CRA-E White Paper

July 2026

AI in Computing Education: Principles and Practical Guidance for Program Leaders

Jonathan Bell (Northeastern University), Michael Hilton (Carnegie Mellon University), Matthew Kam (Google), Denys Poshyvanyk (William & Mary), Borja Sotomayor (The University of Chicago), Chelsea Troy (Mozilla)

Introduction

Generative AI (GenAI) is transforming knowledge work across professions, and computing is no exception. This paper focuses on the educational implications: how we prepare the next generation of computer scientists and software engineers in light of these changes. While educators are already adapting their practices, this technology continues to evolve rapidly, and we believe sustained progress must be grounded in core pedagogical principles.

This paper is designed for academic decision makers such as department chairs, curriculum committees, and program directors who are responsible for helping their programs adapt to an uncertain future. We offer four core principles, rather than prescriptive solutions, for moving forward, recognizing that every institutional context is different. Each program's faculty members should engage in conversations about how best to meet the current and future needs of their students, adapting these recommendations to their own circumstances, student populations, and institutional constraints.

Summary: We organize our guidance around the following core principles:

1. **Learning as process**, not product
2. **Creating clear AI policies** that connect AI usage to learning goals
3. **Evolving assessment practices** to better measure students' learning
4. **Engaging across disciplines** within our colleges and universities

Throughout this document, we emphasize that GenAI integration is not a one-time decision but an ongoing process requiring sustained attention and evidence-based iteration. Additionally, we focus largely on the pedagogical and organizational impacts of AI; while AI integration also involves issues around financial and environmental costs, compliance requirements like FERPA, and equity implications, these are not the primary focus of this paper.

1. Learning as Process, Not Product

When considering outcomes for students in the age of AI, instructors should focus on learning as a process to develop student mastery. Particularly, instructors should focus on two principles backed by decades of research [Ambrose et al., 2010].

1. To develop mastery, students must acquire component skills, practice integrating them, and know when to apply what they have learned. Mastery is not a single ability but a combination of competencies that must be developed individually and then integrated through practice.

2. Goal-directed practice coupled with targeted feedback enhances the quality of students' learning. Practice alone is insufficient; it must be directed toward specific goals and accompanied by feedback that helps students identify gaps between their current performance and the desired outcome.

Instructors should keep these principles in mind, and particularly consider how to best craft educational experiences in a world with GenAI. Instructors should consider the learning processes of the students, and not only evaluate the product of the students' learning, such as the code or report that they submit as part of a project. It has always been the case that producing an artifact does not guarantee student learning, but GenAI, when used incorrectly, can significantly widen the gap between students turning in an artifact and students developing mastery.

GenAI tools are tremendously effective at solving the small, well-specified problems that have traditionally been used to build mastery in computing. This creates a challenge: students can produce correct artifacts without engaging in the goal-directed practice that builds durable skills [Hak et al., 2025]. The problem is not GenAI itself, but the mismatch between tool and purpose. While this is a concern throughout the curriculum, the challenge might be the most acute for early classes.

Addressing this requires instructors to be explicit about which activities are designed to build capacity and why practice matters. Students need opportunities to work through problems, make mistakes, and receive targeted feedback, and GenAI use should be structured to support that learning process rather than shortcut it. Additionally, GenAI can be used to offer more practice opportunities, as well as supplemental feedback that wasn't possible when the instructors were providing all the feedback, particularly in very large classes.

Importantly, valuable practice extends well beyond writing code from scratch—reading, evaluating, debugging, and decomposing problems are increasingly central skills, and there are no shortcuts to building them. In later courses, students can learn to critique, extend, and compare AI-generated code, as well as build larger-scale projects with the aid of GenAI tools [Vadaparty et al., 2024], applying the judgment that earlier practice developed.

2. Creating Clear AI Policies Supporting Learning Goals

Instructors should establish clear policies about when and how students may (and may not) use GenAI. These policies should be firmly grounded in the learning goals of a given class, and there must be an explicit connection between those learning goals and the role GenAI can play in achieving them, as well as how certain uses might undermine those goals. Otherwise, GenAI policies may come across as capricious and arbitrary. For example, a policy that says "you may not use GenAI on this assignment" without explanation can feel like a prohibition for its own sake; the same policy, framed as "this assignment is designed to build your ability to debug code independently, which you will need to evaluate AI-generated code later," connects the constraint to the student's learning.

