Report from Dagstuhl Seminar 25311

Generative AI in Programming Education

Michelle Craig*[1], Paul Denny*[2], Natalie Kiesler*[3], and James Prather*[4]

1 **University of Toronto, CA.** mcraig@cs.toronto.edu
2 **University of Auckland, NZ.** p.denny@auckland.ac.nz
3 **Technische Hochschule Nürnberg, DE.** natalie.kiesler@th-nuernberg.de
4 **Abilene Christian University, US.** james.prather@acu.edu

Abstract

This report documents the program and the outcomes of Dagstuhl Seminar 25311 "Generative AI in Programming Education". During the seminar, we examined the transformative impact of Generative AI on programming education. Because they can solve many introductory tasks given only natural language prompts, AI tools are challenging established approaches to programming education, in which there has been a traditional emphasis on writing small programs and providing (automated) feedback to learners. While these developments raise concerns about student over-reliance and inaccurate feedback, they also open opportunities for new pedagogical practices, such as fostering prompt literacy, adapting curricula, and designing AI-assisted learning tools. The present seminar convened 42 international experts to exchange knowledge, present research, and share innovations through keynotes, lightning talks, and tool demonstrations. Collaborative working groups explored implications for learning outcomes, assessment, equity, human values, and tool design, while identifying directions for systematic evaluation and interdisciplinary research. The seminar successfully established a foundation for a sustained community of practice and set an agenda for advancing programming education in the era of Generative AI.

Seminar July 27 – August 1, 2025 – https://www.dagstuhl.de/25311
2012 ACM Subject Classification Human-centered computing; Social and professional topics → Computing education
Keywords and phrases artificial intelligence, computer programming, computing education, generative ai, large language models
Digital Object Identifier 10.4230/DagRep.15.7.253

1 Executive Summary

Natalie Kiesler (Technische Hochschule Nürnberg, DE)
Michelle Craig (University of Toronto, CA)
Paul Denny (University of Auckland, NZ)
James Prather (Abilene Christian University, US)

Generative AI stands to significantly disrupt education in general and programming education is no exception. In addition, learning to program has several unique requirements and characteristics that require specific approaches. Evidence from the past several decades on how humans learn programming supports the commonly adopted approach of having students write many small programs. Often these are checked, and feedback is provided, by automated assessment tools. However, Generative AI has likely rendered this approach

* Editor / Organizer

obsolete given that easy-to-use tools are now readily available that can solve introductory computing problems with natural language prompts. At the same time, it is well known that the large language models that power Generative AI tools sometimes provide outputs that are either incorrect or inappropriate for the current understanding of a learner, raising concerns around student over-reliance and poor learning outcomes.

Educators are currently taking a variety of approaches, including ignoring the issue. Generative AI is a nascent, yet very rapidly developing field and new challenges and opportunities arise frequently making it extremely difficult for educators to keep pace with developments. Prototype tools that leverage Generative AI to facilitate learning are appearing, however most have yet to be deployed or adopted at scale. New pedagogical approaches are also emerging to foster the development of new kinds of skills, such as effective prompt creation, and new learning resources such as textbooks are appearing that teach programming hand in hand with Generative AI. Such approaches, however, have not yet been evaluated at scale as the field is developing so rapidly.

This Dagstuhl Seminar had the goal to bring together experts and stakeholders in Generative AI and computing education to foster collaboration and to chart a way forward as Generative AI continues to improve and proliferate. It was the goal of this seminar to leverage the experience and knowledge of dozens of programming education experts from around the world to form an enduring community of practice. During the seminar, we started to develop further strategies for incorporating LLMs into programming education and to rigorously evaluate their use and impact. We also explored the following topics in the context of Generative AI in programming education: accessibility; diversity, equality, and inclusion; resources; introductory programming for computer science majors and non-majors; advanced courses (that use programming); curriculum changes; novel pedagogies, approaches and tools; and industry use and changes that may lead to new learning outcomes.

These discussions were informed by participants' prior research and addressed the following objectives:

- Identify current implications of Generative AI on programming education, learning objectives, and curricula.
- Develop recommendations for the pedagogical integration of Generative AI in programming courses.
- Identify and establish interdisciplinary research objectives and questions to investigate Generative AI in programming education.

The seminar was structured into several sections, with all 42 participants actively involved: lightning and keynote talks followed by group brainstorming sessions, and dedicated workshop group sessions. At the beginning of the seminar, every attendee introduced themselves, and the seminar leaders provided an overview of the current GenAI landscape by introducing recent studies, emerging themes, trends, and tools. In addition, we had keynote presentations and discussions on AI in the curriculum, and specifically, Google's AI curriculum was presented. The keynote talks were delivered, as follows:

- Paul Denny: Opening Remarks: Generative AI and the Future of Programming Education
- Natalie Kiesler and James Prather: Beyond the Hype: A Sneak Peek into the Comprehensive Review of Current Trends in Generative AI Research, Teaching Practices, and Tools
- Dennis Bouvier: The Rest of the Robots: GenAI in Upper-level Computing
- Leo Porter and Daniel Zingaro: Learn AI-Assisted Python Programming
- James Prather: A New Curriculum for Computer Science that integrated GenAI: A Collaboration between Google and Academia
- Titus Winters: Generative AI in Software Engineering

We dedicated two more sessions to lightning talks and tool presentations. Prior to the seminar, every attendee had been invited to present recent findings, insights, or tools. This led to 20 lightning talks and tool presentations, which are represented in this report.

The lightning talk and tool sessions were accompanied by brainstorming sessions to identify recent challenges, opportunities, and future directions for research and practice. The brainstorming sessions also helped to form working groups. These subgroups focused on the following aspects: learning outcomes, assessment, social learning, CS1, meta-cognition, access and equity, quality use of GenAI, human values, introductory computing for non-majors, and AI-integrated learning tools. After two check-ins with the subgroups, several of them had merged, so that ultimately, three large working groups remained. The results of their discussions are summarized in this report.

2 Table of Contents

3 Overview of Talks

3.1 Feedback Quality Overview (open and state-of-the-art LLMs)

Imen Azaiz (LMU München, DE)

Joint work of Imen Azaiz, Natalie Kiesler, Sven Strickroth
Main reference Imen Azaiz, Natalie Kiesler, Sven Strickroth: "Feedback-Generation for Programming Exercises With GPT-4", in Proc. of the 2024 on Innovation and Technology in Computer Science Education V. 1, ITiCSE 2024, Milan, Italy, July 8-10, 2024, ACM, 2024.
URL https://doi.org//10.1145/3649217.3653594

It's well known that in introductory computer science courses, not only do many students struggle with learning programming, but educators also often lack the resources to support them effectively, especially when it comes to providing timely, personalized feedback at scale. Large Language Models (LLMs) have shown potential to address this challenge and are increasingly applied in intelligent tutoring and feedback systems. The presentation addressed the research question: *"How can we characterize the AI-generated feedback if provided with a task description and a student solution as input?"*.

Several existing LLMs (e.g., GPT-3.5, GPT-4 Turbo, GPT-o1-preview, Llama3.2-3B, DeepSeek-R1-14B) were evaluated on 55 authentic student submissions from two introductory programming assignments. The feedback was qualitatively analyzed and compared (cf. [1, 2, 3, 4]).

Overall, the models are capable of providing structured, detailed, and personalized feedback, but there are several differences in terms of compliance with specifications, correctness, inconsistencies, and redundancy. Approaches to addressing some of these issues have been discussed, although some may be inherent to the LLM approach.

References

1 Azaiz, Imen; Deckarm, Oliver; Strickroth, Sven (2023, December). *AI-enhanced Auto-Correction of Programming Exercises: How Effective is GPT-3.5?*. International Journal of Engineering Pedagogy (iJEP) 8/13, pp. 67–83, DOI 10.3991/ijep.v13i8.45621.

