Reshaping Undergraduate Computer Science Education in the Generative AI Era (Version 1.0)

This whitepaper is the report of the NUS-Google Workshops
Organized by the AI4SG Lab, NUS School of Computing, SoC Horizons Office, and Google

Executive Summary

Generative AI represents a turning point for Computer Science (CS) education [11]. In recent decades, post-secondary CS education has largely focused on what has been seen as practical software engineering skills: implementation-level programming, debugging, testing, and software design, analysis, and documentation. However, this framing is becoming less tenable as generative AI automates many of these tasks, challenging their centrality in CS education.

Recent findings highlight a growing gap between what universities teach and what the software industry needs [10, 28]. AI coding tools like Claude Code, OpenAI Codex, and Google Gemini can now automate and streamline labor-intensive tasks traditionally assigned to junior roles, such as fixing simple bugs, unit testing, maintaining the build system, writing boilerplate code, and documentation. This shift is disrupting the job market for new graduates, reducing the number of entry-level roles. While the extent to which this contraction is driven by AI versus broader macroeconomic factors (e.g., post-pandemic hiring corrections, rising interest rates) remains an open empirical question, the trend is widely acknowledged across industry and academia. Indeed, the AI-driven transition requires the workforce to develop skills to work and communicate with AI proficiently, effectively, and ethically.

To keep pace with advances in AI technology, CS curricula should consider a shift toward understanding and verifying AI-generated artifacts. This white paper outlines the findings of two international NUS-Google Workshops (November 2025[1] & January 2026[2]) in Singapore, where we convened faculty members, industry practitioners, and students, and proposes a strategic response to reshape how CS should be taught at the undergraduate level. Based on the findings, we identify critical skills that must be preserved and those that are becoming less important. By incorporating these skills as "breadcrumbs," we can provide helpful nudges and engaging exercises within the current curriculum, enhancing learning experiences for everyone. We believe that to effectively prepare future computer science graduates, capable of creating, solving problems, and managing, as well as co-creating, artifacts with AI. It is important to consider a shift in curricula. Emphasizing system design, abstraction, and critical evaluation could greatly enhance their education and readiness for

[1] https://futuredu.ai4sg.org/
[2] https://futuredu.student.ai4sg.org/

the challenges ahead. We propose prerequisites for solutions to reform CS education by fostering AI-native competencies, re-centering fundamental education, enhancing advanced pathways, embracing new pedagogies, and shifting institutional support.

Contents

1 Background **4**
- 1.1 The Historical Context of Computer Science Pedagogy . . . 4
- 1.2 Concurrent Initiatives and Positioning . . . 5
- 1.3 AI Disruptions on Computer Science Education . . . 5

2 Methodology: The Workshops **7**
- 2.1 First Workshop: Faculty-focused Version . . . 7
- 2.2 Second Workshop: Student-focused Version . . . 9
- 2.3 Limitations . . . 10

3 Summary of Outcomes **11**
- 3.1 Faculty Workshop . . . 11
  - 3.1.1 Capabilities of Future Software Engineers . . . 11
  - 3.1.2 Reshaping the Curriculum . . . 12
  - 3.1.3 Challenges with Implementing Curricula Change . . . 14
- 3.2 Student Workshop . . . 14

4 Consensus and Divergences between Faculty and Students **16**
- 4.1 Areas of Consensus . . . 16
- 4.2 Differing Areas of Concern . . . 17

5 Discussion **18**
- 5.1 What Are AI Disruptions in CS Education? . . . 18
- 5.2 How Do Students Learn CS in the Era of AI? . . . 19
- 5.3 Nuanced Implementation of Pedagogical Changes . . . 20

6 Criteria that Any Solution Must Meet **21**
- 6.1 Fostering AI-Native Competencies . . . 21
  - 6.1.1 Working Professionally and Ethically . . . 22
  - 6.1.2 Collaborating with AI . . . 23
- 6.2 Re-centering Fundamental Education . . . 24
- 6.3 Enhancing Advanced Pathways . . . 25
- 6.4 Embracing New Pedagogies and Teaching Strategies . . . 26
- 6.5 Shifting Institutional Support . . . 27

7 Conclusion and Next Steps **29**

8 Whitepaper Contributors **30**

9 Workshop Organizers **31**
9.1 Faculty Workshop Organizers 31
9.2 Student Workshop Organizers 31

10 Faculty Workshop Attendees **32**

11 Student Workshop Attendees **33**

1 Background

1.1 The Historical Context of Computer Science Pedagogy

The current disruption in CS education can be understood within the historical context of code scarcity. Traditionally, the curriculum was built on the premise that writing a program was a labor-intensive manual translation of logical intent into rigid syntactic structures [13, 17]. Consequently, pedagogy emphasized translating ideas into code (or programming mechanics), alongside core foundations such as data structures, algorithms, and system-level abstractions. The implicit assumption was that the struggle to master programming logic was the primary vehicle for developing computational thinking.