Framing policies as guidance about how learning happens, rather than as arbitrary prohibitions, helps students interpret constraints as educational choices and enables programs to establish coherence across the curriculum. When students understand that restrictions exist because the learning comes from the process, not just the output, they are more likely to engage authentically with their coursework (including understanding when AI supports versus undermines that process).

Decisions about GenAI policies should involve faculty governance structures. Computing programs typically have established mechanisms for collective decision-making about curriculum, and these mechanisms should be employed for AI-related decisions. Collective decision-making ensures diverse perspectives are heard and creates broader buy-in for resulting policies. Programs should also anticipate students and faculty who object to GenAI tool use for reasons of privacy, data collection, ethics, or environmental impact, and should establish clear alternatives where appropriate. Addressing these concerns thoughtfully means engaging affected stakeholders, taking their values seriously, and documenting the tradeoffs involved in any policy decision—whether that results in alternatives, accommodations, or a clear explanation of why a particular approach has been chosen. Individual courses and assignments may appropriately adopt different AI policies. However, programs should strive for consistency in how those policies are framed and communicated to students. When every course improvises its own norms, students and faculty face unnecessary confusion; a shared framework for presenting AI policies signals institutional thoughtfulness.

3. Evolving Assessment Practices

If learning is about process, assessment should focus on evaluating the learning process, not just the products resulting from that process. When GenAI can produce working code, correct proofs, functional prototypes, and polished reports, product-only assessment no longer reliably measures what students have learned. Across the curriculum—from theory to HCI to systems—programs should complement product-based assessments with process-based approaches that reveal how students think, design, debug, and reason. This evolution in

assessment is ongoing: as GenAI capabilities change, programs must continually evaluate whether their assessments still measure what matters.

Proctored assessments—in-class exams, live coding demonstrations, oral examinations, and design reviews—directly assess what students know without GenAI assistance. One approach that has proven effective is the Computer-Based Testing Facility (CBTF): a proctored computer lab where students take auto-graded assessments with randomized questions, scheduling their own exam times within a specified window. For example, a University of Illinois facility has proctored over 90,000 exams annually, and quasi-experimental studies demonstrate substantial learning gains from frequent, low-stakes testing [Zilles et al., 2019; Morphew et al., 2019]. Even without dedicated infrastructure, programs can pilot the core idea: students bring laptops to a monitored room for frequent short assessments. The key benefit is enabling the frequent, low-stakes testing that research shows promotes learning [Roediger & Karpicke, 2006], while ensuring valid measurement of foundational skills.

Beyond controlled testing, programs should explore assessments that make the learning process visible:

- *Oral examinations and code walkthroughs*: Students explain their reasoning, trace through code, and respond to follow-up questions. These reveal understanding that submitted artifacts cannot. GenAI tools can make oral exams more scalable—providing preliminary assessment or helping instructors prepare targeted questions based on student submissions.
- *Live problem-solving*: Students work through problems in real-time, with instructors observing their process. This can include debugging exercises, design sessions, or extending unfamiliar code.
- Iterative projects with checkpoints: Rather than evaluating only final submissions, assess intermediate artifacts: design documents, test plans, code reviews, and reflections on changes made. This makes the development process part of what's evaluated.
- *Code review and critique*: Students analyze and improve code—whether peer-written, instructor-provided, or AI-generated. Evaluating AI-generated code is itself a valuable assessment: can students identify subtle bugs, security issues, or poor design choices?
- *AI-assisted practice with structured oversight*: AI tutors trained on course materials can provide students with additional practice opportunities and immediate feedback outside of class. When students' interactions with these tools are visible to instructors, they become part of the learning record—offering insight into how students engage with the material and enabling instructors to identify struggles early.

