Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Factors, benefits, and barriers influencing the adoption of AI tools and software developers' perceptions of AI tools' impact on software sustainability.

Md Sanaul Haque

sanaul.haque@lut.fi

Lappeenranta-Lahti University of Technology

Asif Hassan Russel

Lappeenranta-Lahti University of Technology

Jari Porras

Lappeenranta-Lahti University of Technology

Ari Happonen

Lappeenranta-Lahti University of Technology

Research Article

Keywords: AI-generated code quality, sustainable software engineering, ICT for sustainability, Code generation

Posted Date: November 18th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10360931/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Factors, benefits, and barriers influencing the adoption of AI tools and software developers' perceptions of AI tools' impact on software sustainability.

Abstract

The adoption of AI-based code assistants like ChatGPT and GitHub Copilot is increasingly popular among software developers, aiming to enhance coding efficiency and automate routine tasks. Nevertheless, there is limited empirical data on how these tools affect software quality attributes. For instance, while productivity may improve from a resource standpoint, code quality remains largely unchanged. This research explores how AI-supported development tools influence the sustainability of Scrum-based projects, i.e., software code quality and developer productivity. It employs a mixed-method approach, including a systematic review of 24 studies and 11 semi-structured interviews with experienced industry software developers.

The results show that AI tools can contribute to various aspects of software quality, such as higher code consistency, greater maintainability, and improved defect detection and testing. The characteristics of automated code generation and code suggestions, as well as the generation of test cases through AI, were found to reduce development time and cognitive load and to encourage higher-quality outputs. Nevertheless, issues of reliability, security, and accuracy of code produced by AI, especially in complex development cases, still need to be addressed, and human supervision and validation must be ongoing, e.g., AI generates code which is hard to maintain, or AI produces code so fast and so much in quantity that humans do not have enough motivation and interest.

The research also mentions that AI-assisted development assists sustainable software engineering by maximising resource usage, minimising reuse, and enhancing long-term code quality. This study offers a contribution to the further study of the role of AI in current software engineering and useful insights on the implementation of AI tools in quality-oriented software development by empirically correlating AI tool adoption with software quality results. Future research should focus on quantitatively assessing the effects of AI tools on software quality, using standardised measures such as defect rates, code quality scores, and test coverage. It would also be informative to conduct comparative research on various AI tools in different development settings.

Keywords: AI-generated code quality, sustainable software engineering, ICT for sustainability, Code generation.

1. Introduction

Artificial intelligence (AI) has also become an important disruptive technology in software engineering in recent years, with the introduction of AI-assisted software development systems like ChatGPT, GitHub Copilot, and Tabnine. These tools suggest code in real time, automate common programming tasks, and assist developers with debugging, refactoring, and test generation (Waseem et al., 2024; Davila et al., 2024). Consequently, they have been progressively incorporated into contemporary software development

LUT Group Confidential - Other information (3Y)

processes, particularly in Agile systems like Scrum (Mdallal et al., 2023). Although previous studies have highlighted the productivity benefits of AI-assisted development, there is still little empirical knowledge on the effects of these tools on the quality attributes of software, such as reliability, maintainability, and efficiency in detecting defects (Nguyen and Nadi, 2022; Mehmood et al., 2023).

The quality of software is a key issue in software engineering, where projects are supposed to produce reliable, maintainable and high-performance systems to be delivered in limited time and budget. Conventional development systems tend to be labor-intensive such as manual coding, debugging, and testing that may cause inconsistencies and defects. These challenges can be alleviated with the assistance of AI-based coding assistants, which can automate repetitive tasks, increase the level of code consistency, and improve testing practices (Rajbhoj et al., 2024; Mendes et al., 2024). Nonetheless, even as their usage increases, the evidence is lacking on whether AI tools are always effective on improving the quality of software or creating new risks, including untrustworthy or unsafe code (Khoury et al., 2023).

The current literature on AI-assisted software development is mostly concerned with productivity gains, automation, and developer experience (Waseem et al., 2024; Peras et al., 2023). Although these contributions are useful, they tend to ignore the implications that these contributions have on software quality assurance. As an example, a systematic study of the impact of AI-generated code on maintainability, testing effectiveness, and long-term system reliability has not been conducted (Nguyen and Nadi, 2022). Furthermore, the issues of the correctness, security, and interpretability of AI-generating results are still under-researched, especially in the context of complex development (Klemmer et al., 2024; Khoury et al., 2023). This is particularly imperative in Agile systems like Scrum, which require speed and quality assurance due to the fastness of iterations (Faniran et al., 2017).

Along with quality considerations, more recent discussions have also associated AI-assisted development with software sustainability, shorter development time, maximized resource usage, and increased code efficiency (Freitag et al., 2021; Sikand et al., 2024). Although sustainability is an emerging issue, its connection to the fundamental software quality characteristics needs to be elaborated. In particular, there is no explanation of how the efficiency of developments can translate into quality effects in the long term, including sustainable and stable software systems.

In order to address these gaps, the research paper explores the applications of AI-based development aids in software quality and productivity of developers in Scrum-based environments. The mixed-method approach is taken, which is a systematic literature review (SLR) with semi-structured interviews with professional software developers. The SLR summarizes the available empirical data on the adoption of AI tools, outlines the most important advantages, difficulties, and limitations (Carrera-Rivera et al., 2022; Jalali and Wohlin, 2012). The interview study supports these findings by obtaining real-life developer perspectives and experience on the effects of AI tools on code quality, testing, and development processes (Hove and Anda, 2005).

This research proposal will seek to address the following research questions:

RQ1: What are the key factors, benefits, and barriers influencing the adoption of AI-assisted development tools in software engineering?

LUT Group Confidential - Other information (3Y)

RQ2: How do software developers perceive the impact of AI tools on software sustainability, i.e., software quality and productivity?

This paper has three-fold contributions. First, it presents an extensive synthesis of the literature on AI-assisted software development, with particular emphasis on the implications for software quality. Secondly, it provides practical experience of industry professionals on the advantages and disadvantages of AI tools in the context of practical development. Third, it creates a conceptual connection between AI-assisted development, the quality attributes of software, and sustainable software engineering practices.

The rest of this paper follows the following structure. Section 2 conducts a literature review on AI tools, Scrum practices, and software quality considerations. Section 3 outlines the methodology of the research. Section 4 provides the results and then Section 5 gives a discussion of findings. Lastly, the Section 6 wraps up the research and presents the future research directions.

.2. Literature Review

2.1 AI Applications in Software Development

The conventional software development cycles are usually manual, labor intensive and time consuming. Artificial intelligence (AI) applications like ChatGPT, GitHub Copilot, Tabnine, and CodeWhisperer have been gaining traction as tools to assist software development processes in recent years (Waseem et al., 2024; Mendes et al., 2024). These systems utilize large language models that have been trained on massive code repositories to offer an autosuggestion of code, automate debugging and help in testing and documentation of software (Mohamed et al., 2024).

