Fostering Metacognitive Skills in Programming: Leveraging AI to Reflect on Code*

Giulia Paludo[1,*,†], Alberto Montresor[1]

[1]Department of Computer Science and Information Engineering, University of Trento, Via Sommarive 9, 38121 Trento (TN)

Abstract

The increasing use of AI among students has significant implications for established practices across all disciplines. In the specific case of programming in Computer Science (CS) education, we are observing a debate between the education system which sees AI-generated code as a threat to the learning's quality, and the industry, which expects professionals to best take advantage of AI-assisted programming. In this context, a successful mediation lies in fostering skills such as metacognition and reflective learning to bridge the academic and professional worlds. This paper reviews the literature on AI-assisted practices supporting metacognition and reflective learning. Drawing on this review and the findings from a prior pilot study run by us, we designed the Reflective AI Programming Lab (RAP Lab) where groups of three students collaborate to solve programming tasks using exclusively AI-generated code, with restricted queries and a set of constraints on a designated platform which logs all the interactions between students and the AI. The approach leverages AI-driven feedback and collaboration enhancing dialogical practices as Pair Programming and promoting the development of critical reflection on AI tools in CS. By having students explain in detail their reasoning and structure their solution strategies to a third party (AI), this intervention stimulates metacognition and reflective learning by offering a different perspective on problem solving. In fact, this approach promotes a deeper comprehension of the problem and forces students to clarify and refine their thoughts when articulating their solution strategy. AI serves as an impartial non-judgmental observer, allowing students to explore their mistakes without fear of embarrassment, encouraging a risk-free environment where they are more likely to experiment, learn from their errors, and engage in deeper reflective learning. Although this approach has yet to be validated, it will serve as the basis for more extensive data collection with a larger sample in the upcoming semester.

Keywords

AI-aided Education, Metacognition, AI-aided Programming, Reflective Learning

1. Introduction

Until the summer of Artificial Intelligence, and despite the improvement of new technologies, the educational community believed that certain core practices of many disciplines would still have their hegemony [1]. However, the growing diffusion of AI has challenged this belief, making many crystallised practices in education less effective than in the past and, more importantly, not fully aligned with the needs of the labour market and the changes of socio-economic demands.

Programming in Computer Science (CS) education is among the disciplines most significantly impacted by the introduction of generative AIs capable of writing code. For a long time, coding has been regarded as a core competency and activity in computer science (CS) and software development; recently, the introduction of specific AI assistants for code writing such as GitHub Copilot or ChatGPT elicited a profound discussion about underpinning skills and professions. According to the report by Williams [2] among the uses of AI, 66% of companies who answered to the survey reported using AI assistants for code. Certainly, CS and Software Engineering cannot be reduced to code generation, but having a competent agent for this task drastically changes the role of developers and software engineers

2nd International Workshop on Artificial Intelligence Systems in Education, University of Bolzano - Bolzano, Italy 25-28 November 2024

*Corresponding author.

†These authors contributed equally.

giulia.paludo@unitn.it (G. Paludo); alberto.montresor@unitn.it (A. Montresor)

https://cricca.disi.unitn.it/montresor/ (A. Montresor)

0009-0000-9270-2218 (G. Paludo); 0000-0001-5820-8216 (A. Montresor)

as well as the competencies they need to master [1]. In this respect, the role of education, at high school and university levels, plays a pivotal role in bridging the education and job worlds.

1.1. Context and Problem Statement

Generative AI tools for programming have caught education unprepared, while the industry welcomed this innovation with high enthusiasm, striving to successfully implement it in current processes and have their professional best take advantage of it [3]. Without proper interventions and policies, this debate risks resulting in a future misalignment between young graduates and industry requirements in the CS community [3].

In CS education, many educators perceive AI-generated code as a threat to learning, concerned that students will rely on these tools as shortcuts, bypassing the deeper understanding of underlying concepts [4]. There are fears that the mastery of fundamental coding skills will be diluted, that students may use AI tools to cheat, and that traditional programming exercises and assessments may become obsolete [1]. These concerns are further amplified by studies showing that Large Language Models (LLMs) can produce results comparable to those of top-performing students, raising questions about the role of manual coding skills in educational settings [5].

At the same time, the industry's adoption of AI tools has fundamentally shifted the skillsets expected from future software developers and engineers. Competencies such as problem-solving, critical thinking, and metacognitive skills–rather than merely writing code–are becoming central to professional practice. Without adjustments to educational approaches, there is a risk that students will graduate lacking the skills necessary to collaborate effectively with AI tools in the workplace, exacerbating the disconnect between academic learning and industry needs [6].

This situation calls for a reassessment of how programming is taught in CS education. Rather than seeing AI as a threat, it could be harnessed as a tool for developing higher-order thinking skills such as metacognition and reflective learning. The challenge lies in designing interventions that not only leverage AI's capabilities but also foster a deeper understanding of how these tools can support students' problem-solving processes and encourage critical reflection on their use.

1.2. Contribution

This paper presents a novel pedagogical approach that integrates AI-assisted programming with metacognitive and reflective learning practices, aimed at addressing the educational challenges posed by the increasing use of AI in CS education. The contributions of this work are threefold:

- Literature Review on AI and Metacognitive Practices: We provide an extensive review of the existing literature on AI-assisted practices in education, particularly focusing on how these tools can support metacognition and reflective learning in the context of programming.
- Design of the Reflective AI Programming Lab (RAP Lab): Based on insights from both the literature and a prior pilot study, we propose a structured learning environment, the RAP Lab, where students collaborate to solve programming tasks exclusively using AI-generated code.
- Exploring AI's Role in Promoting Reflective Learning: By situating students in a risk-free environment where AI acts as an impartial observer, the RAP Lab fosters critical thinking and metacognition. Students are required to articulate their problem-solving strategies, offering explanations of their reasoning to a third party (the AI), which in turn prompts them to reflect more deeply on their approaches.