2 Azaiz, Imen; Kiesler, Natalie; Strickroth, Sven (2024). *Feedback-Generation for Programming Exercises With GPT-4*. Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1, ITiCSE 2024, Association for Computing Machinery, pp. 31–37, DOI 10.1145/3649217.3653594.

3 Azaiz, Imen; Kiesler, Natalie; Strickroth, Sven; Zhang, Anni (2025, July). *Open, Small, Rigmarole – Evaluating Llama 3.2 3B's Feedback for Programming Exercises*. International Journal of Engineering Pedagogy (iJEP) 5/15, pp. 57–73, DOI 10.3991/ijep.v15i5.55359.

4 Azaiz, Imen; Felippo, Konrad; Strickroth, Sven (2025). *Small but Competitive – Evaluating DeepSeek-R1 Among Diverse Open LLMs for Formative Programming Feedback*. In: Greubel, André; Strickroth, Sven; Striewe, Michael (eds.): Proceedings of the 7th Workshop "Automatische Bewertung von Programmieraufgaben" (ABP2025), pp. 97–106, DOI 10.18420/abp2025_10. Material available at Zenodo: .

3.2 Developer attitudes towards generative AI

Jamie Benario (Google – Chicago, US)

In this talk, I present recent research at Google and other tech companies that investigate how developers perceive generative AI tools at work. One study that we discuss is research on developer attitudes towards generative AI and their career. The research shows that developers have very complicated feelings, responding with both positive and negative attitudes towards the technology. In the rest of the presentation we discuss related research from around Google and other technology companies that provide insight into these thoughts.

3.3 Generative AI in Upper-level Computing Courses

Dennis Bouvier (United States Air Force Academy, US)

Generative AI (GenAI) is playing an increasingly influential role in computing education across all levels, offering new opportunities to support both teaching and learning. However, its effective integration raises critical concerns related to trust, academic integrity, and broader social and ethical implications. While substantial attention has been given to GenAI use in introductory programming courses (e.g., CS0/CS1), there remains a notable gap in research addressing its application in "upper-level" computing courses such as software engineering, human-computer interaction, algorithms, operating systems, and theoretical computer science.

This presentation gives a brief overview of the early work 2025 ITICSE conference Working Group 2 has drafted for a report that will present two complementary studies: a systematic literature review of GenAI interventions in upper-level computing education, and a survey of computing educators on their practices and perspectives regarding GenAI integration in these contexts.

3.4 What Does It Mean to Program in the Age of AI?

Claus Brabrand (IT University of Copenhagen, DK)

Joint work of Sebastian Nicolajsen, Claus Brabrand
Main reference Sebastian Mateos Nicolajsen, Claus Brabrand: "What Is Programming?", Commun. ACM, Vol. 68(6), pp. 28–30, 2025.
URL https://doi.org//10.1145/3713068

Introduction

Two years ago, while visiting colleagues in Uppsala, we were asked a deceptively simple question: "What does it mean to program?" Despite decades of teaching programming (CS1 and beyond), the question left us momentarily without an answer. What seemed trivial was profoundly unsettling. Today, with the rise of generative AI, the question has only grown more urgent: if programming is just "writing code," is programming itself becoming obsolete?

Historical Perspectives

Definitions of programming have expanded over time:

- **1950s:** programming as drawing up the "schedule" of operations to perform a calculation [1]. (To) program ≈ Calculation and writing a sequence of operations.
- **1970s (Knuth & Dijkstra):** programming as the *art* of composing programs, requiring aesthetics, ingenuity, and creativity [2, 3]. (To) program ≈ Creativity and composition of programs.
- **1985 (Naur):** a program's "life" depends on developers' understanding of how it supports the problem domain [4]. (To) program ∋ Understanding problem & problem domain.
- **2020s (Ko):** programming must account for societal impact: limitations, biases, and ethical considerations [5]. (To) program ∋ Consideration of unintended impact of Programs.

From calculation *to* creativity *to* understanding *to* consideration, the theoretical arc consistently well points beyond (just) "code."

Educator Perspectives

To explore how programming is understood in practice, we conducted a comprehensive study of Danish higher-education programs. Since computing is not mandatory in Danish primary or secondary school, introductory programming courses (CS1) in higher education often represent students' *first formal encounter with programming*: **the educational frontier**. By systematically identifying all such courses nationwide and surveying their educators, we obtained a representative snapshot of how programming is introduced at scale. The results revealed a sharp contrast:

- **What is programming?** educators' definitions of *programming* typically aligned with a narrow view:

  ```
  [design] -implement-> [code]
  ```

- **What is a (good) programmer?** while their definitions of a *good programmer* reflected a broad view spanning the entire problem–solution cycle (with potential iteration):

  ```
  [problem] -analyze-> [spec] -design-> [design]
    -implement-> [code] -evaluate-> [system]
  ```

While no single consensus emerged, the pattern was clear: programming itself was reduced to coding, whereas being a good programmer demanded engagement with the broader cycle.

Tension & Implications

This mismatch is consequential. Students are often implicitly taught that *programming = coding*. In the age of AI, this view invites a dangerous inference: if programming is "just coding," then programming can simply be *outsourced* to generative models. This will not work since programming is inherently more than code generation. Also, it risks depriving students of the chance to grapple with the harder, more enduring aspects of programming: analysis, specification, evaluation, and system-level reasoning.

Call to Action

Our findings echo a broader gap in AI-in-Education research. As highlighted in [6], most empirical evidence addresses the **Solution** domain. Some studies examine **Problem** → **Solution**, but still end in the Solution domain. Virtually none examine the **Problem** domain itself. None explore **Solution** → **Problem**, i.e., evaluating solutions back against the problem domain.

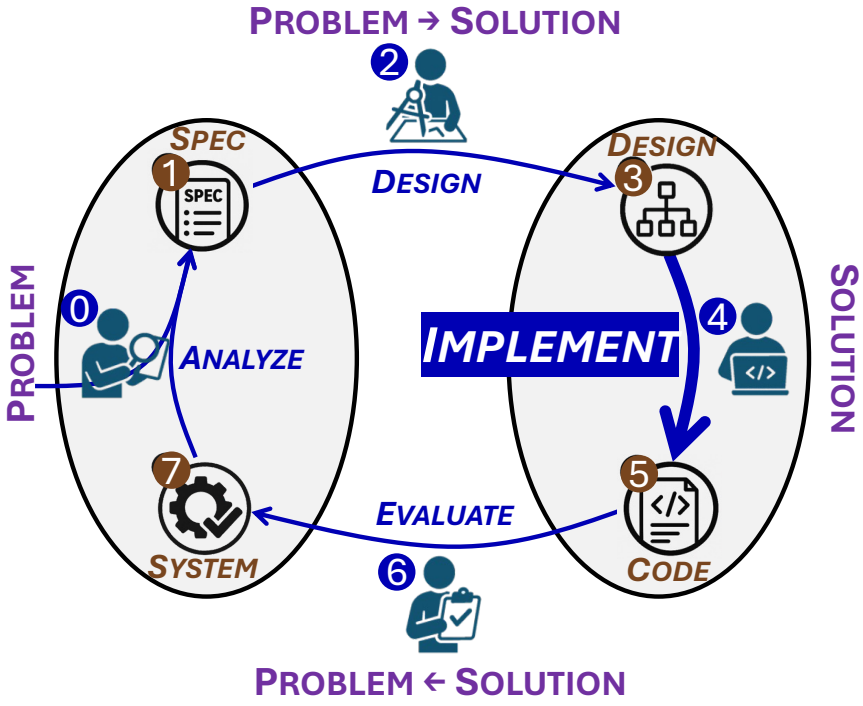

Figure 1 Educators' definitions: narrow vs. broad views of programming.

The way forward is clear. Both education and research must shift attention toward the **Problem domain**:

- **Analyze:** cultivating deep understanding of the problem.
- **Evaluate:** ensuring solutions actually address it.

These stages are hardest to outsource to AI and most crucial for developing durable programming competence.

