Issues in Information Systems
Volume 26, Issue 2, pp. 346-360, 2025

DOI: https://doi.org/10.48009/2_iis_127

Perceptions and challenges of AI-driven code reviews: A qualitative exploration of developer experiences

Sebastian Cataldi, *Middle Georgia State University, scastaldi@gmail.com*

Abstract

AI-driven code review tools represent transformative advances in software development, improving efficiency, productivity, and accuracy in code reviews. Despite these potential benefits, concerns about trust, reliability, and contextual comprehension persist, limiting their widespread adoption. This qualitative study examines the perceptions and challenges faced by software developers regarding AI-driven code review tools. Through semi-structured and thematic analysis involving software developers, technical leads, and architects, the study identifies central themes, including trust in AI-generated recommendations, the impact on developer productivity, ethical considerations, and the need for contextual awareness. While participants acknowledge the efficiency gains and educational value provided by AI tools, skepticism remains regarding the tools' ability to interpret complex business logic and domain-specific scenarios. Participants advocate for enhancements in AI-driven tools, highlighting the need for improved contextual awareness, transparency, ethical integration, and seamless workflow integration. This research adds valuable empirical insights to ongoing discussions in software engineering literature, emphasizing AI-driven code reviews as complementary tools that augment human expertise in software development processes.

Keywords: AI-driven code review, trust in AI, developer perceptions, developer experiences.

Introduction

AI-driven code review tools have emerged as promising innovations designed to improve the efficiency and effectiveness of software development processes. According to Vijayvergiya et al. (2024), modern code review is a collaborative practice where peers review code changes before they are merged into the version control system, ensuring compliance with best practices. Manual code reviews effectively identify defects, enforce coding standards, and facilitate knowledge sharing among developers (Tufano et al., 2021; Rasheed et al., 2024). However, with increasing software complexity and larger projects, maintaining thorough and frequent manual reviews becomes increasingly challenging. To address this challenge, AI-driven tools have been developed to automate key aspects of code review, including the detection of code smells, which are subtle indicators of potential design flaws or deeper issues, the recommendation of refactoring actions, and the prediction of errors based on historical patterns (Almeida et al., 2024; Gerede & Mazan, 2018).

Despite their potential, adoption of these AI-driven code review tools remains slow. Developers express skepticism concerning reliability, contextual accuracy, and ethical considerations associated with AI-

generated feedback (Bird et al., 2023; Ernst & Bavota, 2022). A recurring concern is whether AI can effectively comprehend and handle the nuances of business logic and project-specific contexts, which are critical to meaningful code assessments beyond superficial issues like syntax errors (Bird et al., 2023). Addressing these perceptions and challenges is essential to enhancing trust and integration of AI-driven tools into development workflows.

This qualitative study investigates software developers' perceptions and challenges associated with AI-driven code review tools. Through semi-structured interviews and thematic analysis, this research explores developers' experiences to identify key factors influencing their trust, acceptance, and integration of AI technologies in established development workflows. This investigation provides valuable insights into the strengths, limitations, and potential roles of AI-driven code review tools, emphasizing their capacity as complementary aids rather than replacements for human reviewers. This study contributes to enhancing the future development, adoption, and effective use of AI in software engineering contexts by addressing the following research question:

RQ1: *What are the perceptions and challenges experienced by developers when using AI-driven code review tools?*

Literature Review

Modern Code Review (MCR) plays a crucial role in maintaining and enhancing the security and quality of software. The OWASP Code Review Guide (OWASP Foundation, Inc., 2017) provides a comprehensive framework for secure code reviews, emphasizing their integration into the software development life cycle (SDLC) to identify and mitigate vulnerabilities early, thus enhancing application security. The guide advocates code reviews to promote security awareness and responsibility among development teams, facilitating knowledge sharing and skill development within organizations, leading to more secure software products. Furthermore, Khleel and Nehéz (2020) explore the role of code reviews, contrasting formal inspections with modern code reviews (MCR). While formal inspections are thorough, their cost and rigidity make them less suitable for agile environments. In contrast, supported by tools, MCR offers a flexible, collaborative approach that improves review efficiency and software quality. The study underscores optimizing code review processes by considering technical and non-technical factors to enhance collaboration and learning.

