Generative AI Assisting Programming Learning: Functional Roles, Comparative Analysis, and Emerging Challenges

Heloise Acco Tives Bedin[1], **Madianita Bogo Marioti**[2], **Patricia Jaques**[1]

[1]Federal University of Paraná (UFPR) – Curitiba – PR – Brazil

[2]Lutheran University Center of Palmas (CEULP/ULBRA) – TO – Brazil

heloise.acco@ufpr.br, madianitab@gmail.com patricia@inf.ufpr.br

Abstract. *This study examines the role of generative Artificial Intelligence (AI) in programming education through a systematic literature mapping of 52 studies across seven databases. Our analysis reveals three distinct functional roles for AI: collaborative participant, support tool, and pedagogical mediator. We compare AI-enhanced programming instruction with traditional teaching methods, finding shifts in pedagogical approaches, communication, and cognitive engagement. The research identifies key challenges spanning technical, pedagogical, ethical, and assessment dimensions. We propose a functional classification framework for AI roles and offer recommendations for responsible AI integration in programming education.*

Resumo. *Este trabalho examina, por meio de um mapeamento sistemático da literatura, como a Inteligência Artificial (IA) generativa tem apoiado a aprendizagem de programação. A análise de 52 estudos em sete bases de dados revelou três funções principais da IA: participante colaborativo, ferramenta de suporte e mediador pedagógico. O estudo compara abordagens assistidas por IA com métodos tradicionais, identificando transformações nos modelos pedagógicos, comunicação e engajamento cognitivo. A pesquisa mapeia desafios técnicos, pedagógicos, éticos e avaliativos. Propõe-se uma classificação funcional para os papéis da IA e recomendações para integração responsável no ensino de programação.*

1. Introduction

Research on generative Artificial Intelligence use in computing education has intensified since 2023 [Becker et al. 2023]. These solutions, based on Large Language Models (LLMs), employ Natural Language Processing (NLP) techniques grounded in deep learning, enabling them to autonomously learn grammar, semantics, and pragmatics, as well as to generate a wide variety of content [Zhou et al. 2025].

In programming education, this technological convergence presents unique characteristics. The increasing adoption of generative AI chatbots, such as ChatGPT, and code assistants, such as GitHub Copilot, has been reshaping the landscape of programming education [Vadaparty et al. 2025], transforming how people learn to program and how educators teach introductory computer science courses [Puryear and Sprint 2022]. A transition has emerged from traditional, human-only programming practices to interactions mediated by AI, in which humans and AI systems collaborate on programming

DOI: 10.5753/sbie.2025.12361

tasks [Ma et al. 2023]. This rapid technological incorporation calls for research into its specific pedagogical impacts.

The literature has addressed this phenomenon using a range of terminology that reflects different perspectives on usage. For task-specific support, terms such as AI-assisted programming [Barke et al. 2023], AI-aided programming [Paludo and Montresor 2024], and AI-supported programming education [Yilmaz and Karaoglan Yilmaz 2023] are found. For sustained collaborative interactions, expressions such as AI pair programming [Bird et al. 2022, Chen 2024], human–AI pair programming [Ma et al. 2023, Jiang et al. 2025], AI-powered pair programmer [Al Madi 2023], and AI programming partner [Kuttal et al. 2021] are commonly used. In both contexts, AI offers suggestions, refines solutions, and, in some cases, adapts to the task's context. Despite the differences, these approaches converge in recognizing AI as a mediator in the learning process, with the potential to transform pedagogical and cognitive practices.

Despite significant research growth, the literature reveals important gaps in the investigation of the specific functions that AI assumes in programming education, as well as in the structured comparison of its pedagogical impacts in relation to traditional approaches. Studies that systematically map these aspects and identify emerging challenges are essential to inform evidence-based educational practices. To address these gaps, this study, conducted through a systematic literature mapping (SLM), investigates how generative AI has been applied in programming education, focusing on the identification of its functions, comparison with traditional approaches, and analysis of the main challenges.

To this end, Section 2 presents related work and outlines the originality of this study; Section 3 describes the methodological protocol, including selection criteria, search strategies, and data extraction procedures; Section 4 organizes the findings around the three research questions, discussing their implications and limitations; finally, Section 5 summarizes the study's contributions and offers recommendations for advancing the field.

2. Related Work

The application of generative AI in programming education has attracted growing interest; however, studies focusing on AI functions in collaboration, comparisons with traditional approaches, and specific challenges remain limited. Silva et al. [Silva et al. 2024] conducted a mapping study focused on educational experiences involving generative AI, classifying the studies into five distinct categories and identifying potential benefits such as automated feedback, support for problem-solving, and personalized examples, while also pointing out challenges related to the variability of AI responses.

Manorat et al. [Manorat et al. 2025] conducted a review on the use of AI and machine learning in higher education programming courses, covering intelligent tutoring systems, automated code grading, recommendation engines, and learning support systems. The study presents a functional taxonomy of the systems analyzed and emphasizes the need for further research into cognitive and pedagogical aspects. Liu and Li [Liu and Li 2024] organized a thematic review exploring automatic code generation tools applied to education, highlighting four key dimensions: technical, pedagogical, usability, and ethical. The authors propose a framework to evaluate these tools based on criteria such as adaptability, explainability, and safety.

This study distinguishes itself as a systematic mapping focused on AI-assisted programming with generative AI, proposing a functional taxonomy—collaborative participant, support tool, and pedagogical mediator—and offering a structured comparative analysis between traditional and AI-mediated approaches. Additionally, the study systematizes the main challenges in the field, organized into six categories: technical, pedagogical, communicational, ethical, institutional, and assessment-related. The findings provide evidence of how AI reconfigures traditional collaborative dynamics, contributing to the advancement of future research and the development of responsible educational practices.

3. Methods

A Systematic Mapping Study (SMS), also referred to as a scoping study, is a comprehensive review of primary studies in a specific area, aimed at identifying available evidence on a topic [Kitchenham et al. 2007] and determining whether there are subtopics that require further investigation [Felizardo et al. 2017].

The SLM protocol was adapted from the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) model, originally developed for systematic reviews on the effects of interventions in the healthcare field. However, the model is also recommended for systematic reviews in other areas, provided they involve mixed methods that include quantitative and qualitative studies [Page et al. 2021]. The adoption of the PRISMA protocol aims to ensure transparency, methodological rigor, and reproducibility in conducting this investigation.

Based on the need to deepen understanding of how generative AI has been used in programming education and what current challenges exist in this field, three research questions were defined: **RQ1:** What are the main functions of generative AI systems in programming education? **RQ2:** How does literature compare traditional collaborative programming with AI-assisted approaches? **RQ3:** What are the main challenges identified in the literature concerning programming education assisted by generative AI?

