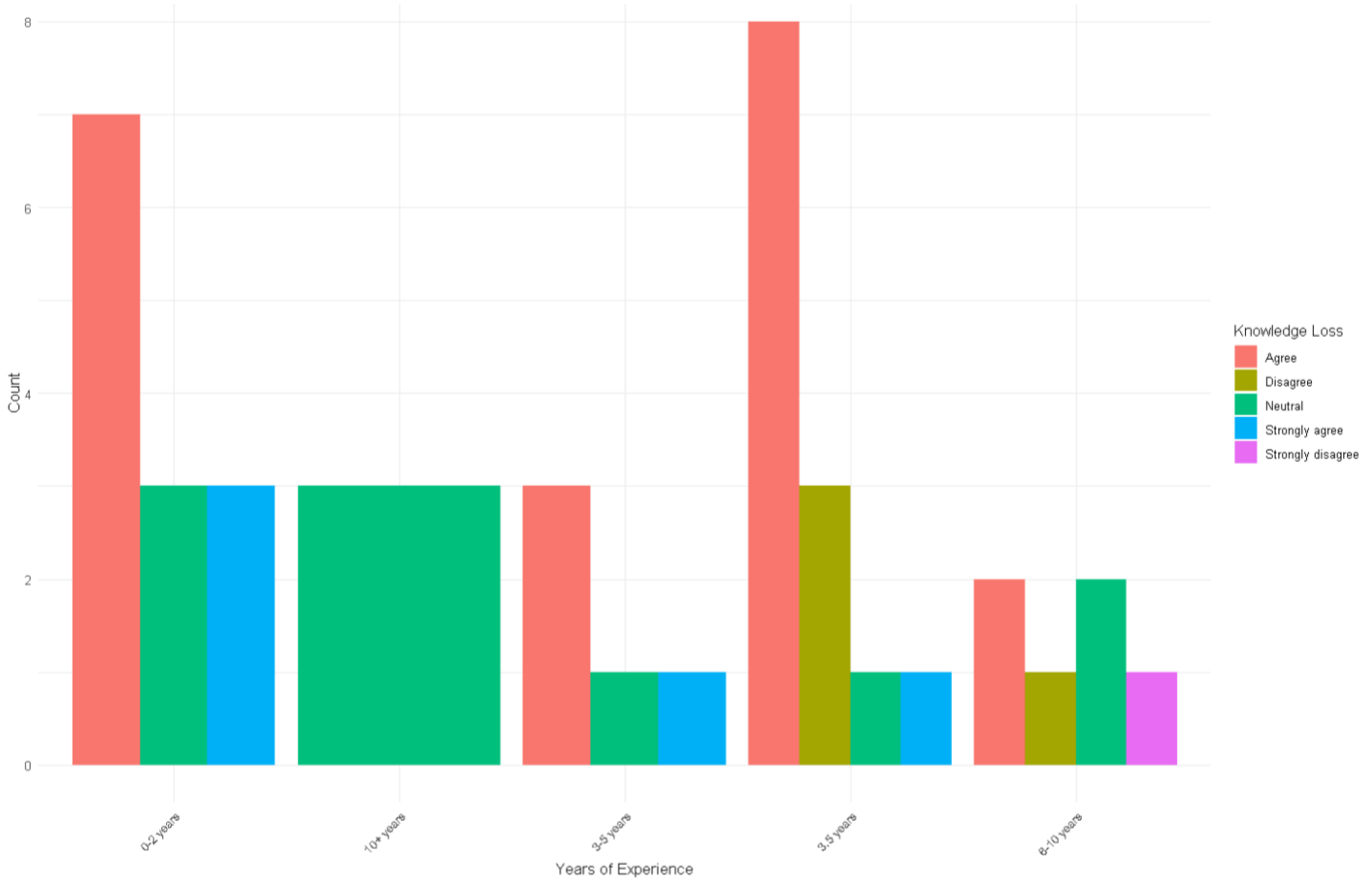
The survey was structured based on the following key principles:

1. **Clarity:** Each question was carefully phrased to ensure clarity and avoid ambiguity, enabling. Participants to provide accurate and thoughtful responses.

2. **Relevance:** All questions were directly tied to the study's hypotheses and objectives, ensuring that the data collected would be actionable and relevant.

3. **Inclusivity:** The survey was designed to accommodate participants from a wide range of roles, ensuring representation across different levels of experience.

