InSight Bulletin: A Multidisciplinary Interlink International Research Journal
Peer Reviewed International, Open Access Journal.
ISSN : 3065-7857 / Website: https://ibrj.us / Volume-3, Issue-1 / January - 2026

Original Article

A Study on the Efficient and Effective Use of Gen-AI Tools in Software Development

Dr. Reena Bharathi[1], Asraar Ahmed Shaikh[2], Zafar Ali Inamdar[3]
[1]Associate Professor, Computer Science, Dept. of Computer Science,
Nowrosjee Wadia College (Autonomous), Pune, Maharashtra, India
[2,3]Student, M.Sc., Computer Science, Dept. of Computer Science,
Nowrosjee Wadia College (Autonomous), Pune, Maharashtra, India

Manuscript ID:
IBMIIRJ -2026-030117

Submitted: 07 Dec. 2025

Revised: 11 Dec. 2025

Accepted: 06 Jan. 2026

Published: 31 Jan. 2026

ISSN: 3065-7857

Volume-3

Issue-1

Pp. 85-96

January 2026

Correspondence Address:
Reena Bharathi,
Associate Professor Dept. of Computer Science,Nowrosjee Wadia College(Autonomous), Pune, Maharashtra, India
Email:
reenabharathi@nowrosjeewadiacollege.edu.in

OPEN ACCESS

Quick Response Code:

Web. https://ibrj.us

DOI: 10.5281/zenodo.18950107

DOI Link:
https://doi.org/10.5281/zenodo.18950107

BY NC SA
Creative Commons

Abstract

Generative AI tools are being widely adopted in software development because they help developer work faster and more efficiently. Most developers believe these tools save time and improve productivity. However, challenges related to productivity gain, code quality, code reliability, developers trust, ethical use, privacy as well as security still remain a major concern. This study offers a quantitative examination of GenAI tool adoption derived from a cross-sectional survey of 209 software development professionals and students. The result shows that GenAI tools increases productivity and reduces development time. Overall, people have a positive view of the code generated by these tools, though they understand that it may need careful review and refinement. One important finding is the gap in trust between less experienced and senior developers. While developers at all levels see the benefits, many senior professionals such team leads and architects are concerned about data security. In fact, over half of them (57.1%) do not trust GenAI tools with sensitive information. This lack of trust among experienced decision makers is one the major obstacle to using GenAI tools more deeply in enterprise projects. This study shows how important it is for GenAI tools to evolve in two ways: by making them more useful and by creating strong, verifiable trust and security framework

Keywords: *Gen-AI, Software Development, Developer productivity, Code quality, Data security, AI code assistance, Developer trust*

Introduction

The landscape of software development is undergoing a significant transformation driven by the increasing prevalence of advanced Generative AI (GenAI) coding assistants [1]. GitHub Copilot and ChatGPT were once just experiments, but now they are important parts of developers' daily work. These systems are meant to speed up coding, make hard programming tasks easier, and also help new developers to get started [2]. They could change the way we think about productivity and how we do software engineering today.

But the quick use of GenAI tools has raised a lot of important issues. Even though everyone agrees that AI-generated code is more efficient, there are still arguments about how reliable, accurate, and high-quality it is [3]. Furthermore, their utilization raises critical concerns regarding developer trust, data privacy, and the safeguarding of proprietary codebases. When developers share code snippets with outside AI services, they trust those services to protect their intellectual property. However, this trust may not always be justified [4].

This study seeks to move beyond anecdotal evidence by providing a quantitative analysis of these issues. The study investigates four key hypotheses:

1. that GenAI tools improve developer productivity;
2. that they enhance the perceived quality of code;
3. that developers trust and rely on the generated code; and
4. That ethical, privacy, and security concerns are a significant factor in their adoption.

How to cite this article:
Bharathi, R., Shaikh, A. A., & Inamdar, Z. A. (2026). A Study on the Efficient and Effective Use of Gen-AI Tools in Software Development. InSight Bulletin: A Multidisciplinary Interlink International Research Journal, 3(1), 85–96. https://doi.org/10.5281/zenodo.18950107

The term *efficient* refers to the ability to achieve desired results with minimal use of resources while the term *effective* refers to the extent to which an intended objective or desired outcome is achieved [5]. Through a survey of 209 developers, students, and technology leaders, this study explores the complex interplay between the perceived benefits and the underlying risks of GenAI tools. The analysis includes a specific focus on how these perceptions differ across professional roles, revealing an "experience-trust divide" that has profound implications for the future of AI in enterprise software development.

Literature Review

Several studies have been carried out on the application of GenAI and how GenAI tools affect software development productivity in firms [6][7][8][9][10][11]. Coutinho et al. (2024) conducted a pilot case study in a large software company to see how toolssuch as. Most participants felt that, these tools were helpful because they save time, support learning and can be used for wide range of tasks. However, some concerns were raised including, the accuracy of the results generated, the need for clear prompts, and security issues. Since the scope of the study was limited, the author suggested conducting more larger and detailed studies to better understand the overall impact of GenAI [10]. Aarti (2024) gave an overview of popular GenAI tools used in software engineering, including GitHub Copilot, ChatGPT, and Amazon CodeWhisperer. The study showed that these tools help developers by automating repetitive work and make coding mode accurate. However, it warned about possible risks like data privacy, ethical issues, and over – reliance on AI. The author concluded that though the GenAI tools enhance the development, organizations should use these tools responsibly and invest in training developers for long-term success [11].

Systematic studies on the potentials and limitations of LLM based software development were also caried out by researchers exploring how GenAI, especially large language models (LLM) could change the future of software development through automation, enhanced productivity, and innovative collaboration frameworks. [12] [13] [14] [15] [16]. Sauvola et al. (2024) They suggested four possible futures (from human-led to AI-led) and talk about how to adopt them, how to use them in different development settings, and problems like ethics, IP, and changing skill needs [16].

Yetiştiren et al. (2023) conducted comparison of the coding abilities of AI tools like ChatGPT, GitHub Copilot, and Amazon CodeWhisperer highlighting that these tools can improve productivity and reduce repetitive work, but the quality and reliability of the generated code vary across tools. The authors stressed that AI should support developers and not replace them and called for better standards to evaluate AI – generated code [17].

Pandey et al. (2024) examined how GitHub Copilot performs in real-world academic and corporate environments. They found that Copilot saved time and improved documentation, especially for simple and repetitive tasks. However, it faced problems with complex projects and multiple files. This study suggested combining AI tools with human code reviews to ensure accuracy and security [18]. Song et al. (2024) studied the use of Copilot in open-source software projects and found that it increased overall contributions and participation. However, it also led to more coordination and integration challenges, showing that higher productivity can sometimes come with added complexity in team collaboration [19].