The outcomes of this approach, though yet to be validated on a large scale, offer a promising direction for the future of programming education, where AI tools are leveraged not merely for efficiency but as catalysts for developing deeper cognitive skills.

2. State of the Art

This section explores the current research landscape on AI and programming education, focusing on how AI-driven tools are reshaping teaching practices and student learning experiences, and on the role of metacognition and reflective learning, examining how these cognitive processes are supported by AI tools.

2.1. Current Practices in Programming Education with AI

The initial steps toward effectively implementing AI for educational purposes in programming focus on tutoring, with chat bot or plugins able to provide feedback and suggestions to the students [7]. However, the most prevalent use of AI by students in personal settings revolves around receiving real-time feedback, correcting typographical errors, and interpreting error messages [8]. These AI-driven functionalities streamline the debugging process by quickly addressing minor issues, allowing developers to avoid spending excessive time on routine error correction.

In a broader context, AI plays a critical role in enhancing programming efficiency by automating routine, time-consuming tasks [9]. This enables developers to shift their focus from repetitive, mechanical aspects of coding to more complex, high-level problem-solving activities, where creativity and critical thinking are paramount.

In this section, we will explore the current practices and implications of AI in programming education research.

2.1.1. AI Tutors and AI Driven Feedback for Programming Education

Emerging applications of LLMs to learn programming include AI tutors, such as CodeHelp [10], which provides real-time guidance on solving exercises for introductory CS education. These methods are already widely used, as they offer on-demand expert help even in large-scale settings, offering personalised learning experiences. In fact, these tools offer convenient interaction and real-time support and can be accessible in the form of a chatbot or a form with different inputs (programming language, problem encountered and error message [10].

AI-driven feedback in programming languages plays an even more important role than in other disciplines, allowing more effective code revision under personalised guidance. This fosters inclusivity not only for diverse learners, but also provides more opportunities for those who are coming from non-CS backgrounds [11]. However, an open discussion on AI-driven feedback, especially when introduced at early stages of learning, raises concerns about its limitations, including the potential for students to become overly reliant on AI for fixing minor issues, hindering their development of autonomy [12]. In this regard, improper use of AI tutors without appropriate guidance can impede the development of the necessary AI literacy skills, preventing students from engaging with AI tools and content with the necessary critical awareness.

Furthermore, despite useful and punctual feedback on errors and suggestions, more often AI lacks pedagogical depth necessary to support the comprehension of concepts beyond the practical help. Beyond AI tutoring and feedback, AI in programming has also been implemented as an upgrade to established techniques in the CS community as pair programming or debugging duck [13].

In *pair programming*, two programmers work together at the same time on a code with two different roles: a driver who is writing the code and a navigator which provides further check and input [12][11]. Instead, the *debugging duck* technique consists of explaining the code line by line to a rubber duck or any object as if capable of understanding. These methods both imply verbal transposition of the program and metal organisation of the strategy employed, resulting in thinking aloud to help the developer understand how to fix the bug. AI upgrades these techniques by creating a more realistic and rich interaction, while making it more accessible as well. In fact, easy access to these tools allow to reinforce practice of conscious revisions and gain the habit of these Socratic approaches to revision even without a teammate [13] [14].

2.1.2. Assessment Challenges and Practice Shifts

Beyond benefits, mentioned practices of generative AI in coding have two main drawbacks: debates about the most adequate assessment [15] and the risk of losing creativity and novelty in programs [16]. Studies have shown that LLMs can emulate the performance of the best students especially when tasked with introductory exercises and this phenomenon poses concerns about the modality and the value of assessment especially when based on the generated code [1]. In this light, traditional assessments relying on code generated by the students' needs to be rethought to accommodate this new reality and stimulate learners in developing new skills.

The previously mentioned practices start finding new solutions to these issues. The case of using more massively AI and "*Explain in Plain English*" (EiPE) questions [17] introduces for students a different task to master, understanding and correctly explaining code in a prompt, and a new view on learning accomplishments for teachers. Despite being sensitive to low complexity exercises and specific programming languages, these original approaches stimulate an attentional shift in addressing the learning goal and its evaluation using LLMs to effectively complement existing pedagogical techniques in CS.

2.1.3. From Code Writers to Code Editors

In the future, we expect developers to write less code themselves, with important consequences in current training and education [18]. As LLMs become more proficient in handling complex tasks and more integrated into the coding workflow, the challenge of education is shifting from solely training *code writers* to also training skilled *code editors*. Although this has immediate advantages on productivity, it is not the same for the long-term impact of future professional code expertise [12, 11] [16]. The comparison with the publishing domain can help in tuning this perspective [19]. In programming, the ability to understand and refine code written by a third party is a sophisticated competence that will become even more essential [10]. This transition can only happen with apt interventions in programming education more focused on the holistic view deeper problem-solving skills, code explanation, strategies comparison rather than the mechanical process of writing code [20].

2.1.4. Explaining Code in Natural Language

Addressing programs through a verbal instruction sequences removes the barriers of code implementation, creating opportunities for deeper focus. Natural language-oriented programming (NLOP) provides numerous advantages, including improved outcome quality, enhanced collaboration, and in a broader sense, democratisation of software development [21].