Badampudi et al. (2023) also provides a comprehensive survey of MCR practices, proposing a research agenda to align academic research with industry practices, thereby enhancing the effectiveness and efficiency of MCR processes. The authors emphasize that MCR practices are essential for improving code quality, reducing post-delivery defects, and facilitating knowledge sharing among developers. Furthermore, Doğan and Tüzün (2022) identify prevalent code review smells in open-source software (OSS) projects, which impact software quality and development efficiency. Addressing these issues can reduce technical debt and improve code review practices, software quality, and collaboration. Finally, Afzali et al. (2023) highlight vulnerabilities in modern web-based code review systems due to inadequate integrity mechanisms, stressing the importance of code reviews in ensuring software quality and security.

Integrating Artificial Intelligence (AI)

Recent studies have emphasized the transformative impact of AI and machine learning on improving the efficiency and quality of code reviews. For instance, Gerede and Mazan (2018) explored how predictive models can determine whether a proposed code change is likely to require revisions. Their findings suggest that such models can help streamline the review process, minimize the number of review iterations, and enhance developer satisfaction by offering actionable insights before the review even begins.

Building on these advancements, Pejic et al. (2023) explore enhancing pull request recommendation systems, particularly in large-scale repositories with extensive developer involvement. Their findings highlight the importance of optimizing reviewer recommendations to manage large-scale repositories effectively, thus supporting better management of reviewer workloads and improving review process efficiency. Additionally, Turzo et al. (2023) propose a novel approach to improving code review analytics by automatically classifying reviewer comments. They developed a deep neural network model that incorporates code context, comment text, and code metrics to categorize review comments into five high-level groups. This classification system helps prioritize more critical feedback, thereby increasing the efficiency and overall effectiveness of the review process.

In a related study, Yin et al. (2023) address the growing complexity and volume of code reviews by introducing an automated model that integrates program structure and code sequences for enhanced learning. Their approach uses a program dependency graph serialization (PDG2Seq) algorithm to transform the structural elements of the code into a unique graph-based sequence, preserving both the semantic and structural characteristics of the program. The study underscores the importance of maintaining contextual information within analysis models and contributes to the development of more robust tools for managing large-scale and complex software projects.

Furthermore, Zydroń and Protasiewicz (2023) explore the automation of code review processes to improve efficiency and accuracy in large-scale software development projects. They highlight the challenge of manually assigning reviewers to pull requests and propose automated techniques utilizing machine learning and social network analysis to recommend suitable reviewers. The results indicate that the proposed automated review system can enhance efficiency and accuracy in code review processes. Integrating natural language processing (NLP) and machine learning techniques, such as pre-trained models such as ChatGPT4, enhances review annotation and accuracy. These findings suggest that automated review systems can increase transparency and accountability, positively impacting project outcomes (Zydroń & Protasiewicz, 2023).

In a comprehensive study, Almeida et al. (2024) explore the application of Artificial Intelligence (AI) in code review processes to enhance the quality and efficiency of software development. Their research demonstrated significant improvements in review efficiency and effectiveness. AICodeReview, a tool developed by the authors, reduced review time, detected more code smells, and facilitated more effective refactoring compared to manual reviews. These findings support the continued development and integration of AI-driven tools in software development workflows, emphasizing the importance of combining automated tools with human expertise for optimal outcomes in code reviews. Integrating AI-based techniques in code review processes offers significant potential for improving overall software quality and development efficiency (Almeida et al., 2024).

Finally, Baumgartner et al. (2024) present an AI-driven pipeline designed to address data clumps in software repositories. Data clumps, or variables frequently appearing together, indicate poor code structure and pose maintenance challenges. The study shows that by integrating LLMs like ChatGPT, the pipeline provides semantic insights that improve refactoring accuracy, address maintenance challenges associated with data clumps, and reduce technical debt. The automated refactoring process enhances overall code quality and maintainability. The study concludes that combining AI-driven techniques with human expertise results in more effective refactoring processes, highlighting the potential of AI in transforming software maintenance practices (Baumgartner et al., 2024).

Ethical Considerations

While the above research demonstrates substantial advancements in modern code review practices, ethical considerations continue to represent significant challenges that require further investigation. Trust, reliability, and contextual accuracy in AI-driven development environments (AIDEs) is crucial for their effective and ethical integration into software development. Ernst and Bavota (2022) discuss the emergence of AIDEs and their transformative potential in software engineering, exemplified by tools like GitHub Copilot, which use large language models such as Codex to automate routine coding tasks and enhance developer productivity through real-time code suggestions. Complementing this discussion, Bird et al. (2023) underscore the necessity of trust in AI-generated code and ethical considerations to adopt these tools effectively. Their research calls for future studies to enhance the reliability, contextual accuracy, and ethical implications of AI-powered programming tools, ensuring they complement developers' workflows and contribute positively to the software development process. Developers must balance the benefits of AI suggestions with potential risks, such as reduced code comprehension and increased security vulnerabilities. The integration of AI in software development underscores the need for new skills, particularly in code review and validation, and understanding the dynamics between developers and AI tools is essential for optimizing their use in real-world settings.