AI-driven solutions have been proven to enhance the efficiency of the development process, automating repetitive workflows and saving human resources and speeding up the coding process (Rajbhoj et al., 2024). Moreover, they help to increase the quality of the code as they can detect possible mistakes and propose more optimized solutions during the development (Mehmood et al., 2023). These features come in especially handy in Agile setups, where fast iterations and constant integration demand efficient and dependable development routines (Peras et al., 2023).

The Scrum model is one of the components of Agile approaches that focuses on the iterative development of a product or service, teamwork, and constant improvement (Faniran et al., 2017). Scrum is a development model that breaks down the development into brief cycles called sprints, which allow teams to deliver incremental improvements, and retain flexibility. By automating repetitive processes and aiding the decision-making process, the introduction of AI tools to Scrum settings can potentially increase the productivity of the team and simplify the development processes (Mdallal et al., 2023). Fig. 1 demonstrate the Scrum workflow, comprising of sprint planning, execution and review cycles.

LUT Group Confidential - Other information (3Y)

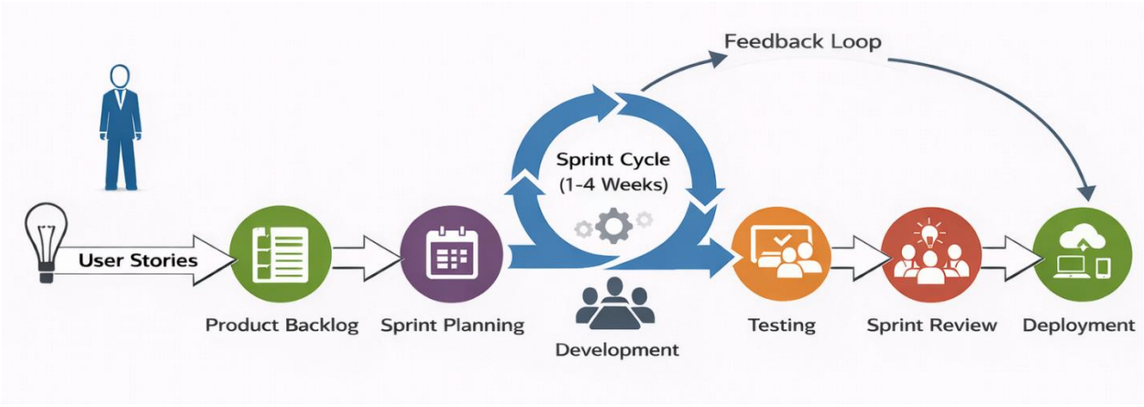


Fig. 1 Scrum process flow illustrating iterative development cycles in Agile environments.

2.2 Impact of AI on Software Quality and Sustainability

Artificial intelligence is starting to be considered as a disruptive tool in software engineering, with a range of impacts on software development lifecycle phases (Duan et al., 2019). The use of AI tools can automate the process of coding, testing, and debugging, thus saving time and enhancing efficiency in the development process (Waseem et al., 2024). These enhancements can have a direct impact on software quality such as maintainability, reliability and performance.

Software sustainability has become a significant factor in contemporary software engineering, with the emphasis on the creation of efficient, maintainable, and environmentally responsible systems (Freitag et al., 2021). Sustainable software development seeks to optimize use of resources, decrease technical debt and provide the capacity to use the system long term. AI tools can help achieve sustainability through streamlining code, minimizing redundancy, and enhancing the efficiency of the overall development (Sikand et al., 2024).

In terms of quality, AI-assisted tools aid in detecting defects and testing efficiency by creating test cases and finding possible bugs early in the development cycle (Sobania et al., 2023). Moreover, AI-based refactoring and code optimization further increase maintainability and minimize the amount of work required, which also leads to quality and sustainability objectives.

However, the association between the use of AI and the quality of software is complicated. Although AI tools are more efficient, there is a question about the validity and trustworthiness of AI-generated code (Khoury et al., 2023). These issues demonstrate the importance of cautious implementation of AI tools in software quality assurance practices.

2.3 AI Tools in Scrum and Software Quality Management

To enhance the Scrum process, such as sprint planning, backlog management, and progress tracking, AI tools are increasingly being incorporated into Agile project management platforms (Kanbur and Kulkarni, 2023). Jira, Smartsheet, and Nifty.ai are tools that deliver AI-driven information that helps teams prioritize work, estimate workloads, and pinpoint possible bottlenecks.

In Scrum settings, AI-based tools are used in managing the quality of software; they are automated to carry out the usual developmental tasks and assist in continuous testing and integration. To illustrate, AI-based tools may produce test cases, spot bugs, and suggest code enhancements, thus improving the

LUT Group Confidential - Other information (3Y)

reliability of software and minimizing defect rates (Mehmood et al., 2023). These features allow development teams to achieve high-quality standards and, at the same time, meet stringent deadlines.

Moreover, AI technologies assist less skilled developers with real-time assistance and code recommendations, which encourages the best practice adherence and enhances consistency of the code (Nguyen and Nadi, 2022). This can be extremely useful in teamwork settings where code quality is vital to long-term maintainability.

Despite these advantages, there is only limited empirical evidence on the efficacy of AI tools in enhancing the quality of software in the context of Scrum. The current literature is mainly concerned with productivity gains instead of quality outcomes that are systematically assessed. This gap underscores the importance of conducting more studies on the role of AI tools in enhancing the quality attributes of software in a real-world development environment. In Scrum activities, AI tools play the following roles as summarized in Table 1.

Table 1. Role of AI tools across Scrum process activities

Scrum Activity	AI Support Function	Example Tools
Sprint Planning	Task prioritization, effort estimation	Smartsheet, Clarizen
Daily Stand-up	Progress tracking, blocker identification	Monday.com, Aconex
Backlog Management	Task organization, prioritization	Jira, Huginn
Work Breakdown	Task decomposition, dependency mapping	Nifty.ai, Wrike
Development	Code generation, auto-completion, refactoring	GitHub Copilot, ChatGPT
Testing	Test case generation, bug detection	Copilot, SonarQube
Product Management	Decision support, roadmap insights	Jira, Nifty.ai

2.4 Research Gap and Problem Statement

The growing use of AI tools in software development has brought tremendous productivity and efficiency in workflow. Nevertheless, scarce empirical studies have been conducted on their effect on software quality aspects, including reliability, maintainability, and effectiveness in detecting defects (Nguyen and Nadi, 2022; Khoury et al., 2023).

The advantages of AI tools in code automation and enhanced developer productivity have been the main topic of previous research (Waseem et al., 2024; Peras et al., 2023). Although such contributions are valuable, they still cannot entirely tackle the challenges related to AI-generated code, such as correctness, security, and consistency (Klemmer et al., 2024).

Furthermore, the topic of AI tools use in the Scrum-based development environment is under-researched, especially regarding its effects on software quality and sustainability. Even though AI tools can be used to improve testing, refactoring, and code optimization, the literature on their usefulness to improve software quality results in the long term is limited.