Researchers have also highlighted various ethical and security concerns originating from the use of GenAI[20][21][22][23]. Pant et al. (2024) performed a grounded theory literature review which focused on how software professionals think about ethics when building AI based systems. Their examination of empirical studies revealed five principal themes: awareness, perception, needs, challenges, and approaches. Their review found that although developers are aware of ethical principles, they often find it difficult to apply them in real projects, due to unclear guidelines, lack of organization support, and limited tools, highlighting the need to make ethics a regular part of AI – driven development and decision- making [23].

Butler et al. (2024) executed a controlled field trial to examine how developers feel about using GenAI tools at work. While developers found these tools helpful, enjoyable, and time-saving, they still questioned their reliability and showed lack of trust. This shows that developers are open to using AI but continue to carefully review and control its outputs [24]. McKinsey (2023) provided their perspective on GenAI in software development and reported that AI tools can boost productivity up to 45% in certain task. However, it also emphasized that organizations must manage change carefully, focus on skill development, and establish strong governance to use AI safely and effectively [25].

Although GenAI tools are widely adopted in software development and are generally perceived to improve efficiency and productivity, existing studies are largely subjective. There is limited empirical evidence that jointly examines productivity gains, perceived code quality, developer trust in AI-generated code, and ethical, privacy as well as security concerns. This paper seeks to bridge this gap by providing a quantitative analysis of these factors and examining their influence on the adoption and use of GenAI tools in software development.

Methodology

This study used a quantitative, cross-sectional survey to understand how software developers and students perceive and use GenAI coding tools. The research focused on four pre-defined hypotheses focused on developer productivity, code quality, trust in AI tools and concerns related to ethics and security

Data Collection and Participants

An online questionnaire was developed and distributed to individuals involved in software development. The survey was active during October 2025. Participants were recruited through a convenience sampling strategy, social media networks such as LinkedIn and WhatsApp. A total of 209 complete responses were collected and used for the analysis. The participant pool was diverse, encompassing a wide range of roles and experience levels, from students and junior developers to senior architects and educators. This heterogeneity allowed for a comprehensive and multi-faceted analysis of GenAI tool adoption.

Hypothesis

The research was guided by four pre-defined hypotheses designed to explore key aspects of GenAI tool adoption:

H1: Use of GenAI tools improves developer productivity.
H2: GenAI tools improve the perceived quality of code.
H3: Developers trust and rely on GenAI-generated code in real-world applications.
H4: Ethical, privacy, and security concerns significantly influence tool adoption.

Survey Instruments

The questionnaire was structured into four primary sections designed to systematically gather data corresponding to the research hypotheses:

1. **Demographics and Professional Background:** This section collected data on the participants' current role, years of programming experience, and primary programming languages.
2. **GenAI Tool Usage:** This section captured information on the specific GenAI tools used, the frequency of use, and the primary tasks for which these tools were employed.
3. **Perceptions of Effectiveness and Quality:** This section contained a series of statements aligned with Hypotheses 1 and 2. Participants were asked to rate their agreement using a 5-point Likert scale, ranging from "Strongly Disagree" (1) to "Strongly Agree" (5).
4. **Trust, Security, and Ethical Concerns:** This final section addressed Hypotheses 3 and 4, covering trust in generated code, data security, and prior experience with security issues.

Hypothesis Mapping

The survey questions were carefully designed to match the research hypothesis, ensuring that the data collected could be clearly analysed and statistically validated.

H1 (Productivity) was assessed using questions on time savings (Q11a: Generative AI tools save me time during coding), productivity improvement (Q11e: These tools improve my productivity), and self-reported productivity changes (Q13: Have you noticed an improvement in your productivity since using GenAI tools?).

H2 (Code Quality) was evaluated through questions on the accuracy and usability of generated code (Q11b: The code generated is mostly accurate and usable) and ratings of code readability and accuracy (Q12a: Code accuracy, Q12b: Code readability).

H3 (Trust and Reliance) was measured by questions on understanding and trusting the code (Q11d: I understand and trust the code generated), reliance on AI for complex problems (Q11f: I rely on AI tools for solving complex problems), trust in data security (Q15: Do you trust generative AI tools not to leak or expose sensitive information from your codebase?), and personal experience with security issues (Q16: Have you ever experienced a security or ethical issue from using an AI code tool (e.g., copying proprietary code, generating vulnerable code)?).

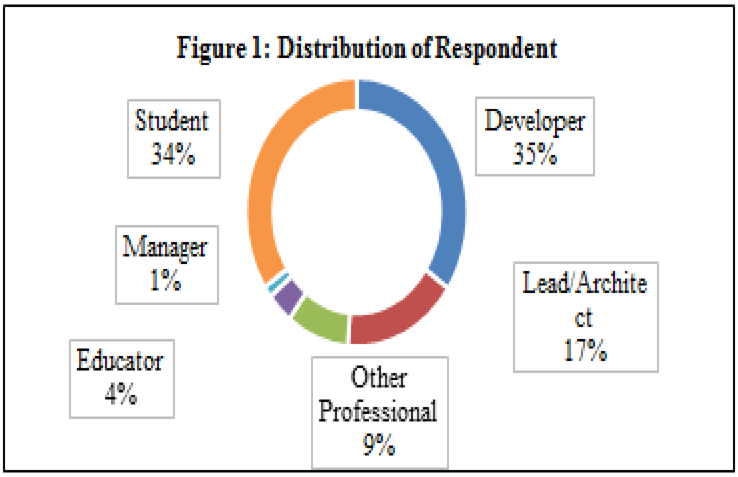
H4 (Ethical & Security Concerns) was investigated by asking about general concerns (Q14: Do you have any concerns about using generative AI tools in software development?) and through the security-focused questions also mapped to H3 (Q15: Do you trust generative AI tools not to leak or expose sensitive information from your code base?, Q16: Have you ever experienced a security or ethical issue from using an AI code tool (e.g., copying proprietary code, generating vulnerable code)?).