Over time, the emphasis has shifted from programming to software engineering. Whereas programming is the discrete act of producing code to solve a specific problem, software engineering involves systematic development, operation, and maintenance of software over time [35]. This includes maintaining codebases, managing complexity at scale, and navigating changing requirements and human collaboration.

Notably, one strand of CS pedagogy that has always operated above the level of syntax is the teaching of type systems and formal verification. Type systems train students to reason about program correctness through abstraction—specifying what a program should do rather than how it produces output—and have long served as a pedagogical bridge between mathematical reasoning and practical software construction. Similarly, formal verification shows that machine-checkable proofs of correctness can provide guarantees that no amount of testing or code review can match. As AI-generated code proliferates, these techniques become rather *more* important: when humans are no longer the primary authors of code, the ability to specify intended behaviour precisely and to verify that implementations conform to those specifications is arguably the most durable technical skill a CS graduate can possess. Yet type systems and verification have historically been treated as advanced or elective topics rather than foundational ones.

Furthermore, new software systems will include AI components that are non-deterministic and may be trained on data distributions that change over the software's deployment lifetime. Evaluation and maintenance of such software systems are typically not taught in standard undergraduate curricula, even as elective topics. The current disruption is occurring rapidly, making these curricular gaps urgent to address.

1.2 Concurrent Initiatives and Positioning

This white paper builds on a rapidly growing body of concurrent initiatives addressing the impact of generative AI on CS education. The ACM/IEEE-CS/AAAI *Computer Science Curricula 2023* (CS2023) [21], endorsed in early 2024, represents the current standard for undergraduate CS curriculum guidelines. CS2023 expanded AI as a core knowledge area—from zero Tier-1 hours in CS2013 to substantial required coverage—and introduced a competency-based framework alongside the traditional knowledge model [15]. Our proposals build upon and extend this framework, particularly in articulating a "verification spectrum" as a graduate competency that CS2023 does not explicitly foreground.

Several parallel efforts overlap with the scope of this white paper. The UC San Diego GenAI in CS Education Consortium[3], offers courses integrating GenAI into CS curricula and has been adopted across North America, Europe, and Africa. The ACM GenAI Task Force[4] published its formal report documenting approaches to teaching programming in the GenAI era. The Computing Research Association's LEVEL UP AI Report (September 2025)[5], produced through 32 roundtables with over 202 experts, charts strategies for expanding AI curricula in the United States.

Our white paper offers several contributions that complement these initiatives. First, our dual-workshop methodology captures both faculty and student perspectives, revealing areas of consensus and divergence that single-stakeholder approaches miss. Second, this paper proposes criteria for program-wide restructuring across all four undergraduate years.

1.3 AI Disruptions on Computer Science Education

The advancement of LLMs and AI-powered coding assistants has altered this landscape [27]. Tools such as Claude Code, OpenAI Codex, and Google Gemini provide automated support for many core tasks that junior engineers[6] have previously performed [31].

Some early empirical studies suggest that AI-assisted code generation can yield productivity gains for certain tasks. For instance, Peng et al. [25] found that developers using GitHub Copilot completed an experimental HTTP server task approximately 55% faster than the control group, though these findings are context-dependent and may not generalise across all development activities. Routine tasks such as writing boilerplate code, refactoring simple functions, generating test cases, and debugging common errors are now handled by AI with increasing competence. As AI increasingly

[3] https://www.teachcswithai.org/

[4] https://acm-education-genai-task-force.github.io/

[5] https://cra.org/new-cra-level-up-ai-report-charts-a-path-to-expand-ai-education-nationwide/