LUT Group Confidential - Other information (3Y)

Thus, this research will help address these gaps by exploring how AI-based development tools are adopted and how they affect sustainability, e.g., software quality and productivity. This study is a combination of a systematic literature review and empirical knowledge of the industry practitioners on the opportunities and challenges of using AI in software development, and this offers a complete view of the opportunities and challenges with respect to AI-assisted software development.

3. Methodology

3.1 Research Design

The research design used in this study is a **mixed-method research design**, which aims to examine the effects of AI-assisted development tools on the quality of software, developer productivity, and sustainability in Scrum-based settings. Mixed-method research is the popular trend of empirical software engineering studies in order to provide a complete literature review and practical practitioner perspectives (Johnson et al., 2007). The study will involve the use of a Systematic Literature Review (SLR) and semi-structured interviews, which will ensure that the methods are triangulated and results validated even more.

The research is a two-stage study. The first step will entail an SLR to identify the main points, benefits, and problems concerning AI tools use in the software development process. The second phase will be semi structured interviews with industry practitioners in order to establish real life experience and validate literature findings. Such a combined approach would enable the creation of a better understanding of AI-aided software development theoretically and practically. Fig. 2 demonstrates the overall research design, the SLR and interview steps.

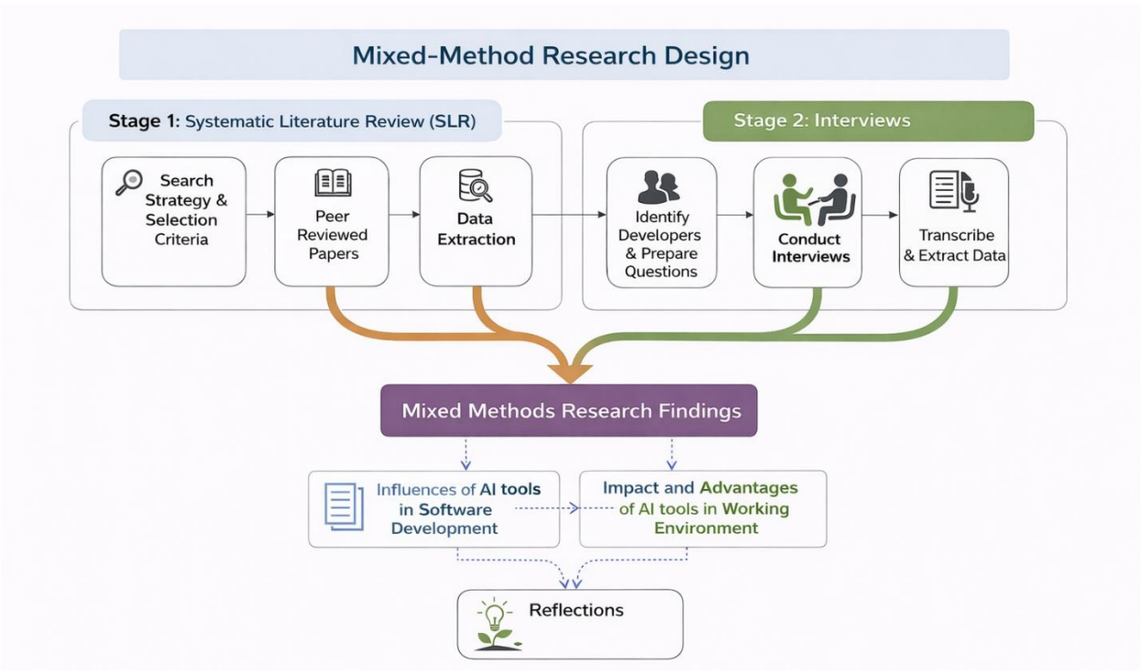


Fig. 2 Research design illustrating the mixed-method approach combining SLR and interviews

3.2 Systematic Literature Review

The SLR adheres to the best practices in evidence-based research to provide transparency, rigor, and reproducibility (Moher et al., 2009; Carrera-Rivera et al., 2022). The reviewing process will entail the

LUT Group Confidential - Other information (3Y)

definition of the search strategies, the implementation of selection strategy, the extraction of relevant data, and synthesizing of findings.

3.2.1 Search Strategy

The major academic databases were searched systematically, such as Web of Science, Scopus, IEEE Xplore, ACM Digital Library, ScienceDirect, and ProQuest. Based on combinations of keywords and Boolean operators related to AI tools, software development, and software quality, search strings were developed. The search strategy is a combination of keywords and Boolean operators to build search queries to find the relevant papers, and the queries are

("AI Tools OR Artificial Intelligence OR AI) AND ("Software OR Software Development") AND ("Benefits OR Advantages")

("AI Tools OR Artificial Intelligence OR AI) AND ("Software OR Software Development") AND ("Software Developer OR Software Engineer")

The keywords were formulated based on the research questions, and they will comprise such terms as AI tools, software development, software quality, and productivity. This will ensure the coverage of the relevant literature (Jalali & Wohlin, 2012).

3.2.2 Study Selection Criteria

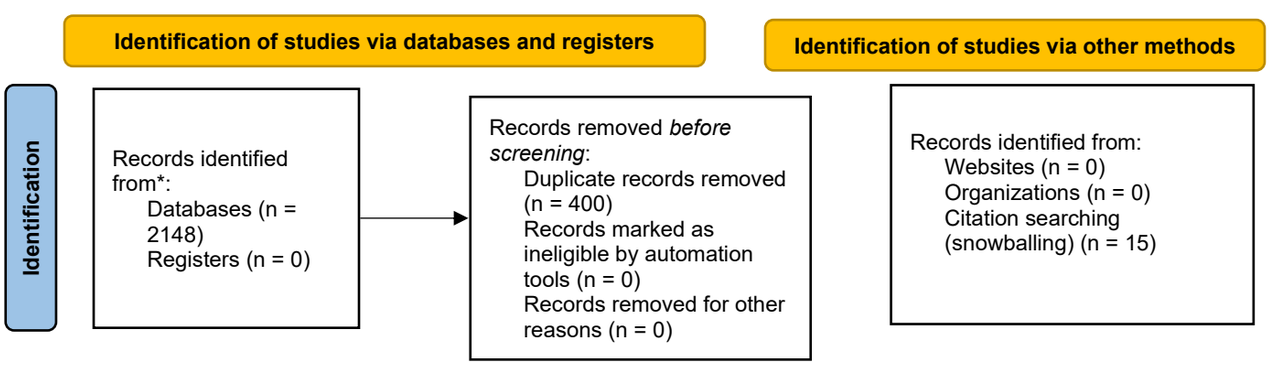
Inclusion and exclusion criteria were applied to ensure the relevance and quality of selected studies. Studies were included if they:

1. Focused on AI tools in software development
2. Addressed software quality, productivity, or developer-related outcomes
3. Were published in peer-reviewed venues

Studies were excluded if they:

1. Focused on non-software domains
2. Lacked methodological clarity
3. were not written in English

The selection process involved identifying 2,148 initial studies, followed by screening based on titles, abstracts, and full texts. Following the removal of duplicates and application of the criteria, 24 studies were retained. Backward snowballing was used to identify additional relevant studies as it is an established method of covering the literature (Jalali & Wohlin, 2012). Fig. 3 shows the process of selecting the studies.