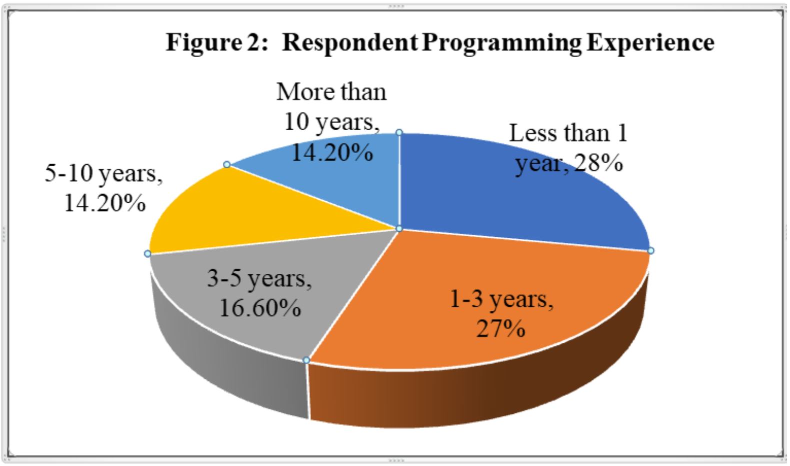
Data Analysis

The collected data was analysed using descriptive and inferential statistics with Python libraries. Non-parametric tests, including the One-Sample Wilcoxon Signed-Rank Test for Likert data and the Chi-Square Goodness-of-Fit Test for categorical data, were employed with a significance level of $p < 0.05$. The analysis included a role-based comparison, for which roles were standardized into five groups. The findings are visualized in the figures and tables below.

Distribution of Respondent Roles

Figure1 illustrates the professional composition of the survey participants.The two largest groups were Students (34.4%) and Developers (34.4%), followed by Leads/Architects (16.7%), Other Professionals (9.1%), and Educators (3.8%), ensuring a diverse range of perspectives in the study.

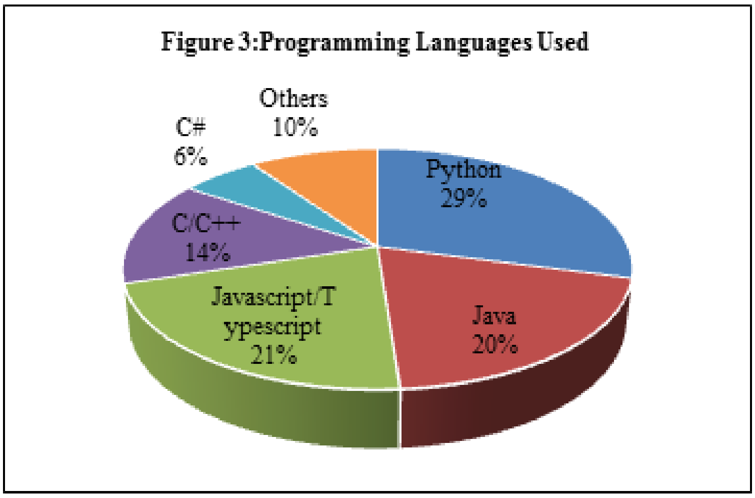




Programming Experience of Survey Respondents

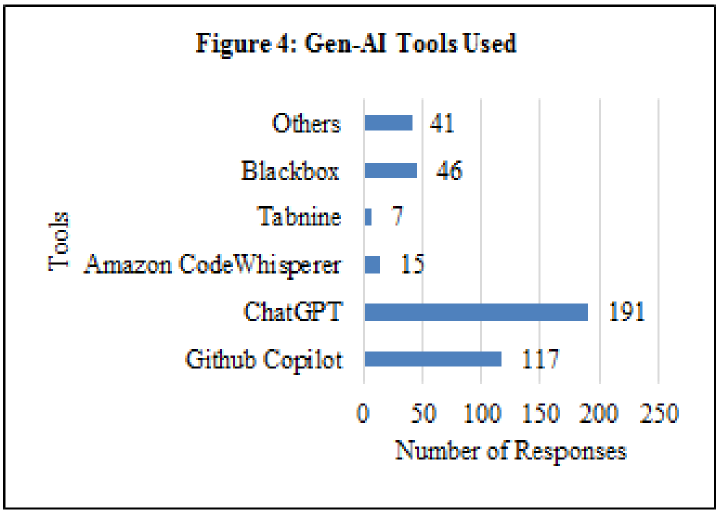
Figure 2 indicates a wide distribution of experience levels, with a significant cohort of early-career professionals (28% with less than 1 year; 27% with 1–3 years) and a substantial presence of senior professionals (14% with 10+ years).

Most Common Programming Languages Used by



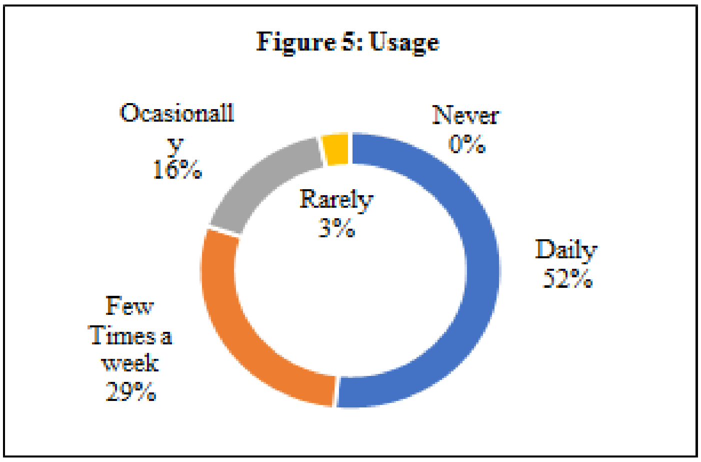
Respondents:

Figure 3 illustrates the prevalence of major languages. Python was the most common, used by 29% of participants, followed by JavaScript / TypeScript (21%) and Java (20%), reflecting strong GenAI adoption within today's most popular language ecosystems. Whereas C/C++ is (14%) and C# (6%) and others (10%)