Explaining code in natural language in fact implies a true understanding of the code generated or reviewed by requiring articulating the purpose of the different pieces [14]. Research indicates that *Natural Language Processing* supports problem-solving and human-machine interaction by requiring a higher level of detail and specification, which reduces errors and improves efficiency [20] . Describing code in natural language formalises and extends the debugging duck technique, commonly used by developers to identify issues in their code. While the practice of explaining code in natural language isn't new, it remains underutilised and is often poorly mastered by novice CS students.

2.1.5. Boosting Code Comprehension

LLMs in programming education can provide more than mere assistance with code generation by fostering critical thinking skills such as better understanding and explaining code [20]. These are transversal skills between academics and professional environments [18]. In fact, collaborative settings in CS with specific approaches like Agile are based on the ability to clearly explain and share the code done to others. Contexts involving multidisciplinary teams and diverse levels of expertise require these abilities also to get both smoother processes within teams [22].

Practices taking advantage of code explanation have been established teaching strategies in programming education. However, traditional mediums revised by a human tutor are not highly efficient or effective especially in large scale settings. Within this purpose, Denny et al. [17] brought this practice in a new approach by combining LLMs generated code with EiPE questions. This method requires students to reverse-engineer AI-generated code by providing a prompt able to exactly replicate the initial code. Engaging students with LLMs in this way provides a clear example of the potential of such tools in this direction of promoting deep understanding of problem and solution, however it presents limitations in terms of complexity of the pieces of code with this technique [23].

2.1.6. Impact and Use Differentiation

Despite the acknowledgement of the many benefits of introducing AI in terms of personalised learning, immediate and punctual support, researchers have analysed the delicate relationships between students' levels of proficiency and AI usage which should orient practices and activities development. A recent study by Zhang and their colleagues [8] observed that higher levels of academic self-efficacy were associated with reduced use of AI tools revealing that confident students are less likely to engage with AI. On the other hand, lower self-efficacy for the academic setting was observed to lead to higher risk of AI dependence[8]. Students with low performance using AI as self-thought have more risks for over reliance and conversely better performing students since they do not feel the need, or risk to miss an enhancement opportunity. Furthermore, utilising AI as a shortcut can result in decreased creativity and lack of authentic learning [8]. These findings support the need for proper interventions in curricula for AI training and literacy to empower students while preventing negative effects on academic performance and cognitive discrepancies.

2.2. Metacognition, Metalearning and Reflective Learning

The simplest way to define metacognition as *"thinking about thinking"* [24] as it refers to the awareness of one's own individual mental abilities [25]. It has two dimensions: knowledge and regulation. While knowledge includes all the ideas learners have about their cognitive performance including strategies and what could influence it, regulation refers to the processes of monitoring ongoing the activities and it includes planning, awareness of the task performance and strategy recall [26]. In our scope, the key aspects of metacognition are awareness and monitoring of reasoning, with the goal of regulating behaviour [27]. More broadly, metacognition helps individuals develop a system of heuristics and methods to better organize and approach problem-solving [28]. From the age of three, metacognition supports learning processes at different levels recalling specific strategies or monitoring our processes for efficiency [25]. The complexity and heterogeneity of this definition makes it difficult to frame with high precision from a scientific perspective [29].

However, educational psychology has accumulated evidence to support the importance of metacognition for enhancing and supporting the learning process, especially when it comes to self-directed learning, a crucial skill within the 21st century technological revolutions.

In education, specific activities and practices can help learners fully benefit from metacognition by making it explicit. For example, while we may naturally generalize a learned strategy to a new scenario, metacognitive interventions enhance the ability to consciously retrieve, compare and apply the appropriate strategy, leading to more efficient decision making and higher quality of the outcome[30]. This process also fosters greater awareness of the thought process and reasoning, allowing learners to refine and upgrade previous knowledge for future applications [31]. Metacognition and related concepts, such as Higher Order Thinking Skills (HOTS) which include critical thinking, problem solving and motivation, have become fundamental in evolving work environments that demand high-quality, original information processing [32].

In recent years, new constructs have emerged in the literature to specifically address metacognition within the context of learning, most notably through the concept of metalearning. While metacognition is more universal and encompasses the aspects of knowledge and regulation of cognitive strategies[30],