Human Oversight

The evolving landscape of code review processes highlights the interaction between automated tools and human oversight. Whether through peer reviews in distributed environments, automated task integration, or the use of advanced AI models, the emphasis remains on enhancing efficiency and effectiveness while maintaining the critical role of human expertise. Dos Santos and Nunes (2018) investigate the effectiveness of peer code review in distributed software development (DSD) using objective data from code repositories and subjective data from developer surveys. The study emphasizes the importance of considering technical and non-technical factors in DSD. While automated tools enhance the review process, they cannot replace the need for active human participation. The balance between review thoroughness and efficiency is critical, especially in DSD contexts. By combining empirical data and subjective insights, the study provides a comprehensive understanding of code review effectiveness, highlighting the nuanced requirements of peer code reviews in distributed environments (Dos Santos & Nunes, 2018). In a related study, Baumgartner et al. (2024) demonstrate how the automated refactoring process enhances overall code quality and maintainability. Their findings underscore that combining AI-driven techniques with human expertise results in more effective refactoring processes, highlighting AI's potential to transform software maintenance practices (Baumgartner et al., 2024).

The literature illustrates significant advances in software quality facilitated by Modern Code Review (MCR) and AI-driven tools. According to Keary (2017), integrating structured security-focused code review practices into the software development life cycle (SDLC) enables early vulnerability detection, enhances software security, and fosters collaborative knowledge sharing among development teams. Khleel and Nehéz (2020) highlight MCR's adaptability for agile environments compared to traditional inspections. AI tools like AICodeReview and GitHub Copilot further enhance review efficiency and developer learning but raise ethical concerns around transparency and trust (Ernst & Bavota, 2022; Bird et al., 2023). Human oversight remains critical, particularly in complex and distributed contexts (Dos Santos & Nunes, 2018; Baumgartner et al., 2024). While AI-driven code reviews have transformative potential, their success depends on ethical integration and a balanced partnership with human expertise.

Furthermore, Turzo (2023) proposes improving modern code reviews (MCR) effectiveness by automating tasks, such as reviewer selection and bug identification, to reduce time and resources spent. Integrating these models with static analysis tools can identify and suggest potential solutions for code defects that might otherwise be missed. This approach aims to streamline code reviews, making them more efficient

while relying on human oversight to ensure the quality of the automated tools. Furthermore, Kotsiantis et al. (2024) explore AI-assisted programming, focusing on utilizing code embeddings and transformers to enhance software development tasks. These technologies reduce manual coding efforts and minimize errors, making software development more efficient. Kotsiantis et al.(2024) highlights the importance of addressing the limitations and challenges of current AI technologies. It emphasizes the need for collaboration between AI researchers and software developers to advance AI-assisted programming, suggesting that these tools will be widely adopted in integrated development environments (IDEs), playing a crucial role in the evolution of software development.

Methodology

This study employs a qualitative research design to explore the perceptions and challenges faced by software developers when using AI-driven code review tools. Thematic analysis was chosen as the analytical method because of its strength in capturing complex and nuanced experiences that are often missed by quantitative approaches (Creswell, 2013). For data analysis, we employed an inductive thematic analysis using Quirkos. The process began with initial open coding of interview transcripts. Codes were then grouped and refined iteratively through collaborative discussions among the researchers. Using Quirkos to identify co-occurring themes and patterns, which were validated across participants to ensure conceptual coherence. Quirkos' interactive interface facilitated theme development through visual mapping and clustering of related codes. This helped identify relationships among themes and supported iterative refinement throughout the coding process.

Instrument

The instrument consisted of a semi-structured interview guide with open-ended questions organized into six sections. Section one focused on background and experience, section two explored perceptions, section three addressed challenges, section four gathered suggestions for improvement, section five examined future and ethical considerations, and section six included closing questions. All interviews were conducted live via Microsoft Teams. This approach enabled in-depth research and proved effective in detailing the experiences of participants and understanding the context in which these experiences occurred (Guest et al., 2006).

Sampling

Semi-structured interviews were conducted with 10 participants, including software developers (n = 3), senior developers (n = 4), team leads (n = 2), and a software architect (n = 1). All participants had been using AI-driven code review tools for at least six months. The sample size was determined based on the principle of data saturation, which is reached when no new themes or insights emerge from additional interviews (Guest et al., 2006). This approach supports the collection of rich, meaningful data while maintaining research efficiency.