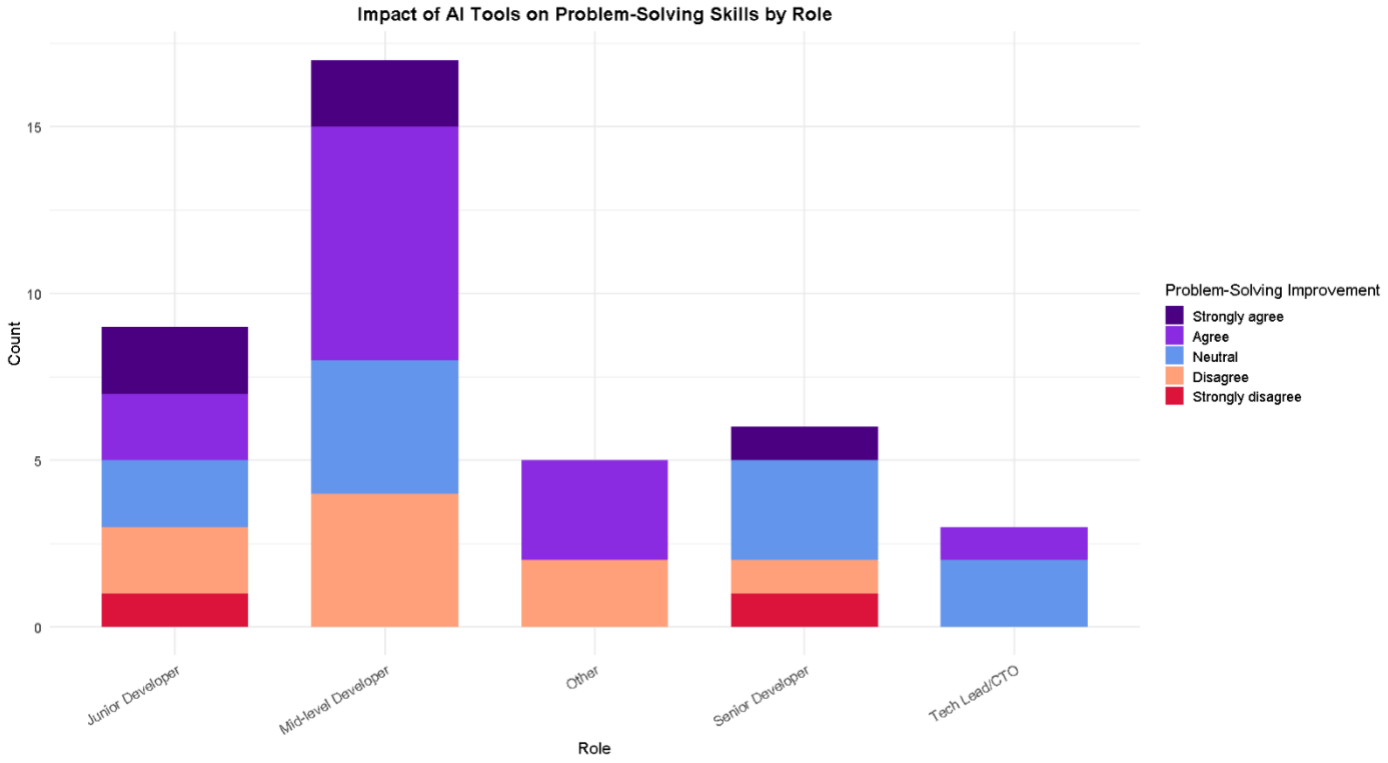
4. **Likert Scale Responses:** To standardize responses, most questions utilized a five-point Likert scale, ranging from **"Strongly Disagree" (1)** to **"Strongly Agree" (5).**