This set of questions was designed to support discussions on the transformations introduced by the use of generative AI in programming learning processes. These three questions enable a focused mapping both on identifying the roles currently played by generative AI and on the pedagogical changes resulting from its adoption, in contrast to traditional practices, while describing the main obstacles to be overcome for advancement of the field.

To support the selection and quality of analyzed evidence, specific inclusion and exclusion criteria were applied. To be selected, a study should explicitly address the use of generative AI in programming learning. Works unrelated to the central theme, publications in languages other than Portuguese and English, studies published before 2020, publications that were not complete articles, and duplicate works were excluded. Although systematic mappings traditionally prioritize peer-reviewed publications, the inclusion of the ArXiv repository was justified by the emerging nature of the field, where relevant studies are first made available as preprints and leading journals accept submissions previously available on ArXiv, ensuring broader coverage of the most recent studies with the same quality criteria applied to other publications. The 5-year timeframe (2020-2025), more restrictive than usual in systematic mappings (10-15 years), was established considering the emerging nature of generative AI applied to programming.

The search for publications was conducted across seven databases relevant to the fields of Computer Education and Artificial Intelligence: ACM DL, DBLP, IEEE Xplore, Science Direct, Springer, SOL, and ArXiv. The search was configured by adapting the search string[1] to available parameters in each database's search options, adjusting syntax to accommodate specific requirements and operators of each platform while preserving the structure of essential elements. The specific string adaptations for each database are documented in the Supplementary Material[2] to ensure research transparency and reproducibility.

The selection process was conducted independently by two researchers, following sequential filters. In the first filter, titles and abstracts were analyzed to verify exclusion criteria. In the second filter, articles were read to verify inclusion criteria and relevance to research questions. The Porifera tool was used to assist in constructing the search string, defining criteria, importing results from databases, and recording filtering decisions. The results of the applied filters are also detailed in the Supplementary Material.

Information extraction was supported by the Google NotebookLM tool for preliminary identification of relevant elements in the texts. The extraction prompts were designed, tested, and iteratively refined, with the final prompts used and raw results documented in the Supplementary Material. The quality criterion established was that the volume of relevant information automatically extracted should exceed 90% of the content obtained through complete reading and manual extraction of a set of 8 papers (approximately 15% of the selected studies). To ensure that critical information was not omitted, the researchers conducted manual verification of the selected papers, complementing and correcting the automatic extractions when necessary.

4. Results

The systematic search across the seven selected databases identified a total of 495 publications. The distribution of works by database was: Scopus with 236 studies (47.7%), DBLP with 97 (19.6%), ScienceDirect with 77 (15.6%), IEEE with 34 (6.9%), arXiv with 19 (3.8%), ACM with 18 (3.6%), and SOL with 14 (2.8%). After removing 86 duplicates (17.37% of the initial total), 409 unique publications remained for analysis.

Based on the inclusion and exclusion criteria defined in Section 3, the execution of the first filter (title and abstract analysis) rejected 305 studies (61.62% of the initial total), leaving 104 publications considered potentially relevant. In the second filter, after in-depth reading of the works, an additional 52 studies were excluded (10.51% of the initial total), resulting in 52 publications selected for final analysis (10.51% of the initial total identified). Information extraction was supported by the Google NotebookLM tool, with the prompts used and raw extraction results presented in the Supplementary Material, and the 52 selected works are available in a public folder[3]. The following sections present the

[1]The base search string used was: ((“programming” AND (“pair” OR “collaborative” OR “in teams”) OR “social coding”)) AND (“artificial intelligence” OR “AI” OR “generative” OR “large language model” OR “LLM” OR “code assistant” OR “automatic code completion” OR “code completion”) AND (“teaching” OR “learning” OR “education” OR “instruction” OR “computer education”).

[2]Supplementary Material available at: `https://encurtador.com.br/umdQ5`

[3]Folder with the 52 selected works available at: `https://drive.google.com/drive/folders/1oxLii0LC6VVlH3zZvqJuYeNIWMbL8XcW?usp=sharing`

responses developed for each of the three research questions that guided this systematic mapping.

4.1. What are the main functions of generative AI systems in programming education?

The analysis of the selected studies identified three main functions performed by generative AI in programming education: **Collaborative participant** — AI systems that actively participate in programming tasks, assuming driver/navigator roles [Williams et al. 2000] and engaging in continuous dialogue; **Support tool** — AI systems activated on-demand to provide specific assistance such as code generation or debugging help, without continuous interaction; and **Pedagogical mediator** — AI systems designed to facilitate learning through adaptive scaffolding, progress monitoring, and personalized feedback.

Table 1 presents the consolidated classification of the studies according to the function performed, as described above. The first column indicates the AI function category and the number of SMS studies assigned to each category, while the second column lists the references of the studies classified under each function.

Table 1. Distribution of studies according to identified AI functions in programming education

Classification (count)	Studies
AI as a collaborative participant in pair programming (23)	[Al Madi 2023, Imai 2022, Robe et al. 2022, Manfredi et al. 2023, Vadaparty et al. 2025, Domingo 2023, Chen 2024, Bird et al. 2022, Górecki 2024, Zhang et al. 2024, Hassany et al. 2024, Ma et al. 2024, Zhang et al. 2023, Dos Santos and Cury 2023, Kuttal et al. 2021, Ma et al. 2023, Valový 2023, Wang et al. 2025, Valový and Buchalcevova 2023, Robe and Kuttal 2022, Wei et al. 2024, Jiang et al. 2025, Kazemitabaar et al. 2023]
AI as a support tool (23)	[Phung et al. 2023, Piccolo et al. 2023, Carvalho and Oliveira 2024, Asare et al. 2023, Puryear and Sprint 2022, Barke et al. 2023, Dibia et al. 2023, Banić et al. 2023, Rao et al. 2024, Simaremare et al. 2024, Moradi Dakhel et al. 2023, Yilmaz and Karaoglan Yilmaz 2023, Groothuijsen et al. 2024, Amoozadeh et al. 2024, De Silva et al. 2024, Bassner et al. 2024, Nguyen and Nadi 2022, Jiang et al. 2023, Chan et al. 2023, Rasnayaka et al. 2024, Feng et al. 2024a, Sarkar et al. 2022, Zhou et al. 2025]
AI as a pedagogical mediator (6)	[Norton et al. 2024, Zhang et al. 2025, Yang et al. 2025, Paludo and Montresor 2024, Tang et al. 2024, Feng et al. 2024b]

Figure 1 provides an integrated view of the main challenges, benefits, and specific characteristics associated with each function, offering a holistic perspective on the pedagogical potential and limitations of each identified approach.