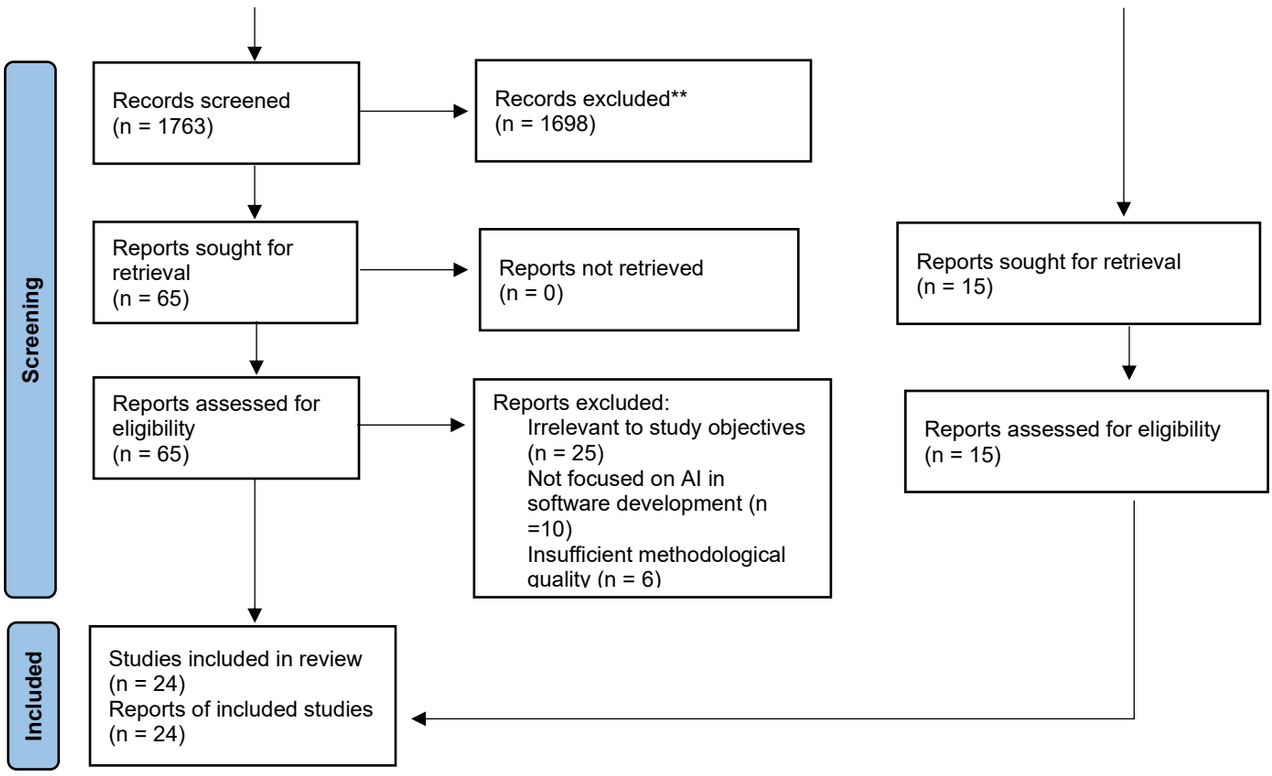


Fig. 3 PRISMA flow diagram of identifying, screening, eligibility and inclusion of studies in the systematic literature review.

3.2.3 Data Extraction and Analysis

Structured analysis was carried out to extract relevant data in a structured spreadsheet, such as the characteristics of studies, AI tools applied, reported benefits, limitations, and quality-related results. The thematic analysis was used to analyze the extracted data and, in this method, the patterns are identified, coded, and grouped using qualitative data (Vaismoradi et al., 2016).

The analysis process included:

1. Open coding to identify initial concepts
2. Axial coding to group related categories
3. Theme development to capture key insights

This approach enabled the identification of recurring themes related to software quality, productivity, and AI tool adoption.

3.3 Interview Study

3.3.1 Sampling and Participants

Semi-structured interviews with software developers currently employed in industry and using the AI tools in Scrum-based projects were carried out to supplement the results of the SLR. Participants were sampled by a purposive approach to include individuals with pertinent experience in software development and those who were familiar with AI-assisted tools.

LUT Group Confidential - Other information (3Y)

The respondents were located using professional sites like LinkedIn. Among 27 people who were initially identified, 16 individuals were invited to take part, since they met the selection criteria. Eleven participants accepted to participate in the study. These respondents are a mix of different positions such as software engineers, senior developers and technical leads, and different experience in the industry.

3.3.2 Data Collection

The interviews were to be held in semi-structured format and, thus, it was both consistent and flexible in data collection (Hove and Anda, 2005). The interviews took between 30-50 minutes and were done through online communication tools.

The interview protocol included questions focusing on:

- Usage of AI tools in development workflows
- Perceived impact on software quality (e.g., reliability, testing)
- Productivity improvements and workflow efficiency
- Challenges and limitations of AI-generated code

All interviews were conducted with participants' informed consent and were audio-recorded for accuracy.

3.3.3 Data Analysis

Thematic analysis was employed to analyze interview data, which was conducted in a systematic manner through coding and development of themes (Vaismoradi et al., 2016). The transcripts were analyzed to extract the key trends and findings addressing the quality of the software, experience of the developers, and sustainability.

The coding was conducted in an iterative manner, and themes were developed out of the data. This strategy helped to make sure the analysis reflected both anticipated and unforeseen results.

3.4 Validity and Reliability Considerations

To ensure the rigor and credibility of the study, several validity measures were implemented:

- **Construct Validity:** Multiple data sources (SLR and interviews) were used to capture different dimensions of AI tool adoption, improving the comprehensiveness of findings.
- **Internal Validity:** Systematic procedures, including structured coding and thematic analysis, were used to minimize researcher bias (Vaismoradi et al., 2016).
- **External Validity:** Although the sample size is limited, participants were selected from diverse professional backgrounds, improving the generalizability of insights.
- **Reliability:** The study follows a transparent and replicable methodology, including clearly defined search strategies and analysis procedures (Moher et al., 2009).

3.5 Ethical Considerations

The study was conducted in compliance with ethical considerations. The participants were aware of the objective of the research and gave consent before participating. To guarantee confidentiality and privacy,

LUT Group Confidential - Other information (3Y)

data were anonymized. Participation was voluntary, and participants were allowed to withdraw at any stage of the study.