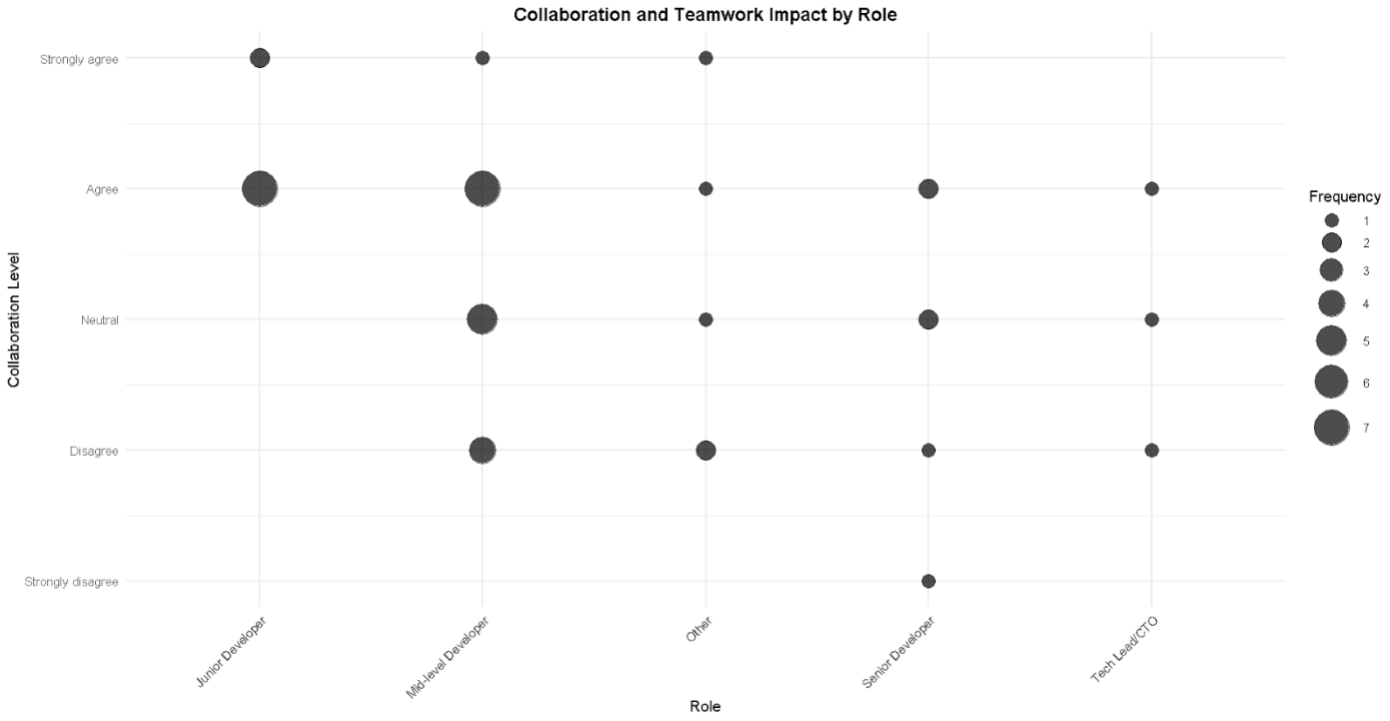
Şekil 1 Knowlede Use

Figure 1 demonstrates the varying perceptions of AI tools' impact on knowledge retention across different experience levels among software developers. Junior developers (0-2 years of experience) predominantly report a significant decrease in their deep knowledge of programming languages, as evidenced by the majority of responses falling under "Strongly Agree" and "Agree." This supports the hypothesis that heavy reliance on AI tools during the early stages of a developer's career may hinder the development of foundational skills and critical problem-solving abilities. Mid-level developers (3-5 years of experience), on the other hand, exhibit a more balanced view, with fewer extreme responses. This suggests that mid-level developers may have adapted to AI usage, balancing its benefits with the need for skill development. Senior developers (6+ years of experience) largely disagree or remain neutral about knowledge loss, reflecting their ability to leverage AI tools effectively without compromising their expertise. These findings highlight a key insight: while AI tools can accelerate productivity, their impact on skill development and knowledge retention varies significantly based on a developer's experience, emphasizing the importance of guided AI tool adoption and training to prevent long-term skill erosion.

Figure 2- This stacked bar chart illustrates the impact of AI tools on problem-solving skills across various roles in software development. Junior developers predominantly perceive AI tools as highly beneficial, with a significant portion of responses falling under "Agree" and "Strongly Agree." This indicates that AI tools might help streamline their tasks and enhance logical thinking. Mid-level developers show a more balanced distribution of opinions, with many responses clustered around "Neutral" and "Agree." This suggests that while AI tools are perceived as helpful, their impact might vary depending on individual expertise and use cases. Senior developers and Tech Leads/CTOs, on the other hand, exhibit a notable increase in "Neutral" and "Disagree" responses, reflecting a more critical or less impactful view of AI tools for their problem-solving abilities. This pattern highlights how the perception of AI tools' utility evolves with experience and role complexity, emphasizing the need for tailored AI integration strategies across different levels of expertise and responsibility.