[6] Throughout this white paper, we use the term *"junior engineer"* in two related senses. In industry, it refers to an entry-level software engineering role whose responsibilities centre on well-scoped, lower-complexity tasks: fixing simple bugs, writing boilerplate code, generating test cases, maintaining build systems, and producing documentation. As a metaphor for AI, it captures the observation that current AI coding tools can perform these same tasks with increasing competence, yet consistently fall short on higher-order skills such as system architecture, verification, security reasoning, and long-term maintainability [2]. The educational implications of this dual usage are significant: if AI occupies the functional role of a junior engineer, then the traditional apprenticeship pathway through which graduates developed expertise by performing precisely these tasks is disrupted.

automates the translation of logic into programs, the bottleneck of software development is shifting. Software engineers can now offload the burden of rote syntax to AI [36], allowing them to focus their efforts on higher-order engineering skills.

However, productivity gains from AI tools may be context-dependent and subject to systematic overestimation. A randomized controlled trial by Becker et al. [5] found that experienced open-source developers actually took 19% longer to complete real tasks when using AI tools, despite believing they were 20% faster, which shows a striking gap between perception and reality. These findings suggest that the productivity benefits of AI coding tools may be most pronounced for less experienced developers or less complex codebases, rather than uniformly applicable. Emerging empirical evidence suggests that the productivity gains from AI coding tools may come at a significant cost to software quality and maintainability. He et al. [18] conducted a large-scale causal study of Cursor adoption across open-source GitHub projects using a difference-in-differences design. While they found that Cursor adoption led to a statistically significant increase in short-term development velocity, this speedup was transient. More critically, adoption was associated with a substantial and persistent increase in static analysis warnings and code complexity: factors that the authors identify as drivers of long-term velocity slowdown. These findings underscore a central tension: AI tools can accelerate code production, but without rigorous human oversight, the resulting code accumulates technical debt that erodes the initial productivity gains.

Thus, the central challenge for CS education is not whether students will use AI, but how they will learn to work with and supervise it effectively. Doing so still requires today's students to learn a strong conceptual understanding of computation and software systems. Historically, code production served as a mechanism for students to develop, exercise, fine-tune, and demonstrate their understanding of and ability to apply these concepts. Today, however, as AI systems increasingly generate code, students get fewer and fewer of these experiences.

Most critically, students' use of AI is undermining their intellectual skills. When a class assigns a programming assignment to a student who uses AI to generate the program (whether instructed to do so or not), the student gets less practice in problem-solving and critical thinking, and research shows that these habits disappear from students' practices remarkably quickly [8, 9, 16, 22, 33]. If universities graduate students without the skills to think critically and problem-solve about software systems (whether written by AI or not), how can they responsibly manage such systems? That is the problem that universities must now address. This concern is supported by recent causal evidence. Bastani et al. [3] conducted a randomized controlled trial with nearly 1,000 students and found that unrestricted GPT-4 access, while improving practice performance by 48%, led to a 17% decline in exam scores when AI was subsequently removed. Crucially, a guardrail version designed to provide hints rather than direct answers mitigated these negative effects—suggesting that the pedagogical design of AI integration, not AI use per se, determines learning outcomes.



Figure 1: Our workshops are initiatives by the AI4SG Lab, at the NUS School of Computing, and Google

2 Methodology: The Workshops

In response to advances in AI, we sought to gather voices from a variety of stakeholders in CS education. In the first workshop, we brought together CS faculty members and IT industry practitioners from around the world to identify problems, assumptions, and opportunities arising from the advent of AI, and to brainstorm the fundamental needs and solutions for the future of CS education. The second workshop included current CS students and recent graduates in CS-related fields as participants to solicit views from junior practitioners and to understand their lived experiences as the generation most impacted by this AI paradigm shift.

2.1 First Workshop: Faculty-focused Version

In November 2025[7], we first organized a three-day workshop that brought together CS faculty members and IT industry practitioners from around the world. The workshop had 30 attendees, with a diverse range of participants from North America, Asia, Europe, and Oceania.

Day 1: The workshop commenced with a warm-up discussion for early arrivals on the first day. This session was dedicated to identifying foundational concepts that have become increasingly critical, as well as subjects that may now be considered “obsolete” or warrant reduced emphasis. Specifically, attendees were divided into three separate groups to discuss the topic across four categories: KEEP, CUT, GAP, and THREAT.