4. Results

4.1 Findings from Systematic Literature Review (RQ1)

The systematic literature review revealed four key themes associated with the use of AI-assisted development tools and their implications on software quality: (1) quality improvement and efficiency, (2) support of routine development processes, (3) inability to cope with complex tasks, and (4) reliability and security. Fig. 4 shows the key benefits and barriers that were identified during the SLR.

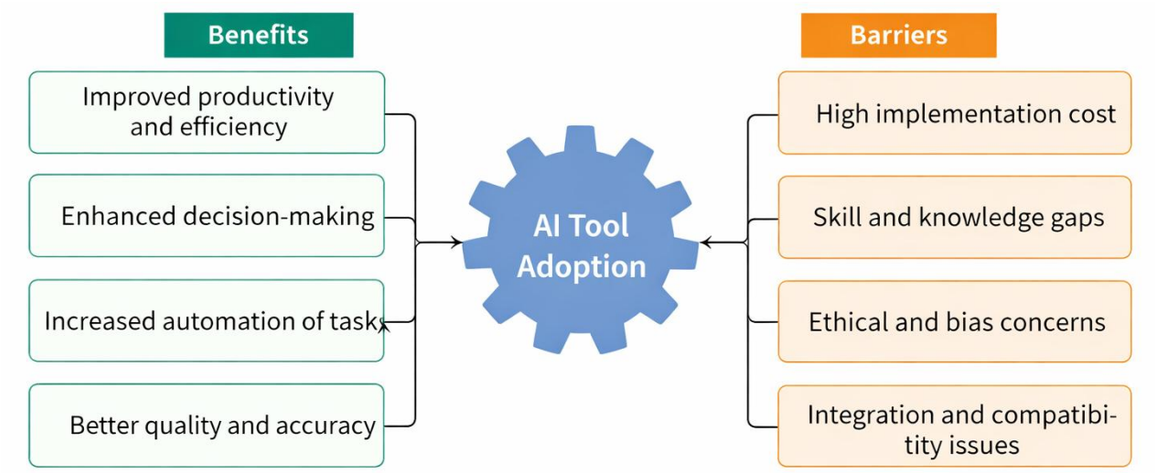


Fig. 4 Key themes identified from SLR: benefits and barriers of AI tool adoption.

Theme 1: Quality Enhancement and Efficiency Gains

One of the most prominent conclusions made throughout the reviewed literature is that AI-based tools can be used to achieve several quality attributes of software, especially its consistency in its code, sustainability, and efficiency in its defect detection. GitHub Copilot and ChatGPT help developers by writing syntactically correct code, providing fix suggestions, and aiding in refactoring code (Waseem et al., 2024; Rajbhoj et al., 2024).

Code generation and review processes generated by AI can improve the quality of code by minimizing human mistakes and increasing structural consistency. Also, this is supported by automated documentation and refactoring support, which enhance maintainability and allow developers to create cleaner and reusable code (Mendes et al., 2024). According to several studies, AI tools enable automated test creation and debugging, enhance test coverage, and fault detection rates (Mehmood et al., 2023; Sobania et al., 2023).

Additionally, AI tools can alleviate the cognitive burden on developers by minimizing the use of external searches and repetitive problem-solving, and AI indirectly leads to high-quality outputs (Peras et al., 2023)

Theme 2: AI as Support for Routine Development Activities

AI tools can be used as effective augmentative systems that help in daily coding, such as code completion, syntax correction, and boilerplate generation. The support is especially useful in case of Agile settings

LUT Group Confidential - Other information (3Y)

like Scrum, where quick iterations need effective management of repetitive processes (Faniran et al., 2017).

In the context of software quality, automation of routine tasks aids in keeping the coding standards and consistency as well as decreasing the variability that the practice of manual coding adds. Moreover, AI tools assist more novice developers, helping them to follow best practices and the overall reliability of the code (Nguyen and Nadi, 2022).

Theme 3: Limitations in Handling Complex Development Tasks

Despite their advantages, AI tools have shortcomings in their ability to process complex or situation-specific programming tasks. Some articles emphasize that AI-generated results might not be contextually aware and result in incomplete or sub-optimal solutions (Waseem et al., 2024).

These restrictions have implications on the quality of software especially their correctness and reliability. Multi-condition logic and domain-specific applications typically involve manual validation and refinement of AI-generated code (Khoury et al., 2023). Unless adequately monitored, the use of AI tools can bring in inconsistencies or latent defects.

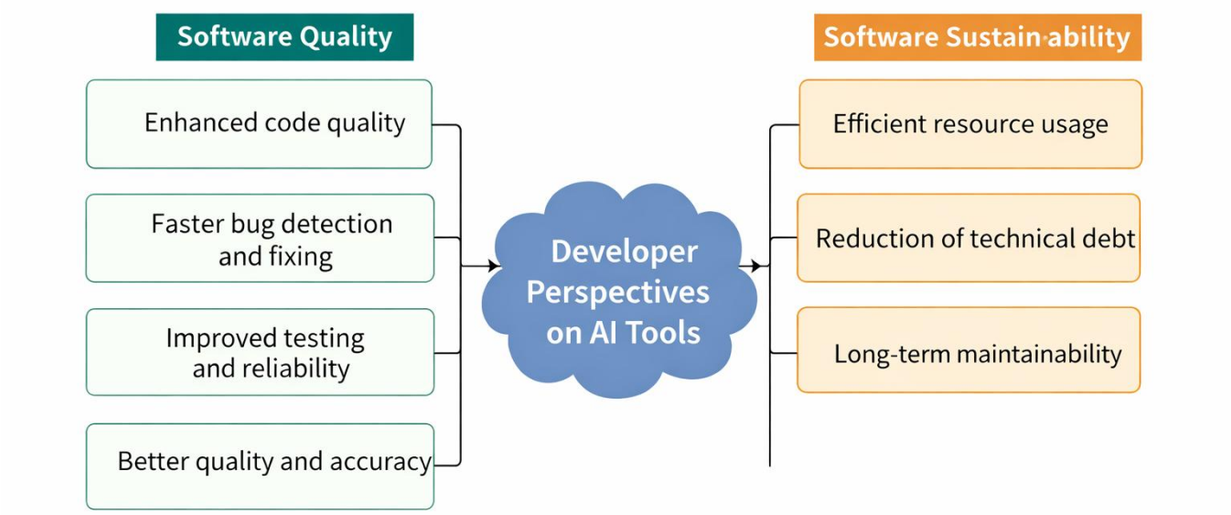
Theme 4: Reliability and Security Concerns

A critical concern identified in the literature is the **reliability and security of AI-generated code**. Code generated by AI can be syntactically correct and semantically invalid or have security vulnerabilities (Khoury et al., 2023).

The problems of non-uniformity of outputs, the inability to understand the context, and the possible creation of insecure code patterns directly affect the attributes of software reliability and security (Klemmer et al., 2024). Such results highlight the importance of human-in-the-loop validation and adoption of AI tools to the current software quality assurance procedures.