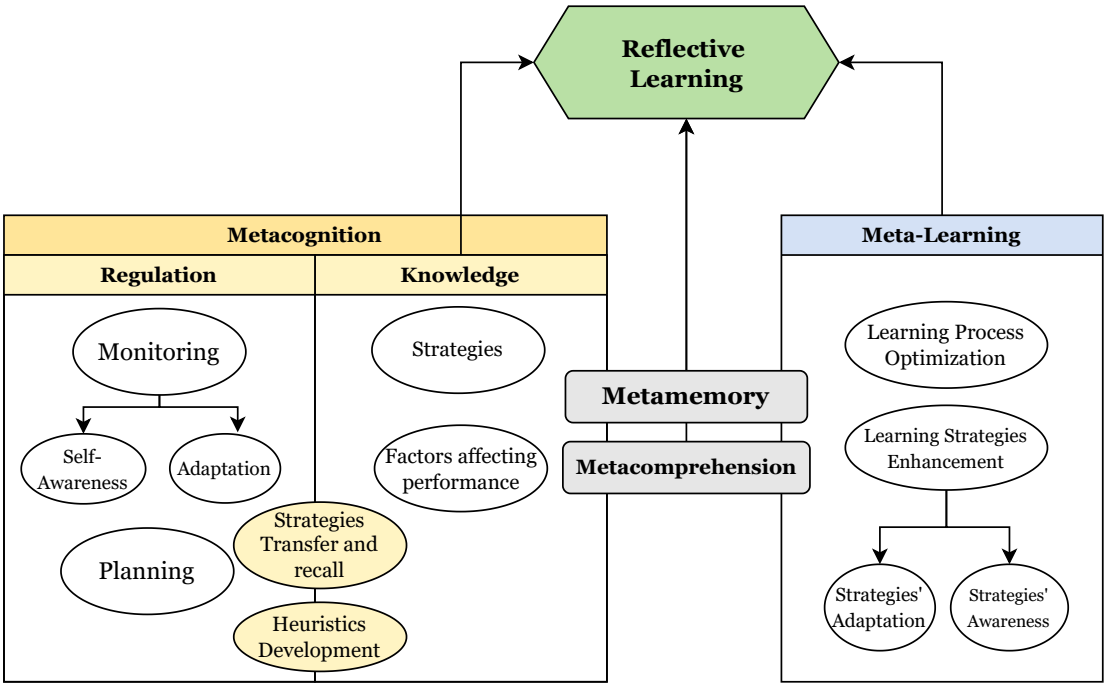

Figure 1: The interplay between metacognition, metalearning and reflective learning.

metalearning focus specifically on the understanding of how one learns, emphasizing processes that enhance learning to optimize both current and future learning experiences [33]. Metalearning provides a framework for understanding, adapting and optimizing the learning process, highlighting the powerful interplay between strategies and self-awareness in learning.

Lastly, the concept of Reflective Learning emerges as a practical approach in the educational context to promote the active use of metacognitive strategies, by providing tasks that students need to navigate meaningfully [34] [27] which cannot be accomplished through mechanical or repetitive methods.

The triad of metacognition, metalearning and reflective learning (Figure 1) express its highest value when brought to informal settings, where fostering these concepts leads to improved outcomes and more meaningful learning experiences [34] [27].

2.3. Previous Pilot

We conducted a previous pilot study with an intensive two-day programming challenge called *"Hack-aprompt"*, with 39 students with different backgrounds in CS and Software Engineering, coming from two universities (Trento and Innsbruck). Students were organized in groups of three, equipped with one laptop each, to solve programming problems using only AI-generated code. All the interactions between students and the AIs were logged; furthermore, pre and post questionnaires have been completed by most of the participants, focusing the aspects of problem-solving skills, approach to the problem, AI literacy and metacognitive skills; finally, two groups voluntarily agreed to switch on video recording and screen sharing.

Data analysis revealed notable effects from such an intensive interaction with AI. The activity led to significant improvements in AI literacy, clarity in code communication within group settings, and metacognitive awareness. We also observed improvements in specific problem-solving skills, such as problem decomposition and solution planning. Qualitative data further highlighted activity's benefits in fostering deeper understanding of problems in the preliminary stages and a more reflective transferable to autonomous practice [35] .

3. Metacognition and CS Education

There is an increasing emphasis on encouraging students to focus on deeper code comprehension and explanation, moving beyond mere information retrieval and superficial data processing. This approach aligns with the concepts of metacognition and *higher order thinking skills* (HOTS), which are key to enhancing both the learning experience and its outcomes [36].

The integration of AI into learning processes, along with shifts in educational strategies, has a profound impact on these concepts. By engaging with AI, students expand their learning environments beyond traditional content-based approaches. In this dynamic, students not only learn from AI, but AI also adapts to their interactions, creating a mutually adaptive learning experience.

This interaction transforms programming education by accelerating the learning process, providing more immediate feedback and support, and enabling students to tackle complex problems earlier, even before they have fully mastered a programming language and its underlying theoretical concepts. Through this hands-on approach, as students transition from thinking like *code writers* to *code editors*, the need for metacognitive abilities grows significantly [37].

Although often not prioritised in CS education, metacognition and the related metalearning concept are central to effective learning. These skills help students become more self-aware and reflective, while more importantly enabling them to adapt and apply apt strategies more effectively in their reasoning processes [14] [27]. The impact of advanced metacognitive skills in academic settings was observed by numerous studies. Metacognition-based interventions have shown positive effects at both the personal and performance levels, with reports of increased confidence and preparedness correlating higher GPAs, outperforming peers trained with traditional approaches [38, 39]

From a practical perspective, advanced metacognitive skills become crucial when students engage in more complex cognitive tasks that require them to externalize their individual reasoning to an interlocutor with no previous knowledge or ability to interpret–similar to the *debugging duck* technique [40].

In this context, the shift from traditional code writing to the implementation of AI assistance fosters improved code writing supported by more sophisticated metacognitive reflection and tailored feedback [41]. In the specific case of our pilot, an intensive and exclusive use of AI generated code obtained in a limited queries chat modality drives a deeper level interaction with content since the students have to rethink more in depth to the problem to best explain it and evaluate AI's code with a critical eye. Compared to other approaches mentioned, this technique, engaged with medium complexity exercises, turns routine writing tasks into opportunities for critical thinking, self-assessment and exploration of new strategies.

3.1. Metacognitive Strategies in CS Education through AI

Introducing metacognitive practices in CS and programming education involves designing activities, tasks, and exercises that shift the focus from the implementation phase to decomposition, evaluation and editing/debugging. These strategies aim to foster awareness and reflection during learning, encouraging students to develop habits they can apply across various contexts [42] (Figure 2).