Conclusion

We do not argue for a single hegemonic definition of programming. Instead, we call for an explicit conversation – with colleagues, with students, and with society. Programming cannot be reduced to the narrow slice of coding. It encompasses the broader, intangible qualities of problem analysis, domain understanding, evaluation, and societal consideration – the very human qualities that will remain vital, probably even more so, even in the age of AI.

References

1 A. F. Blackwell. What is programming? In *Proceedings of the 14th Workshop of the Psychology of Programming*, page 14. Citeseer, 2002.
2 D. E. Knuth. Computer programming as an art. In *ACM Turing Award Lectures*, 1974.
3 E. W. Dijkstra. A short introduction to the art of programming. Vol. 4. Technische Hogeschool, Eindhoven, 1971.
4 P. Naur. Programming as theory building. *Microprocessing and Microprogramming*, 15(5), 1985.
5 A. J. Ko et al. It is time for more critical CS education. *Communications of the ACM*, 63(11), Nov. 2020.

6 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 IV, S. Strickroth, and D. Zingaro. Beyond the hype: A comprehensive review of current trends in generative AI research, teaching practices, and tools. In *Proc. ITiCSE-WGR 2024*, pp. 1–39, Milan, Italy, 2024. doi:10.1145/3689187.3709614.

3.5 Howzat? Appealing to Expert Judgement for Evaluating Human and AI Next-Step Hints for Novice Programmers

Neil Brown (King's College London, GB)

Joint work of Neil Brown, Paul Denny, Juho Leinonen
Main reference Neil C. C. Brown, Pierre Weill-Tessier, Juho Leinonen, Paul Denny, Michael Kölling: "Howzat? Appealing to Expert Judgement for Evaluating Human and AI Next-Step Hints for Novice Programmers", ACM Trans. Comput. Educ., Vol. 25(3), pp. 1–43, 2025.
URL https://doi.org//10.1145/3737885

Motivation: Students learning to program often reach states where they are stuck and can make no forward progress – but this may be outside the classroom where no instructor is available to help. In this situation, an automatically generated next-step hint can help them make forward progress and support their learning. It is important to know what makes a good hint or a bad hint, and how to generate good hints automatically in novice programming tools, for example using Large Language Models (LLMs).

Method and participants: We recruited 44 Java educators from around the world to participate in an online study. We used a set of real student code states as hint-generation scenarios. Participants used a technique known as comparative judgement to rank a set of candidate next-step Java hints, which were generated by Large Language Models (LLMs) and by five human experienced educators. Participants ranked the hints without being told how they were generated. The hints were generated with no explicit detail given to the LLMs/humans on what the target task was. Participants then filled in a survey with follow-up questions. The ranks of the hints were analysed against a set of extracted hint characteristics using a random forest approach.

Findings: We found that LLMs had considerable variation in generating high quality next-step hints for programming novices, with GPT-4 outperforming other models tested. When used with a well-designed prompt, GPT-4 outperformed human experts in generating pedagogically valuable hints. A multi-stage prompt was the most effective LLM prompt. According to a fitted random forest model, the two most important factors of a good hint were length (80–160 words being best), and reading level (US grade nine or below being best). Offering alternative approaches to solving the problem was considered bad, and we found no effect of sentiment.

Conclusions: Automatic generation of these hints is immediately viable, given that LLMs outperformed humans – even when the students' task is unknown. Hint length and reading level were more important than several pedagogical features of hints. The fact that it took a group of experts several rounds of experimentation and refinement to design a prompt that achieves this outcome suggests that students on their own are unlikely to be able to produce the same benefit. The prompting task, therefore, should be embedded in an expert-designed tool.

3.6 Opening Remarks: Generative AI and the Future of Programming Education

Paul Denny (University of Auckland, NZ)

Joint work of Natalie Kiesler, Michelle Craig, James Prather, Paul Denny

The seminar opened with reflections on the rapid and transformative developments in Generative AI and their implications for programming education. This is an area marked by both excitement and uncertainty, with leaders from academia and industry expressing diverging views. These range from predictions that traditional university education, particularly in technical disciplines, may become obsolete, to optimism that AI will enhance learning and professional productivity. These perspectives were noted as mirroring recent debates in Communications of the ACM, where opposing viewpoints have argued either for the end of programming as it is currently understood or for a new era of AI-augmented programming practice.

The organisers of the seminar, Natalie Kiesler, Michelle Craig, James Prather, and Paul Denny, were excited at the prospect of fostering collaboration among researchers and practitioners from computing education, software engineering, HCI, and related fields. Participants were invited to explore how Generative AI is reshaping the ways that programming is taught, and how pedagogical approaches, assessment practices, and curricula need to evolve to ensure that AI supports rather than replaces meaningful learning in programming.

The organisers paid a special tribute to Dr Brett Becker, whose early work and vision were instrumental in the establishment of this seminar. Dr Becker, who passed away in October 2024, was remembered with deep appreciation for his contributions to the field and his role in bringing this community together.

3.7 Never in my Wildest Dreams: GenAI Agent-Based Software Development

Christopher D. Hundhausen (Oregon State University – Corvallis, US)

URL http://tiny.cc/GenAIAgents

Agentic GenAI tools, such as Copilot Agent, have been recently integrated into IDEs such as Visual Studio Code. Using thse tools, software developers can orchestrate software development by describing software requirements to a GenAI agent in plain English. The agent then implements the requirements by directly modifying a codebase. This process resembles a computational steering model with a human in the loop to monitor development progress and steer the agent toward the desired solution. I reflect on one month of intensive use of a GenAI agent, identifying high-level abstractions that a developer uses in lieu of computer code. I propose a set of general student learning outcomes that computing educators might use to design and learning activities to facilitate this form of software development and assess student learning. A video demo of my interactions and reflections can be found at `http://tiny.cc/GenAIAgents`.

3.8 Edugator: An AI-enabled Tool for Creating and Delivering Interactive Computing Content

Amanpreet Kapoor (University of Florida – Gainesville, US)

Joint work of Marc Diaz, Dustin Karp, Prayuj Tuli, Amanpreet Kapoor
Main reference Marc Diaz, Dustin Karp, Prayuj Tuli, Amanpreet Kapoor: "Edugator: An AI-enabled Tool for Creating and Delivering Interactive Computing Content", in Proc. of the 56th ACM Technical Symposium on Computer Science Education V. 2, SIGCSE TS 2025, Pittsburgh, PA, USA, 26 February 2025 – 1 March 2025, p. 1732, ACM, 2025.
URL https://doi.org//10.1145/3641555.3705025

Edugator is a browser-based, AI-enabled tool designed to help instructors of introductory computing courses create and deliver interactive educational content. It streamlines the content authoring process by incorporating generative AI models into both the creation and delivery stages. Using this tool, instructors can create bespoke interactive computing lessons and programming problems by providing a prompt and a few clicks. They can also author templates and test cases in programming languages such as C++, Java, C, and Python. Additionally, instructors can validate programming problems by running them against an auto-generated solution, allowing them to refine the problems before releasing it to students, preventing misinformation or ambiguity. Students can complete lessons and solve programming problems in a browser-based text editor receiving immediate feedback. They can also interact with a large language model-powered AI chatbot that scaffolds a student on how to approach the problem without giving out solutions. Edugator is built using modern web frameworks and the goal of the tool is to accelerate the adoption of automated assessment tools by minimizing the challenges instructors face with such tools. It also supports Learning Tools Interoperability (LTI), allowing seamless integration with learning management systems (LMS). Tools like Edugator streamline assessment authoring and delivery for Instructors while supporting student learning by promoting learning-by-doing and providing meaningful, personalized feedback. More information about the tool can be found at `https://edugator.app/`.