Data Collection

The data was analyzed using thematic analysis, a method for identifying, analyzing, and reporting patterns in qualitative data (Creswell, 2013). The interviews were transcribed and the initial codes were generated systematically throughout the dataset, reviewed and categorized into themes. The themes were then refined and validated to reflect the essence of the data in relation to the research questions. A thematic map was also created to illustrate the relationship between themes (Creswell, 2013).

Participants were selected using purposive sampling to ensure a diverse representation of roles in software development, including junior developers, team leads, and software architects, aimed to capture a broad range of perspectives on AI-driven code review tools.

Results

To address the research question, this study conducted a qualitative analysis of interview data from 10 participants. The analysis was performed using Quirkos, a qualitative data analysis tool that facilitated the systematic coding, organization, and visualization of findings. This process enabled a detailed exploration of developers' perceptions and the challenges they face when using AI-driven code review tools. The analysis identified six key themes: efficiency, learning, trust, challenges, collaboration, and future expectations, providing a comprehensive understanding of how these tools influence software development workflows.

Table 1. Themes

Title	Total Codes
Learning and Knowledge	2
Educational Tool	7
Challenges	3
Hallucinations	4
Context Awareness	5
Code Consistency	6
Data Privacy Issues	3
Bias	1
Incorrect or Irrelevant Feedback	14
Impact on Developer Skills	4
Trust	22
AI vs. Traditional Code Reviews	6
Efficiency and Speed	7
Potential to Evolve	15
Transparency	6
Ethics and Security Concerns	11
Human-AI Collaboration	14
Skepticism	13
Reliability in Code Review	14
Lack of Context Understanding	17
Future Expectations and Improvements	20
Productivity	19
Reduce Workload	12
Integration in Workflows	5
Total Codes	**230**

The thematic analysis revealed six central themes: efficiency and productivity, learning and knowledge enhancement, trust and reliability, challenges and limitations, collaboration and workflow integration, and future expectations. Figure 1 highlights the most frequently discussed terms, visually reinforcing the prominence of these themes in the data (Appendix B, Figure 1).

Efficiency and Productivity

Participants highlighted that AI-driven tools reduced manual code review time and allowed focus on complex tasks. For example, one participant explained, "Compared to humans, the AI is very thorough when looking at the code base and the changes within a pull request" (Appendix A, Participant 01). Another participant emphasized time savings as a key benefit, "This automated code review process saves a lot of our time... it is pretty neat and pretty good" (Appendix A, Participant 05). Similarly, one developer observed, "It goes to that review instantly within 5 minutes... improving productivity for one developer and the whole team" (Appendix A, Participant 08). The thoroughness of AI in detecting errors and suggesting improvements was a recurring theme, boosting productivity. Participants also noted that AI tools minimized human error, ensured consistency, and enabled faster development cycle iterations. This theme was especially relevant in agile development contexts, where fast iteration cycles and time-sensitive deliverables amplify the benefits of automation.

Learning and Knowledge Enhancement

Participants expressed that AI-driven code reviews reinforced coding principles and provided educational value through detailed explanations. As one developer described, "Having explained chunks of code is helping me learn at a faster pace than if I tried to go out and read articles" (Appendix A, Participant 02). Another noted, "It reinforces the fundamentals" when revisiting older code (Appendix A, Participant 01). A third participant shared, "My biggest interest... is learning more about the coding frameworks I'm working on" (Appendix A, Participant 02). This theme was prevalent, with many participants recognizing AI tools as ongoing learning aids that helped them adopt best practices. Additionally, participants highlighted that AI tools exposed them to alternative coding methods and encouraged them to stay updated with evolving standards. The guidance provided by AI tools was seen as particularly valuable for junior developers, providing mentoring support and accelerating their learning curve.

Trust and Reliability

While many participants acknowledged AI's reliability in detecting errors, they also expressed skepticism regarding its ability to understand complex business logic and project-specific contexts. For instance, one participant said, "It's around 85% to 90%. I cannot completely trust it because... it does not work as expected" (Appendix A, Participant 06). Similarly, one participant noted, "I haven't seen where it's blatantly incorrect, but I've seen irrelevant suggestions" (Appendix A, Participant 02). This shows that despite some perceived reliability, participants remained cautious. Developers consistently emphasized the importance of human oversight, advocating for a balanced approach in which AI acts as an assistant rather than a replacement. These concerns were particularly salient in projects with high business logic complexity, where developers relied more on domain knowledge than syntax-level correctness. Ethical concerns emerged prominently across interviews, particularly regarding data privacy and intellectual property. These concerns underscore the need for transparent data handling policies and clear boundaries on AI training data, aligning with calls for ethical accountability in enterprise AI tools (Bird et al., 2023).