Paradoxically, GenAI can help shift assessment toward process. GenAI tools can generate varied

problem sets tailored to individual students, making each assessment unique. They can create more complex and engaging project scenarios that would be impractical to design manually at scale. They can provide preliminary feedback on student work, freeing instructor time for higher-value interactions like oral examinations and design reviews. And they can help students engage in more ambitious projects—where the real learning happens in integration, debugging, and iteration rather than in writing boilerplate code.

A particular concern is that students may overestimate their abilities when GenAI has helped them complete tasks, believing they understand more than they do because they produced correct outputs [Prather et al., 2024]. Process-based assessments provide a corrective: when students must explain their reasoning or work through problems live, both they and their instructors gain accurate information about what they actually understand. This calibration is valuable not just for grading, but for helping students develop accurate self-assessment—a skill they will need throughout their careers.

4. Engaging Beyond the Department

Individual educators should not face the challenges of AI in education by themselves, and should engage with colleagues in their departments and across disciplines, with industry practitioners who are integrating AI into professional workflows, and with researchers studying how these tools actually perform. If students are expected to use AI tools effectively, instructors should engage with AI tools themselves. Faculty cannot credibly guide students without firsthand experience—understanding both capabilities and limitations, developing intuitions about when AI assistance helps versus hinders. Programs should consider identifying early adopters in their faculty who can stay current with rapidly evolving tools and help the rest of the department build expertise.

Computing programs in particular have an opportunity and obligation to engage with other programs across the university on these shared challenges. Every discipline is grappling with questions about AI: What does "AI literacy" mean, and how should it be taught? What constitutes a reasonable AI policy for coursework in fields where GenAI assistance may be appropriate in different ways than in CS? What ethical concerns should be considered when using GenAI tools? Computing faculty have domain expertise that can inform these conversations, while other disciplines bring perspectives on how AI intersects with their own pedagogical goals and professional standards. Defining "AI literacy" in specific disciplines is one concrete example: computing faculty understand the technical capabilities and limitations of GenAI systems, while faculty in other disciplines understand what their students need to know to use GenAI responsibly in fields such as nursing, journalism, business, or law. Working together to articulate shared learning outcomes (both across programs and within a discipline) produces better results than either group working alone. By engaging in these cross-disciplinary conversations, computing programs contribute to a collaborative culture where faculty expertise shapes institutional responses to GenAI.

This collaborative spirit should extend beyond individual institutions to the broader computing education community. Programs should document what they tried, what worked, and what didn't—both for their own future reference and to contribute to a shared body of knowledge. Publication in computing education venues, presentations at conferences, and informal sharing through professional networks all accelerate collective progress. The computing education research community has studied assessment validity, academic integrity, and skill development extensively; programs making significant changes should consult this literature and, ideally, contribute to it by studying the effects of their changes. No program needs to solve these problems alone, and the challenges we face are shared across institutions worldwide. Several community efforts are already producing useful resources, including the CRA's Level Up summits on computing education, the ACM Task Force on AI and CS Education, and the GenAI in CS Education Consortium (teachCSwithAI.org).

As programs adapt, they should ground their efforts in evidence rather than intuition. Before making changes, they should articulate what success looks like and define metrics aligned with learning goals. These goals should be established before data collection to prevent post-hoc rationalization. They should seek regular feedback from multiple stakeholder groups about their experiences: faculty who observe their classroom dynamics, students who experience the curriculum firsthand, alumni who can speak to their professional preparation, and employers who observe and evaluate graduates' performance. Each group has access to different information about whether program changes are achieving their intended goals. AI integration also introduces practical considerations—financial and environmental costs, compliance requirements like FERPA, and equity implications when access to AI tools is uneven—that programs should navigate thoughtfully, modeling the resource-conscious decision-making we hope to instill in our students.

Conclusion

Public narratives around AI often confuse GenAI's ability to produce artifacts with its capacity to replace computing professionals, and even computing education itself. This conflation overlooks what makes computing disciplines difficult. In software engineering, it is requirements discovery, architectural tradeoffs, specification, verification, performance, and long-term maintainability. In HCI, it is understanding user needs and contexts, designing appropriate studies, and interpreting ambiguous results. In theoretical computer science, it is identifying which problems matter and recognizing when a proof approach will generalize. AI can generate outputs quickly but does not eliminate the need to frame problems well, evaluate whether solutions fit their context, or exercise disciplinary judgment [Yahav, 2022]. Faster artifact production via GenAI tools can push students into these higher-order concerns earlier, as evaluation and integration become the bottleneck rather than production. This creates an opportunity to strengthen curricula around the judgment and practices that define each discipline—the work that remains hard regardless of how artifacts are produced.