3.9 Inevitable AI? Reconsidering the "Inevitable" Integration of Generative AI in Computing Education

Hieke Keuning (Utrecht University, NL)

One of the objectives of this Dagstuhl Seminar, and of many other initiatives on Generative AI, is to develop recommendations for the pedagogical integration of GenAI in programming courses. I aim to challenge this seemingly "inevitable" integration.[1]

Large Language Models are built on human labor exploitation, stolen data, and are detrimental to the environment [2]. Unfortunately, this is usually an afterthought in many papers and talks. It is impossible to ethically justify their use at such a large scale. A huge amount of money is involved, and companies' interests are clearly not in human learning

[1] https://leonfurze.com/2025/04/28/the-myth-of-inevitable-ai/

[11]. The hype is unprecedented. The main goal of the major AI companies is automation, ultimately replacing much of human work. Turning to the consequences of its use, there is evidence of reduced critical thinking when knowledge workers use GenAI [12]. The benefits for programmers are debatable[2]; a recent study has shown that AI makes developers slower, although they thought it was the opposite [9].

So is the promise of revolutionizing education false? Current meta-reviews are not convincing [3], and many claims are unsupported by evidence [4]. Teachers spend a lot of time "reviewing, repairing and sometimes completely reworking AI-produced outputs" [8]. In computing education, researchers have observed a "widening gap" between weaker and stronger students [1] and signs of "social erosion" because students no longer ask each other for help [5]. While many new tools have emerged, they are often not built on known pedagogy or learning theory [7].

In the end, do we decide, or the technology [13]? "AI is unavoidable, but not inevitable".[3] Pretending it is not there does not make any sense, but we can take the lead in deciding what to do with it. We can take a step back and resist the hype [6]. Put pedagogy first, provide guardrails and scaffolding for authentic learning experiences. When building tools, only use LLMs for specific use cases, combining them with proven techniques (e.g. [10]). Finally, let us go back to the real problems at hand, and focus on community and learning.

References

1. Prather, J., Reeves, B., Leinonen, J., MacNeil, S., Randrianasolo, A., Becker, B., Kimmel, B., Wright, J. & Briggs, B. The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers. *Proceedings Of The 2024 ACM Conference On International Computing Education Research*. pp. 469-486 (2024)
2. Muldoon, J., Graham, M. & Cant, C. Feeding the machine: the hidden human labour powering AI. (Canongate Books,2024)
3. Weidlich, J., Gašević, D., Drachsler, H. & Kirschner, P. ChatGPT in Education: An Effect in Search of a Cause. *Journal Of Computer Assisted Learning*. **41** (2025)
4. Kohn, T. From Imitation Games to Robot-Teachers: A Review and Discussion of the Role of LLMs in Computing Education. *Journal Of Computer Assisted Learning*. **41** (2025)
5. Hou, I., Man, O., Hamilton, K., Muthusekaran, S., Johnykutty, J., Zadeh, L. & MacNeil, S. 'All Roads Lead to ChatGPT': How Generative AI is Eroding Social Interactions and Student Learning Communities. *Proceedings Of The 30th ACM Conference On Innovation And Technology In Computer Science Education*. pp. 79-85 (2025)
6. Rudolph, J., Ismail, F., Tan, S. & Seah, P. Don't believe the hype. AI myths and the need for a critical approach in higher education. *Journal Of Applied Learning And Teaching*. **8**, 6-27 (2025)
7. Topali, P., Haelermans, C., Molenaar, I. & Segers, E. Pedagogical considerations in the automation era: A systematic literature review of AIEd in K-12 authentic settings. *British Educational Research Journal*. (2025)
8. Selwyn, N., Ljungqvist, M. & Sonesson, A. When the prompting stops: exploring teachers' work around the educational frailties of generative AI tools. *Learning, Media And Technology*. pp. 1-14 (2025)
9. Becker, J., Rush, N., Barnes, E. & Rein, D. Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity. *ArXiv Preprint ArXiv:2507.09089*. (2025)

[2] https://blog.glyph.im/2025/06/i-think-im-done-thinking-about-genai-for-now.html
[3] https://marcwatkins.substack.com/p/ai-is-unavoidable-not-inevitable

10 Birillo, A., Artser, E., Potriasaeva, A., Vlasov, I., Dzialets, K., Golubev, Y., Gerasimov, I., Keuning, H. & Bryksin, T. One step at a time: Combining llms and static analysis to generate next-step hints for programming tasks. *Proceedings Of The 24th Koli Calling International Conference On Computing Education Research*. pp. 1-12 (2024)

11 Olson, P. Supremacy: AI, ChatGPT, and the Race that Will Change the World. (St. Martin's Press,2024)

12 Lee, H., Sarkar, A., Tankelevitch, L., Drosos, I., Rintel, S., Banks, R. & Wilson, N. The impact of generative AI on critical thinking: Self-reported reductions in cognitive effort and confidence effects from a survey of knowledge workers. *Proceedings Of The 2025 CHI Conference On Human Factors In Computing Systems*. pp. 1-22 (2025)

13 Padiyath, A. Do I Have a Say in This, or Has ChatGPT Already Decided for Me?. *XRDS*. pp. 52-55 (2024)

3.10 Beyond the Hype: A Sneak Peek into the Comprehensive Review of Current Trends in Generative AI Research, Teaching Practices, and Tools

Natalie Kiesler (Technische Hochschule Nürnberg, DE) and James Prather (Abilene Christian University, US)

Joint work of James Prather, Juho Leinonen, Natalie Kiesler, Jamie Gorson Benario, Sam Lau, Stephen MacNeil, Narges Norouzi, Simone Opel, Vee Pettit, Leo Porter, Brent N. Reeves, Jaromír Savelka, David H. Smith, Sven Strickroth, Daniel Zingaro

Main reference James Prather, Juho Leinonen, Natalie Kiesler, Jamie Gorson Benario, Sam Lau, Stephen MacNeil, Narges Norouzi, Simone Opel, Vee Pettit, Leo Porter, Brent N. Reeves, Jaromír Savelka, David H. Smith, Sven Strickroth, Daniel Zingaro: "Beyond the Hype: A Comprehensive Review of Current Trends in Generative AI Research, Teaching Practices, and Tools", in Proc. of the 2024 Working Group Reports on Innovation and Technology in Computer Science Education, ITiCSE 2024, Milan, Italy, 8 July 2024, pp. 300–338, ACM, 2024.

URL https://doi.org//10.1145/3689187.3709614

Generative AI (GenAI) keeps advancing rapidly, and the literature in computing education is expanding almost as quickly. Initial responses to GenAI tools were mixed between panic and utopian optimism. Many were quick to point out the opportunities and challenges of GenAI. Researchers reported that these new tools are capable of solving most introductory programming tasks and are causing disruptions throughout the curriculum. These tools can write and explain code, enhance error messages, create resources for instructors, and even provide feedback. In 2024, new research started to emerge, focusing on the effects of GenAI usage in the computing classroom. At the same time, a new class of tools is being developed that can provide personalized feedback to students on their programming assignments or teach both programming and prompting skills. With the literature expanding so rapidly, an ITiCSE working group aimed to summarize and explain what is happening on the ground in computing classrooms. In our lightning talk presentation, we provided the results of a systematic literature review; a survey of educators and industry professionals; and interviews with educators using GenAI in their courses, educators studying GenAI, and researchers who create GenAI tools to support computing education. The triangulation of these methods and data sources helps expand the understanding of GenAI usage and perceptions at this critical moment for our community.

