Applied Human Factors and Ergonomics (AHFE2025), Vol. 199, 2025, 1231–1242
https://doi.org/10.54941/ahfe1006934

Large Language Models in Programming: A Meta-Analysis of Tools, Users, and Human-Computer Interaction Themes

Daniel M. Olivares, Charles Bennington, and Abigail Skillestad

Gonzaga University, School of Engineering & Applied Science, Spokane, WA 99258-0102, USA

ABSTRACT

The integration of Large Language Models (LLMs) such as OpenAI's Codex and ChatGPT into programming environments has significantly transformed software development practices, influencing productivity, education, and user experience. This meta-analysis integrates findings from empirical studies and product-level evaluations of widely adopted commercial tools (e.g., GitHub Copilot, Amazon CodeWhisperer, Tabnine, Sourcegraph Cody) alongside innovative academic prototypes. We explore how different user groups, including novice programmers, expert developers, researchers, and computer science educators, engage with and experience these LLM-based programming tools. The synthesis highlights key human-computer interaction (HCI) themes including trust calibration, cognitive load management, interface modalities (inline versus chat interactions), and balancing automation with user control. While findings indicate substantial productivity benefits, improved workflow integration, and enhanced learning support, persistent challenges related to code correctness, user over-reliance, and ethical concerns remain significant. This integrated analysis offers comprehensive design recommendations and outlines implications for future development, research practices, and education.

Keywords: Artificial intelligence, Software development, Programming tools, Human-computer interaction, User experience, Software education, Developer productivity, Meta-analysis

INTRODUCTION

The rapid advancement and adoption of large language models (LLMs) capable of generating and assisting with code have reshaped the software development landscape. Foundational models such as OpenAI's Codex and ChatGPT have enabled the development of modern programming assistants that integrate directly into coding environments ("ChatGPT," 2024; "OpenAI Codex," 2025). Tools such as GitHub Copilot ("GitHub Copilot," 2025), Amazon CodeWhisperer ("CodeWhisperer," 2025), and Tabnine ("Tabnine," 2025) have evolved from experimental utilities into integral components of modern software engineering workflows. At the same time, academic prototypes continue to explore innovative user

interactions and pedagogical applications, providing valuable insight into future developments in programming tool design.

Despite their popularity and transformative potential, current understanding of LLM-based assistants remains fragmented because much of the literature focuses on human-computer interaction and user-experience questions (e.g., Barke et al., 2023; Vaithilingam et al., 2022), while another body of work focuses on product-level capability surveys and feature comparisons (Heller, 2024; Sugi, 2024). This divide makes it difficult to form a comprehensive view of usability, learning outcomes, and professional implications.

This paper presents a meta-analysis synthesizing empirical user studies, HCI findings, and detailed feature comparisons from leading commercial and academic programming assistants. By combining these complementary perspectives, we aim to bridge critical gaps in understanding how novices, experts, researchers, and educators, engage with LLM-based tools and how these interactions influence software development practices and education. The analysis addresses several key questions. How do distinct user groups benefit from or struggle with these tools? How do varying interface designs impact user effectiveness and trust? What design strategies can align tool capabilities with user needs and professional expectations?

The paper proceeds as follows. Section 2 outlines the methodology and analytical framework used for the meta-analysis. Section 3 maps the landscape of commercial and academic programming assistants. Section 4 presents synthesized findings for each user group. Section 5 discusses major HCI themes and resulting design recommendations, and Section 6 offers a discussion of implications. Section 7 concludes with directions for future research and development.

METHODOLOGY

Literature Selection

Our meta-analysis employs a systematic literature selection process focusing on recent studies and authoritative sources from 2021 through early 2025. Sources include prominent academic databases including ACM Digital Library, IEEE Xplore, arXiv, and other reputable repositories. Inclusion criteria encompassed empirical studies, user-centered evaluations, systematic literature reviews, and significant technical reports relevant to LLM-based programming tools.

Additionally, we incorporated high-quality grey and pre-publication sources, including official tool documentation, reputable industry reviews, technical blogs, and early-access papers or preprints (e.g., arXiv manuscripts and conference papers “in press”). These materials were selected for their credibility, timeliness, and technical detail. Drawing on them made it possible to include the most up-to-date information about tool capabilities, interface models, and practical implications before those details appear in formal publications.