Metacognitive strategies incorporate scaffolding and self-regulation techniques [24] using tasks such as explaining their reasoning, thinking aloud and editing outcomes with a focus on deeper understanding. For instance, students might be asked to discuss the process they followed, either because required by the task itself (as in our pilot's activity) or via prompts aligned with Bloom's taxonomy for cognitive skills [37] with questions like "*Can I identify patterns in my actions?*" or "*Which were the strategies employed for this outcome?*" [41, 39, 19]. These guided reflections help students recognize patterns and approaches, allowing them to identify areas of improvement, develop an organized problem-solving strategy, and ultimately foster the conditions for self-awareness and confidence in learning.

According to Cornoldi [25], an intervention can effectively address metacognition by involving the following aspects:

- discussion of perspectives and beliefs;

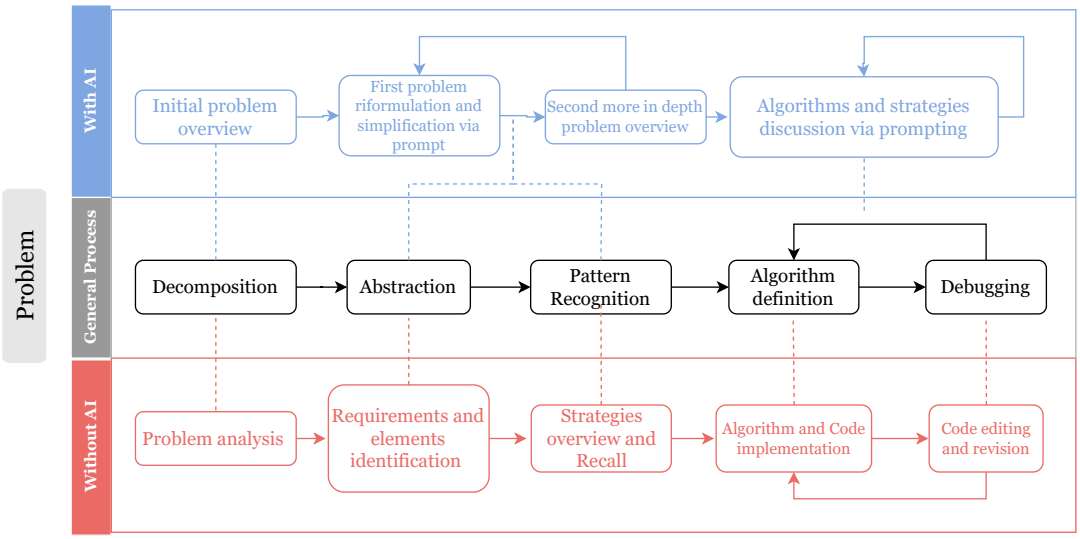

Figure 2: Comparison of the process behind a programming task or problem with and without AI

- analysis of errors;
- analysis and intentional attention to the application of strategies;
- the individual is oriented to master the competence and become proficient rather than to the performance,;
- stimulation of the ability of getting into others' thoughts.

A deep engagement with AI offers a comprehensive opportunity to integrate all the previously mentioned strategies through the outlined techniques. Specifically, translating the envisioned solution into prompts to guide AI output, rather than writing code manually, entails:

- discussing the code expected with what received addressing strengths and weaknesses;
- analysing errors or noncompliance with exercises constraints;
- carefully evaluating appropriate strategies and how guiding LLMs toward the right approach;
- trying to understand the LLMs approach to forecast misunderstandings and adapt prompting.

3.2. Prompt Engineering to Enhance Metacognitive Skills

Human-AI interaction in this context centers on prompt engineering and the user's ability to differentiate between useful and inaccurate AI-generated code. It also involves steering the AI model toward greater accuracy by crafting precise and effective prompts [19].

The role of prompt engineering in learning is thus twofold: it fosters deeper analysis of one's own thinking while uncovering the capabilities and limitations of AI-generated content, encouraging users to critically evaluate the results to meet their specific requirements [23].

With this foundation, using prompts to generate AI code becomes a tool for developing students' metacognitive skills, as it requires them to articulate their envisioned solutions through prompts. While this process may not be inherently unique, it challenges metacognition, especially when interacting with AI leads to unexpected results or misunderstood instructions. In tasks with ambiguous elements, such instances are frequent, prompting students to reassess even the smallest details they may have overlooked when coding independently [8].

In addition to evaluating their solutions, these reflections focus on understanding why specific results are produced, how to refine prompt formulations to better guide the model, and identifying the weaknesses in the generated output.

Current models often generate inefficient code (where efficiency refers to the program's ability to achieve the desired outcome in the smallest amount of time), requiring human intervention to identify areas for improvement and enhance code quality [20]. This process sharpens editing skills and fosters a deeper understanding, making students more proficient at analyzing code and devising strategies when coding independently.

On one hand, integrating AI extensively into coding can make the process seem much easier; on the other, inefficiencies, challenges with prompt engineering, and mistakes made by LLMs often lead users to prefer manual coding. However, the editing process offers several valuable learning outcomes, reinforcing comprehension and analysis through active engagement with both the tasks and AI tools. This interaction emphasizes the discussion of perspectives and assumptions, a more rigorous analysis of errors, careful application of strategies, and a focus on mastering skills rather than merely achieving performance. It also encourages students to consider the thoughts and approaches of others—an ability individuals develop from childhood but which is often overlooked or underemphasized in educational settings [41].

4. The Reflective Programming AI Lab: Overview, Applications and Educational Implications

Building on recent findings in the literature and insights from the pilot, we designed the new *Reflective AI Programming Lab* (RAP Lab), which will be implemented in both introductory and advanced programming courses. The lab is structured to provide a practical and reflective learning experience, featuring a set of exercises that must be solved exclusively using AI-generated code, under supervision and with specific guidelines.