Analysis of the selected studies characterizes each function based on methodological approaches, technologies employed, and observed learning outcomes.

Studies classifying **AI as a collaborative participant** mostly describe systems developed by the researchers themselves, often integrating LLMs through commercial *Application Programming Interfaces* (APIs), such as GPT. These systems were designed to operate in a continuous, adaptive, and dialogic manner throughout the programming activity.

The functions performed include: (i) automatic code sharing and regeneration based on the collaborative context [Feng et al. 2024a]; (ii) acting as a virtual driver or navigator in immersive environments [Górecki 2024, Manfredi et al. 2023]; (iii) executing tasks of explanation, correction, and validation in iterative debugging cycles [Ma et al. 2024]; (iv) simulating human pairing dynamics through structured dialogue

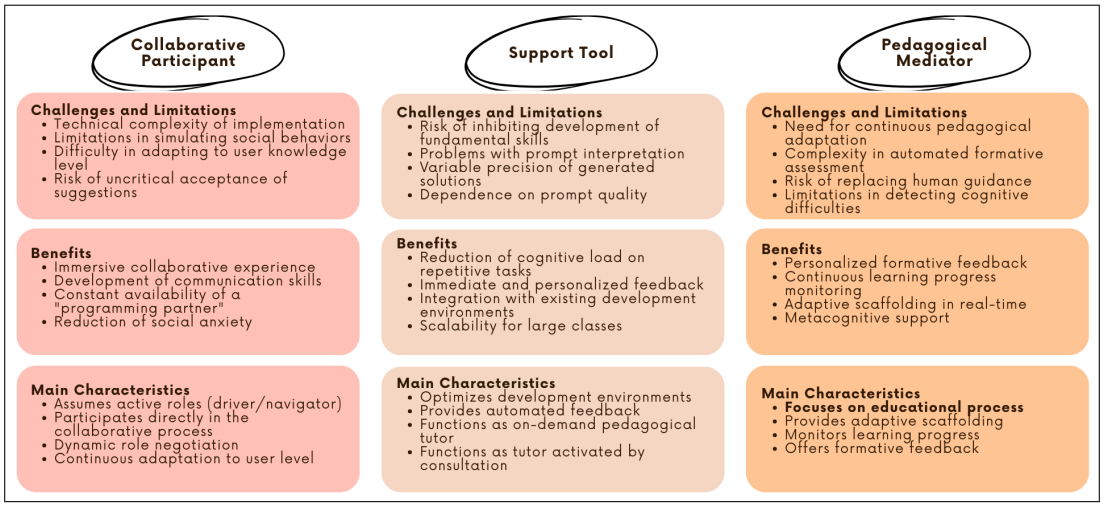

Figure 1. Functions performed by generative AI in programming education

[Robe and Kuttal 2022, Jiang et al. 2025]; and (v) offering continuous real-time code suggestions [Bird et al. 2022, Imai 2022].

The learning impacts include improved productivity, with a higher number of lines of code produced [Imai 2022], and reduced social anxiety compared to working with human peers [Kuttal et al. 2021]. Comparative studies indicate that students paired with AI achieved performance levels equivalent to those of human pairs [Robe et al. 2022], although some reported a lack of dynamic engagement, describing the experience of working with AI as “talking to our own thoughts” [Simaremare et al. 2024].

Studies describing **AI as a support tool** characterize its use as occasional, reactive, and centered on resolving doubts, providing conceptual explanations, or generating code on demand. In this category, AI is triggered by the student at specific moments during the programming activity, without maintaining continuous interaction.

The functions include: (i) code generation based on natural language descriptions through specific prompts [Puryear and Sprint 2022, Nguyen and Nadi 2022]; (ii) offering suggestions such as intelligent autocompletion or exploration of alternatives [Barke et al. 2023]; (iii) automatic creation of exercises and educational content [Chan et al. 2023]; (iv) automated debugging triggered after compilation failures [De Silva et al. 2024]; and (v) explanation of concepts and provision of code snippets [Yilmaz and Karaoglan Yilmaz 2023, Groothuijsen et al. 2024].

GitHub Copilot is the most widely studied tool, integrated into environments such as VS Code to provide real-time suggestions. Studies reported that it was capable of solving most introductory tasks [Puryear and Sprint 2022]. The impacts include reduced time to complete tasks [Rasnayaka et al. 2024], improved algorithmic thinking by freeing time from routine tasks [Yilmaz and Karaoglan Yilmaz 2023], and enhanced conceptual understanding [Groothuijsen et al. 2024]. However, some studies identified risks of overdependence and limitations in adapting to specific contexts [Moradi Dakhel et al. 2023].

Studies that classify **AI as a pedagogical mediator** describe systems specifically designed to facilitate the learning process through adaptive *scaffolding*, which consists of

structured and temporary support tailored to the student's level, progress monitoring, and personalized formative *feedback*, with an emphasis on developing cognitive and metacognitive skills.

The functions include: (i) providing contextualized hints and guidance based on the current state of the code [Yang et al. 2025]; (ii) facilitating the process through the five stages of computational thinking [Zhang et al. 2025]; (iii) promoting reflective learning through critical editing of generated code [Paludo and Montresor 2024]; (iv) offering personalized feedback at multiple levels of difficulty [Norton et al. 2024]; and (v) delivering tutoring adapted to the specific context of the course [Feng et al. 2024b].

The reported impacts indicate promising outcomes in metacognitive development. Students exhibited fewer off-task behaviors, increased focus on iterative refinement, and broader exploration of cognitive dimensions [Zhang et al. 2025]. The RAP Lab significantly enhanced metacognitive awareness and deepened students' understanding of problems [Paludo and Montresor 2024], while CourseAssist stood out by providing responses considered more useful, accurate, and pedagogically appropriate than those generated by generic models such as GPT-4 [Feng et al. 2024b].

The diversity of studies shows that, although the support tool function still predominates in the literature, an increasing number of proposals explore AI as a collaborative partner in the coding process. At the same time, a third perspective is emerging that positions AI as a pedagogical mediator, focusing on the educational process and the development of metacognitive skills. This evolution suggests a hybrid learning ecosystem, where different AI approaches may be integrated in a complementary way to address the diverse needs of programming teaching and learning.

4.2. How does literature compare traditional collaborative programming with AI-assisted approaches?

The analyzed literature provides evidence for comparing traditional collaborative programming practices (pair programming) and those mediated by generative artificial intelligence. Most selected studies do not perform direct or systematic comparisons between these approaches. Nevertheless, available descriptions allow for identification of differences in learning experiences.