Analytical Framework

We adopted a qualitative thematic analysis approach to integrate findings across diverse sources systematically. Initially, individual papers and reports were reviewed to identify common themes related to user experiences, interface interactions, learning outcomes, productivity impacts, and HCI considerations. These themes were organized by user group: novices, expert developers, researchers, and educators, to capture nuanced interactions and outcomes.

Variations among sources were examined and reconciled through comparison and discussion in the relevant parts of the analysis. This iterative process focused on how findings converged or diverged across studies representing different user groups. Emphasis was placed on identifying recurring relationships between user experiences, interface characteristics, and learning or productivity outcomes. Through this comparison, broader trends and consistent patterns were identified, forming a thematic framework connecting evidence from user research with technical and design features of LLM-based programming assistants, providing a grounded basis for discussing tool design implications and future work directions.

LANDSCAPE OF LLM-BASED PROGRAMMING TOOLS

Comparative Summary

A comparative summary table is presented below to highlight key features and intended user audiences of prominent LLM-based programming tools:

Table 1: Comparative summary table.

Tool	Interface Model	Key Features	Target Audience
GitHub Copilot	Inline, Chat	General code generation, ease of use	IT professionals, developers, students
Amazon CodeWhisperer	Inline	Security scanning, vulnerability checks	AWS developers, technicians
Tabnine	Inline, Chat	Customizable models, privacy-focused	IT professionals, developers
Sourcegraph Cody	Inline, Chat	Deep repository context, codebase navigation	IT professionals, developers
Claude Code	Inline, Chat	Agent-style workflow, repository awareness	Developers, researchers, IT professionals
Academic Prototypes	Varied (Interactive, Chat)	Test-driven development, pedagogical guidance	Researchers, educators

Commercial Tools

Widely adopted commercial LLM-based programming tools include GitHub Copilot ("GitHub Copilot," 2025), Amazon CodeWhisperer ("CodeWhisperer," 2025), Tabnine ("Tabnine," 2025), Claude Code ("Claude Code," 2025), and Sourcegraph Cody ("Cody," 2025). These tools primarily offer inline code completion, chat-based interactions, and additional capabilities such as security scanning, reference tracking, and contextual code understanding. GitHub Copilot, for instance, provides inline suggestions and basic chat interactions that enhance coding speed and reduce context switching. Amazon CodeWhisperer integrates security scanning to warn users of potential vulnerabilities in suggested code, aiming to enhance code quality and safety. Tabnine offers customizable model selection and privacy-focused deployments, appealing to enterprise users who prioritize data security. Claude Code introduces agent-style workflow functions and repository awareness that support collaborative development and integration with version-control systems. Sourcegraph Cody emphasizes deep repository context, allowing developers to navigate and query large codebases efficiently.

Academic and Research Prototypes

Academic prototypes explore innovative and diverse interactions, including proactive AI assistance, test-driven code generation, and interactive explanations within notebook environments (Chen et al., 2024; Fakhoury et al., 2024; Mcnutt et al., 2023). These prototypes aim to push the boundaries of user interactions by with features like test case generation, enhanced debugging support, and pedagogically driven hints and explanations. Notable examples include systems integrated into Jupyter notebooks ("Project Jupyter," 2025) that assist data scientists by suggesting analytical next steps, facilitating iterative development, and reducing cognitive load during complex data-analysis tasks (Mcnutt et al., 2023).

USER EXPERIENCE FINDINGS

Novice Programmers and Students

Novice programmers frequently leverage LLM-based tools for immediate feedback, syntax correction, and learning support (Finnie-Ansley et al., 2022; Kazemitabaar et al., 2024a). Empirical studies indicate that these tools improve novices' ability to complete programming tasks independently and foster greater confidence in coding (Kazemitabaar et al., 2024a; Vaithilingam et al., 2022). However, novices often struggle to critically evaluate AI-generated suggestions, leading to potential over-reliance and superficial learning (Prather et al., 2024; Vaithilingam et al., 2022). Studies suggest that structured guidance, such as scaffolded learning prompts or integrated code-validation mechanisms, improves novice outcomes by reinforcing critical thinking and ensuring meaningful engagement with the programming concepts (Kazemitabaar et al., 2024a; Prather et al., 2024; Stamper et al., 2024).