3.11 Evidence of Learning Loss and Teaching Fundamentals

Colleen Lewis (University of Illinois Urbana-Champaign, US)

Joint work of Binglin Chen, Colleen M. Lewis, Matthew West, Craig Zilles
Main reference Binglin Chen, Colleen M. Lewis, Matthew West, Craig Zilles: "Plagiarism in the Age of Generative AI: Cheating Method Change and Learning Loss in an Intro to CS Course", in Proc. of the Eleventh ACM Conference on Learning @ Scale, L@S '24, p. 75–85, Association for Computing Machinery, 2024.
URL https://doi.org//10.1145/3657604.3662046

In my lightning talk I discussed two projects that are relevant to the teaching and learning of computer science (CS) in the age of Generative AI. The first project looks at learning loss from student plagiarism within an introductory programming course [1]. We collected data in an intro Python course for non-CS majors before and after the wide availability of ChatGPT. We developed four indicators of plagiarism. We found that these indicators appearing on homework submissions were correlated with lower performance on a final exam taken within a testing facility, while controlling for a student's performance on an exam early in the semester. This speaks to the ways in which plagiarism appears to decrease learning and the age of Generative AI provides easy opportunities for plagiarism. The second project looks at using physical objects to help students reason about foundational ideas within programming [2, 3]. This includes variable types, variable assignment, and function calls. This work builds on taken for granted practices within mathematics education of using physical objects to help students learn about quantity. Resources for this project are available at CSTeachingTIps.org/3D.

References

1 Chen, B., Lewis, C. M., West, M., & Zilles, C. (2024). Plagiarism in the Age of Generative AI: Cheating Method Change and Learning Loss in an Intro to CS Course. Learning at Scale 2024. https://doi.org/10.1145/3657604.36620
2 Lewis, C. M. (2021). Physical Java Memory Models: A Notional Machine. ACM SIGCSE Proceedings. 52(1). https://doi.org/10.1145/3408877.3432477
3 Lewis, C. M., Hernandez, M., Kuo, A., McDowell, H., Roller, H. (2025). Experience Report: Physical Models of Java Inheritance. SIGCSE 2025. Pittsburg, PA. 10.1145/3641554.3701871

3.12 Disciplinary Identity and Design Methods for GenAI

Kevin Lin (University of Washington – Seattle, US)

Joint work of Kevin Lin, Alannah Oleson, Anna Batra, Iris Zhou, Suh Young Choi, Chongjiu Gao, Yanbing Xiao, Sonia Fereidooni

One promise of generative AI for programming is that it can help us build software by quickly translating specifications into implementations. If we follow the premise, then skills like analyzing the qualities of specifications and evaluating the correctness of implementations could be more important to emphasize. But this presumes we know what software we want to build in the first place. Our choices shape not only what skills students learn but also their disciplinary values interpretation: "a process by which students reflect on the values of a disciplinary domain, as well as who they are and might become in relation to the domain" [1].

My scholarship has explored redesigning technologies as a way to shape disciplinary values interpretation by drawing on design methods such as iterative design [2, 3]. Iterative design is a software development practice that involves prototyping, testing, analyzing, and refining technology. Instead of assigning students a complete specification of a program to implement and focusing on evaluating the qualities of the final product, students could showcase their software development process over time. We can then ask questions about each step of the process.

Generative AI is often framed as a productivity tool, but what does productivity free us to do? Teaching students design methods could empower them to ask bigger questions about their work and challenge them to reflect on what exactly they hope to achieve in their future computing careers [4].

References

1 Sepehr Vakil. 2020. "I've Always Been Scared That Someday I'm Going to Sell Out": Exploring the relationship between Political Identity and Learning in Computer Science Education. In Cognition and Instruction, 38(2), 87–115. https://doi.org/10.1080/07370008.2020.1730374
2 Alannah Oleson, Meron Solomon, Christopher Perdriau, Amy Ko. 2023. Teaching Inclusive Design Skills with the CIDER Assumption Elicitation Technique. ACM Trans. Comput.-Hum. Interact. 30, 1, Article 6 (February 2023), 49 pages. https://doi.org/10.1145/3549074
3 Kevin Lin. 2024. An Invitation to Reimagine: Empowering Students to Redesign Computing Problems and Artifacts. Invited talk. https://kevinl.info/an-invitation-to-reimagine/
4 Anna Batra, Iris Zhou, Suh Young Choi, Chongjiu Gao, Yanbing Xiao, Sonia Fereidooni, Kevin Lin. 2024. "It Can Relate to Real Lives": Attitudes and Expectations in Justice-Centered Data Structures & Algorithms for Non-Majors. In Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2024). Association for Computing Machinery, New York, NY, USA, 88–94. https://doi.org/10.1145/3626252.3630754

3.13 APFEL – Adaptive Programming Feedback for E-Learning

Dominic Lohr (Universität Erlangen-Nürnberg, DE)

Main reference Dominic Lohr, Marc Berges, Abhishek Chugh, Michael Striewe: "Adaptive Learning Systems in Programming Education: A Prototype for Enhanced Formative Feedback", in Proc. of the DELFI 2024 - Die 22. Fachtagung Bildungstechnologien der Gesellschaft für Informatik e.V., DELFI 2024, Fulda, Germany, September 9-11, 2024, LNI, Vol. P-356, Gesellschaft für Informatik e.V., 2024.
URL https://doi.org//10.18420/DELFI2024_57

In my talk, I presented the tool APFEL, an adaptive learning environment for programming education that integrated GenAI in a controlled and transparent wa.

Unlike systems that leave it to LLMs to "decide" what kind of feedback to generate or provide and *when*, APFEL ensures that the feedback process is traceable and understandable for learners – if they want to know "Why did I get this feedback here?"

At its core, APFEL uses a symbolic/rule-based layer to determine the type of feedback that should be provided, based on empirical findings and pedagogical concepts and principles. This context is then used to systematically assemble a prompt, which is further enriched with additional context from a domain model and a learner model using RAG. This ensures that the resulting feedback is personalized and also didactically grounded and traceable.

3.14 Who is the author?

Andrew James Luxton-Reilly (University of Auckland, NZ)

Humans are considered to be the agents responsible for actions when they use tools to complete those actions. When we say "I cut down a tree", we attribute the act to the human rather than the axe. Similarly, when we create code using a tool such as GenAI, we should attribute the act of creation to the human rather than the machine.

Although this view challenges some academics, publishers such as ACM have already decided that only humans can be authors, and they are responsible for the product of GenAI use.

We conclude that users of GenAI tools are the authors of the output produced. Since these tools are ubiquitous, authentic assessment tasks should allow students to use GenAI, and the work product should be graded in the same manner as any other student work. To operationalise this approach, we adopt a two-lane approach in which we make a distinction between secure and insecure assessments.

For secure assessment, we may prohibit the use of GenAI to determine what the student can achieve independent of tools. For insecure assessment, students are permitted to use GenAI and must be graded with the view that they are the author, and are responsible for the product that they produce using tools.

3.15 All Roads Lead to ChatGPT: The Negative Impacts on Learning Communities

Stephen MacNeil (Temple University – Philadelphia, US)

Joint work of Irene Hou, Owen Man, Kate Hamilton, Srishty Muthusekaran, Jeffin Johnykutty, Leili Zadeh, Stephen MacNeil
Main reference Irene Hou, Owen Man, Kate Hamilton, Srishty Muthusekaran, Jeffin Johnykutty, Leili Zadeh, Stephen MacNeil: "'All Roads Lead to ChatGPT': How Generative AI is Eroding Social Interactions and Student Learning Communities", in Proc. of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE 2025, Nijmegen, The Netherlands, 27 June 2025 - 2 July 2025, pp. 79–85, ACM, 2025.
URL https://doi.org//10.1145/3724363.3729024

Generative AI provides students with valuable support even late at night when their friends are asleep. However, as students rely more heavily on these tools for help, instead of their friends, what will be the impact? Based on semi-structured interviews with 17 undergraduate computing students, we found that help-seeking requests are now often mediated by generative AI, with students frequently redirecting questions from their peers to generative AI instead of providing assistance themselves. This has the potential to undermine relationships and create a transactional classroom environment where friends never have a chance to form and learning communities erode. Students also reported feeling increasingly isolated from their peers and demotivated as a result. Our work is call to action to refocus our efforts on fostering vibrant learning communities and centering social interactions in an age of automation.