Şekil 2 Problem Solving

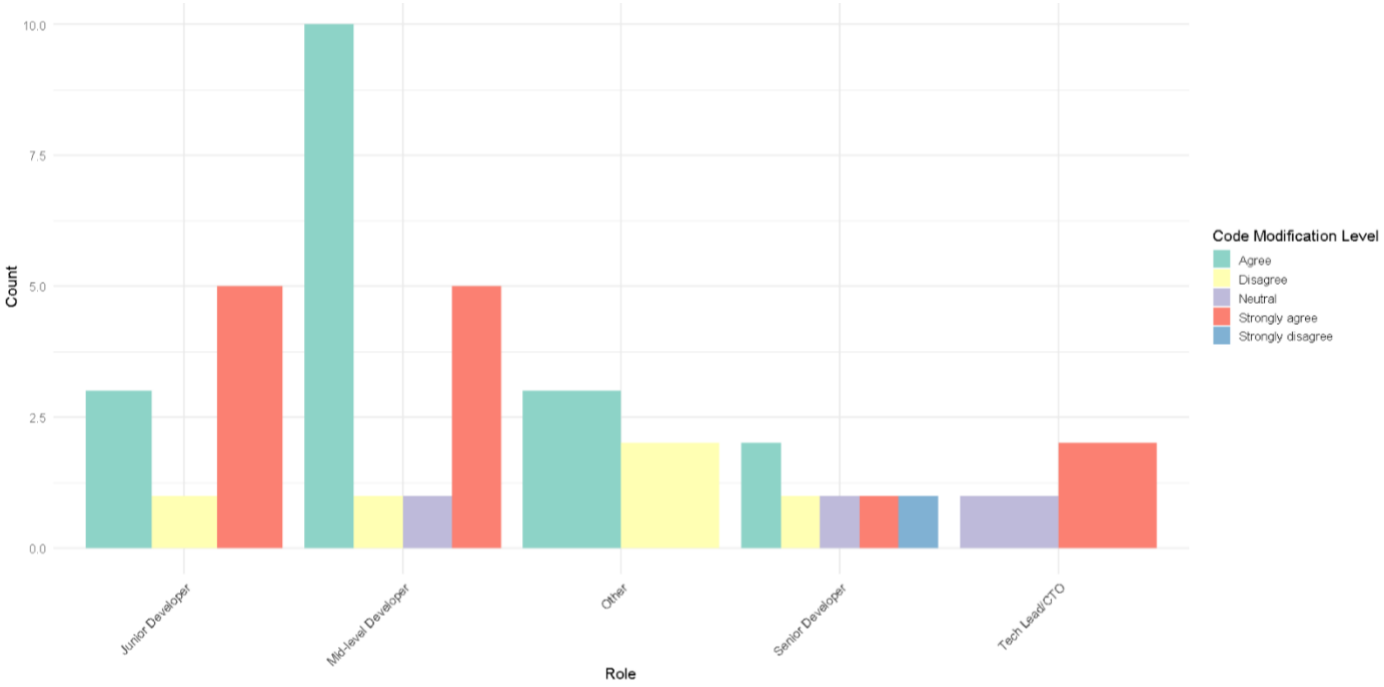


Şekil 3 Collobaration and teamwork

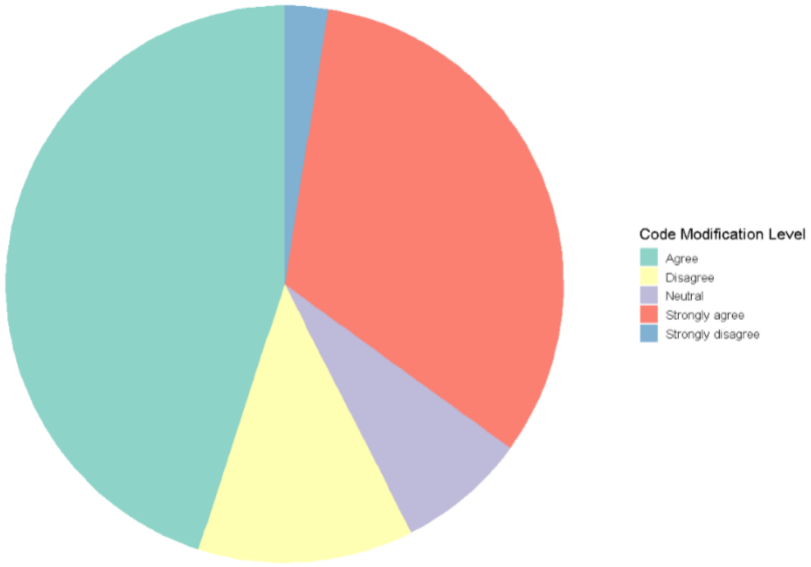
Figure 3 -The analysis of collaboration and teamwork reveals notable differences in how AI tools are perceived across roles in software development. Junior developers overwhelmingly believe that AI tools enhance teamwork, with responses heavily concentrated around **"Agree"** and **"Strongly Agree."** This could be attributed to their reliance on AI for assistance with shared tasks and communication, which might reduce barriers in collaborative environments. Tech Leads and CTOs also hold a predominantly positive view, likely valuing AI tools for aligning team efforts and facilitating project management at a strategic level. On the other hand, senior developers exhibit a more critical stance, with many responses leaning toward **"Neutral"** or **"Disagree."** This skepticism might stem from their reliance on interpersonal skills and established workflows, which AI tools cannot fully replicate. These patterns suggest that while AI tools are seen as enablers of collaboration, their perceived utility is shaped by the specific expectations and responsibilities tied to each role. This calls for a nuanced approach to integrating AI tools, ensuring they address the unique needs of different roles while fostering seamless collaboration.

Figure 4-5 The **pie chart** reveals that the majority of participants agree or strongly agree that AI-generated code often requires significant modification or rework, indicating a general perception that AI tools, while accelerating coding processes, frequently produce output that needs adjustments. The bar chart complements this by showing how these perceptions vary by role. Junior developers predominantly agree or strongly agree, suggesting they may struggle more with adapting AI-generated code due to their limited experience. Mid-level developers exhibit a more balanced distribution, reflecting their transitional stage where they begin integrating AI tools more effectively. Senior developers lean towards neutral or disagree, implying that their advanced skills allow them to minimize the need for extensive modifications. Leadership roles, such as Tech Leads/CTOs, primarily agree, likely due to their strategic use of AI tools to standardize or optimize team outputs rather than for direct coding tasks.