Professional Developers

For professional developers, LLM-based tools enhance productivity by automating routine tasks, offering quick solutions to coding challenges, and reducing cognitive load (Barke et al., 2023; Liang et al., 2024; Weisz et al., 2025). Experienced users typically demonstrate greater proficiency in evaluating AI-generated suggestions, thus maintaining high standards of code quality (Barke et al., 2023). Professional developers express ongoing concerns about code correctness, security risks, and potential disruptions in workflow due to inaccurate suggestions (Dakhel et al., 2023; Sandoval et al., 2023). Tools that integrate deep contextual awareness and provide transparency in the reasoning behind code suggestions are particularly valued for reducing uncertainty and fostering appropriate trust (Heller, 2024; Sugi, 2024).

Professionals also report that LLM-based assistants reshape collaborative workflows by serving as an "always-available pair programmer" (Barke et al., 2023; Weisz et al., 2025). These assistants can free up cognitive resources, allowing developers to focus on higher-level design decisions while delegating simple code and repetitive tasks to the AI (Liang et al., 2024). However, several studies underscore that blind acceptance of AI-generated code introduces technical liabilities and creates subtle bugs that may be overlooked in early testing (Moradi Dakhel et al., 2023; Sandoval et al., 2023).

Another critical consideration for professionals is workflow alignment. Professional developers value tools that integrate with established version-control systems, continuous-integration and continuous delivery (CI/CD) pipelines, and security policies (Weisz et al., 2025). For example, CodeWhisperer's automated vulnerability detection resonates with teams prioritizing DevSecOps practices, while Tabnine's customizable deployment options appeal to organizations with strict intellectual property and privacy requirements ("CodeWhisperer," 2025; Heller, 2024). Claude Code further supports integration with version-control workflows, which benefits collaborative development environments.

Overall, professional developers view LLMs as augmentative rather than substitutive collaborators. They value tools that provide transparency, rationale, and insight into model reasoning. This underscores the importance of building interfaces that not only deliver accurate code but also provide explainable and accountable AI partnerships (Liang et al., 2024; Sugi, 2024).

Researchers

Researchers benefit uniquely from LLM-based tools, using them to facilitate experimental coding, streamline data analysis, and enhance teaching methodologies (Chen et al., 2024; Mcnutt et al., 2023). For researchers, assistants embedded in interactive environments such as Jupyter notebooks markedly improve productivity and analytical depth by suggesting next-step analyses and easing iterative exploration (Mcnutt et al., 2023). These tools allow researchers to digest information in a conversational style, asking

the LLM questions directly, instead of parsing through dense academic and technical readings.

In "Guidance for Researchers and Peer-Reviewers on the Ethical Use of LLMs in Scientific Workflows," Watkins discusses how large language models may assist researchers and editors in the peer review process by flagging grammatical or stylistic errors, improving clarity, and enhancing readability. This allows human reviewers to focus more on the substance of a manuscript. The article also cautions that LLM outputs can include errors or bias and emphasizes the importance of transparency, disclosure, and clear standards to protect confidentiality and uphold research integrity (Watkins, 2024). Conversely, LLMs can hinder the research process and are not recommend for fact-checking (Dierickx et al., 2024; Narayanan Venkit et al., 2025; Quelle and Bovet, 2024).

Educators

Educators adopt classroom-oriented systems to demonstrate coding concepts, automate formative assessment, and create interactive learning experiences that maintain student engagement (Kazemitabaar et al., 2024b; Lyu et al., 2024). Nevertheless, ethical issues, including academic integrity, equitable access, and appropriate attribution, remain pressing concerns that require clear institutional guidelines and ongoing oversight (Becker et al., 2023; Prather et al., 2024). Recent educator-centered research further highlights that many of these concerns extend beyond technical errors to encompass broader social and pedagogical harms, including shifts in student motivation and the erosion of teacher agency (Feng et al., 2025; Harvey et al., 2025).

Educators also face challenges because most LLM-based tools were not originally developed for educational use (Lieb and Goel, 2024), which can lead to negative outcomes when classroom needs are overlooked. At the same time, LLMs offer an aspirational goal of providing personalized tutoring for individual students (Becker et al., 2023; Lieb and Goel, 2024; Stamper et al., 2024), a capability that has long been out of reach. Recent work in Artificial Intelligence in Education further argues that LLM-based tutoring systems should be grounded in feedback theories from the learning sciences to support both effectiveness and equity in educational settings (Stamper et al., 2024).