3.16 Prompt Programming

Victor-Alexandru Padurean (MPI für Software Systems – Saarbrücken, DE)

Joint work of Victor-Alexandru Padurean, Paul Denny, Alkis Gotovos, Adish Singla
Main reference Victor-Alexandru Padurean, Paul Denny, Alkis Gotovos, Adish Singla: "Prompt Programming: A Platform for Dialogue-based Computational Problem Solving with Generative AI Models", in Proc. of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1, ITiCSE 2025, Nijmegen, The Netherlands, 27 June 2025 - 2 July 2025, pp. 458–464, ACM, 2025.
URL https://doi.org//10.1145/3724363.3729094

Computing students increasingly rely on generative AI tools for programming assistance, often without formal instruction or guidance. This highlights a need to teach students how to effectively interact with AI models, particularly through natural language prompts, to generate and critically evaluate code for solving computational tasks.

To address this, we developed a novel platform for prompt programming that enables authentic dialogue-based interactions, supports problems involving multiple interdependent functions, allows manual code edits of generated code, and offers on-request execution of generated code.

Data analysis from students using the tool in introductory programming courses shows high engagement, with most prompts occurring in multi-turn dialogues. Furthermore, students were highly selective about the code they chose to execute. Reflection data also indicates that the majority of students preferred a balanced mix of natural language prompting and manual code editing when tackling harder problems.

3.17 Learn AI-Assisted Python Programming

Leo Porter (University of California – San Diego, US) and Daniel Zingaro (University of Toronto Mississauga, CA)

The advent of Generative AI tools is rapidly transforming the landscape of software development, presenting both challenges and opportunities for computer science education. In this talk, we address the critical need to adapt introductory programming curricula to this new reality. We share the journey of creating the textbook "Learn AI-Assisted Python Programming" and developing a corresponding CS1 course that embraces AI tools. The core of our approach is to shift the focus from traditional coding syntax to a new set of essential skills for the modern developer, including prompt engineering, critical code evaluation, testing, debugging, and problem decomposition in an AI-assisted environment.

We provide a detailed account of our pilot "CS1-LLM" course at UC San Diego, which served over 500 students. We discuss the course's design, which featured open-ended, creative projects across various domains to keep students engaged. We conclude with practical lessons learned from our teaching experience and an overview of our ongoing work to support educators and build a broader community around integrating AI into computer science education.

3.18 A New Curriculum for Computer Science that integrated GenAI: A Collaboration between Google and Academia

James Prather (Abilene Christian University, US)

Google's Tech Exchange program has been shaping and building engineers for many years. The advent of GenAI caused some panic among the creators, and it became a critical question about what to do. It was decided that the curriculum must be entirely reimagined. Google partnered with over a dozen universities to rework its curriculum to be GenAI forward from the very beginning. The redesigned courses were CS1, CS2, Applied Algorithms, Software Engineering, and Project Management. The results show that students engaged, worked hard, and learned traditional coding and computer science concepts. Students also increased their general self-efficacy over the test semester as well as their GenAI-specific self-efficacy. We believe this curriculum represents an effective first step in redesigning our courses to account for this fast-growing technology.

3.19 Experiences from an AI Task Force at a Large Institution

Karen Reid (University of Toronto, CA)

Joint work of The University of Toronto Task Force on Generative AI
Main reference The University of Toronto Task Force on Generative AI: "Report: Toward an AI-ready university – University of Toronto." (June 2025). Retrieved September 19, 2025.
URL https://ai.utoronto.ca/u-of-t-ai-task-force/report/

The University of Toronto Task Force on AI explored the impact of generative AI on all aspects of the University. I served on the main task force and co-chaired the Teaching and Learning Working group. During our consultations we heard many faculty despairing that unsupervised assessments no longer promote student development and learning since students can use GenAI tools to complete much of the work. The working group report highlights the need for GenAI literacy programming for both faculty and students, the importance of reconsidering learning outcomes across the curriculum, the challenge of providing formative feedback at scale, and noted that classroom dynamics are shifting as students and faculty alike bring their AI avatars to class with them. The task force reports recommend that the university invest in AI technology so that faculty and staff may experiment with and develop tools to support student learning. As we refined recommendations, we kept coming back to the idea that "good teaching is still good teaching." In other words, best teaching practices remain resilient in the age of GenAI. Ultimately, the task force focused on the university as an "institution dedicated to the development of human potential." Key takeaways include the need for greater transparency with students about how learning happens and a renewed determination to foster social interaction among students to enhance their learning.

3.20 AISOP – Using AI to Leverage E-Portfolios in Teaching

Daniel Schiffner (DIPF – Frankfurt am Main, DE)

Joint work of Daniel Schiffner, Wolfgagng Müller
Main reference Alexander Gantikow, Andreas Isking, Paul Libbrecht, Wolfgang Müller, Sandra Rebholz: "On the Creation of Classifiers to Support Assessment of E-Portfolios", in Proc. of the IEEE International Symposium on Multimedia, ISM 2023, Laguna Hills, CA, USA, December 11-13, 2023, pp. 297–302, IEEE, 2023.
URL https://doi.org//10.1109/ISM59092.2023.00057

In order to make the learning process visible, E-Portfolios have been utilized for many years. However, for both, teachers and students, this is considered an time-consuming and tedious task. As part of the project AISOP at the PH-Weingarten, these tasks have been simplified using AI methodologies.

Teachers define a concept map, which the tool uses to evaluate the created portfolios. Students get feedback on their current state before going into the examination, which only bases on the portfolio.

As code, it looks for the student like this:

```
while(learning) {
  write_portfolio() //here reflection with learning happens
  gather_experience()
  update_and_evaluate_using_ai() // concept map and AI tools are triggered
  get_feedback()
}
prepare_for_exam()
do_exam()
```

For the teachers, it could be represented as follows:

```
define_concept_map()
while(learning) {
  assign_tasks()
  gather_insights_from_ai() //using dashboards, etc.
  provide_additional_feedback() // human in the loop
}
check_portfolios_with_ai()
//prior to examination, check for missing entries, unclear documentation, etc.
do_exam_using_portfolio()
```

The project was defined and started in the pre-GPT era, with ChatGPT disrupting a lot of development. Many possible improvements exist, as higher quality evaluation is now possible. Currently, the project is being finished, and the idea pushed and feedback on it's implementation as a general tool collected. We hope that the tool gets the opportunity to be finished and provided to other teachers. This will facilitate a different type of assessment that cannot be simply solved by memorization and and requires some time and effort for students. The time spent helps students to better understand larger concepts and emphasizes continuous learning

3.21 Generative AI in Software Engineering

Titus Winters (Adobe – New York, US)

A summary of recent trends, stats, and beliefs regarding the use of generative AI within the tech industry.

3.22 Accessibility of GenAI Tools with Screen Readers

Daniel Zingaro (University of Toronto Mississauga, CA)

In this talk, I discuss two separate GenAI accessibility topics.

First, that there's an unequivocal pro of GenAI tools (in all of the needed pro-con discussions!): GenAI-powered tools are making visual-related accessibility improvements that I never thought possible. They are describing videos, making inaccessible apps accessible, helping with usability concerns on websites, describing graphs and other diagrams from papers, etc. Many of these challenges have been understood but not solved for decades ("add alt tags to your images!"); GenAI is allowing us to go beyond a mere description of visual material to full-on interaction.

Second, that we as a community need to ensure that the GenAI tools we ask our students are indeed accessible! In particular, I described the inherent accessibility of commandline-based GenAI tools that otherwise maintain a user's existing workflow (rather than requiring them to use IDEs that may take weeks to learn and that themselves may not be accessible).