To systematize this comparative analysis, the observed aspects were organized into six categories: (1) organizational aspects, (2) pedagogical aspects, (3) socio-emotional aspects, (4) cognitive engagement and metacognitive strategies, (5) methodological characteristics of studies, and (6) communication. These categories emerged from a hybrid approach that combined existing frameworks from the literature with inductive analysis of the collected data. The technical, pedagogical, and ethical dimensions were initially inspired by the framework proposed by Liu and Li [Liu and Li 2024], while the communication and socio-emotional categories were identified during thematic analysis of the primary studies, reflecting specific characteristics of generative AI tools in the educational context. These categories are detailed in Table 2, which summarizes the main contrasts between the two approaches.

Regarding the **organizational aspects**, the main change is the elimination of the need for coordination between two students. Systems such as *Generative Co-Learners* (GCL) use AI agents to act as co-learners in asynchronous environments

Table 2. Comparison between traditional pair programming and AI-assisted programming, organized into six analytical categories

Category	Traditional	Assisted by Generative AI
Organizational Aspects		
Pair formation	Pairing between students based on affinity, performance, or availability [Robe and Kuttal 2022].	Elimination of the need for a human partner through autonomous agents [Manfredi et al. 2023]; role simulation and AI pairing in immersive environments [Górecki 2024].
Interaction environment	In-person or remote environments with synchronous tools [Robe and Kuttal 2022]; teacher mediation and direct interactions [Robe et al. 2022].	Examples include asynchronous environments with AI integrated into IDEs [Imai 2022]; presence of simulated co-learners [Wang et al. 2025]; interaction via avatars in mixed reality (MR) [Górecki 2024].
Pedagogical Aspects		
Learning outcomes	Peer collaboration fosters mutual learning and code quality [Banić et al. 2023].	Error reduction [Puryear and Sprint 2022]; significant initial learning gains [Sarkar et al. 2022]; exposure to new languages and libraries [Bird et al. 2022].
Role distribution	Alternation between driver and navigator, with active contribution from both [Al Madi 2023].	AI mostly assumes the role of driver [Bird et al. 2022]; the human acts as navigator, evaluating and reformulating prompts [Paludo and Montresor 2024].
Socio-emotional Aspects		
Engagement and collaboration	Peer interaction fosters idea exchange and emotional engagement [Robe and Kuttal 2022]; Hawthorne effect observed [Valový and Buchalcevova 2023].	Students report lower emotional involvement [Dibia et al. 2023]; tendency to uncritically accept AI suggestions [Barke et al. 2023]; perception of isolation and loss of empathy [Dos Santos and Cury 2023]; reduced peer collaboration [Groothuijsen et al. 2024].
Cognitive Engagement and Metacognitive Strategies		
Reflection and critical thinking	Peer discussions promote reasoning articulation and mutual review [Ma et al. 2023].	Prompt engineering as a reflective strategy [Paludo and Montresor 2024]; analysis of AI suggestions and debugging of errors [Ma et al. 2024]; potential for personalized support [Zhang et al. 2025].
Methodological Characteristics of the Studies		
Research approaches	Qualitative studies focused on natural language-mediated interactions and observation of collaborative behaviors [Al Madi 2023, Imai 2022].	Investigation of interaction flow with automated agents and validation criteria for AI suggestions [Barke et al. 2023, Dibia et al. 2023]; strategies such as debugging and prompt engineering [Ma et al. 2024].
Communication		
Communication style	Direct interaction-based communication, with greater fluency in conceptual discussions [Robe and Kuttal 2022].	More explicit and structured communication, adapted to AI perception [Robe et al. 2022]; focus on clearly formulated prompts and objectives [Dibia et al. 2023]; reduced mutual conceptual negotiation [Ma et al. 2023]; limitations in joint meaning-making [Groothuijsen et al. 2024].

[Wang et al. 2025], while mixed reality approaches allow for the complete replacement of the physical presence of partners [Manfredi et al. 2023]. This flexibility increases accessibility and enhances the scalability of the practice.

In terms of **pedagogical aspects**, empirical evidence shows a shift from the "learn-to-code" model to a "code-to-learn" paradigm. While traditional programming approaches typically require understanding the problem before writing code, in AI-assisted interactions, students often begin by generating code automatically and then seek to understand its functionality afterward [Valový and Buchalcevova 2023]. Additionally, there is a transformation in the nature of the predominant cognitive task: the activity of code creation (*Create*), typical of traditional approaches, is replaced by tasks involving the use and modification of existing code (*Use-Modify*), enabling a greater focus on analysis and comprehension, but also demanding distinct skills in critical reading and debugging [Kazemitabaar et al. 2023].

In the **socio-emotional aspects**, studies reveal that participants felt that pairing students with generative AIs lacked dynamism and did not foster engagement

[Simaremare et al. 2024], due to the absence of genuine collaboration between team members when AI is used as the navigator [Groothuijsen et al. 2024]. However, students reported ease of communication with the AI [Dos Santos and Cury 2023], suggesting potential benefits for less confident learners.

Cognitive engagement is transformed through prompt engineering as a metacognitive strategy, fostering deep reflection on one's own thinking and revealing the capabilities and limitations of AI-generated content [Paludo and Montresor 2024]. Evidence shows that correcting ChatGPT errors involves iterative questioning and promotes deeper cognitive processes [Zhang et al. 2025], in contrast to the spontaneous verbalization typical of traditional programming.

In terms of **communication**, there is a shift from implicit to explicit modes, where developers working with agents use a higher frequency of questions and clarifications, along with formal and explicit communication about role-switching—contrasting with the natural fluency of human interaction [Robe et al. 2022].

Thus, the evidence converges to show that traditional programming emphasizes discussions that support reasoning articulation, with cooperative behavior being key to success [Ma et al. 2023]. AI-assisted programming promotes autonomy and flexibility but shifts the core activity toward verifying and debugging generated code [Sarkar et al. 2022], requiring new mediation strategies and critical integration of the technology into the educational process.

4.3. What are the main challenges identified in the literature regarding AI-assisted programming education?

By analyzing the challenges reported in the literature on AI-assisted programming education, it becomes possible to guide efforts toward overcoming them and enhancing the benefits of these technologies. Six main categories were defined to organize these challenges: (1) technical, (2) pedagogical (including cognitive aspects), (3) communication and interaction, (4) ethical and social, (5) institutional and implementation-related, and (6) assessment-related. A summary of these challenges is presented in Figure 2.