Challenges and Limitations

A notable challenge identified by participants was the AI's struggle with business logic, leading to improper suggestions. As one participant explained, "These tools recommend deprecated features... it does not understand new features" (Appendix A, Participant 04). Another added, "The higher you go, the more context it loses" (Appendix A, Participant 01). A different developer pointed out, "Once you address some

comments... you might receive double the comments in return" (Appendix A, Participant 09). Contradictory feedback from AI tools created additional workload, and concerns about training data quality and ethical implications were often raised, highlighting widespread concerns. Developers reported that AI tools sometimes generated false positives, requiring careful review and offsetting some time-saving benefits. Ethical concerns included data privacy, bias in AI training models, and potential over-reliance on automated systems, which participants believed could undermine critical thinking and collaborative code review practices.

Collaboration and Workflow Integration

Participants had mixed experiences with AI tools. While some valued quick feedback that reduced reliance on human reviewers, others felt AI hindered peer discussions. One participant shared, "Whenever there is an issue, it tags the team member, and we usually collaborate and talk about that issue" (Appendix A, Participant 05). Another reflected, "The more we use it, the more ingrained it becomes in the workflow" (Appendix A, Participant 03). A recurring suggestion was improving AI's contextual awareness for better workflow integration. Participants also noted that AI tools could disrupt established review processes by introducing conflicting suggestions that required mediation. However, many acknowledged that AI tools streamlined repetitive tasks when integrated effectively, allowing human reviewers to focus on high-impact code assessments, thus enhancing team efficiency.

Future Expectations

Participants expected AI tools to continue as complementary aids to human reviewers, with hopes for improvements that would reduce manual intervention. As one participant envisioned, "In the future, it will just write a code and then give it to us... and then making sure it looks good" (Appendix A, Participant 05). Another shared, "Yes, it will evolve... we'll simply be inputting prompting with business requirements" (Appendix A, Participant 02). One participant noted, "I think it's good, but I wouldn't rely solely on code reviews" (Appendix A, Participant 10). Several participants highlighted continuous training and adaptation of AI tools as essential, underlining the need for development. Participants also expressed optimism about the future of AI-driven code review tools, such as improved contextual understanding, adaptive learning from project-specific code bases, and enhanced security features. The need for customizable AI-driven code review tools tailored to specific project needs was also stressed, with participants hoping for more transparent and explainable AI operations to build trust and improve adoption.

Discussion

The results of this qualitative study contribute to the existing literature by highlighting both positive perceptions and ongoing challenges faced by developers when using AI-driven code review tools. Consistent with prior research, the participants acknowledged multiple advantages associated with integrating AI-driven tools, including enhanced efficiency, productivity, and opportunities for skill enhancement (Almeida et al., 2024; Rasheed et al., 2024; Tufano et al., 2021). Participants reported that AI-driven tools effectively reduced their workload by quickly identifying errors, suggesting relevant refactoring options, and automating repetitive tasks. These observations align closely with Almeida et al. (2024), who found that AI tools significantly reduced the time required for code reviews while improving overall code quality. Similarly, Gerede and Mazan (2018) demonstrated that AI-based predictive models increased review efficiency and improved developer satisfaction by proactively identifying code issues requiring review.

However, despite these perceived benefits, the study findings underscore persistent skepticism among developers, particularly regarding AI's limitations in handling complex business logic and domain-specific

contexts. Participants expressed doubts about the AI tool's ability to interpret deeper layers of contextual information, echoing concerns documented by Bird et al. (2023), who noted developers' reluctance to fully trust AI-generated suggestions due to perceived inadequacies in understanding intricate project-specific details. Ernst and Bavota (2022) further emphasized the importance of trust and reliability in AI-driven development environments, advocating that human oversight remains essential, especially when handling complex or sensitive software tasks.

Moreover, ethical considerations emerged as a significant dimension shaping developer perceptions. Developers expressed concerns about data privacy, inherent biases in AI training datasets, and the risk of excessive reliance on automation, which could undermine the critical thinking and collaborative practices of developers. These ethical challenges mirror concerns discussed extensively in prior studies, including Baumgartner et al. (2024), who underscored the necessity of transparent and explainable AI recommendations to mitigate bias and promote accountability. The developers' apprehensions around ethical issues, particularly data privacy and bias, align with broader ethical discourses highlighted in recent literature (Bird et al., 2023; Ernst & Bavota, 2022).