3.23 Human-AI Interaction Challenge: How to Ensure Continued Growth of a Human as an Expert?

Jaromír Šavelka (Carnegie Mellon University – Pittsburgh, US)

Traditional software development (SD) involved minimal AI and a lot of code writing and reading. While AI supported SD has been characterized by the use of manually triggered AI to do some of the code writing the recently emerged agentic SD enables AI to handle complex tasks such as planning, testing, and refactoring. Human developers still play a crucial role in defining goals, reviewing, fixing outcomes, and guiding the process. When the agent-generated codebase falls short on, e.g., functional or stylistic requirements. One can ask the agents to modify something. One can even suggest how to go about it (e.g., where in the codebase; what library to use). And one can iterate ... At some point, one may give up and modify the code themselves. Hence, (1) there will likely be less natural opportunities for one to directly write and read code; (2) a developer will likely be responsible for handling increasingly larger and complex codebases; (3) the same or even greater expertise in software engineering, including reading and writing code, might be needed. Then, how can we ensure

that future software engineers go about their everyday work in a way that systematically hones their expertise? In conclusion, there appears to be a fundamental human-AI interaction challenge: How do we design interactions between agentic SD frameworks and humans to ensure continued growth of a human as an expert?

4 Working groups

4.1 "Andy's Axe" as a Guiding Principle for our Stance on AI-in-Education (for Computing)

Ibrahim Albluwi (Princess Sumaya University for Technology – Amman, JO), Dennis Bouvier (United States Air Force Academy, US), Claus Brabrand (IT University of Copenhagen, DK), Michelle Craig (University of Toronto, CA), Rodrigo Duran (Federal Institute of Brasília, BR), Christopher D. Hundhausen (Oregon State University – Corvallis, US), Kevin Lin (University of Washington – Seattle, US), Andrew James Luxton-Reilly (University of Auckland, NZ), Leo Porter (University of California – San Diego, US), Karen Reid (University of Toronto, CA), David H. Smith IV (Virginia Polytechnic Institute – Blacksburg, US), Claudia Szabo (University of Adelaide, AU), Shubbhi Taneja (Worcester Polytechnic Institute, US), Michel Wermelinger (The Open University – Milton Keynes, GB), Titus Winters (Adobe – New York, US), and Daniel Zingaro (University of Toronto Mississauga, CA)

One of the many outcomes of Dagstuhl Seminar 25311 is the following collective *educational stance* on how to appropriately navigate the "AI-in-Education Crisis" (as it has been dubbed).

Historically, technological innovation has always preceded ethics and legislation. We argue the metaphor of *"Andy's Axe"* serves as a guiding principle informing the future educational stance on GenAI. As the analogy goes: We say: "*Andy* cut down the tree." We don't say: "An *Axe* cut down the tree." Nor do we say that the tree was cut as a result of an *"Andy–Axe collaboration."* Pressed for more detail, we would say that Andy cut down the tree, *using* the Axe. Andy's *Axe* is just another *tool*; although one that commands our attention. *Andy* is the one responsible for: (1) [in]appropriate *use* of the tool; *and* (2) [un]intended *consequences* of their actions (whether using the tool or not).

Generative Artificial Intelligence (GenAI) is here, students will use it – whether we (attempt to) ban it or not – and they will need it in their future careers as professionals in the software industry. The alternative is educational institutions "forcing" students through *irrelevant programmes* and, ultimately, "producing" *irrelevant graduates* for the AI-empowered industry. Hence, we should not deprive future generations of students access to, and competences in, *effective*, *critical*, and *ethical use* of AI. Assuming this educational stance – *Andy* is responsible for Andy's *use* of Andy's *Axe* – the following picture emerges on the (event) horizon:

Let us *use* AI for *tedious lower-level* code writing under human guidance (the human wielding the metaphorical axe). This frees the human (student or professional) to focus more on *higher-level issues such as: problem solving*, *design*, *ethics*, *fairness*, *privacy*, *safety*, *security*, *usability*, *modularity*, *effectiveness*, *sustainability*, *trustworthiness*, and overall software *quality*.

If the adoption of AI, in any way, mirrors the adoption of the personal computer, what may come is a *rise*, not in *productivity*, but in *quality*. In any case, for this to work, we need to teach our future students how to *effectively*, *critically*, and *ethically* wield the Axe: GenAI.

4.2 GenAI in Programming Education: Hypes, Hoaxes, and Hopes

Carolin Hahnel (Ruhr-Universität Bochum, DE), Jamie Gorson Benario, Hieke Keuning (Utrecht University, NL), Natalie Kiesler (Technische Hochschule Nürnberg, DE), Tobias Kohn (KIT – Karlsruher Institut für Technologie, DE), Dennis Komm (ETH Zürich, CH), Colleen Lewis (University of Illinois Urbana-Champaign, US), Dominic Lohr (Universität Erlangen-Nürnberg, DE), Brent Reeves (Abilene Christian University, US), Jaromír Šavelka (Carnegie Mellon University – Pittsburgh, US), Jacqueline Staub (Universität Trier, DE), and Christina Weers (Goethe-Universität – Frankfurt am Main, DE)

Discussed Problems

The general theme of the working group centered on the question of what drives the hype surrounding the use of generative AI (GenAI) for educational and occupational purposes, and whether GenAI will actually deliver on its promises [3]. Two of these promises concern the future efficiency of writing code and GenAI's positive impact on learning processes and outcomes, both of which are possibly overly optimistic. We identified several strands requiring further research: (1) the effect of GenAI use on critical learning opportunities; (2) the pedagogy behind tool use and the potential gains and losses associated with its implementation; (3) the impact of GenAI use on personal and professional development; and (4) the ethical implications of GenAI use for educational purposes.

Strand 1. How does the use of GenAI change critical learning opportunities?

The concept of *desirable difficulties* in learning processes refers to challenges that can initially be effortful and frustrating but ultimately enhance understanding and personal growth [1]. The use of GenAI for learning raises questions about whether it reduces these valuable struggles. For instance, over-reliance on GenAI could create an illusion of competence, causing learners to believe they understand a topic simply because they can generate a response without truly comprehending the underlying concepts. Consequently, GenAI use may benefit advanced learners and experts by providing quick access to information, while novices may require the struggle to develop foundational understanding and critical thinking skills. Thus, these higher-order thinking skills will play a more central role in an era in which GenAI is ubiquitous.

Education may need to shift further toward teaching students how and why to learn and how to apply their knowledge effectively. This may require rethinking learning goals. In general, the focus may need to shift "from the destination to the journey." For example, self-driving cars may eventually make learning to drive obsolete. But if the same were true for, say, "reasoning," should it consequently be excluded from what we consider worth learning altogether? It is essential to define what constitutes a valuable outcome, whether personal growth, societal benefit, or environmental impact.

More research is needed to determine the concrete advantages of using GenAI in education. For example, anecdotal and empirical evidence suggest that students appreciate GenAI for providing starting points or templates [4], which can help overcome the initial hurdle of beginning to work on a task. In contrast, concern may remain that GenAI use could reduce potential creative benefits of being bored or "facing a blank page," both of which may stimulate innovative and creative thinking [5]. Overall, we need to consider how AI practices and newly developed GenAI tools relate to educational theory (see also a recent article by Hazzan and Erez [2]).

Strand 2. What do we gain and lose by using GenAI in learning processes?

Using GenAI tools may enhance but also diminish certain skills. It is crucial to quantify which skills are developed or lost through GenAI use in order to achieve a deeper understanding of how these tools can be leveraged or should be avoided. Choosing a *human-value-first* approach towards GenAI use might not focus on productivity in this context of learning, but rather emphasize the importance of growth.

One benefit of using GenAI in learning processes is the potential for providing immediate learning support, such as when using LLM-based tutors. However, concerns remain about the adaptability and integration of these tools into the learning process. Rigorous studies are needed to quantify growth, productivity, and other measures to determine whether AI is fulfilling its promises.

Strand 3. How does GenAI use affect personal and professional identity development?

In the long run, the use of GenAI may significantly impact identity development and *self-concept*. For instance, software engineers may experience an identity crisis by feeling threatened in their self-concept when their employers enforce the use of GenAI.

Widespread encouragement of GenAI use may instill diffuse and multifaceted fears among employees, encompassing concerns about job security, missing out or being left behind, and diminished self-value. Employees may worry about becoming dependent on GenAI or question whether failing to keep up with GenAI tools will make them inefficient and obsolete. Professionals who took pride in being selected for their unique skills may feel devalued when they see that GenAI can perform similar tasks. Thus, the use of GenAI can potentially downgrade one's sense of self-worth, challenging their professional identity and self-concept and leading to a sense of diminished self-value.