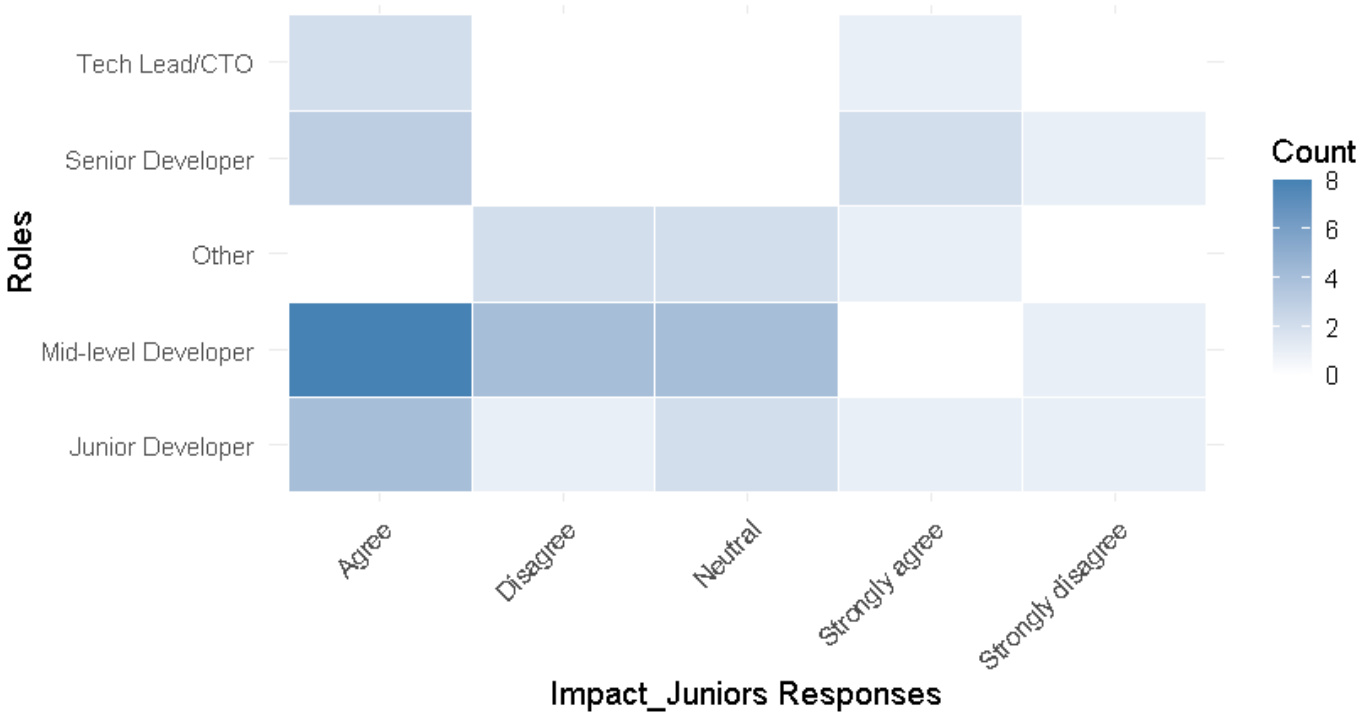
These findings highlight the evolving nature of AI tool usage across roles. Junior developers could benefit from targeted training on debugging and refining AI outputs, while mid-level developers might need best practices to optimize their workflows. For senior developers and leaders, tailoring AI solutions to align with strategic goals can further enhance their productivity. This underscores the need for role-specific approaches to maximize the potential of AI tools in software development workflows.