DESIGN AND HCI INSIGHTS

Trust Calibration

Appropriate trust calibration is essential in LLM-based programming tools (Barke et al., 2023; Vaithilingam et al., 2022). Transparent explanations, uncertainty indicators, and clearly communicated limitations help users build accurate mental models of tool capabilities and prevent over-reliance or under-utilization (Ibrahim et al., 2025; Moradi Dakhel et al., 2023; Sugi, 2024).

Trust calibration is not only about preventing blind reliance but also about considering appropriate skepticism. Tools that provide rationales for their outputs, cite training data sources, or communicate uncertainty enable users

to engage critically with suggestions (Moradi Dakhel et al., 2023; Sugi, 2024; Vaithilingam et al., 2022). Studies have shown that when explanations accompany code recommendations, users are more likely to detect errors and are less likely to accept faulty completions (Liang et al., 2024; Sandoval et al., 2023).

Cognitive Load Management

Tools should reduce cognitive load through seamless integration into existing workflows and by providing context-aware suggestions (Barke et al., 2023; Mcnutt et al., 2023; Weisz et al., 2025). This involves minimizing interruptions and maintaining user engagement by anticipating and addressing user needs (Liang et al., 2024).

While reducing cognitive load is a clear advantage of LLM integration, poorly timed or overly detailed interventions can instead increase cognitive effort. Effective designs should incorporate adaptive interfaces that sense user intent and adjust assistance accordingly. For instance, concise completions are most effective during routine coding, while richer, explanatory dialogues support deeper learning in educational contexts (Liang et al., 2024; Mcnutt et al., 2023).

Interface Modalities

Combining inline suggestions with interactive chat-based explanations provides flexible interaction modalities that support diverse coding scenarios and user preferences (Barke et al., 2023; Heller, 2024; Sugi, 2024). Designers should ensure smooth transitions between these modes to preserve workflow continuity and minimize context-switching (Liang et al., 2024).

Balance Between Automation and User Control

Designs must strike a careful balance between automation and user control (Barke et al., 2023; Moradi Dakhel et al., 2023). Integrated verification features, such as auto-generated tests or security scans, and easily reversible overrides give developers continuous oversight and help ensure high-quality code outcomes (Fakhoury et al., 2024; Sandoval et al., 2023).

Developers consistently emphasize the need for oversight mechanisms. Features such as auto-generated unit tests (Fakhoury et al., 2024; Ouedraogo et al., 2024), built-in security scans ("CodeWhisperer," 2025; Sandoval et al., 2023), and "explain this suggestion" functions promote LLM accountability and ensure that automation complements the user rather than replaces human judgement. Future designs should move towards interaction models where humans remain the primary decision-makers and LLMs manage mechanical or repetitive subtasks.

DISCUSSION

The findings of this meta-analysis indicate that the impact of large language model (LLM)-based programming tools depends not just on what the tools can do, but how users integrate them into their work or learning processes. Across all user groups, a common pattern emerges: LLMs can either enhance

or undermine user performance based on the degree of reflection and critical engagement involved.

For novice programmers, these tools can be both supportive and risky. Many students rely on LLMs to check syntax or explore alternative approaches, which can build confidence and improve task completion (Kazemitabaar et al., 2024a). However, when students lean too heavily on the model to generate solutions, they lose opportunities for conceptual understanding and independent problem solving (Prather et al., 2024). Systems providing structured guidance or scaffolding show promise in promoting learning without removing challenge (Stamper et al., 2024). This design shift helps move LLMs from automatic solution providers to interactive learning partners that reinforce understanding and skill development.

For professional developers, the key challenge is integrating LLMs productively within established workflows effectively. Developers appreciate how LLMs can reduce repetitive coding tasks and help test new ideas (Weisz et al., 2025), but the most effective use comes when the LLM is treated as a collaborator rather than a shortcut. When developers review, verify, and adapt model-generated suggestions instead of adopting them immediately, the AI becomes a second reviewer rather than an unquestioned authority (Barke et al., 2023). Future features such as explanations and testing prompts can further align these assistants with professional standards and collaborative team practices (Fakhoury et al., 2024).