The RAP Lab follows a similar approach to the activity tested in the pilot but is conducted every two weeks. In each lab session, a pair or triad of students, rotating throughout the semester, are given a setup consisting of one computer with a pre-configured platform on which they will work.

The core activity involves solving programming exercises, aligned in content and complexity with the theoretical course lectures, exclusively using AI-generated code. These exercises include specific constraints, such as a limited number of queries to submit and restrictions on using certain functions. Students will work with various programming languages and different LLMs, encouraging them to explore beyond their comfort zones through hands-on engagement. The activity is entirely voluntary and not graded.

This approach aims to enhance their flexibility, metacognitive skills, and technical competence. To support this goal, students will also be prompted with metacognitive and metalearning reflection questions based on Bloom's taxonomy, which they will answer in a form after each session. These questions are designed to summarize the lab experience while fostering self-awareness and building the habit of reflective learning.

To better connect the lab sessions with the main lectures, all submitted queries will be recorded in a log, providing the instructor with valuable data for curriculum adjustments and class discussions.

Lastly, although the lab is not graded, there is a need to track students' progress. To achieve this, a self-report questionnaire will be administered at the beginning, middle, and end of the semester. The data collected will assess students' perceived improvements, expertise, and learning across various dimensions of the activity, including individual skills, technical competencies, AI literacy, and group work.

The integration of AI into CS education, particularly in programming, has significant implications for both curriculum and pedagogy.

As suggested by Hutson and Plate [19], there is a need to reconfigure educational approaches to systematically incorporate revision and editing skills, emphasizing the ability to critically assess and evaluate AI-generated content and code. Adjusting curricula in this direction promotes enduring proficiency over mere performance, while highlighting verbal reasoning, communication skills, and providing opportunities to train contextual judgment.

In this direction, encouraging students to internalize a Socratic approach empowers them to apply this method of questioning beliefs and perspectives to problems across various disciplines. Our proposal, which merges EipE and prompt engineering, provides effective tools for enhancing internal reasoning and fostering greater self-awareness. It also helps students learn how to learn, cultivating the ability to continuously improve their learning strategies [41].

In this context, theoretical knowledge is put into practice through discussion and deeper re-elaboration [29]. By employing tools and pedagogies rooted in the Socratic approach, content is thoroughly reworked in alignment with the principles of metacognitive interventions. This transformation shifts AI into IA–*Intelligence Augmentation* [14]–thereby accelerating and enhancing the quality of the learning process.

A key implication is the need to design practical learning experiences that go beyond mere code generation. Implementing methods that facilitate explicit discussion and application of theoretical knowledge, either individually or in small groups [29], and mediated by AI and editing tasks, fosters genuine and focused inquiry.

Moreover, leveraging AI in this way can promote inclusion by lowering implementation barriers, enabling all students, especially those with difficulties, to develop relevant skills and contribute meaningfully in group settings [43] . On a broader scale, AI can empower individuals to understand code without needing to be experts, thereby democratizing access to programming knowledge and fostering more informed and conscious users.

5. Conclusions

Despite the potential threat to academic integrity and quality in the field of CS, AI has been shown to significantly enhance productivity and efficiency. This underscores the need for a thoughtful adaptation of curricula to adequately prepare students for the demands of their future careers.

Although the literature on this topic remains fragmented, the strategic use of AI can positively impact higher-order cognitive processes, particularly at the meta-levels of learning, such as metacognition, metalearning, and reflective learning. Empowering students to become more aware of their thought processes and to articulate their reasoning with clarity not only improves learning outcomes and performance but also fosters the development of critical skills. In an ever-evolving technological landscape, basic technical competencies may no longer suffice. Instead, reasoning flexibility, deeper comprehension, and the ability to edit and refine ideas will be essential for future professionals.

The implications for CS education include modernizing established practices with AI integration and providing supervised opportunities to develop AI literacy. Overall, the findings of this paper suggest that curricula should be adjusted to meet emerging demands by emphasizing the development of individual heuristics and problem-solving strategies, rather than focusing heavily on mechanical, "*apply the rule*" exercises. AI-assisted tools can play a key role in successfully implementing these approaches.

The RAP lab embodies these principles by designing activities with AI that prioritize Socratic interactions, placing less emphasis on code generation and focusing more on EIPe [17] and verbal reasoning. The potential of such experiences lies in fostering deeper engagement during the phases of comprehension and solution definition, with a stronger reliance on metacognition and higher-order thinking skills.

In conclusion, strengthening the role of the editor encourages the ability to compare, elaborate, and synthesize multiple perspectives, sparking new avenues of creativity while fostering AI literacy to prevent over-reliance on AI.

References

[1] M. Daun, J. Brings, How ChatGPT Will Change Software Engineering Education, in: Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1, ACM, Turku Finland, 2023, pp. 110–116. doi:10.1145/3587102.3588815.