Day 2: Moving into the main workshop on the second day, the objective shifted to architecting the core content ("what") of the new curriculum, progressing from problem statements to defining guiding principles and developing concrete models. The attendees were divided into five groups to address a general question: **What are the core competencies for a Computer Science graduate in 2030?** Each group were also assigned theme-specific questions. Group members were assigned based on expertise and diversity to ensure a range of perspectives were represented in each group.

- Foundational & Theoretical Skills (Group 1): *What core CS theory, topics, or knowledge cannot be replaced?*
- The New Software Engineer Role (Group 2): *What practical skills (code review, systems integration, testing, prompt engineering) define the new SE role?*
- AI-Native Competencies (Group 3): *What new, specific AI skills are required (e.g., applied ML, AI systems design)?*

[7] https://futuredu.ai4sg.org/



Figure 2: Photo from Day 2. The faculty workshop features multiple rounds of group brainstorming activities.

- Professional & Ethical Skills (Group 4): *What soft skills (ethics, security, communication, collaboration) are now paramount?*
- New Pedagogies & Teaching Strategies (Group 5): *How and what we teach to foster these new competencies?*

Day 3: The final day of the workshop continued the discussion of topics stemming from the conclusions reached on Day 2. The results revealed overlapping outcomes among groups. Consequently, we reorganized the groups to facilitate a more focused discussion. The central objective of the discussion was to determine the structure of the four-year curriculum, drawing upon the established "Graduate Profile" as the guiding framework.

- Group 1: The New Core (Years 1-2): This group redesigned the introductory sequence. The group brainstormed solutions for teaching CS101 in the era when AI can write the code.
- Group 2: Advanced Pathways (Years 3-4): The group was assigned to propose solutions for teaching computer science for new advanced tracks or specializations.
- Group 3: The AI-as-Collaborator Policy: The group was assigned to define AI's role in education, the AI usage policy, and new assessment methods that are AI-resistant or AI-inclusive.
- Group 4: The Ethics & Societal Impact Thread: The group designed a guideline for ethics, security, and societal impact that integrates into every year of the curriculum, beyond just one course.

In the afternoon session on Day 3, attendees gathered in groups based on their topics of interest to draft proposals to reform the CS curriculum.

- Foundational Research for the Next-Generation SE (Group 1): Aligned with Day 1's "Foundational Skills" and "New SE Role", this group defined the new science required to build, verify, and secure the complex AI-native systems that graduates will be asked to create (e.g., verifiable systems, AI-driven SE, ethical auditing).
- The Co-evolving Talent & Infrastructure Pipeline (Group 2): This group redesigned the academia-industry interface beyond internships to create a dynamic, shared ecosystem for both talent and technology (e.g., new collaborative models, shared infrastructure, continuous feedback loop).
- The Research Agenda for AI-augmented Pedagogy (Group 3): This group moved beyond a simple "best practices" guide to define the fundamental research needed to build, deploy, and understand AI's role in education (e.g., cognitive impact, assessment innovation, co-educator development).
- The Faculty & Institutional Action Plan (Group 4): To complement Day 1's vision, this group envisioned institutional change and faculty development, together with potential barriers, policies, incentives, and resource allocation.

2.2 Second Workshop: Student-focused Version

As a follow-up, in January 2026[8], we convened a one-day workshop to explore the evolving landscape of computer science education in the age of AI. The event brought together 60 participants, including students and recent graduates from across the globe, with approximately half of the cohort representing Singapore-based institutions. This diverse mix of local and international perspectives provided a comprehensive foundation for tackling the complexities of modernizing CS curricula. Attendees were divided into eight working groups (Groups 1–8) assigned to four critical discussion themes:

- Challenges & Response (Groups 1-2): Focused on disruption and institutional adaptation.
- CS Education Fundamentals (Groups 3-4): Targeted the foundational undergraduate years 1 and 2.
- Advanced Pathways (Groups 5-6): Addressed the specialized needs of undergraduate years 3 and 4.
- Profile of the Modern CS Graduate (Groups 7-8): Concentrated on graduate-level competencies and industry alignment.

To ground the sessions, each group was provided with a tailored set of questions based on the findings from our inaugural workshop. These prompts guided the attendees through intense

[8]https://futuredu.student.ai4sg.org/

brainstorming and collaborative problem-solving, culminating in a series of group presentations where each team shared their synthesized findings.