This study also reveals practical implications for improving AI-driven code review tools. Participants called for improvements in the contextual awareness and accuracy of AI tools, advocating for increased customization capabilities, transparency in recommendation processes, and seamless workflow integration. These findings are consistent with recommendations from the OWASP Code Review Guide (Keary, 2017), which emphasizes the integration of security best practices and transparency into software development processes. Furthermore, these findings align with studies by Pejic et al. (2023) and Zydroń and Protasiewicz (2023), which advocate for more contextually aware and customizable AI tools capable of adapting dynamically to specific project environments to optimize reviewer effectiveness and enhance overall software quality. These findings align with recent research emphasizing that AI systems should support rather than replace human judgment. Concerns about transparency and trust echoed throughout participant responses, reflecting broader themes in responsible AI design (Baumgartner et al., 2024).

Participants also expressed future expectations for AI-driven code review tools, highlighting the need for continuous improvement in adaptive learning capabilities, better contextual awareness, and more robust integration into existing development workflows.). These expectations resonate with suggestions from Almeida et al. (2024) and Rasheed et al. (2024), who have recommended ongoing refinement of AI's capabilities through deeper contextual learning and integration into established human-driven processes.

In summary, while the benefits of AI-driven code review tools in software engineering practices are recognized, there are still significant concerns about contextual understanding, trustworthiness, and ethical considerations. The current study reinforces the need for AI tools to evolve towards greater transparency, contextual sensitivity, and seamless integration into existing workflows, achieving a balanced partnership between automation and human expertise.

To enhance the practical utility of AI-driven code review tools, designers should consider integrating contextual signals from project history, commit patterns, or developer role. Participants expressed a desire for explainable feedback mechanisms, such as linking suggestions to real-world code examples or highlighting rationale behind recommendations. Enabling this kind of transparency may not only improve adoption but also foster trust, especially in high-stakes or regulated development environments.

Conclusion

This qualitative research explored the perceptions and challenges of software developers using AI-driven code review tools. The literature review highlighted existing knowledge regarding AI's ability to improve code review efficiency, productivity, and skills while underscoring prevalent concerns regarding trust, contextual understanding, and ethical implications (Almeida et al., 2024; Bird et al., 2023; Ernst & Bavota, 2022; Tufano et al., 2021).

Semi-structured interviews were conducted with software developers, technical leads and solution architects using thematic analysis. The thematic analysis revealed six key themes: efficiency and productivity, learning and knowledge enhancement, trust and reliability, challenges and limitations, collaboration and workflow integration, and future expectations. Participants acknowledged that AI-driven tools reduce repetitive tasks, increased productivity, and improved continuous learning. However, skepticism persisted regarding the tools' limitations in contextual understanding, ethical concerns, and reliability, underscoring the continued necessity of human oversight.

The discussion aligned these findings with previous studies, highlighting persistent challenges related to trust, contextual understanding, and ethical considerations when integrating AI tools in software development (Bird et al., 2023; Ernst & Bavota, 2022). Consistent with research advocating for human oversight in AI-driven software engineering practices (Baumgartner et al., 2024), this study underscores the necessity of balanced human-AI collaboration rather than complete automation.

In summary, this research provides evidence supporting a hybrid model, emphasizing the complementary role of AI-driven code review tools and human expertise. Future research should address AI limitations, especially contextual understanding and ethical transparency, aligning technological advances with ethical standards in software engineering.

References

Afzali, H., Torres-Arias, S., Curtmola, R., & Cappos, J. (2023). Towards verifiable web-based code review systems. *Journal of Computer Security*, *31*(2), 153–184. https://doi.org/10.3233/JCS-210098

Almeida, Y., Albuquerque, D., Filho, E. D., Muniz, F., de Farias Santos, K., Perkusich, M., Almeida, H., & Perkusich, A. (2024). AICodeReview: Advancing code quality with AI-enhanced reviews. *SoftwareX*, *26*, 101677. https://doi.org/10.1016/j.softx.2024.101677

Badampudi, D., Unterkalmsteiner, M., & Britto, R. (2023). Modern Code Reviews—Survey of Literature and Practice. *ACM Trans. Softw. Eng. Methodol.*, *32*(4), 107:1-107:61. https://doi.org/10.1145/3585004

Baumgartner, N., Iyenghar, P., Schoemaker, T., & Pulvermüller, E. (2024). AI-Driven Refactoring: A Pipeline for Identifying and Correcting Data Clumps in Git Repositories. *Electronics (2079-9292)*, *13*(9), 1644. https://doi.org/10.3390/electronics13091644

Bird, C., Ford, D., Zimmermann, T., Forsgren, N., Kalliamvakou, E., Lowdermilk, T., & Gazit, I. (2023). Taking Flight with Copilot: Early insights and opportunities of AI-powered pair-programming tools. *Queue*, *20*(6), Pages 10:35-Pages 10:57. https://doi.org/10.1145/3582083

Creswell, J. W. (2013). *Qualitative inquiry and research design: Choosing among five approaches* (3rd ed). SAGE Publications.