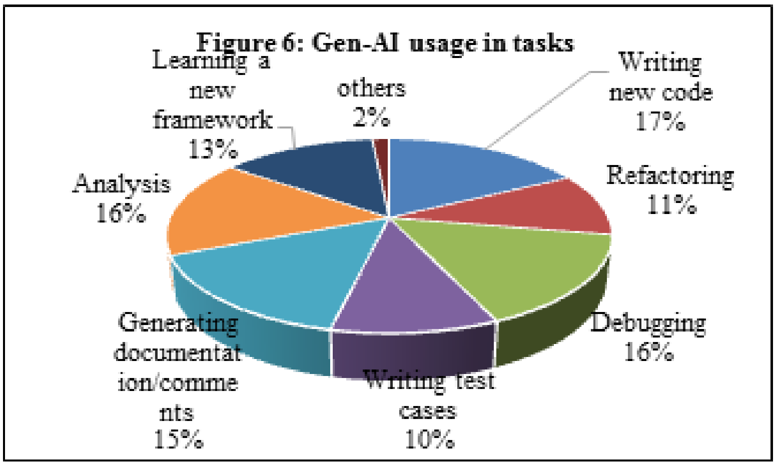
Prevalence of Different Generat

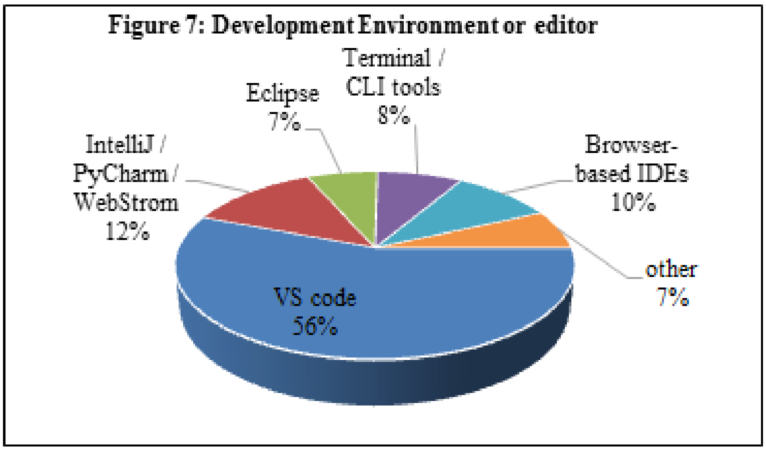
ve AI Tools:
Figure 4 ranks the tool popularity amongst the 209 respondents. ChatGPT is the dominant tool, utilized by an overwhelming 91% of the sample. Specialized coding assistants like GitHub Copilot are also popular, used by 56%, suggesting a common practice of using both generalist and specialist tools. Another tool which is mostly used in BlackBox (22%) and others are (19%)



Frequency of GenAI Tool Use in Development Workflows:
Figure 5 illustrates the deep integration of GenAI into daily work. A significant majority (81%) use these tools frequently, either "Daily" (52%) or "A few times a week" (29%), confirming that for most users, GenAI is a staple rather than an occasional aid. The study also shows that 19% of the respondent use it either occasionally (16%) and Rarely (3%)

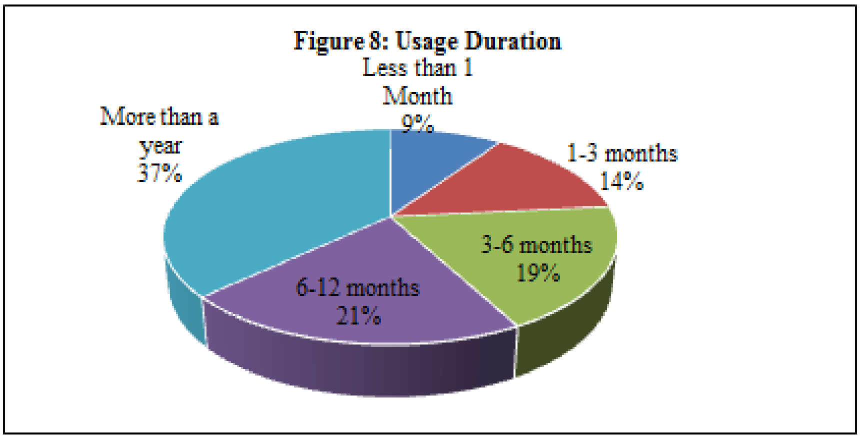
Most Frequent Tasks Performed with GenAI Tools
Figure 6 highlights GenAI's versatility. The most common tasks were Debugging (16%), Writing new code (17%), and Learning a new framework / library (13 %), analysis (16%), demonstrating its role across the entire development lifecycle.

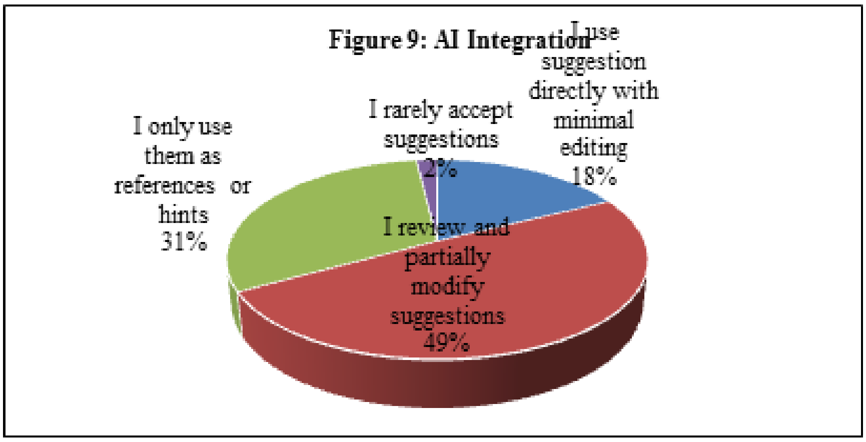




Popular Development Environments for GenAI Tool Integration:
Figure 7 illustrates where developers make use of GenAI. VS Code is the clear leader, used by 56% of respondents. Integrated environments like IntelliJ / PyCharm / WebStorm also showed significant use at 12% while others tools such as Eclipse, CGI based and browser-based tools are also used.

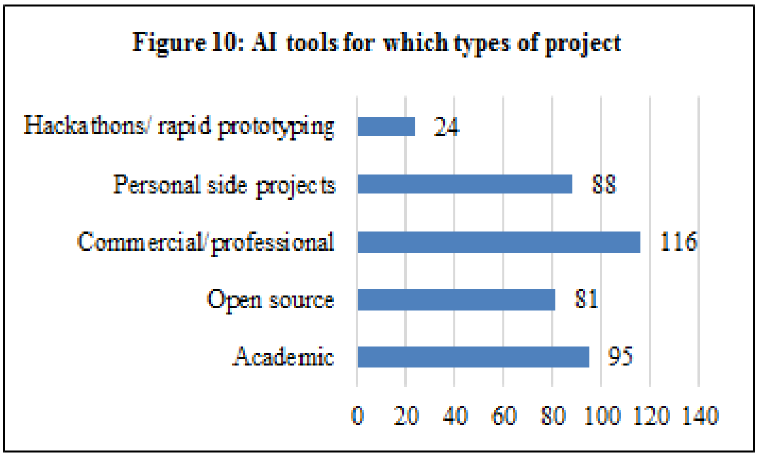
How Respondents Have Been Using AI-Assisted Coding Tools: Figure 8 indicates a maturing user base. A majority of users (37%) have been using these tools for more than a year, with 21% having over six months of experience, suggesting perceptions are based on long-term usage.