Pointing out ways of how to use and possibly when to avoid AI while preserving and cultivating one's identity, self-worth, and self-value within the profession could be crucial. Factors such as sense of belonging, self-efficacy, self-concept, and exploration of causes and comparators driving fears and anxieties may become more relevant in CS education than they are at this point in time.

Strand 4. How can GenAI be used ethically?

The use of GenAI presents several inherent ethical challenges; most pressingly, do we want to participate in a fundamentally unethical system? The value of using GenAI must be carefully considered for every context and application.

Our discussion also touched on the broader issue of ethical hypocrisy, such as the environmental impact of activities like flying and using ChatGPT. It was noted that ethics often involve making choices, and our responsibility as educators includes informing students and guiding them to understand the trade-offs involved in using GenAI so they can make

conscious decisions. Human behavior is often a result of negotiations with others, and education should support the comprehensive personal development of students. This involves identifying important values and teaching students how to uphold them, even in the face of disruptive technological advancements, such as GenAI tools.

Conclusions

Using GenAI in education may present both opportunities and challenges. Based on our discussion, we identified several areas of research that require further investigation. These include (i) gathering sound empirical evidence to evaluate the advantages and disadvantages of GenAI use and developing strategies to integrate GenAI into educational practices in a pedagogically meaningful way, while preserving the benefits of *learning struggle* and *critical thinking*; (ii) quantifying the impacts of GenAI use, developing guidelines for its integration, and exploring how GenAI learning processes can align with *human values* and *growth*; (iii) identifying and developing strategies to support individuals in maintaining a positive *self-concept* and fostering their *professional identity* in the face of technological change; and (iv) identifying educational and psychological mechanisms that encourage students to develop a more responsible and *value-driven approach* towards GenAI use.

References

1 Bjork, E. L. and Bjork, R. A. (2011). Making things hard on yourself, but in a good way: Creating desirable difficulties to enhance learning. In M. A. Gernsbacher, R. W. Pew, L. M. Hough, and J. R. Pomerantz (eds.), *Psychology and the Real World: Essays Illustrating Fundamental Contributions to Society*, pp. 56–64. Worth Publishers. https://psycnet.apa.org/record/2011-19926-008

2 Hazzan, O. and Erez, Y. (2025). Rethinking computer science education in the age of GenAI. *ACM Transactions on Computing Education*, *25*(3). https://doi.org/10.1145/3732792

3 Kohn, T. (2025). From imitation games to robot-teachers: A review and discussion of the role of LLMs in computing education. *Journal of Computer Assisted Learning*, *41*(3), e70043. https://doi.org/10.1111/jcal.70043

4 Scholl, A. and Kiesler, N. (2024). How novice programmers use and experience ChatGPT when solving programming exercises in an introductory course. In *2024 IEEE Frontiers in Education Conference (FIE)*, Washington, DC, USA, pp. 1–9, IEEE. https://doi.org/10.1109/FIE61694.2024.10893442.

5 Zeißig, A., Kansok-Dusche, J., Fischer, S. M., Moeller, J., and Bilz, L. (2024). The association between boredom and creativity in educational contexts: A scoping review on research approaches and empirical findings. *Review of Education*, *12*(1), e3470. https://doi.org/10.1002/rev3.3470

4.3 We're at a Crossroads: How GenAI Presents Challenges to Equity and Inclusion in Computing Education

Earl Huff (University of Texas – Austin, US), Laura E. Brown (Michigan Technological University – Houghton, US), and Daniel Schiffner (DIPF – Frankfurt am Main, DE)

Advances in Generative AI (GenAI) have changed the way we view programming and programming education. GenAI's ability to write usable, working code has now sparked conversations and research regarding how computer science courses should be taught. Students

are utilizing tools such as ChatGPT to produce coded solutions to tasks, but still require the ability to read and comprehend the code. Current research examines the implications of GenAI tools across the breadth and depth of CS courses and what it means for future iterations of CS curricula. What is lacking, however, is research addressing GenAI's impact on future software developers on two fronts. On one front, we need to consider how differential access to GenAI tools and education in K-12 among specific populations can widen the gap in terms of academic progression in higher education, especially for historically marginalized groups. There is already a gap in access to curricula and teachers, as seen with the AP Computer Science Principles. GenAI may further widen this gap due to several factors: a lack of a curriculum adapted for GenAI, a shortage of teachers skilled in using, advising, and understanding GenAI, and limited access to computers and AI tools in schools. These factors can create disadvantageous situations for students entering college to pursue a degree in CS. On the other front, industry hiring practices are evolving unevenly, with AI being used both by candidates and evaluators, often without clear standards or awareness of the implications. To what extent may companies utilize GenAI tools to aid in the evaluation of candidates, and how will they try to mitigate bias in the hiring process? What are candidates allowed to delegate to an AI if applying for a position and what training do they require for it?

This position paper argues that without systemic changes to the entire education through hiring pathway, AI risks reinforcing existing biases, introducing new biases, and rendering traditional assessments obsolete. We call for a rethinking of programming education across all levels, so that students acquire the competencies to utilize AI, and avoid the creation of new biases and unethical systems by gatekeeping newcomers who do not have access to special tools.

Participants

- Ibrahim Albluwi
Princess Sumaya University for Technology – Amman, JO
- Imen Azaiz
LMU München, DE
- Jamie Benario
Google – Chicago, US
- Dennis Bouvier
United States Air Force Academy, US
- Claus Brabrand
IT University of Copenhagen, DK
- Laura E. Brown
Michigan Technological University – Houghton, US
- Neil Brown
King's College London, GB
- Michelle Craig
University of Toronto, CA
- Paul Denny
University of Auckland, NZ
- Rodrigo Duran
Federal Institute of Brasília, BR
- Carolin Hahnel
Ruhr-Universität Bochum, DE
- Earl Huff
University of Texas – Austin, US
- Christopher D. Hundhausen
Oregon State University – Corvallis, US
- Amanpreet Kapoor
University of Florida – Gainesville, US
- Hieke Keuning
Utrecht University, NL
- Natalie Kiesler
Technische Hochschule Nürnberg, DE
- Tobias Kohn
KIT – Karlsruher Institut für Technologie, DE
- Dennis Komm
ETH Zürich, CH
- Juho Leinonen
Aalto University, FI
- Colleen Lewis
University of Illinois Urbana-Champaign, US
- Kevin Lin
University of Washington – Seattle, US
- Dominic Lohr
Universität Erlangen-Nürnberg, DE
- Andrew James Luxton-Reilly
University of Auckland, NZ
- Stephen MacNeil
Temple University – Philadelphia, US
- Victor-Alexandru Padurean
MPI für Software Systems – Saarbrücken, DE
- Leo Porter
University of California – San Diego, US
- James Prather
Abilene Christian University, US
- Brent Reeves
Abilene Christian University, US
- Karen Reid
University of Toronto, CA
- Daniel Schiffner
DIPF – Frankfurt am Main, DE
- Jan Schneider
DIPF – Frankfurt am Main, DE
- Adish Singla
MPI-SWS – Saarbrücken, DE
- David H. Smith IV
Virginia Polytechnic Institute – Blacksburg, US
- Jacqueline Staub
Universität Trier, DE
- Sven Strickroth
LMU München, DE
- Claudia Szabo
University of Adelaide, AU
- Shubbhi Taneja
Worcester Polytechnic Institute, US
- Christina Weers
Goethe-Universität – Frankfurt am Main, DE
- Michel Wermelinger
The Open University – Milton Keynes, GB
- Titus Winters
Adobe – New York, US
- Daniel Zingaro
University of Toronto Mississauga, CA
- Jaromír Šavelka
Carnegie Mellon University – Pittsburgh, US