For researchers and educators, LLMs open new opportunities for teaching and collaboration. Researchers benefit from assistants embedded in analytical environments that support iteration and reproducibility, although continued attention to bias and data integrity remains essential (Mcnutt et al., 2023; Watkins, 2024). Educators, meanwhile, face a related challenge in guiding students towards responsible and meaningful use. Best results occur when instructors describe these systems as learning aids that promote reasoning and reflection rather than as tools for generating direct answers (Harvey et al., 2025; Kazemitabaar et al., 2024b).

Overall, findings across user groups emphasize that the effectiveness of LLMs is determined not by automation but by how individuals choose to engage it. Future systems should encourage user curiosity, explanation, and reflection, allowing users to benefit from the AI's strengths while maintaining agency and accountability (Liang et al., 2024). Ultimately, LLMs are not replacements for human reasoning, but they can enhance it. When used cautiously and correctly, learning, creativity, and productivity can be elevated across all levels of expertise. The future challenge is to design and use LLMs in ways that strengthen, rather than erode the skills that make human programming effective in the first place.

CONCLUSION

This meta-analysis offers a broad perspective on the current state and impacts of LLM-based programming tools, synthesizing empirical insights, detailed tool evaluations, and key HCI considerations. We identified clear benefits across user groups, including improved productivity, enhanced learning opportunities, and streamlined research and teaching processes. However, significant challenges remain concerning code correctness, managing cognitive load, and appropriately calibrating user trust.

Future research should prioritize longitudinal studies that examine the long-term effects of LLM-tool use on programming skills and productivity, detailed assessments of collaborative coding environments that integrate AI assistance, and deeper investigations into ethical and legal implications. Additionally, advancing design strategies that tailor tools to specific user needs, enhance contextual understanding, and provide greater transparency in output generated by AI systems will be crucial for maximizing user benefits and minimizing potential drawbacks. Pursuing these research avenues will help refine LLM-based programming assistants and ensure they better support developers, researchers, and educators across evolving professional and educational contexts.

REFERENCES

Barke, S., James, M. B., Polikarpova, N., 2023. Grounded Copilot: How Programmers Interact with Code-Generating Models. Proc. ACM Program. Lang. 7, 78:85-78:111. https://doi.org/10.1145/3586030

Becker, B. A., Denny, P., Finnie-Ansley, J., Luxton-Reilly, A., Prather, J., Santos, E. A., 2023. Programming Is Hard - Or at Least It Used to Be: Educational Opportunities and Challenges of AI Code Generation, in: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1, SIGCSE 2023. Association for Computing Machinery, New York, NY, USA, pp. 500–506. https://doi.org/10.1145/3545945.3569759

Chen, J., Lu, X., Du, Y., Rejtig, M., Bagley, R., Horn, M., Wilensky, U., 2024. Learning Agent-based Modeling with LLM Companions: Experiences of Novices and Experts Using ChatGPT & NetLogo Chat, in: Proceedings of the 2024 CHI Conference on Human Factors in Computing Systems, CHI '24. Association for Computing Machinery, New York, NY, USA, pp. 1–18. https://doi.org/10.1145/3613904.3642377

Claude Code [WWW Document], 2025. URL: https://claude.ai.

CodeWhisperer is becoming a part of Amazon Q Developer - CodeWhisperer [WWW Document], 2025. URL: https://docs.aws.amazon.com/codewhisperer/latest/userguide/whisper-legacy.html.

Cody | AI coding assistant from Sourcegraph [WWW Document], 2025. URL: https://sourcegraph.com/cody.

Dierickx, L., Van Dalen, A., Opdahl, A. L., Lindén, C.-G., 2024. Striking the Balance in Using LLMs for Fact-Checking: A Narrative Literature Review, in: Preuss, M., Leszkiewicz, A., Boucher, J.-C., Fridman, O., Stampe, L. (Eds.), Disinformation in Open Online Media, Lecture Notes in Computer Science. Springer Nature Switzerland, Cham, pp. 1–15. https://doi.org/10.1007/978-3-031-71210-4_1

Fakhoury, S., Naik, A., Sakkas, G., Chakraborty, S., Lahiri, S. K., 2024. LLM-Based Test-Driven Interactive Code Generation: User Study and Empirical Evaluation. IEEE Transactions on Software Engineering 50, 2254–2268. https://doi.org/10.1109/TSE.2024.3428972

Feng, Y., Zhao, T., Zu, Y., Tavares, A., Gomes, T., Xu, H., 2025. An Explorative Investigation into Leveraging LLMs to Predict University Students' Learning Motivation, in: Proceedings of the Annual Meeting of the Cognitive Science Society.