Typical Methods for Integrating AI Suggestions: Figure 9 illustrates a dominant "human-in-the-loop" workflow. The most common approach, chosen by 49% of developers, is to "review and partially modify suggestions. 31% says "Only use them as references or hints"" Only 18% reported using suggestions directly with minimal editing, underscoring a pattern of critical collaboration

10. Types of Projects Where Gen AI Tools Are Used: Figure 10 highlights GenAI application across different contexts. Adoption is highest in Commercial / professional projects (55%). However, usage in Academic project is nearly as high at 45%, demonstrating a significant leap from experimental to professional environments. Even open source have the high usage of 38%



11. Likert

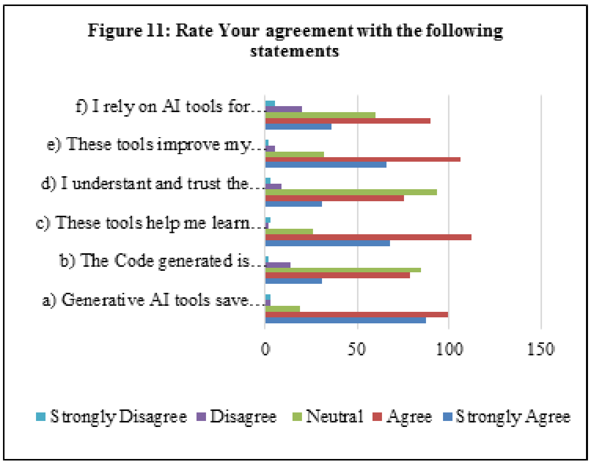


Figure 11: Rate Your agreement with the following statements

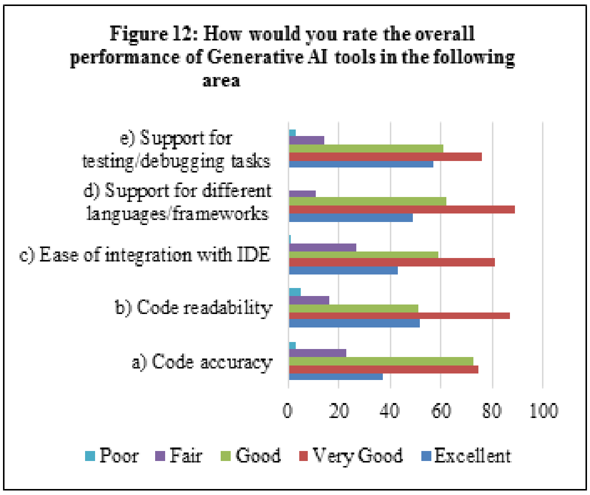


Figure 12: How would you rate the overall performance of Generative AI tools in the following area

Figure 11 highlights agreement of the respondents to questions. It can be seen from the above figure 11 that an overwhelming 184 of respondents Agreed or Strongly Agreed with the statement "Generative AI tools save time during coding". For "code generated is mostly accurate and usable" most people agreed (78) but there were more who voted as neutral (85). The highest number of respondents agrees that AI tools help to learn faster (178). Among the 209 respondent 105 strongly agree / agree that they understand the code generated but a large number (92) were neutral. For improving productivity (170) strongly agree / agree with the statement. Most of the respondent rely on AI tools for solving complex problem (124) whereas (64) are neutral

Figure 12 illustrates that an overwhelmingly 183 respondents says that the code accuracy is Good while only (26) saying its fair or poor. It also highlights that the 188 respondents are able to read the code whereas only 21 have issues in reading. 181 respondents feel it is easy to integrate the code with IDE and 198 respondents say Gen-AI tools support different languages and framework while 192 respondents say that Gen-AI tools supports testing and debugging tasks

12. Self-Reported Change in Productivity Since Using GenAI Tools:

Figure 13 provides powerful, direct evidence supporting Hypothesis 1.

It can be seen that a remarkable 93% of users reported that their productivity has either "Slightly" (36%) or "Significantly" (57%) increased while only 2% reported a decrease.

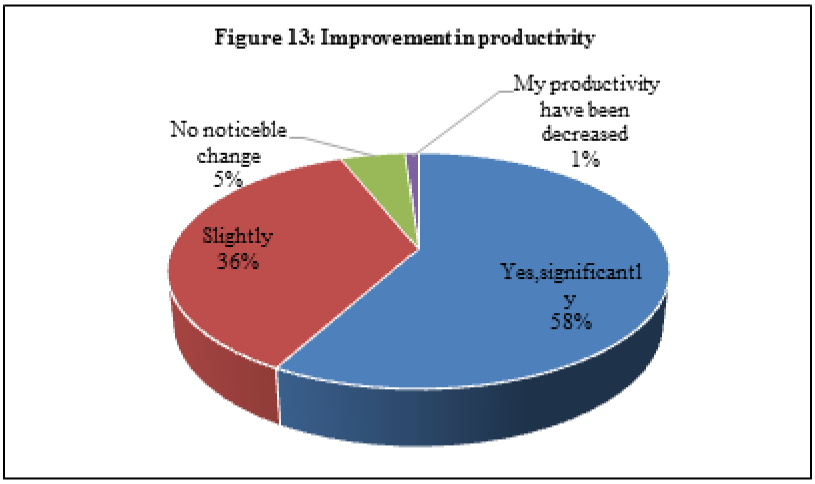


Figure 13: Improvement in productivity

13. Using a subscription of Gen-AI Tools

Figure 14 illustrates that most of the respondents (61%) use the
free Gen-AI tool and (29%) have the subscription whereas (9%) are planning to subscribe

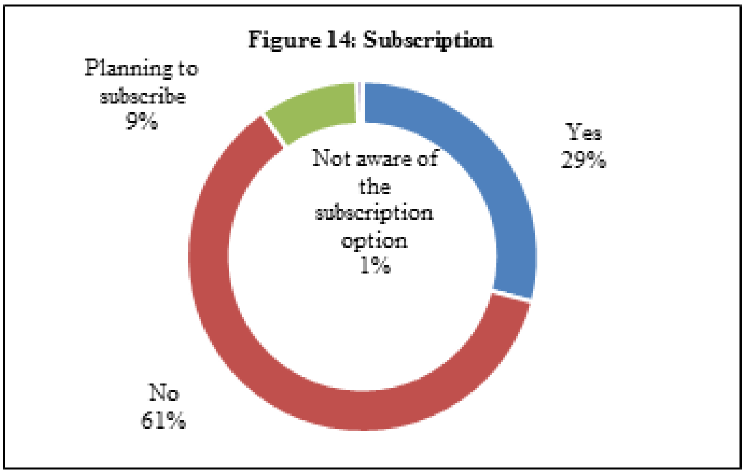


Figure 14: Subscription

4. Hypothesis Testing

The statistical analysis of the 209 survey responses yielded significant insights into each of the four hypotheses. The findings confirm the perceived benefits of GenAI tools in terms of productivity and quality, but also reveal a deep-seated and statistically significant level of distrust, particularly concerning data security.