4.2 Findings from Interview Study (RQ2)

The major advantages of AI tools based on developer perspectives are illustrated in Fig. 5.



LUT Group Confidential - Other information (3Y)

Fig. 5 Developer perspectives on AI tools and their contribution to software quality and sustainability.

The analysis of the interview showed that there were four major themes that describe the perceptions that developers have towards AI tools and how they influence the quality of software and the development practices.

Theme 1: Improvement in Code Quality and Development Efficiency

Participants consistently report that AI tools are improving the quality of code; they suggested correct recommendations, refactored code, and assisted in automated reviews. Developers observed that AI-assisted code is able to reduce redundant code, enhance readability, and foster consistent coding.

Moreover, AI tools have been discovered to enhance testing activities by coming up with test cases and detecting possible defects at an early stage of development. This helps to enhance efficiency in defect detection and the overall software reliability, which is in line with the results of previous studies (Mehmood et al., 2023; Sobania et al., 2023).

Theme 2: Automation of Repetitive Tasks and Workflow Optimization

The developers emphasized that AI tools automate repetitive processes such as boilerplate code, documentation, and even basic debugging. This automation reduces the number of individuals that need to work manually and allows developers to focus on more valuable and critical work.

Manually, the fewer repetitions, the fewer human-related errors and the more consistent and standardized code, which is more sustainable and resistant to changes in the long term (Mendes et al., 2024).

Theme 3: Contribution to Sustainable and Maintainable Software

The participants stressed that AI tools can help to develop maintainable and efficient code, which can facilitate the sustainability of the software in the long term. The use of AI tools can reduce resource usage and cost of development by promoting code reuse, optimization and efficient design patterns (Sikand et al., 2024).

The refactoring with the support of AI improves the organization and maintenance of the code, such that software systems are dynamic and can be extended over time.

Theme 4: Concerns Regarding Reliability and Over-reliance

Despite the benefits, the participants expressed their doubts regarding the reliability of AI-generated outcomes, particularly in complex scenarios. The AI tools may provide false or sub-optimal solutions, which should be critically evaluated.

Also, excessive dependence on AI tools can influence critical thinking and problem-solving skills of developers, which can affect the quality of their code in the long-term. These issues are consistent with previous studies that emphasize the importance of human supervision in AI-assisted development (Klemmer et al., 2024). The most important themes that were identified during the analysis of the interview, as well as their resolutions to software quality, are presented in Table 2.

LUT Group Confidential - Other information (3Y)

Table 2. Summary of interview insights on AI tool usage

Theme	Software Quality Implication	Key Insight from Interviews	Representative Observation
Code Quality Improvement	Maintainability, Consistency	AI tools improve code structure, readability, and reuse	"AI reduces redundant code and improves structure"
Testing Efficiency	Reliability, Defect Detection	Automated test generation enhances early bug detection	"AI helps generate test cases and detect errors faster"
Workflow Optimization	Consistency, Efficiency	Automation reduces repetitive tasks and human error	"Less manual work improves overall development quality"
Maintainability & Sustainability	Maintainability, Scalability	AI supports refactoring and long-term code usability	"AI helps write cleaner, reusable, and maintainable code"
Reliability Concerns	Reliability, Correctness	AI outputs require validation, especially for complex tasks	"AI-generated code is helpful but needs careful review"

4.3 Summary of Findings

Altogether, the results point to the fact that AI-based development tools positively influence the quality of software, in particular regarding:

- Improved code consistency and maintainability
- Enhanced defect detection and testing efficiency
- Reduced human errors through automation

However, these benefits are accompanied by challenges related to:

- Reliability and correctness of AI-generated code
- Security vulnerabilities
- Limitations in complex problem-solving

These results highlight the idea that, despite the ability to enhance the quality of software using AI tools, their application should be properly implemented with human expertise and the current quality assurance tools, which is consistent with the existing empirical data (Nguyen & Nadi, 2022; Khoury et al., 2023). The correlation of AI capabilities and software quality attributes is presented in Table 3.

Table 3. Mapping AI-assisted features to software quality attributes

AI Capability	Software Quality Attribute	Impact on Quality
Code generation	Maintainability	Improves code structure and reuse

LUT Group Confidential - Other information (3Y)

Auto code suggestions	Consistency	Reduces variability in coding style
Code refactoring	Maintainability	Enhances readability and scalability
Automated testing	Reliability	Improves fault detection
Bug detection	Defect detection	Reduces error rates
Documentation support	Maintainability	Facilitates easier updates
Debugging assistance	Reliability	Accelerates issue resolution

5. Discussion

5.1 Interpretation of Key Findings

This study presents empirical results demonstrating that AI-assisted development tools significantly influence the software quality aspects, namely maintainability, code consistency, and defect detection efficiency. The results are premised on the synthesis of the literature review and the interview data which suggest that AI tools can be used as quality enablers provided, they are implemented in the development workflow appropriately.

The enhancement of the maintainability and consistency of codes is one of the most remarkable discoveries. AI applications can help developers create structured and standardized code, minimizing variability and enhancing readability. This is consistent with the previous studies that show that AI-based tools assist in code refactoring and optimization, which results in a better maintainability (Mendes et al., 2024; Waseem et al., 2024). Moreover, better documentation and re-use patterns of codes lead to long-term sustainability of software.

The research also identifies the effectiveness of AI tools in improving testing processes and faults. The debugging and test generation are automated, enabling software developers to detect faults at earlier stages of the software development lifecycle, enhancing reliability. The results align with previous research that has revealed that AI-supported tools can enhance testing efficiency and fault detection rates (Mehmood et al., 2023; Sobania et al., 2023).

Nonetheless, the findings also suggest that AI tools have new quality risks (especially, concerning correctness, reliability, and security). AI-generated code can be semantically false or contextually unsuitable, as well as syntactically correct (Khoury et al., 2023). This is particularly important in more complex or domain-specific applications where erroneous output can cause latent defects or vulnerabilities. These results underpin the role of human supervision as an indispensable element of AI-assisted development (Klemmer et al., 2024). The overall influence on the quality of software and productivity due to the use of AI tools is summarized in Fig. 6.