Doğan, E., & Tüzün, E. (2022). Towards a taxonomy of code review smells. *Information and Software Technology*, *142*, 106737. https://doi.org/10.1016/j.infsof.2021.106737

Dos Santos, E. W., & Nunes, I. (2018). Investigating the effectiveness of peer code review in distributed software development based on objective and subjective data. *Journal of Software Engineering Research and Development*, *6*(1), 14. https://doi.org/10.1186/s40411-018-0058-0

Ernst, N. A., & Bavota, G. (2022). AI-Driven Development Is Here: Should You Worry? *IEEE Software*, *39*(2), 106–110. https://doi.org/10.1109/MS.2021.3133805

Gerede, Ç. E., & Mazan, Z. (2018). Will it pass? Predicting the outcome of a source code review. *Turkish Journal of Electrical Engineering & Computer Sciences*, *26*(3), 1343–1353. https://doi.org/10.3906/elk-1707-173

Guest, G., Bunce, A., & Johnson, L. (2006). How Many Interviews Are Enough?: An Experiment with Data Saturation and Variability. *Field Methods*, *18*(1), 59–82. https://doi.org/10.1177/1525822X05279903

Keary, E. (2017). *OWASP Code Review Guide*. OWASP Code Review Guide. https://owasp.org/www-project-code-review-guide/

Khleel, N. A. A., & Nehéz, K. (2020). Tools, Processes and Factors Influencing of Code Review. *Multidiszciplinaris Tudomanyok*, *10*(3), 277–284. https://doi.org/10.35925/j.multi.2020.3.33

Kotsiantis, S., Verykios, V., & Tzagarakis, M. (2024). AI-Assisted Programming Tasks Using Code Embeddings and Transformers. *Electronics (2079-9292)*, *13*(4), 767. https://doi.org/10.3390/electronics13040767

Pejic, N., Radivojevic, Z., & Cvetanovic, M. (2023). Helping Pull Request Reviewer Recommendation Systems to Focus. *IEEE Access*, *11*, 71013–71025. https://doi.org/10.1109/ACCESS.2023.3292056

Rasheed, Z., Sami, M. A., Waseem, M., Kemell, K.-K., Wang, X., Nguyen, A., Systä, K., & Abrahamsson, P. (2024). AI-powered Code Review with LLMs: Early Results. *arXiv.Org*. https://doi.org/arXiv:2404.18496v1

Tufano, R., Pascarella, L., Tufano, M., Poshyvanyk, D., & Bavota, G. (2021). *Towards Automating Code Review Activities* (No. arXiv:2101.02518). arXiv. https://doi.org/10.48550/arXiv.2101.02518

Turzo, A. K. (2023). Towards Improving Code Review Effectiveness Through Task Automation. *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, 1–5. https://doi.org/10.1145/3551349.3559565

Turzo, A. K., Faysal, F., Poddar, O., Sarker, J., Iqbal, A., & Bosu, A. (2023). *Towards Automated Classification of Code Review Feedback to Support Analytics* (No. arXiv:2307.03852). arXiv. https://doi.org/10.1109/esem56168.2023.10304851

Vijayvergiya, M., Salawa, M., Budiselić, I., Zheng, D., Lamblin, P., Ivanković, M., Carin, J., Lewko, M., Andonov, J., Petrović, G., Tarlow, D., Maniatis, P., & Just, R. (2024). AI-Assisted Assessment of Coding Practices in Modern Code Review. *Proceedings of the 1st ACM International Conference on AI-Powered Software*, 85–93. https://doi.org/10.1145/3664646.3665664

Yin, Y., Zhao, Y., Sun, Y., & Chen, C. (2023). Automatic Code Review by Learning the Structure Information of Code Graph. *Sensors (14248220)*, *23*(5), 2551. https://doi.org/10.3390/s23052551

Zydroń, P. W., & Protasiewicz, J. (2023). Enhancing Code Review Efficiency – Automated Pull Request Evaluation using Natural Language Processing and Machine Learning. *Advances in Sciences and Technology*, *17*(4), 162–167. https://doi.org/10.12913/22998624/169576

Appendix A
Selected Participant Quotes by Theme

Efficiency and Productivity

"Compared to humans, the AI is very thorough when looking at the code base and the changes within a pull request. The review could be of high quality, depending on how much time you dedicate." - Participant 01.