4.1. Hypothesis 1: Use of GenAI tools improves developer productivity-This hypothesis was tested using the following questions:

Q11e (Improves productivity): p-value = 4.80e-29
Q13 (Productivity improvement distribution): p-value = 1.00e+00

Interpretation:

Low p-values support H1, indicating a statistically significant agreement that GenAI tools improve productivity
Q11a: Rate your agreement: Generative AI tools save my time during coding.
Q11e: Rate your agreement: These tools improve my productivity.
Q13: Have you noticed an improvement in your productivity since using GenAI tools?

Table 1. Mean agreement scores for Productivity, Quality and Trust by role

Role	productivity	quality	trust	Respondent Count
Developer	4.04	3.43	3.49	72
Educator	3.88	3.50	3.38	8
Lead / Architect	4.23	3.71	3.74	35
Other Professional	4.26	3.53	3.32	19
Student	4.00	3.68	3.68	72

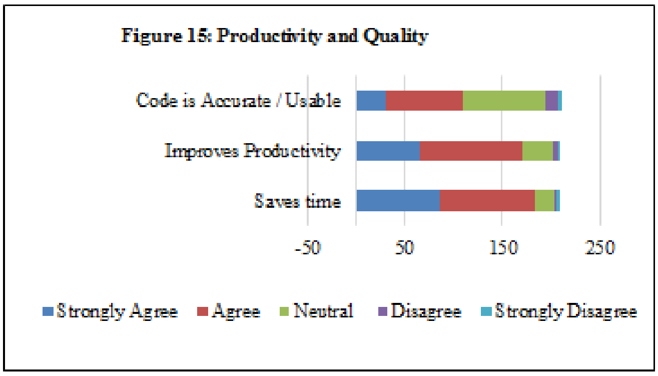


Figure 15: Productivity and Quality

The data provides strong support for H1. As shown in Figure 15, respondents showed statistically significant agreement that GenAI tools save time and improve productivity (p < 0.001 for both Q11a and Q11e). This positive sentiment was consistent across all professional roles, with mean agreement scores for productivity above 4.0 (Agree) for most groups (see Table 1 and Figure 17).

Result

Q11a (Saves time): p-value = 4.15e-31

Agreement with Statements on Productivity and Quality (n=209):

Figure 15 visualizes the overwhelmingly positive sentiment regarding GenAI benefits. For "Improves productivity," 78% of respondents agreed or strongly agreed. For "Code is accurate and usable," 60% agreed or strongly agreed. A Wilcoxon Signed-Rank test confirmed these positive sentiments were statistically significant (p<0.0001 for both)

4.2. Hypothesis 2: GenAI tools improves the perceived quality of code

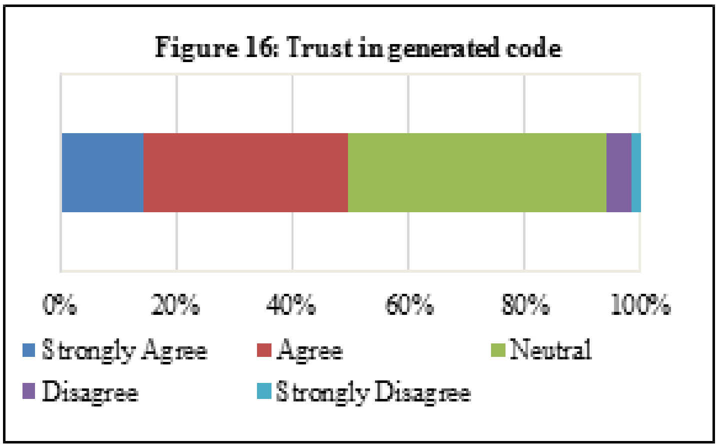
This hypothesis was tested using the following questions:
Q11b: Rate your agreement: The code generated is mostly accurate and usable.
Q12a: How would you rate the overall performance of generative AI tools in... Code accuracy?
Q12b: How would you rate the overall performance of generative AI tools in... Code readability?

The data provides support for H2. As visualized in Figure 15, there was statistically significant agreement that the generated code is accurate and usable (p < 0.001). This perception of high quality was consistent across all roles (Table 1), indicating a broad consensus on this benefit.

Result
Q11b (Accurate and usable): p-value = 1.13e-15
Q12a (Code accuracy rating): p-value = 1.35e-13
Q12b (Code readability rating): p-value = 1.92e-18
Interpretation: Low p-values support H2, indicating a statistically significant positive perception of code quality.

4.3. Hypothesis 3: Developers trust and rely on GenAI-generated code

This hypothesis was tested using the following questions
Q11d: Rate your agreement: I understand and trust the code generated.
Q11f: Rate your agreement: I rely on AI tools for solving complex problems.
Q15: Do you trust generative AI tools not to leak or expose sensitive information from your codebase?



Result
Q11d (Understand and trust code): p-value = 1.23e-15
Q11f (Rely on for complex problems): p-value = 9.09e-15
Q15 (Trust not to leak data distribution): p-value = 1.00e+00

Interpretation:
GenAI platforms with sensitive data.
This profound security epticism directly challenges the notion of broad-based trust and is most pronounced among senior roles (Figure 18).

The results for H3 are paradoxical and reveal the "experience-trust divide." While a majority of users trust the functional correctness of the code (Figure 16, p < 0.001), this trust does not extend to data security. As detailed in Table 3, nearly half of all developers (44.4%) and a majority of leads/architects (57.1%) explicitly do not trust

The Wilcoxon tests check for positive trust/reliance. The Chi-square test shows if the strong opinions on data leakage are random. Low p-values here challenge H3, revealing significant distrust in security.
Distribution of Agreement on Trust in Functional Correctness of AI-Generated Code: Figure 16 illustrates a clear positive skew, with a majority of respondents (51%) agreeing or strongly agreeing that they trust the functional output of the code. The median response was significantly greater than neutral (p < 0.001), supporting the first part of Hypothesis 3.

4.4. Hypothesis 4: Ethical, privacy and security concerns significantly influence tool adoption.

This hypothesis was tested using the following questions:
Q14: Do you have any concerns about using generative AI tools in software development?
Q15: Do you trust generative AI tools not to leak or expose sensitive information from your codebase?
Q16: Have you ever experienced a security or ethical issue from using an AI code tool?

The data provides overwhelming support for H4. The widespread security distrust found in H3 is a key factor. Furthermore, as shown in Figure 16, the most frequently cited concerns were "Data privacy" and "Security risks," selected by 59.3% and 50.2% of respondents, respectively. These results confirm that ethical and security issues are a primary factor influencing developer adoption and usage patterns.