Finnie-Ansley, J., Denny, P., Becker, B. A., Luxton-Reilly, A., Prather, J., 2022. The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming, in: Proceedings of the 24th Australasian Computing Education Conference, ACE '22. Association for Computing Machinery, New York, NY, USA, pp. 10–19. https://doi.org/10.1145/3511861.3511863

GitHub Copilot · Your AI pair programmer [WWW Document], 2025. GitHub. URL: https://github.com/features/copilot.

Harvey, E., Koenecke, A., Kizilcec, R. F., 2025. "Don't Forget the Teachers": Towards an Educator-Centered Understanding of Harms from Large Language Models in Education, in: Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems. Presented at the CHI 2025: CHI Conference on Human Factors in Computing Systems, ACM, Yokohama Japan, pp. 1–19. https://doi.org/10.1145/3706598.3713210

Heller, M., 2024. Review: Tabnine AI coding assistant flexes its models. InfoWorld. URL: https://www.infoworld.com/article/3484857/review-tabnine-ai-coding-assistant-flexes-its-models.html.

Ibrahim, L., Collins, K. M., Kim, S. S. Y., Reuel, A., Lamparth, M., Feng, K., Ahmad, L., Soni, P., Kattan, A. E., Stein, M., Swaroop, S., Sucholutsky, I., Strait, A., Liao, Q. V., Bhatt, U., 2025. Measuring and mitigating overreliance is necessary for building human-compatible AI. https://doi.org/10.48550/arXiv.2509.08010

Introducing ChatGPT [WWW Document], 2024. URL: https://openai.com/index/chatgpt/.

Kazemitabaar, M., Hou, X., Henley, A., Ericson, B. J., Weintrop, D., Grossman, T., 2024a. How Novices Use LLM-based Code Generators to Solve CS1 Coding Tasks in a Self-Paced Learning Environment, in: Proceedings of the 23rd Koli Calling International Conference on Computing Education Research, Koli Calling '23. Association for Computing Machinery, New York, NY, USA, pp. 1–12. https://doi.org/10.1145/3631802.3631806

Kazemitabaar, M., Ye, R., Wang, X., Henley, A. Z., Denny, P., Craig, M., Grossman, T., 2024b. CodeAid: Evaluating a Classroom Deployment of an LLM-based Programming Assistant that Balances Student and Educator Needs, in: Proceedings of the CHI Conference on Human Factors in Computing Systems. pp. 1–20. https://doi.org/10.1145/3613904.3642773

Liang, J. T., Yang, C., Myers, B. A., 2024. A Large-Scale Survey on the Usability of AI Programming Assistants: Successes and Challenges, in: Proceedings of the IEEE/ACM 46th International Conference on Software Engineering, ICSE '24. Association for Computing Machinery, New York, NY, USA, pp. 1–13. https://doi.org/10.1145/3597503.3608128

Lieb, A., Goel, T., 2024. Student Interaction with NewtBot: An LLM-as-tutor Chatbot for Secondary Physics Education, in: Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, CHI EA '24. Association for Computing Machinery, New York, NY, USA, pp. 1–8. https://doi.org/10.1145/3613905.3647957

Lyu, W., Wang, Y., Chung, T. (Rachel), Sun, Y., Zhang, Y., 2024. Evaluating the Effectiveness of LLMs in Introductory Computer Science Education: A Semester-Long Field Study, in: Proceedings of the Eleventh ACM Conference on Learning @ Scale, L@S '24. Association for Computing Machinery, New York, NY, USA, pp. 63–74. https://doi.org/10.1145/3657604.3662036

Mcnutt, A. M., Wang, C., Deline, R. A., Drucker, S. M., 2023. On the Design of AI-powered Code Assistants for Notebooks, in: Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems, CHI '23. Association for Computing Machinery, New York, NY, USA, pp. 1–16. https://doi.org/10.1145/3544548.3580940