Şekil 4 Code Modification Bar Chart



Şekil 5 Collobaration and teamwork Pie Chart



Şekil 6 Heat Map aggrement for Juniors Learning Opportunities

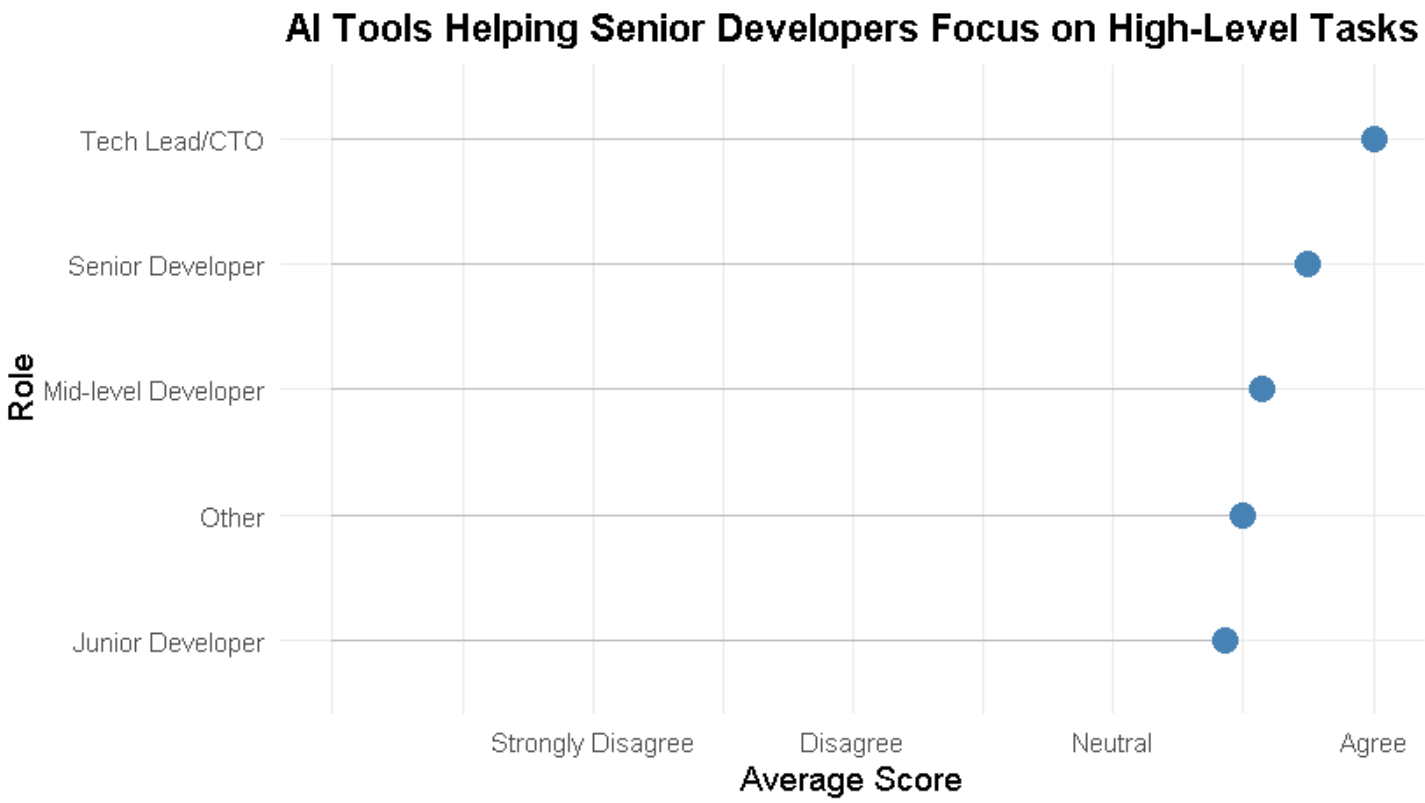
Figure 6 - The survey data on **"AI tools negatively impact junior developers by limiting their learning opportunities"** reveals mixed perceptions, with a significant portion of junior developers agreeing or strongly agreeing, suggesting a belief that reliance on AI tools may hinder foundational skill development and deep problem-solving engagement. This perspective aligns with concerns that juniors might prioritize AI-generated solutions over learning the intricacies of coding, potentially affecting their long-term growth. Conversely, some responses, particularly from senior developers and Tech Leads, reflect disagreement, emphasizing that AI tools can serve as valuable learning aids, providing immediate feedback and accelerating task completion. However, the negative impact can be further supported by arguing that over-reliance on AI diminishes critical thinking and debugging experience, crucial for skill development. Balancing the use of AI tools with structured learning opportunities could address these concerns, ensuring juniors benefit from AI without compromising their growth. This dual perspective highlights the need for mindful integration of AI tools in junior developers' workflows to maximize their potential benefits while mitigating risks.

Figure 7- The lollipop chart illustrates the average perception of how AI tools help senior developers focus on high-level tasks across various roles in software development. Senior developers and Tech Leads/CTOs exhibit the highest average scores, predominantly in the **"Agree"** and **"Strongly Agree"** categories. This indicates that these groups view AI tools as effective in automating repetitive tasks, freeing their time to concentrate on strategic and complex problem-solving. This alignment with their responsibilities suggests that AI tools are particularly beneficial for roles that demand high-level thinking and decision-making.

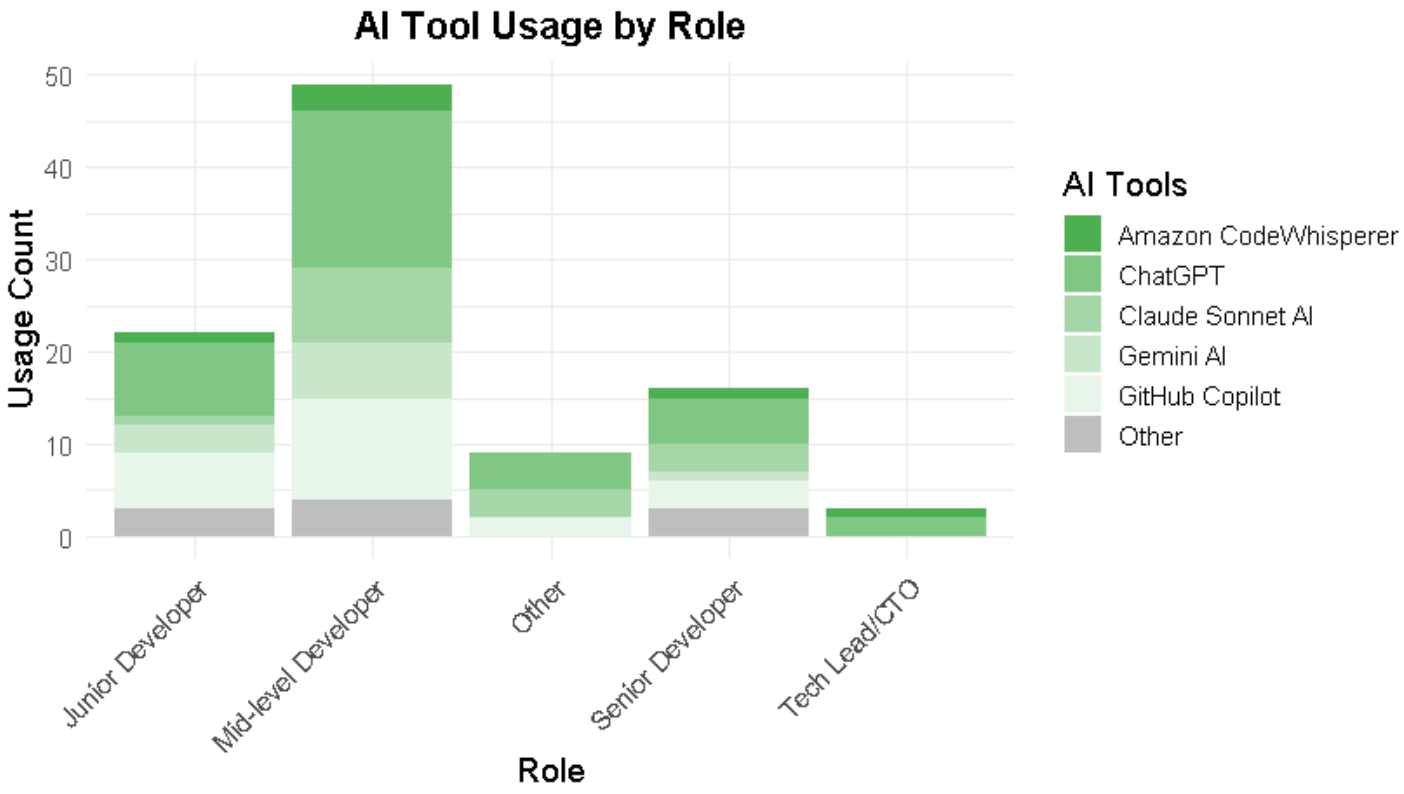
Mid-level developers display moderate scores, reflecting their transitional phase where they balance repetitive tasks with increasing exposure to high-level responsibilities. The variability in their responses may stem from differing levels of familiarity and reliance on AI tools within this group.