Most Frequently Cited Concerns Regarding GenAI Tool Usage (n=209)

Table 2 highlights the top concerns selected by respondents. "Data privacy" was the most cited concern, selected by 124 participants (59.3%), followed by "Security risks" (105 participants, 50.2%), directly supporting the significance of Hypothesis 4.

Table 2. Security concerns

Survey Questions	Responses
Data privacy when the code is sent to the server	124
Security risks from using AI- generated code	105
Copyright issues/ Code reuse	88
Intellectual property violation	82
Lack of transparency in how suggestions are generated	70
Ethical concerns about the student use	65

Interpretation: This hypothesis is qualitative but supported by the high frequency of concerns. The Chi-square test on Q15 (from H3) provides statistical backing, showing these concerns are a significant, non-random factor.

Comparative Mean Agreement Scores by Professional Role : Figure 17 directly illustrates the "experience-trust divide." While mean scores for Productivity (M > 3.8) and Quality (M > 3.4) are consistently high across all roles, the mean score for Trust shows greater variation, with Leads/Architects (M = 3.74) and Students (M = 3.68) reporting higher functional trust than Developers (M = 3.49).

Table 3. Distribution of trust in Data Security by role (%)

Role Group	No	Not Sure	Yes
Developer	44.4	30.6	25.0
Educator	37.5	50.0	12.5
Lead/ Architect	57.1	22.9	20.0
Other Professional	52.6	36.8	10.5
Student	31.9	52.8	15.3

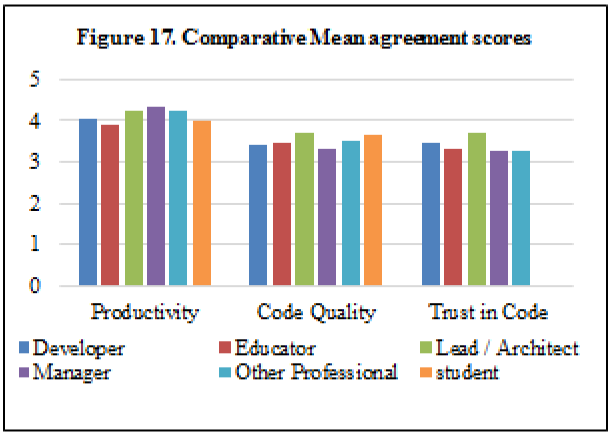


Figure 17. Comparative Mean agreement scores

Distribution of Trust in Data Security by Professional Role (n=209):
Table 3 illustrates a detailed breakdown of the "experience-trust divide." It contrasts the high percentage of Leads/Architects who explicitly distrust GenAI with their data (57.1% "No") against the high percentage of Students who are uncertain (52.8% "Not sure"). A Chi-Square test confirmed that the distribution of these opinions is statistically significant ($p < 0.001$).

Discussion

Our findings present a dual narrative for the role of GenAI in software development, characterized by a tension between utility and security. This study confirms the transformative potential of GenAI tools while simultaneously highlighting the significant barriers to their full-scale adoption, leading to what we term the "Developer's Dilemma."

1. **The Unanimous embrace of productivity and quality**

The results clearly show that GenAi tools are widely seen as improving both productivity and code quality, which helps explain why they are being adopted so rapidly. Similar studies were carried out which shows clear productivity gains from tools like GitHub Copilot [2]. Developers across roles- from students to senior architects- agreed on these benefits, suggesting that GenAI tools are useful for everyone. This broad agreement highlights why GenAI has become an important part of modern development workflows.

2. **The paradox of trust: Functional confidence vs Security suspicion**

One the key finding of this study is the paradoxical nature in developer trust. While developers generally trust GenAI tools to produce correct code, they remain highly concerned about data security. In other words, developers may rely on AI for coding tasks but hesitate to share sensitive information with it. This concern matches earlier researches that raised warnings about security risks [3],[4]. Nearly half of our respondents (45%) do not trust these tools with sensitive data and 38% are uncertain.

This lack of trust represents the core “Developer’s Dilemma” and remains the main barrier to deeper enterprise adoption of GenAI tools

3. **The experience – Trust divide: Senior developer as security gatekeepers**

The role-based analysis shows that concerns about data security increase with experience and responsibility, creating what we call an “experience-trust divide”. Senior professionals such as tech leads and architects are the most cautious because they are directly responsible for system security and protecting company data. Their concerns reflect practical risk assessment rather than abstract fear.

These senior people act as an organizational gatekeeper. While students and individual developers may use GenAI tools to boost personal productivity, leads and architects decide whether such tools are approved at scale. Their strong lack of trust-57.1% saying they would not trust GenAI tools with sensitive data-remains the biggest barrier to using these tools as fully trusted, enterprise-level systems.

4. Implications and the path forward

The findings of this study have clear implications for three key stakeholders:

For Organizations: GenAI tools can clearly improve productivity, but they also introduce real risks. Companies need clear rules about how these tools should be used especially around what data can and cannot be shared. The strong concerns raised by senior professionals also show a growing need for private, on-premise, or high secure GenAI solutions that organizations can fully trust.

For GenAI Tool Developers: The message from the development community is clear: security and transparency are not optional features but core requirements. To succeed in enterprise environments, tool creators must inform how data is been handled, protected and owned. Building trust through strong privacy guarantees is just as important as improving the accuracy of AI models.

For Developers and Educators: GenAI tools should be used thoughtfully, not blindly. Developers must carefully review and test AI-generated code before using it in real systems. For students-who tend to be less aware of security risks-educators play a key role in teaching both how to use these tools effectively and how to understand their limitations and potential risks.

5. Limitations of study

It is important to acknowledge the limitations of this study. The use of a convenience sampling strategy may introduce selection bias, as individuals with strong opinions on GenAI may have been more likely to respond. Furthermore, this study is a cross-sectional snapshot; the GenAI field is evolving rapidly, and perceptions may change as the technology matures. Finally, our data captures perceived productivity and quality, which may not perfectly correlate with objective, measurable metrics.

Conclusion

This study shows that although GenAI coding tools are widely praised for improving productivity and code quality, their wider adoption is held back by strong concerns about data security. These concerns are especially strong among senior developers and architects, who are responsible for protecting systems and intellectual property. As a result, the future of AI is software engineering depends not only on better algorithms, but also on building trust. Until the tension between productivity and security-the “developer’s dilemma”-is addressed, the full potential of GenAI is enterprise environments will remain limited

Future Work and Enhancement

While this study offers a strong snapshoot of current developer views, the fast-changing nature of GenAI makes ongoing research essential. Based on our findings, several directions for future work stand out.