Several studies highlight **technical** limitations, such as inconsistencies in AI behavior and the generation of faulty code. GPT-4, for instance, tends to make extensive changes to code, not always respecting the original context provided by the student [Phung et al. 2023]. Problems have been identified with the integration of tools like Copilot into different *Integrated Development Environments* (IDEs), server instability, and difficulties in controlling the suggestions generated [Zhang et al. 2023]. These models may produce insecure or technically incorrect code without indicating uncertainty, requiring constant validation [Górecki 2024, Bird et al. 2022]. There is also low explainability of suggestions and an inability to handle complex or infrequent tasks [Chen 2024, Zhang et al. 2024].

In the **pedagogical** domain, the challenges encompass both instructional and cognitive aspects. Students have shown behaviors of passive acceptance of AI suggestions and a tendency toward overreliance, compromising their autonomy [Dibia et al. 2023]. Cognitively, long responses increase mental load and hinder the fluency of reasoning [Barke et al. 2023]. Difficulties in reviewing, understanding, or modifying the generated code represent additional barriers, especially among learners with limited prior knowl-

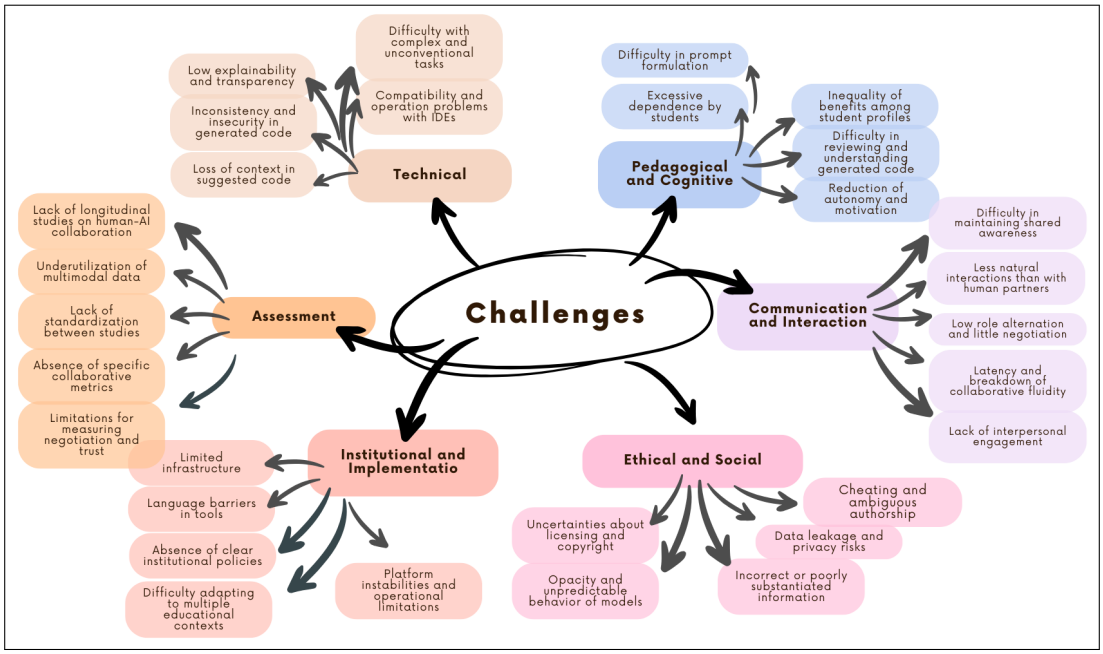

Figure 2. Challenges in AI-assisted programming education.

edge [Ma et al. 2023, Kazemitabaar et al. 2023]. Studies also indicate inequality in perceived benefits, with more experienced students tending to benefit more from these tools [Kazemitabaar et al. 2023].

Regarding the challenges of **communication and interaction**, exchanges with AI agents are perceived as less natural and more structured than those between humans. There is a lower occurrence of interruptions, debates, and negotiated solutions [Groothuijsen et al. 2024]. The use of multiple agents may lead to confusion regarding the authorship of suggestions [Wang et al. 2025]. Difficulties in maintaining shared awareness in AI-mediated collaborative environments have been documented [Feng et al. 2024a]. In this context, the term shared awareness refers to the ability to maintain a common and dynamic understanding of the task state, performed actions, and participants' intentions, ensuring that AI contributions are aligned with the interaction history and the current focus of the activity [Feng et al. 2024a].

Ethical and social issues involve risks to academic integrity, such as cheating on exams and ambiguous authorship of AI-generated code [Carvalho and Oliveira 2024]. Concerns have been raised about the transparency of AI models, which often operate as "black boxes" [Jiang et al. 2025]. Data leakage incidents have been identified, including cases where suggestions contained personal information [Bird et al. 2022, Zhang et al. 2023]. Uncertainties about licensing and copyright of generated code further highlight the need for clear institutional guidelines.

From an **institutional and implementation** perspective, the main issues include infrastructure limitations, language barriers in tools predominantly available in English [Phung et al. 2023, Simaremare et al. 2024], and the absence of clear institutional guidelines [Carvalho and Oliveira 2024]. Recurring operational limitations have also been reported, which compromise the stable adoption of these tools [Zhou et al. 2025,

Zhang et al. 2023].

In the **assessment** dimension, there is a wide diversity of metrics originally designed for traditional contexts, which fail to capture fundamental aspects of collaborative interactions between humans and AI agents. The literature highlights the absence of specific metrics to qualify human-AI collaboration [Dibia et al. 2023], the lack of standardized instruments that enable consistent comparisons [Kazemitabaar et al. 2023], and the underuse of multimodal data [Domingo 2023]. There is also a recognized need for longitudinal investigations and labeled datasets documenting real interactions between humans and AI agents [Robe et al. 2022].

Unlike technical problems, which may be mitigated through technological advances, the reported challenges emphasize the need for research that considers the heterogeneity of contexts in the formulation of public policies and the definition of research agendas, in order to avoid exacerbating educational inequalities.

4.4. Discussion

This mapping reveals transformations in programming education dynamics when mediated by generative AI, highlighting both the potential and the challenges for educational practice. The distinction between AI as a *collaborative participant*, *support tool*, or *pedagogical mediator* represents a conceptual evolution in the field.

These transformations challenge established assumptions about programming learning. The shift from a "learn-to-code" to a "code-to-learn" model [Valový and Buchalcevova 2023] represents a paradigmatic change that, while reducing entry barriers and facilitating familiarization with new technologies, may compromise central elements of human collaboration. The reduction in socio-emotional exchanges and mutual conceptual negotiation [Groothuijsen et al. 2024, Ma et al. 2023] contrasts with the reflective nature of programming learning and AI use, where discussions and productive conflicts tend to foster deep learning. New forms of cognitive engagement are also emerging, mediated by prompt engineering and the critical evaluation of automated suggestions [Paludo and Montresor 2024].