Junior developers report the lowest average scores, which can be attributed to their limited involvement in high-level tasks. Their primary focus on learning and foundational tasks might reduce their exposure to the benefits of AI tools in strategic contexts. Additionally, juniors may perceive AI tools more as learning aids rather than productivity enhancers for high-level work.

These patterns suggest that the utility of AI tools is closely tied to the nature of each role's tasks, emphasizing the need for tailored integration and training programs to maximize their impact across all experience levels.



Şekil 7 Lollipop Chart Focusing High Level Task

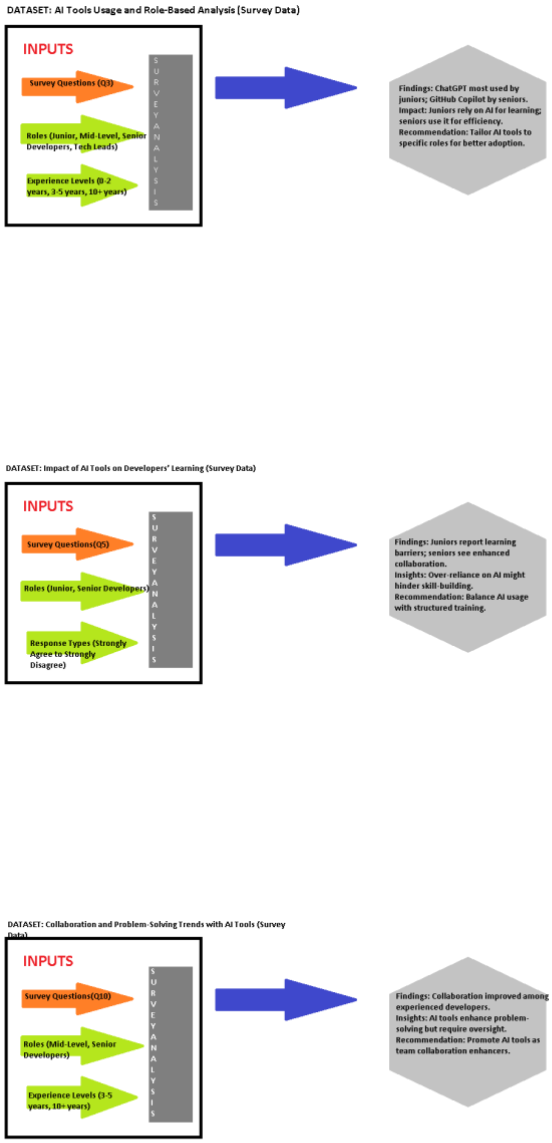


Şekil 8 AI Usage Bar Chart

Figure 8- The stacked bar chart reveals distinct patterns in AI tool usage among different roles. ***ChatGPT*** *emerges as the most widely used tool across all roles, especially among junior and mid-level developers. This dominance can be attributed to its accessibility, ease of use, and versatility, allowing developers to quickly generate code snippets, debug, or explore new concepts. Moreover, ChatGPT's conversational interface makes it particularly appealing for less experienced developers seeking immediate guidance.* ***GitHub Copilot****, the second most popular tool, is heavily favored by senior developers and Tech Leads. This preference likely stems from its seamless integration with popular IDEs and its ability to provide context-aware code suggestions, making it a powerful productivity enhancer for experienced developers. On the other hand,* ***Claude Sonnet AI*** *and similar lesser-known tools have the lowest usage rates. This can be explained by their limited market penetration, narrower feature sets, or lack of integration with widely used development environments. Additionally, awareness and familiarity with these tools may be lower among developers, further reducing their adoption.*