[2] Williams, T. Some Companies are Already Replacing Workers with ChatGPT, Despite Warnings it Shouldn't be Relied on for 'anything important., Fortune (2023).
[3] T. Clear, Å. Cajander, A. Clear, R. Mcdermott, A. Bergqvist, M. Daniels, M. Divitini, M. Forshaw, N. Humble, M. Kasinidou, S. Kleanthous, C. Kultur, G. Parvini, M. Polash, T. Zhu, A Plan for a Joint Study into the Impacts of AI on Professional Competencies of IT Professionals and Implications for Computing Students, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 2, ACM, Milan Italy, 2024, pp. 757–758. doi:10.1145/3649405.3659527.
[4] T. P. Tate, S. Doroudi, D. Ritchie, Y. Xu, M. W. Uci, Educational Research and AI-Generated Writing: Confronting the Coming Tsunami, 2023. doi:10.35542/osf.io/4mec3.
[5] J. Prather, P. Denny, J. Leinonen, B. A. Becker, I. Albluwi, M. Craig, H. Keuning, N. Kiesler, T. Kohn, A. Luxton-Reilly, S. MacNeil, A. Petersen, R. Pettit, B. N. Reeves, J. Savelka, The Robots Are Here: Navigating the Generative AI Revolution in Computing Education, in: Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education, ACM, Turku Finland, 2023, pp. 108–159. doi:10.1145/3623762.3633499.
[6] M. Folmeg, I. Fekete, R. Koris, Towards identifying the components of students' AI literacy: An exploratory study based on Hungarian higher education students' perceptions, Journal of University Teaching and Learning Practice 21 (2024). doi:10.53761/wzyrwj33.
[7] M. Verleger, J. Pembridge, A Pilot Study Integrating an AI-driven Chatbot in an Introductory Programming Course, in: 2018 IEEE Frontiers in Education Conference (FIE), IEEE, San Jose, CA, USA, 2018, pp. 1–4. doi:10.1109/FIE.2018.8659282.
[8] D. Zhang, M. A. Bin Ahmadon, S. Yamaguchi, Human-AI Pair Programming by Data Stream and Its Application Example, in: 2022 IEEE International Conference on Consumer Electronics-Asia (ICCE-Asia), IEEE, Yeosu, Korea, Republic of, 2022, pp. 1–4. doi:10.1109/ICCE-Asia57006.2022.9954649.
[9] T. Weber, M. Brandmaier, A. Schmidt, S. Mayer, Significant Productivity Gains through Programming with Large Language Models, Proceedings of the ACM on Human-Computer Interaction 8 (2024) 1–29. doi:10.1145/3661145.
[10] P. Denny, V. Kumar, N. Giacaman, Conversing with Copilot: Exploring Prompt Engineering for Solving CS1 Problems Using Natural Language, in: Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1, ACM, Toronto ON Canada, 2023, pp. 1136–1142. doi:10.1145/3545945.3569823.
[11] G. Manfredi, U. Erra, G. Gilio, 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 Engineering, ACM, Oulu Finland, 2023, pp. 450–454. doi:10.1145/3593434.3593952.
[12] Q. Ma, T. Wu, K. Koedinger, Is AI the better programming partner? Human-Human Pair Programming vs. Human-AI pAIr Programming, arXiv preprint arXiv:2306.05153 (2023). doi:10.48550/ARXIV.2306.05153.
[13] S. Ojeda-Ramirez, S. Rismanchian, S. Doroudi, Learning About AI to Learn About Learning: Artificial Intelligence as a Tool for Metacognitive Reflection, 2023. doi:10.35542/osf.io/64ekv.
[14] H. Hassani, E. S. Silva, S. Unger, M. TajMazinani, S. Mac Feely, Artificial Intelligence (AI) or Intelligence Augmentation (IA): What Is the Future?, AI 1 (2020) 143–155. doi:10.3390/ai1020008.
[15] M. Liffiton, B. Sheese, J. Savelka, P. Denny, CodeHelp: Using Large Language Models with Guardrails for Scalable Support in Programming Classes, 2023. doi:10.48550/ARXIV.2308.06921, version Number: 1.
[16] R. Garcia, A. Csizmadia, J. L. Pearce, B. Alshaigy, O. Glebova, B. Harrington, K. Liaskos, S. J. Lunn, B. MacKellar, U. Nasir, R. Pettit, T. Prickett, S. Schulz, C. Stewart, A. Zavaleta Bernuy, All for One and One for All - Collaboration in Computing Education: Policy, Practice, and Professional Dispositions, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 2, ACM, Milan Italy, 2024, pp. 763–764. doi:10.1145/3649405.3659530.
[17] P. Denny, D. H. Smith, M. Fowler, J. Prather, B. A. Becker, J. Leinonen, Explaining Code with a Purpose: An Integrated Approach for Developing Code Comprehension and Prompting Skills, in:

Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, ACM, Milan Italy, 2024, pp. 283–289. doi:10.1145/3649217.3653587.

[18] J. Prather, J. Leinonen, N. Kiesler, J. G. Benario, S. Lau, S. MacNeil, N. Norouzi, S. Opel, V. Pettit, L. Porter, B. N. Reeves, J. Savelka, D. H. Smith, S. Strickroth, D. Zingaro, How Instructors Incorporate Generative AI into Teaching Computing, in: Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 2, ACM, Milan Italy, 2024, pp. 771–772. doi:10.1145/3649405.3659534.

[19] J. Hutson, D. Plate, Human-ai collaboration for smart education: Reframing applied learning to support metacognition, Faculty Scholarship (2023).

[20] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. D. Lago, T. Hubert, P. Choy, C. d. M. d'Autume, I. Babuschkin, X. Chen, P.-S. Huang, J. Welbl, S. Gowal, A. Cherepanov, J. Molloy, D. J. Mankowitz, E. S. Robson, P. Kohli, N. de Freitas, K. Kavukcuoglu, O. Vinyals, Competition-Level Code Generation with AlphaCode, arXiv preprint arXiv:2203.07814 (2022). doi:10.48550/ARXIV.2203.07814, publisher: [object Object] Version Number: 1.

[21] A. Beheshti, Natural Language-Oriented Programming (NLOP): Towards Democratizing Software Creation, 2024. doi:10.48550/ARXIV.2406.05409, version Number: 1.