- **Qualitative 0** more objective evidence. For example, a controlled experiment comparing developers who use GenAI tools with those who do not on the same complex tasks could measure differences in code complexity, bug rates, and adherence to security best practices.
- **0 Developers:** The data shows that students use GenAI tools frequently. Further research should examine how this affects skill development over time-whether early use helps students learn faster through examples, or whether it weakens code problem-solving and debugging skills

Acknowledgment

The authors would like to express their sincere gratitude to all the software professionals, students, educators, and industry practitioners who participated in the survey and generously shared their time and insights. Their responses were invaluable to understanding real-world perspectives on the use of Generative AI tools in software development. The authors also acknowledge the support and encouragement provided by the Department of Computer Science, Nowrosjee Wadia College (Autonomous), Pune, which facilitated the completion of this study.

Financial support and sponsorship

Nil.

Conflicts of interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Reference

1. Ziegler, A., Svyatkovskiy, A., & Neubig, G. (2022). Large language models for code: A systematic review and empirical analysis. Proceedings of the 44th International Conference on Software Engineering (ICSE).
2. Peng, S., Kalliamvakou, E., Czerwonka, P., & Bird, C. (2023). The productivity effects of generative AI assistance: The case of GitHub Copilot. Science, 381(6657), 1-8
3. Perry, N., O'Dea, R., & Assael, M. (2023). Do users write more insecure code with AI assistants? Proceedings of the ACM SIGSAC Conference on Computer and Communications Security.
4. Pearce, H., Ahmad, B., Tan, B., Dolan-Gavitt, B., & Karri, R. (2022). Asleep at the keyboard? Assessing the security of GitHub Copilot’s code contributions. 2022 IEEE Symposium on Security and Privacy (SP).

5. ISO, ISO 9000:2015—Quality management systems—Fundamentals and vocabulary. Geneva, Switzerland: International Organization for Standardization, 2015.
6. Pereira, G. V., Jackson, V., Prikladnicki, R., van der Hoek, A., Fortes, L., Araújo, C., ... & Ramos, D. (2025, April). Exploring GenAI in Software Development: Insights from a Case Study in a Large Brazilian Company. In 2025 IEEE/ACM 47th International Conference on Software Engineering: Software Engineering in Practice (ICSE-SEIP) (pp. 330-341). IEEE.
7. Petrovska, O., Clift, L., Moller, F., & Pearsall, R. (2024, January). Incorporating generative AI into software development education. In Proceedings of the 8th Conference on Computing Education Practice (pp. 37-40).
8. Nguyen- Duc, A., Cabrero- Daniel, B., Przybylek, A., Arora, C., Khanna, D., Herda, T., ... & Abrahamsson, P. (2025). Generative artificial intelligence for software engineering—A research agenda. Software: Practice and Experience, 55(11), 1806-1843.
9. Cornide-Reyes, H., Monsalves, D., Durán, E., Silva-Aravena, F., & Morales, J. (2025, May). Generative Artificial Intelligence in Agile Software Development Processes: A Literature Review Focused on User eXperience. In International Conference on Human-Computer Interaction (pp. 228-246). Cham: Springer Nature Switzerland.
10. Coutinho, M., Marques, L., Santos, A., Dahia, M., França, C., & de Souza Santos, R. (2024, July 15–16). The role of generative AI in software development productivity: A pilot case study. In Proceedings of the 1st ACM International Conference on AI-PoweredSoftware(AIware'24)(pp.1–8).ACM. https://doi.org/10.1145/3664646.3664773
11. Aarti. (2024). An overview of generative AI tools and their evolving role in modern software engineering. International Journal of Innovative Research in Computer Science and Software Engineering, 10(3), 45–52. Retrieved from https://www.ijfmr.com/papers/2024/3/23271.pdf
12. Xia, C. S., Deng, Y., Dunn, S., & Zhang, L. (2024). Agentless: Demystifying llm-based software engineering agents. arXiv preprint arXiv:2407.01489.
13. Manish, S. (2024). An autonomous multi-agent llm framework for agile software development. International Journal of Trend in Scientific Research and Development, 8(5), 892-898.
14. Michelutti, C., Eckert, J., Monecke, M., Klein, J., & Glesner, S. (2024, November). A Systematic Study on the Potentials and Limitations of LLM-assisted Software Development. In 2024 2nd International Conference on Foundation and Large Language Models (FLLM) (pp. 330-338). IEEE.
15. He, J., Treude, C., & Lo, D. (2025). LLM-Based Multi-Agent Systems for Software Engineering: Literature Review, Vision, and the Road Ahead. ACM Transactions on Software Engineering and Methodology, 34(5), 1-30.
16. Sauvola, J., Tarkoma, S., Klemettinen, M., Riekki, J., & Doermann, D. (2024). Future of software development with generative AI. Automated Software Engineering, 31(26). https://doi.org/10.1007/s10515-024-00426-z
17. Yetiştiren, B., Özsoy, I., Ayerdem, M., & Tüzün, E. (2023). Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. arXiv. https://arxiv.org/abs/2304.10778 arXiv
18. Pandey, R., Singh, P., Wei, R., & Shankar, S. (2024). Transforming Software Development: Evaluating the Efficiency and Challenges of GitHub Copilot in Real-World Projects. arXiv. arXiv
19. Song, F., Agarwal, A., & Wen, W. (2024). The Impact of Generative AI on Collaborative Open-Source Software Development: Evidence from GitHub Copilot. arXiv. arXiv
20. Chakraborty, S., Burgueño, L., Moreno, N., Troya, J., & Muñoz, P. (2025). Mind the Ethics! The Overlooked Ethical Dimensions of GenAI in Software Modeling Education. arXiv preprint arXiv:2509.13896.
21. Atemkeng, M., Hamlomo, S., Welman, B., Oyetunji, N., Ataei, P., & Fendji, J. L. K. (2024). Ethics of software programming with generative AI: is programming without generative AI always radical?. arXiv preprint arXiv:2408.10554.
22. Donvir, A., & Sharma, G. (2025, January). Ethical Challenges and Frameworks in AI-Driven Software Development and Testing. In 2025 IEEE 15th Annual Computing and Communication Workshop and Conference (CCWC) (pp. 00569-00576). IEEE.
23. Pant, A., Hoda, R., Tantithamthavorn, C., & Turhan, B. (2024). Ethics in AI through the practitioner's view: A grounded theory literature review. Empirical Software Engineering, 29(67). Springer Link
24. Butler, J., Suh, J., Haniyur, S., & Hadley, C. (2024). Dear Diary: A randomized controlled trial of Generative AI coding tools in the workplace. arXiv. https://arxiv.org/abs/2410.18334 arXiv
25. McKinsey Digital. (2023). Unleashing developer productivity with generative AI. McKinsey. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/unleashing-developer-productivity-with-generative-ai McKinsey & Company+1