For educators and developers, the findings highlight the importance of preserving essential elements of human collaboration. Strategies include: (1) implementing hybrid approaches that combine human–human interaction with AI support; (2) developing structured protocols for the critical evaluation of automated suggestions to prevent passive acceptance [Dibia et al. 2023]; (3) providing specific teacher training for identifying signs of uncritical dependence [Barke et al. 2023] and promoting metacognitive reflection; and (4) designing activities that explicitly require conceptual negotiation among students prior to using AI. The observed trend toward more adaptive and socially responsive agents represents an opportunity to develop more effective systems that balance automated assistance with the promotion of intellectual autonomy.

This study presents limitations that should guide future interpretations. The rapid evolution of AI technologies may have rendered some analyzed tools obsolete. The absence of direct experimental comparisons in primary studies limits the ability to establish robust causal relationships, and the predominance of studies in English may introduce cultural bias, especially considering the inferior performance of AI tools in multilingual contexts [Simaremare et al. 2024]. Additionally, the use of Google NotebookLM for ini-

tial information extraction, although followed by rigorous manual verification by the researchers, constitutes a potential threat to validity, as automated processing may have resulted in the omission of relevant information during the preliminary analysis phase.

In the Brazilian context, studies indicate that tools based on generative AI have supported programming learning by providing individualized support to students, even in large classes, and by helping address initial conceptual difficulties often associated with dropout rates [Carvalho and Oliveira 2024]. Empirical evaluations have shown that generative AI solutions offer clear responses, functional code, and helpful guidance for novice students, fostering personalized learning [Carvalho and Oliveira 2024]. Evidence also highlights the risks of uncritical acceptance of AI-generated responses by students, even when incorrect, reinforcing the importance of strategies aimed at critical evaluation and the development of metacognitive skills [Deus et al. 2024, Filho et al. 2023].

5. Conclusion

This mapping study investigated how generative AI has been incorporated into programming education, based on the analysis of 52 selected studies. The main outcome was the proposition of a classification that distinguishes three functional roles of AI: collaborative participant, support tool, and pedagogical mediator. The identification of a shift from a "learn-to-code" to a "code-to-learn" model, accompanied by changes in communication (more structured and less spontaneous), cognitive engagement (with the emergence of prompt engineering), and socio-emotional dynamics (with reduced interpersonal negotiation), is relevant for understanding the evolution of the field, offering comprehensive systematization of pedagogical implications of generative AI in programming education.

The identified challenges go beyond technical limitations, encompassing risks of uncritical dependence, the weakening of genuine collaboration, and the lack of appropriate metrics for human-AI interaction. These findings demonstrate that effective integration requires hybrid approaches that preserve essential elements of programming education without AI.

For educational practice, the following are recommended: (1) specific teacher training for mediating hybrid learning environments; (2) development of structured protocols for the critical evaluation of automated suggestions; and (3) implementation of pedagogical models that systematically alternate between human collaboration and AI assistance. For researchers, the suggested future agenda includes the development of collaborative metrics tailored to human-AI interaction and investigations across diverse educational contexts.

Despite limitations from rapid technological evolution and scarce direct experimental comparisons, this study establishes theoretical and empirical foundations for the responsible integration of generative AI into programming education. The success of this integration will depend on coordinated efforts among researchers, educators, and developers to balance the benefits of automated assistance with the development of intellectual autonomy and the preservation of the social dimension of learning.

Declaration on the Use of Generative AI Tools in the Review and Writing Process

In developing this article, the authors used two generative AI tools: (1) Google NotebookLM to support preliminary information extraction from selected studies, as detailed in the methods section, with subsequent verification and manual complementation by the researchers; (2) ChatGPT, based on the GPT-4o language model, specifically to support textual review and translation into English. After applying the tools, the authors reviewed and adjusted the content as necessary. It is important to emphasize that the authors assume full responsibility for the final content of the publication.

Acknowledgment

Coordination for the Improvement of Higher Education Personnel (CAPES) - Program of Academic Excellence (PROEX).

References

Al Madi, N. (2023). How readable is model-generated code? examining readability and visual inspection of github copilot. In *Proceedings of the 37th IEEE/ACM International Conference on Automated Software Engineering*, ASE '22, New York, NY, USA. Association for Computing Machinery.

Amoozadeh, M., Nam, D., Prol, D., Alfageeh, A., Prather, J., Hilton, M., Srinivasa Ragavan, S., and Alipour, A. (2024). Student-ai interaction: A case study of cs1 students. In *Proceedings of the 24th Koli Calling International Conference on Computing Education Research*, Koli Calling '24, New York, NY, USA. Association for Computing Machinery.

Asare, O., Nagappan, M., and Asokan, N. (2023). Is github's copilot as bad as humans at introducing vulnerabilities in code? *Empirical Software Engineering*, 28(129).

Banić, B., Konecki, M., and Konecki, M. (2023). Pair programming education aided by chatgpt. In *2023 46th MIPRO ICT and Electronics Convention (MIPRO)*. IEEE.

Barke, S., James, M. B., and Polikarpova, N. (2023). Grounded copilot: How programmers interact with code-generating models. *Proc. ACM Program. Lang.*, 7(OOPSLA1).

Bassner, P., Frankford, E., and Krusche, S. (2024). Iris: An ai-driven virtual tutor for computer science education. In *Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2024, page 394–400, New York, NY, USA. Association for Computing Machinery.

Becker, B. A., Craig, M., Denny, P., Keuning, H., Kiesler, N., Leinonen, J., Luxton-Reilly, A., Malmi, L., Prather, J., and Quille, K. (2023). Generative ai in introductory programming. *Computer Science Curricula 2023, Curricular Practices Volume*.

Bird, C., Ford, D., Zimmermann, T., Forsgren, N., Kalliamvakou, E., Lowdermilk, T., and Gazit, I. (2022). Taking flight with copilot: Early insights and opportunities of ai-powered pair-programming tools. *ACM Queue*, 20(6):35–57.

Carvalho, S. and Oliveira, I. C. (2024). O uso e o potencial de inteligências artificiais no aprendizado de algoritmos e programação de computadores. In *Anais do XXXV*

Simpósio Brasileiro de Informática na Educação, pages 3049–3058, Porto Alegre, RS, Brasil. SBC.

Chan, W. K., Yu, Y. T., Keung, J. W., and Lee, V. C. (2023). Toward ai-assisted exercise creation for first course in programming through adversarial examples of ai models. In *2023 IEEE 35th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 132–136.