F,igure 9- The bar chart reveals a clear trend in the survey responses regarding the usage of AI tools. The majority of participants selected **"Agree"** and **"Strongly Agree"**, indicating a widespread positive perception of AI tools among respondents. This suggests that AI tools are largely seen as beneficial for enhancing productivity and simplifying repetitive tasks. A smaller portion selected **"Neutral"**, reflecting either limited experience with AI tools or ambivalence about their impact. Interestingly, there were no responses for **"Disagree"** or **"Strongly Disagree"**, highlighting the absence of significant negative sentiment toward AI tools in this sample. These findings demonstrate strong adoption and satisfaction with AI tools, particularly in roles where automation and efficiency are critical. However, the small presence of neutral responses suggests an opportunity to explore ways to better integrate AI tools for hesitant or inexperienced users. This overall positivity reinforces the importance of continuing to innovate and tailor AI solutions to meet diverse needs in software development.

Graphical Abstract



Conclusion

The findings of this study highlight the transformative role of AI tools in modern software development, showcasing both their potential and their limitations. AI tools, such as ChatGPT and GitHub Copilot, have become integral in automating repetitive coding tasks, improving productivity, and enabling developers to focus on more complex challenges. The widespread adoption of these tools, particularly among junior and mid-level developers, underscores their accessibility, ease of use, and ability to bridge knowledge gaps. However, this reliance also raises concerns about the diminishing opportunities for skill development and critical problem-solving, especially for less experienced developers.

The results indicate that junior developers frequently perceive AI tools as beneficial for task execution and logical thinking but may inadvertently prioritize AI-generated solutions over foundational learning. Mid-level developers present a more balanced view, reflecting a transitional stage in their careers where they learn to integrate AI tools effectively without over-reliance. Senior developers and Tech Leads, in contrast, leverage these tools strategically, focusing on high-level problem-solving and team management while maintaining their expertise and critical thinking skills.

One of the key insights from this analysis is the role-specific adoption of AI tools. ChatGPT, as the most widely used tool, is favored for its versatility and conversational interface, appealing especially to junior developers. On the other hand, GitHub Copilot's integration with development environments makes it a preferred choice for senior developers seeking context-aware code suggestions. However, lesser-known tools like Claude Sonnet AI lag in adoption due to limited awareness and narrower functionalities.

This study emphasizes the need for a balanced and role-specific approach to integrating AI tools into software development workflows. While AI tools offer significant productivity gains, fostering structured learning opportunities and providing training on the effective use of these tools is crucial to prevent over-reliance and ensure long-term skill development. By addressing these challenges, AI tools can continue to enhance software development practices while supporting the diverse needs of developers across all roles and experience levels.

REFERENCES

[1] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer, "The impact of AI on developer productivity: Evidence from GitHub Copilot," arXiv preprint, last modified February 13, 2023. [Online]. Available: https://ar5iv.labs.arxiv.org/html/2302.06590. Accessed: Oct. 24, 2024.

B. Martinović and R. Rozić, "Impact of AI tools on software development code quality," EasyChair Preprint No. 13562, University of Mostar, Mostar, June 6, 2024. [Online]. Available: https://easychair.org/publications/preprint/13562. Accessed: Oct. 24, 2024.

D. Sadler, "Humans do it better: GitClear analyzes 153M lines of code, finds risks of AI," Arc Talent Blog, Jan. 30, 2024. [Online]. Available: https://arc.dev/talent-blog/impact-of-ai-on-code/. Accessed: Oct. 24, 2024.

• D. Cotroneo, et al., "Automating the correctness assessment of AI- generated code for security contexts," The Journal of Systems and Software, vol. 216, 2024, Art. no. 112113. [Online].Available: https://doi.org/10.1016/j.jss.2024.112113. Accessed: Oct. 24, 2024.

• G. Recupito, et al., "Technical debt in AI-enabled systems: On the prevalence, severity, impact, and management strategies for code and architecture," The Journal of Systems and Software, vol. 216, 2024, Art. no. 112151. [Online]. Available:https://doi.org/10.1016/j.jss.2024.112151. Accessed: Oct. 24, 2024.

• K. Misiejuk, R. Kaliisa, and J. Scianna, "Augmenting assessment with AI coding of online student discourse: A question of reliability," Computers and Education: Artificial Intelligence, vol. 6, 2024, Art. no. 100216. [Online]. Available: https://doi.org/10.1016/j.caeai.2024.100216. Accessed: Oct. 24, 2024.

• D. Howell, "Can AI code generation really replace human developers?" ITPRO, Mar. 28, 2024. [Online]. Available: https://www.itpro.com/technology/artificial-intelligence/can-ai-code- generation-really-replace-human-developers. Accessed: Oct. 24, 2024.

• "Conceptualising Software Development Lifecycle for Engineering AI Planning Systems," Proceedings of the 2024 IEEE Conference on Artificial Intelligence, IEEE, 2024. [Online]. Available:https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=10164777. Accessed: Oct. 24, 2024.