"This automated code review process saves a lot of our time. The comments we are getting are good; sometimes there are hallucinations; otherwise, I think it is pretty neat and pretty good." - Participant 05.

"Whenever we make any change, we commit anything, and it goes to that review instantly within 5 minutes. It returns the feedback that we can change it at the same time, improving productivity for one developer and the whole team." - Participant 08.

"It helps us reduce the friction because another person may be unavailable for code reviews every time." - Participant 08.

"I take AI tools advantage every day. My day-to-day development activities have been improved." - Participant 07.

Learning and Knowledge Enhancement

"I've learned a lot from using the AI-driven code review tools. Having explained chunks of code is helping me learn at a faster pace than if I tried to go out and read articles. It's huge." - Participant 02.

"Even with the older code bases I wrote a couple of years ago, I'm back to them, making fixes and seeing everything I like. When the AI goes through it, it's like, this could have been done XYZ or a little bit better. It reinforces the fundamentals." - Participant 01.

"My biggest interest in AI and code review is learning more about the coding frameworks I'm working on, so any additional capabilities it could have to teach the developer would be valuable to me." - Participant 02.

"It gives us more quick feedback rather than relying on the other developers. So this helps us reduce the friction." - Participant 08.

Trust and Reliability

"I believe that when it comes to trust, as well as code quality, code structure, and overall logic, it can be fully trusted. However, it cannot be trusted in the context of the application regarding what that pull request entails." - Participant 03.

"It's around 85% to 90%. I cannot completely trust it because, in my experience, the code breaks when I try to implement the same right, and it does not work as expected." - Participant 06.

"They are reliable at finding small human errors... But for smaller things, like error handling and commit code logs, you have to check and see what it's doing for everything that's more about functional logic." - Participant 09.

"I haven't seen where it's blatantly incorrect, but I've seen irrelevant suggestions. I think that might also be my opinion, having more context of the code." - Participant 02.

"From what I've seen, I haven't noticed the tool suggesting anything that would introduce a vulnerability. However, sometimes it can nitpick about minor details." - Participant 10.

Challenges and Limitations

"These tools recommend deprecated features. Since the model was probably trained sometime in the past, it does not understand new features and doesn't yet fully understand the syntax." - Participant 04.

"The higher you go, the more context it loses. It's very good for methods, functions, and classes, but adding a higher microservice level would be better. It loses some of that context." - Participant 01.

"It does not understand the context. You will have to specify exactly whatever the output comes right. You'll have to modify it on top of it, and then you'll have to use it." - Participant 07.

"The challenges specifically tied to my current work involve finishing up a pull request for refactoring an entire microservice... Once you address some comments and push your changes, you might receive double the comments in return." - Participant 09.

"Sometimes, I see that it goes into hallucination. It keeps on telling you and giving you the same feedback." - Participant 08.

Collaboration and Workflow Integration

"One good thing we have noticed is that whenever there is an issue, it tags the team member, and we usually collaborate and talk about that issue." - Participant 05.

"If we could use AI to verify facts continually and not have to rely on the human brain to recall what was said or what the truth is, I think that would be valuable for collaboration." - Participant 02.

"The more we use it, the more ingrained it becomes in the workflow... Using ChatGPT, AI, RAG, and Copilot simultaneously has made the workflow significantly faster." - Participant 03.

"We can ask other developers what they think of this feedback, how we can implement these suggestions, and how to improve these suggestions." - Participant 06.

Future Expectations

"In the future, it will just write a code and then give it to us, and then it is just a matter of plugging in and testing it and then making sure it looks good." - Participant 05.

"Yes, it will evolve, and I expect that AI will eventually write all the code, and we'll simply be inputting prompting with business requirements." - Participant 02.

"I think it’s good, but I wouldn't rely solely on code reviews. I would use them to augment the team to identify potential gaps." - Participant 10.

"I do see it becoming more accessible; it's moving really fast. It's just the way it's evolving. It's actually going to progress a lot faster." - Participant 09.

"AI will be available in each tool. So AI will be invited to each of the developing tools they are working on." - Participant 07.

Appendix B

Figures

Figure 1. Word Cloud