Moradi Dakhel, A., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C., Jiang, Z. M. (Jack), 2023. GitHub Copilot AI pair programmer: Asset or Liability? J. Syst. Softw. 203. https://doi.org/10.1016/j.jss.2023.111734

Narayanan Venkit, P., Laban, P., Zhou, Y., Mao, Y., Wu, C.-S., 2025. Search Engines in the AI Era: A Qualitative Understanding to the False Promise of Factual and Verifiable Source-Cited Responses in LLM-based Search, in: Proceedings of the 2025 ACM Conference on Fairness, Accountability, and Transparency. Presented at the FAccT '25: The 2025 ACM Conference on Fairness, Accountability, and Transparency, ACM, Athens Greece, pp. 1325–1340. https://doi.org/10.1145/3715275.3732089

OpenAI Codex [WWW Document], 2025. URL: https://openai.com/codex/.

Ouedraogo, W. C., Kabore, K., Tian, H., Song, Y., Koyuncu, A., Klein, J., Lo, D., Bissyande, T. F., 2024. LLMs and Prompting for Unit Test Generation: A Large-Scale Evaluation, in: Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering. Presented at the ASE '24: 39th IEEE/ACM International Conference on Automated Software Engineering, ACM, Sacramento CA USA, pp. 2464–2465. https://doi.org/10.1145/3691620.3695330

Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., Kimmel, B., Wright, J., Briggs, B., 2024. The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers, in: Proceedings of the 2024 ACM Conference on International Computing Education Research - Volume 1, ICER '24. Association for Computing Machinery, New York, NY, USA, pp. 469–486. https://doi.org/10.1145/3632620.3671116

Project Jupyter [WWW Document], 2025. URL: https://jupyter.org.

Quelle, D., Bovet, A., 2024. The perils and promises of fact-checking with large language models. Frontiers in Artificial Intelligence 7, 1341697.

Sandoval, G., Pearce, H., Nys, T., Karri, R., Garg, S., Dolan-Gavitt, B., 2023. Lost at C: a user study on the security implications of large language model code assistants, in: Proceedings of the 32nd USENIX Conference on Security Symposium, SEC '23. USENIX Association, USA, pp. 2205–2222.

Stamper, J., Xiao, R., Hou, X., 2024. Enhancing LLM-Based Feedback: Insights from Intelligent Tutoring Systems and the Learning Sciences, in: Olney, A. M., Chounta, I.-A., Liu, Z., Santos, O. C., Bittencourt, I. I. (Eds.), Artificial Intelligence in Education. Posters and Late Breaking Results, Workshops and Tutorials, Industry and Innovation Tracks, Practitioners, Doctoral Consortium and Blue Sky. Springer Nature Switzerland, Cham, pp. 32–43. https://doi.org/10.1007/978-3-031-64315-6_3

Sugi, Y., 2024. The anatomy of an AI coding assistant [WWW Document]. URL: https://sourcegraph.com/blog/anatomy-of-a-coding-assistant.

Tabnine AI Code Assistant | private, personalized, protected [WWW Document], 2025.. Tabnine. URL: https://www.tabnine.com/.

Vaithilingam, P., Zhang, T., Glassman, E. L., 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models, in: Extended Abstracts of the 2022 CHI Conference on Human Factors in Computing Systems, CHI EA '22. Association for Computing Machinery, New York, NY, USA, pp. 1–7. https://doi.org/10.1145/3491101.3519665

Watkins, R., 2024. Guidance for researchers and peer-reviewers on the ethical use of Large Language Models (LLMs) in scientific research workflows. AI Ethics 4, 969–974. https://doi.org/10.1007/s43681-023-00294-5

Weisz, J. D., Kumar, S. V., Muller, M., Browne, K.-E., Goldberg, A., Heintze, K. E., Bajpai, S., 2025. Examining the Use and Impact of an AI Code Assistant on Developer Productivity and Experience in the Enterprise, in: Proceedings of the Extended Abstracts of the CHI Conference on Human Factors in Computing Systems, CHI EA '25. Association for Computing Machinery, New York, NY, USA, pp. 1–13. https://doi.org/10.1145/3706599.3706670