[22] M. Biasutti, S. Frate, Group metacognition in online collaborative learning: validity and reliability of the group metacognition scale (GMS), Educational Technology Research and Development 66 (2018) 1321–1338. doi:10.1007/s11423-018-9583-0.

[23] S. MacNeil, A. Tran, D. Mogil, S. Bernstein, E. Ross, Z. Huang, Generating Diverse Code Explanations using the GPT-3 Large Language Model, in: Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 2, ACM, Lugano and Virtual Event Switzerland, 2022, pp. 37–39. doi:10.1145/3501709.3544280.

[24] J. H. Flavell, Metacognition and cognitive monitoring: A new area of cognitive–developmental inquiry, American Psychologist 34 (1979) 906–911. doi:10.1037/0003-066X.34.10.906.

[25] C. Cornoldi, Metacognizione e apprendimento, Il mulino, Bologna, 2002. OCLC: 1404999933.

[26] S. R. Raiyan, M. N. Faiyaz, S. M. J. Kabir, M. Kabir, H. Mahmud, M. K. Hasan, Math Word Problem Solving by Generating Linguistic Variants of Problem Statements, 2023. doi:10.48550/ARXIV.2306.13899, version Number: 1.

[27] A. F. D. Kittel, T. Seufert, It's all metacognitive: The relationship between informal learning and self-regulated learning in the workplace, PLOS ONE 18 (2023) e0286065. doi:10.1371/journal.pone.0286065.

[28] S. Hennessy, P. Murphy, The Potential for Collaborative Problem Solving in Design and Technology, International Journal of Technology and Design Education 9 (1999) 1–36. doi:10.1023/A:1008855526312.

[29] J. Perry, D. Lundie, G. Golder, Metacognition in schools: what does the literature suggest about the effectiveness of teaching metacognition in schools?, Educational Review 71 (2019) 483–500. doi:10.1080/00131911.2018.1441127.

[30] J. Metcalfe, A. P. Shimamura (Eds.), Metacognition: knowing about knowing, first mit press paperback edition ed., The MIT Press, Cambridge, Massachusetts, 1996. OCLC: 1241904980.

[31] M. V. J. Veenman, B. H. A. M. Van Hout-Wolters, P. Afflerbach, Metacognition and learning: conceptual and methodological considerations, Metacognition and Learning 1 (2006) 3–14. doi:10.1007/s11409-006-6893-0.

[32] R. Rianti, Z. A. Aziz, M. Aulia, INCORPORATING HIGHER ORDER THINKING SKILLS INTO ENGLISH SUMMATIVE ASSESSMENTS, English Review: Journal of English Education 12 (2024) 353–360. doi:10.25134/erjee.v12i1.9301.

[33] K. D. Salzman, K. Allen, K. McAnally, Differences in the detail: Metacognition is better for seen than sensed changes to visual scenes, Consciousness and Cognition 112 (2023) 103533. doi:10.1016/j.concog.2023.103533.

[34] A. F. D. Kittel, T. Seufert, It's all metacognitive: The relationship between informal learning and self-regulated learning in the workplace, PLOS ONE 18 (2023) e0286065. doi:10.1371/journal.

pone.0286065.
[35] F. F. Giulia Paludo, Agnese Del Zozzo, A. Montresor, Beyond coding with ai: an educational intervention and a research methodology for cs education, in: Proceedings of the 6th Conference of High Education Learning Methodologies and Technologies Online, Rome, Italy, 2024, submitted.
[36] L. Sulistiyani, H. Habiddin, Y. Yahmin, HOTS & Problem-Based Learning (PBL) with blended learning, J-PEK (Jurnal Pembelajaran Kimia) 7 (2022) 1–8. doi:10.17977/um026v7i12022p001.
[37] I. Goldstein, S. Papert, Artificial intelligence, language, and the study of knowledge, Cognitive Science 1 (1977) 84–123. doi:10.1016/S0364-0213(77)80006-2.
[38] H. J. Swanson, A. Ojutiku, B. Dewsbury, The Impacts of an Academic Intervention Based in Metacognition on Academic Performance, Teaching and Learning Inquiry 12 (2024). doi:10.20343/teachlearninqu.12.12.
[39] Western Colleges, Inc., A. Famarin, Metacognition in General Science to Improve the Academic Performance of Students, Pedagogy Review: An International Journal of Educational Theories, Approaches and Strategies 2 (2024) 109–118. doi:10.62718/vmca.pr-ijetas.2.1.SC-0624-028.
[40] K.-W. Han, E. Lee, Y. Lee, The Impact of a Peer-Learning Agent Based on Pair Programming in a Programming Course, IEEE Transactions on Education 53 (2010) 318–327. doi:10.1109/TE.2009.2019121.
[41] M. Guzdial, Providing Students with Computational Literacy for Learning About Everything, in: S.-C. Kong, H. Abelson (Eds.), Computational Thinking Education in K–12, The MIT Press, 2022, pp. 29–48. doi:10.7551/mitpress/13375.003.0005.
[42] S. Skalicky, S. A. Crossley, D. S. McNamara, K. Muldner, Identifying Creativity During Problem Solving Using Linguistic Features, Creativity Research Journal 29 (2017) 343–353. doi:10.1080/10400419.2017.1376490.
[43] R. M. Marra, D. J. Hacker, C. Plumb, Metacognition and the development of self-directed learning in a problem-based engineering curriculum, Journal of Engineering Education 111 (2022) 137–161. doi:10.1002/jee.20437, publisher: Wiley.