Chen, T. (2024). The impact of ai-pair programmers on code quality and developer satisfaction: Evidence from timi studio. In *Proceedings of the 2024 International Conference on Generative Artificial Intelligence and Information Security*, GAIIS '24, page 201–205, New York, NY, USA. Association for Computing Machinery.

De Silva, D., Jayasinghe, P., Illesinghe, A., Mallawaarachchi, D., Vithanage, C., and Vidhanaarachchi, S. (2024). Coderookie: Educational java programming environment for beginners. *Lecture Notes in Networks and Systems*, 1013 LNNS:243 – 253.

Deus, W., Marcolino, A., Avellar, G., Oliveira, K., and Barbosa, E. (2024). Avaliando resoluções de exercícios introdutórios de programação na era das ias generativas: Um estudo de caso com o chatgpt. In *Anais do XXXV Simpósio Brasileiro de Informática na Educação*, pages 1645–1657, Porto Alegre, RS, Brasil. SBC.

Dibia, V., Fourney, A., Bansal, G., Poursabzi-Sangdeh, F., Liu, H., and Amershi, S. (2023). Aligning offline metrics and human judgments of value for code generation models.

Domingo, C. (2023). Recording multimodal pair-programming dialogue for reference resolution by conversational agents. In *Proceedings of the 25th International Conference on Multimodal Interaction*, ICMI '23, page 731–735, New York, NY, USA. Association for Computing Machinery.

Dos Santos, O. L. and Cury, D. (2023). Challenging the confirmation bias: Using chatgpt as a virtual peer for peer instruction in computer programming education. In *Proceedings - Frontiers in Education Conference, FIE*.

Felizardo, K. R., Nakagawa, E. Y., Fabbri, S. C. P. F., and Ferrari, F. C. (2017). *Revisão sistemática da literatura em engenharia de software: teoria e prática*. Elsevier, Rio de Janeiro, Brasil, 1 edition.

Feng, L., Yen, R., You, Y., Fan, M., Zhao, J., and Lu, Z. (2024a). Coprompt: Supporting prompt sharing and referring in collaborative natural language programming. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, page 1–21. ACM.

Feng, T., Liu, S., and Ghosal, D. (2024b). Courseassist: Pedagogically appropriate ai tutor for computer science education. In *Proceedings of the 2024 on ACM Virtual Global Computing Education Conference V. 2*, SIGCSE Virtual 2024, page 310–311, New York, NY, USA. Association for Computing Machinery.

Filho, L. P., Souza, T., and Paula, L. (2023). Análise das respostas do chatgpt em relação ao conteúdo de programação para iniciantes. In *Anais do XXXIV Simpósio Brasileiro de Informática na Educação*, pages 1738–1748, Porto Alegre, RS, Brasil. SBC.

Górecki, J. (2024). Pair programming with chatgpt for sampling and estimation of copulas. *Comput. Stat.*, 39(6):3231–3261.

Groothuijsen, S., van den Beemt, A., Remmers, J. C., and van Meeuwen, L. W. (2024). Ai chatbots in programming education: Students' use in a scientific computing course and consequences for learning. *Computers and Education: Artificial Intelligence*, 7:100290.

Hassany, M., Brusilovsky, P., Ke, J., Akhuseyinoglu, K., and Narayanan, A. B. L. (2024). Engaging an llm to explain worked examples for java programming: Prompt engineering and a feasibility study. In *CEUR Workshop Proceedings*, volume 3840.

Imai, S. (2022). Is github copilot a substitute for human pair-programming? an empirical study. In *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, ICSE '22, page 319–321, New York, NY, USA. Association for Computing Machinery.

Jiang, L., Bin Ahmadon, M. A., Zhang, D., Fukada, T., and Yamaguchi, S. (2023). Synergyai: An human-ai pair programming design tool based on program net. In *2023 11th International Conference on Information and Education Technology (ICIET)*, pages 210–214.

Jiang, L., Yamaguchi, S., and Bin Ahmadon, M. A. (2025). Synergyai: A human–ai pair programming tool based on dataflow †. *Information (Switzerland)*, 16(3).

Kazemitabaar, M., Chow, J., Ma, C. K. T., Ericson, B. J., Weintrop, D., and Grossman, T. (2023). Studying the effect of ai code generators on supporting novice learners in introductory programming. In *Proceedings of the 2023 CHI Conference on Human Factors in Computing Systems*, page 1–23. ACM.

Kitchenham, B. et al. (2007). Guidelines for performing systematic literature reviews in software engineering. Technical report, Technical report, ver. 2.3 ebse technical report. ebse.

Kuttal, S. K., Sedhain, A., and AuBuchon, J. (2021). Designing a gender-inclusive conversational agent for pair programming: An empirical investigation. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 12797 LNAI:59 – 75.

Liu, J. and Li, S. (2024). Toward artificial intelligence-human paired programming: A review of the educational applications and research on artificial intelligence code-generation tools. *Journal of Educational Computing Research*, 62(5):1385 – 1415.

Ma, Q., Shen, H., Koedinger, K., and Wu, S. T. (2024). How to teach programming in the ai era? using llms as a teachable agent for debugging. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 14829 LNAI:265 – 279.

Ma, Q., Wu, T., and Koedinger, K. (2023). Is ai the better programming partner? human-human pair programming vs. human-ai pair programming.

Manfredi, G., Erra, U., and Gilio, G. (2023). A mixed reality approach for innovative pair programming education with a conversational ai virtual avatar. In *Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engi-*

neering, EASE '23, page 450–454, New York, NY, USA. Association for Computing Machinery.

Manorat, P., Tuarob, S., and Pongpaichet, S. (2025). Artificial intelligence in computer programming education: A systematic literature review. *Computers and Education: Artificial Intelligence*, 8:100403.

Moradi Dakhel, A., Majdinasab, V., Nikanjam, A., Khomh, F., Desmarais, M. C., and Jiang, Z. M. J. (2023). Github copilot ai pair programmer: Asset or liability? *Journal of Systems and Software*, 203:111734.

Nguyen, N. and Nadi, S. (2022). An empirical evaluation of github copilot's code suggestions. In *2022 IEEE/ACM 19th International Conference on Mining Software Repositories (MSR)*, pages 1–5.

Norton, V., Honda, F., Pessoa, M., and Pires, F. (2024). Codex: auxiliando estudantes com tea em programação por meio de um ambiente virtual de aprendizagem (ava) integrado à um llm. In *Anais Estendidos do XIII Congresso Brasileiro de Informática na Educação*, pages 155–158, Porto Alegre, RS, Brasil. SBC.