The principles in this paper are intended to guide immediate action on that opportunity, without the need to wait for a consensus on the impact of GenAI on the discipline. Programs should begin

by mapping their curriculum to identify which courses build foundational skills and which focus on professional practice, and use that map to determine where GenAI usage should be restricted, structured, or encouraged at students' different development stages. They should audit existing assessments for validity, and pilot low-overhead proctored assessments to complement take-home work. They should establish a program-wide GenAI policy with consistent categories and clear connections to learning goals, reducing student confusion and faculty burden. Additionally, they should create opportunities for faculty to develop firsthand experience with GenAI tools, building the expertise needed to guide students effectively.

There is much that this paper does not address: what to remove from existing curricula to make room for new material, which topics will grow in importance as GenAI capabilities expand, and how to empower non-CS majors to create software in this new landscape. As the amount of what becomes possible without writing code from scratch continues to grow, these questions will demand their own sustained attention.

Success will require ongoing collaboration within programs, across institutions, and with industry, as we collectively discover best practices for an uncertain future. No program needs to navigate this alone, and the challenges we face are shared. By working together, grounding decisions in evidence, and remaining committed to student learning, computing education can adapt to the AI era while preserving what makes it valuable.

As Turing Award winner and AI pioneer Herb Simon said: "Learning results from what the student does and thinks, and only from what the student does and thinks. The teacher can advance learning only by influencing what the student does to learn." While the integration of AI into the discipline of CS does bring its challenges, we should not lose focus on our students and their learning. This will guide us as we move forward. If we focus on what students need to do, the rest will follow.

Thanks to Reviewers:

- Nate Derbinsky
- Matt Handley
- Q McCallum
- Yonatan Kogan
- Julia Mossbridge
- Evan Peck
- JD Pirtle
- Leo Porter
- Kelly Shaw

References

Ambrose, S. A., Bridges, M. W., DiPietro, M., Lovett, M. C., & Norman, M. K. (2010). *How Learning Works: Seven Research-Based Principles for Smart Teaching*. Jossey-Bass.

Hak, J., Johnson, N., Amoozadeh, M., Alipour, A., and Chattopadhyay, S. 2025. Observing Without Doing: Pseudo-Apprenticeship Patterns in Student LLM Use. In Proceedings of the 25th Koli Calling International Conference on Computing Education Research (Koli Calling '25). Association for Computing Machinery, New York, NY, USA, Article 28, 1–11. https://doi.org/10.1145/3769994.3770027

Prather, J., Reeves, B. N., Leinonen, J., MacNeil, S., Randrianasolo, A. S., Becker, B. A., Kimmel, B., Wright, J., & Briggs, B. (2024). The Widening Gap: The Benefits and Harms of Generative AI for Novice Programmers. In *ACM Conference on International Computing Education Research (ICER)*, 469–486.

Roediger III, Henry L., and Jeffrey D. Karpicke. "Test-enhanced learning: Taking memory tests improves long-term retention." Psychological Science 17.3 (2006): 249-255.

Tomkin, Jonathan H., et al. "Evidence that communities of practice are associated with active learning in large STEM lectures." International Journal of STEM Education 6.1 (2019): 1-15.

Vadaparty, A., Zingaro, D., Smith, D. H., Padala, M., Alvarado, C., Gorson Benario, J., & Porter, L. (2024). CS1-LLM: Integrating LLMs into CS1 Instruction. In Proceedings of the 2024 on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2024). Association for Computing Machinery, New York, NY, USA, 297–303. https://doi.org/10.1145/3649217.3653584
Yahav, E. (2022). The Fundamental Theorem of Program Synthesis. https://x.com/yahave/status/1660729517210476544

Zilles, Craig B., et al. "Every University Should Have a Computer-Based Testing Facility." CSEDU (1). 2019.