LUT Group Confidential - Other information (3Y)

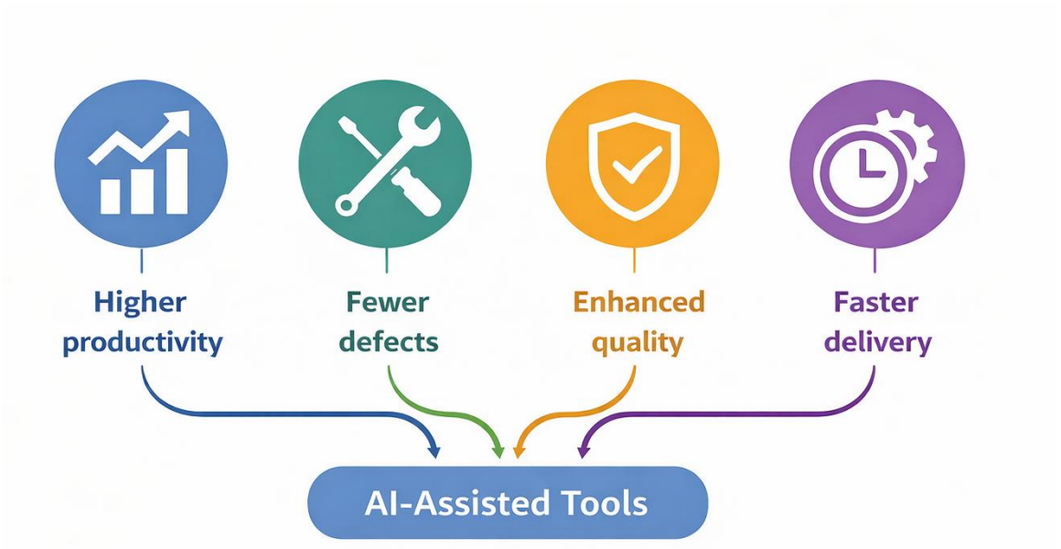


Fig. 6 Impact of AI-assisted tools on software quality and productivity.

5.2 Comparison with Existing Literature

This research aligns with other previous investigations that report the productivity advantages of AI-assisted development tools (Waseem et al., 2024; Rajbhoj et al., 2024). Nevertheless, this research goes beyond previous research and clearly connects the use of AI tools to the quality of the output of the software, not just the increase in efficiency.

Past research has demonstrated that programming assistants like GitHub Copilot can not only increase the speed of the development process but also help to write tests (Nguyen and Nadi, 2022; Mehmood et al., 2023). This research validates such findings but adds additional evidence to show their implications in quality attributes like maintainability, reliability and defect detection.

Also, the issues of inconsistent outputs and security risks found in previous research are justified by the practical conclusions of the current research (Khoury et al., 2023; Klemmer et al., 2024). As opposed to most previous studies that may only be based on experimental or survey-based designs, this study integrates literature reviews with practical practice of practitioners, giving a more holistic view of AI tools in practice-based software development settings.

5.3 Implications for Software Quality Engineering

The findings of this study have several important implications for both researchers and practitioners:

1. **Integration with Quality Assurance Processes:** AI tools should be integrated within established software quality assurance frameworks to ensure reliability and correctness.
2. **Human-in-the-Loop Validation:** Continuous human oversight is essential to validate AI-generated outputs and mitigate potential risks (Klemmer et al., 2024; Glas et al., 2026).
3. **Focus on Maintainability and Testing:** Organizations should leverage AI tools to improve code maintainability, automated testing, and defect detection (Mehmood et al., 2023).
4. **Developer Training and Awareness:** Developers should be trained to critically evaluate AI-generated code to avoid over-reliance and ensure quality standards.

LUT Group Confidential - Other information (3Y)

These implications suggest that AI tools can be optimally utilized as augmentative systems to complement, but not to substitute, human experience in software engineering.

5.4 Limitations of the Study

This study has several limitations that should be acknowledged. First, the interview sample is very small (11 participants), and the findings can only be extrapolated to a certain degree. However, the selection of the participants was made on a broad scale of professional backgrounds and in a way overcomes this weakness.

Second, the study focuses primarily on the Scrum-based development environments, and the findings may not be wholly applicable to other methods such as DevOps or Waterfall (Faniran et al., 2017). Third, conclusions are drawn more on qualitative experiences and a future research can combine quantitative values such as density of defects and maintainability indexes to provide more empirical data.

5.5 Future Research Directions

Further research is required on the security consequences of AI-generated code and the creation of frameworks of safe and reliable AI-assisted programming (Khoury et al., 2023). The discussion of the usage of AI in other development practices, including the DevOps and continuous integration pipelines, is also a viable prospective study. The next wave of research must involve quantitative determination of the effects of AI tools on the quality of software, using standardized measures like defect rates, score on code quality, and test coverage. It would also be informative to conduct comparative research on various AI tools in different development settings.

6. Conclusion

This paper has examined the impact of AI-based development tools on software quality, developer productivity, and software development sustainability in Scrum-based software development projects. The systematic literature review, combined with the results of the empirical research based on the experience of the industry practitioners, makes the research complete and enables having a clear understanding of the effect of AI tools on the existing software engineering practice.

The findings confirm that AI tools are very useful to enhance the quality of software features, particularly the consistency of the code and maintainability and the effectiveness of detecting defects. The enhanced development processes and reduced human efforts are linked with automated code generation, refactoring, and testing, which reflects prior research findings (Smolic et al., 2026; Waseem et al., 2024; Mehmood et al., 2023).

Simultaneously, the paper also points out severe issues concerning the dependability, accuracy, and safety of AI-written code (Khoury et al., 2023). These outcomes emphasize the point that AI tools should not and cannot replace developers, as they are collaborative systems that should augment human capabilities.

Comprehensively, this study adds to the existing literature on AI in software engineering by explicitly connecting AI tool usage to the quality of the resulting software. It also offers some practical advice to organizations aiming to incorporate AI tools in their development cycle without compromising the quality of the software.

LUT Group Confidential - Other information (3Y)

Funding acquisition declaration

This study has received no funding for conducting research.

Data availability declaration

Data can be gathered upon requesting the authors.

Competing interest declaration

This study has no competing interests.

References

- Ayoola, B. (2023). Evaluation of stakeholder mapping and personas for sustainable software development. In IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 270–272). Doi: 10.1109/ICSE-Companion58688.2023.00073

- Bambazek, P., Groher, I., & Seyff, N. (2023). Application of the sustainability awareness framework in agile software development. In IEEE 31st International Requirements Engineering Conference (RE) (pp. 264–274). *Doi: 10.1109/RE57278.2023.00034*

- Bhandari, K., Kumar, K., & Sangal, A. (2023). Artificial intelligence in software engineering: Perspectives and challenges. In Third International Conference on Secure Cyber Computing and Communication (ICSCCC) (pp. 133–137). Doi: 10.1109/ICSCCC58608.2023.10176436.