Page, M. J., McKenzie, J. E., Bossuyt, P. M., Boutron, I., Hoffmann, T. C., Mulrow, C. D., Shamseer, L., Tetzlaff, J. M., Akl, E. A., Brennan, S. E., Chou, R., Glanville, J., Grimshaw, J. M., Hróbjartsson, A., Lalu, M. M., Li, T., Loder, E. W., Mayo-Wilson, E., McDonald, S., McGuinness, L. A., Stewart, L. A., Thomas, J., Tricco, A. C., Welch, V. A., Whiting, P., and Moher, D. (2021). The prisma 2020 statement: an updated guideline for reporting systematic reviews. *BMJ*, 372.

Paludo, G. and Montresor, A. (2024). Fostering metacognitive skills in programming: Leveraging ai to reflect on code. In *CEUR Workshop Proceedings*, volume 3879.

Phung, T., Pădurean, V.-A., Cambronero, J., Gulwani, S., Kohn, T., Majumdar, R., Singla, A., and Soares, G. (2023). Generative ai for programming education: Benchmarking chatgpt, gpt-4, and human tutors. In *ICER '23: Proceedings of the 2023 ACM Conference on International Computing Education Research*, ICER '23, page 41–42, New York, NY, USA. Association for Computing Machinery.

Piccolo, S. R., Denny, P., Luxton-Reilly, A., Payne, S. H., and Ridge, P. G. (2023). Evaluating a large language model's ability to solve programming exercises from an introductory bioinformatics course. *PLoS Computational Biology*, 19(9):e1011511.

Puryear, B. and Sprint, G. (2022). Github copilot in the classroom: learning to code with ai assistance. *J. Comput. Sci. Coll.*, 38(1):37–47.

Rao, L., Liu, S., and Liu, A. (2024). Program segment testing for human-machine pair programming. *Int. J. Softw. Eng. Knowl. Eng.*, 34(10):1565–1591.

Rasnayaka, S., Wang, G., Shariffdeen, R., and Iyer, G. N. (2024). An empirical study on usage and perceptions of llms in a software engineering project. In *Proceedings of the 1st International Workshop on Large Language Models for Code*, LLM4Code '24, page 111–118, New York, NY, USA. Association for Computing Machinery.

Robe, P. and Kuttal, S. K. (2022). Designing pairbuddy—a conversational agent for pair programming. *ACM Transactions on Computer-Human Interaction*, 29(4).

Robe, P., Kuttal, S. K., AuBuchon, J., and Hart, J. (2022). Pair programming conversations with agents vs. developers: challenges and opportunities for se community. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, ESEC/FSE 2022, page 319–331, New York, NY, USA. Association for Computing Machinery.

Sarkar, A., Gordon, A. D., Negreanu, C., Poelitz, C., Ragavan, S. S., and Zorn, B. (2022). What is it like to program with artificial intelligence?

Silva, T., Vidotto, K., Tarouco, L., and Silva, P. (2024). Potencialidades do uso de inteligência artificial generativa como apoio ao ensino de programação. In *Anais do XXXV Simpósio Brasileiro de Informática na Educação*, pages 1942–1956, Porto Alegre, RS, Brasil. SBC.

Simaremare, M., Pardede, C., Tampubolon, I., Manurung, P., and Simangunsong, D. (2024). Pair programming in programming courses in the era of generative ai: Students' perspective. In *2024 31st Asia-Pacific Software Engineering Conference (APSEC)*, pages 507–511.

Tang, X., Wong, S., Pu, K., Chen, X., Yang, Y., and Chen, Y. (2024). Vizgroup: An ai-assisted event-driven system for collaborative programming learning analytics. In *Proceedings of the 37th Annual ACM Symposium on User Interface Software and Technology*, UIST '24, New York, NY, USA. Association for Computing Machinery.

Vadaparty, A., Geng, F., Smith, D. H., Benario, J. G., Zingaro, D., and Porter, L. (2025). Achievement goals in cs1-llm. In *Proceedings of the 27th Australasian Computing Education Conference*, ACE '25, page 144–153, New York, NY, USA. Association for Computing Machinery.

Valový, M. (2023). Psychological aspects of pair programming: A mixed-methods experimental study. In *EASE '23: Proceedings of the International Conference on Evaluation and Assessment in Software Engineering*. ACM.

Valový, M. and Buchalcevova, A. (2023). The psychological effects of ai-assisted programming on students and professionals. In *2023 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 385–390.

Wang, T., Wu, T., Liu, H., Brown, C., and Chen, Y. (2025). Generative co-learners: Enhancing cognitive and social presence of students in asynchronous learning with generative ai. *ACM on Human-Computer Interaction, Volume 9, Issue 1*, 9(1).

Wei, Z., Lee, A. T., Lee, V. C., and Chan, W.-K. (2024). Toward ai-facilitated learning cycle in integration course through pair programming with ai agents. In *2024 36th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 1–5.

Williams, L., Kessler, R., Cunningham, W., and Jeffries, R. (2000). Strengthening the case for pair programming. *IEEE Software*, 17(4):19–25.

Yang, Y., Liu, J., Zamfirescu-Pereira, J., and DeNero, J. (2025). Pensieve discuss: Scalable small-group cs tutoring system with ai. In *SIGCSE TS 2025 - Proceedings of the 56th ACM Technical Symposium on Computer Science Education*, volume 2, page 1669 – 1670.

Yilmaz, R. and Karaoglan Yilmaz, F. G. (2023). The effect of generative artificial intelligence (ai)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. *Computers and Education: Artificial Intelligence*, 4:100147.

Zhang, B., Liang, P., Zhou, X., Ahmad, A., and Waseem, M. (2023). Demystifying practices, challenges and expected features of using github copilot. *International Journal of Software Engineering and Knowledge Engineering*, 33(11-12):1653 – 1672.

Zhang, H., Cheng, W., Wu, Y., and Hu, W. (2024). A pair programming framework for code generation via multi-plan exploration and feedback-driven refinement. In Filkov, V., Ray, B., and Zhou, M., editors, *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering, ASE 2024, Sacramento, CA, USA, October 27 - November 1, 2024*, pages 1319–1331. ACM.

Zhang, L., Jiang, Q., Xiong, W., and Zhao, W. (2025). Effects of chatgpt-based human–computer dialogic interaction programming activities on student engagement. *Journal of Educational Computing Research*.

Zhou, X., Liang, P., Zhang, B., Li, Z., Ahmad, A., Shahin, M., and Waseem, M. (2025). Exploring the problems, their causes and solutions of ai pair programming: A study on github and stack overflow. *Journal of Systems and Software*, 219:112204.