- Carrera-Rivera, A., Ochoa, W., Larrinaga, F., & Lasa, G. (2022). How to conduct a systematic literature review: A quick guide for computer science research. MethodsX, 9. Doi: 10.1016/j.mex.2022.101895

- Cohen, L., Manion, L., & Morrison, K. (2017). Research methods in education (8th ed.). Routledge. ISBN: 9781138209886

- Davila, N., Wiese, I., Steinmacher, I., Silva, L. L. D., Kawamoto, A., Favaro, G. J. P., & Nunes, I. (2024). An industry case study on adoption of AI-based programming assistants. In IEEE/ACM

LUT Group Confidential - Other information (3Y)

46th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 92–102). Doi: 10.1145/3639477.3643648

- Duan, Y., Edwards, J. S., & Dwivedi, Y. K. (2019). Artificial intelligence for decision making in the era of big data: Evolution, challenges and research agenda. *International Journal of Information Management, 48*, 63–71. Doi: 10.1016/j.ijinfomgt.2019.01.021

- Faniran, V. T., Badru, A., & Ajayi, N. (2017). Adopting Scrum as an agile approach in distributed software development: A review of literature. In International Conference on Next Generation Computing Applications (NextComp) (pp. 36–40). Doi:10.1109/NEXTCOMP.2017.8016173

- Freitag, C., Berners-Lee, M., Widdicks, K., Knowles, B., Gordon, S. B., & Friday, A. (2021). The real climate and transformative impact of ICT: A critique of estimates, trends, and regulations. *Patterns*. Doi: 10.1016/j.patter.2021.100340

- Glas, M., Nirschl, C., Lanyado, B., van Niekerk, J. (2026). Insecure by design? A human-centric security perspective on AI-assisted software development. Computers and Security. 164, 104842. Doi: 10.1016/j.cose.2026.104842

- Hove, S. E., & Anda, B. (2005). Experiences from conducting semi-structured interviews in empirical software engineering research. In IEEE International Software Metrics Symposium (METRICS). Doi: 10.1109/METRICS.2005.24

- Imai, S. (2022). Is GitHub Copilot a substitute for human pair-programming? An empirical study. In IEEE/ACM 44th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion) (pp. 319–321). Doi: 10.1145/3510454.3522684

- Jalali, S., & Wohlin, C. (2012). Systematic literature studies: Database searches vs. backward snowballing. In ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (pp. 29–38). Doi: 10.1145/2372251.23722

- Johnson, R. B., Onwuegbuzie, A. J., & Turner, L. A. (2007). Toward a definition of mixed methods research. *Journal of Mixed Methods Research, 1*(2), 112–133. Doi: 10.1177/1558689806298224

- Kanbur, M., & Kulkarni, O. P. C. (2023). Creative AI in software project management. In International Conference on Futuristic Technologies (INCOFT). Doi: 10.1109/INCOFT60753.2023.10425234

LUT Group Confidential - Other information (3Y)

- Khoury, R., Avila, A. R., Brunelle, J., & Camara, B. M. (2023). How secure is code generated by ChatGPT? In IEEE International Conference on Systems, Man, and Cybernetics (SMC). Doi: 10.1145/3658644.3690283

- Klemmer, J. H., Horstmann, S. A., Patnaik, N., Ludden, C., Burton, C., & Powers, C. (2024). Using AI assistants in software development: A qualitative study on security practices and concerns.

- Mdallal, R., Alsawalmeh, D., Jarad, A., & Odeh, A. (2023). AI-powered conceptual model for Scrum framework. In International Conference on Computing and Information Technology (ICCIT) (pp. 597–604). Doi: 10.1109/ICCIT58132.2023.10273881

- Mehmood, S., Janjua, U. I., & Ahmed, A. (2023). From manual to automatic: The evolution of test case generation methods and the role of GitHub Copilot. In International Conference on Frontiers of Information Technology (FIT) (pp. 13–18). Doi: 10.1109/FIT60620.2023.00013

- Mendes, W., Souza, S., & Souza, C. D. (2024). You're on a bicycle with a little motor: Benefits and challenges of using AI code assistants. In IEEE/ACM Conference on Cooperative and Human Aspects of Software Engineering (CHASE). Doi: 10.1145/3641822.3641882

- Mohamed, S., Parvin, S., & Parra, A. E. (2024). Chatting with AI: Deciphering developer conversations with ChatGPT. In IEEE/ACM International Conference on Mining Software Repositories (MSR) (pp. 187–191). Doi: 10.1145/3643991.364507

- Moher, D., Liberati, A., Tetzlaff, J., & Altman, D. G. (2009). Preferred reporting items for systematic reviews and meta-analyses: The PRISMA statement. *Annals of Internal Medicine, 151*(4), 264–269. Doi: 10.1371/journal.pmed.1000097

- Nguyen, N., & Nadi, S. (2022). An empirical evaluation of GitHub Copilot's code suggestions. In IEEE/ACM International Conference on Mining Software Repositories (MSR). Doi: 10.1145/3524842.3528470

- Peras, D., Stapić, Z., & Matijević, M. (2023). AI techniques and tools in agile software development: Preliminary research. Proceedings of the Central European Conference on Information and Intelligent Systems.

- Purwoko, J. W., Abdullah, T., Wijaya, B. A. A., Gunawan, S., & Saputra, K. E. (2023). Analysis of ChatGPT potential: Transforming software development with AI chatbots. In International Conference on Networking, Electrical Engineering, Computer Science, and Technology (IConNECT) (pp. 36–41). Doi: 10.1109/IConNECT56593.2023.10327087

LUT Group Confidential - Other information (3Y)

- Rajbhoj, A., Somase, A., Kulkarni, P., & Kulkarni, V. (2024). Accelerating software development using generative AI: ChatGPT case study. In Innovations in Software Engineering Conference (ISEC). Doi: 10.1145/3641399.3641403

- Şahin, F. (2022). Artificial intelligence, game theory, programming languages and training methods. In International Conference on Artificial Intelligence of Things (ICAIoT). Doi: 10.1109/ICAIoT57170.2022.10121899

- Sikand, S., Mehra, R., Sharma, V. S., Kaulgud, V., Podder, S., & Burden, A. P. (2024). Do generative AI tools ensure green code? An investigative study. In IEEE/ACM International Workshop on Responsible AI Engineering (RAIE). Doi: 10.48550/arXiv.2506.08790

- Sobania, D., Briesch, M., Hanna, C., & Petke, J. (2023). An analysis of the automatic bug fixing performance of ChatGPT. In IEEE/ACM International Workshop on Automated Program Repair (APR) (pp. 23–30). Doi: 10.48550/arXiv.2301.08653

- Smolic, E., Brcic, M., Hobor, L., Kovac, M. (2026). AI-Assisted Unit Test Writing and Test-Driven Code Refactoring: A Case Study. arXiv:2604.03135. Doi: 10.48550/arXiv.2604.03135

- Vaismoradi, M., Turunen, H., & Bondas, T. (2016). Theme development in qualitative content analysis and thematic analysis. *Journal of Nursing Education and Practice, 3*(1), 1–10. Doi: 10.5430/jnep.v6n5p100

- Waseem, M., Das, T., Ahmad, A., Liang, P., Fahmideh, M., & Mikkonen, T. (2024). ChatGPT as a software development bot: A project-based study. In International Conference on Evaluation of Novel Approaches to Software Engineering (ENASE) (pp. 406–413). Doi:10.5220/0012631600003687

LUT Group Confidential - Other information (3Y)