2.3 Limitations

We view these workshops as a starting point for an ongoing, evidence-driven conversation rather than a definitive conclusion. However, we acknowledge several limitations of our workshops. First, workshop participants were self-selected and likely already engaged with questions of AI in education, introducing potential selection bias. Second, while the faculty workshop included international participants from North America, Asia, Europe, and Oceania, the student workshop drew approximately half its cohort from Singapore-based institutions, limiting geographic generalizability. Third, workshop findings reflect participants' perceptions and proposals rather than empirically validated outcomes; the proposed solutions remain to be tested. Fourth, industry perspectives were represented primarily through individual practitioners rather than systematic employer surveys, and a broader range of industry input would strengthen the findings.

3 Summary of Outcomes

3.1 Faculty Workshop

3.1.1 Capabilities of Future Software Engineers

The discussion on Day 1 focused on preparing students for a world where AI has the capabilities of a "junior engineer", as defined in Section 1.2. Participants generally agreed that the educational focus should shift towards higher-level skills. Three key examples raised were abstraction, verification, and taking responsibility for systems.

- *Abstraction*: Abstract computational thinking gains importance when code production becomes cheaper. The ability to understand computational structures that span programming languages, architect systems, and analyze complex AI models and their limitations is what differentiates skilled software engineers from mere programmers.
- *Verification:* Participants agreed that AI models effectively act as junior software engineers, capable of producing code but often making suboptimal architectural decisions. Verification thus serves as a crucial skill in which humans play the role of a "senior software engineer". Importantly, “verification” in this context encompasses a spectrum of rigor: from informal code review and testing, through static analysis and type-system-guided development, to full formal verification using proof assistants. This verification spectrum aligns with the three-level framework proposed in the formal methods education literature [14], which argues that such thinking can be embedded within existing courses without requiring additional curricular expansion.
- *Taking responsibility for systems:* With AI lowering the barrier to building systems, participants raised the legal, moral, and ethical responsibilities associated with such systems as issues that remain squarely in the human domain. Ethical responsibilities include designing systems for human betterment, such as AI augmentation over replacement. Legal responsibilities include potential accreditation and liability for systems, particularly when AI generates code for safety-critical applications. As systems become increasingly easy to produce through automation, the need for deliberate, thoughtful, and responsible system design and production gains renewed importance.

The consensus was that as creating code becomes cheaper, the ability to discern quality, ensure security, and maintain ethical standards becomes even more valuable. Teaching students to operate across the verification spectrum is also essential if graduates are to take genuine responsibility for AI-generated systems. Consequently, the traditional pathway in which junior engineers gradually build expertise through progressively complex tasks may need to be compressed or restructured. Students will likely need to engage with architectural thinking and integration challenges earlier in their education, though doing so effectively will require carefully scaffolded learning experiences rather than a simple bypass of foundational stages.

3.1.2 Reshaping the Curriculum

Shifting the Core (Years 1-2). The key shift is to move introductory CS learning goals from low-level syntax and rote coding to higher-order computational thinking grounded in computing fundamentals, while focusing human-centric skills that cannot be automated. Key features include, but are not limited to:

- *Embracing AI integration:* AI tools are intentionally integrated from the start to enable students to work on larger, more motivating projects, thereby reducing low-level workload and avoiding early student demotivation.
- *Computational Thinking and theory:* The new core seeks to foster fundamental and theoretical skills, emphasizing Computational Thinking as its foundation. It focuses on the enduring importance of core CS theory, such as abstraction, decomposition, and algorithmic reasoning, and fundamentals such as problem-solving, scientific reasoning, and learning agility. It proposes teaching up the abstraction ladder, allowing students to delegate low-level coding tasks to AI (after learning the fundamentals) and instead focus on problem specification and problem solving. Type-driven development and lightweight formal specification (such as preconditions, postconditions, and contracts) offer an analogous pedagogical mechanism for achieving this shift: they train students to express intent precisely and to reason about correctness at the specification level rather than the syntax level, which is a similar type of skill required to direct and verify AI-generated code effectively.
- *Human-centric skills:* At the same time, crucially, soft skills such as communication, collaboration, ethics, and societal reasoning are important core CS competencies, rather than peripheral courses. These are vital for understanding and communicating “what to build” and its broader implications; students must be equipped to justify systems and assess their impact, rather than just implementing them.

Integrating AI into Advanced Pathways (Years 3-4). Participants agreed that AI could be a thread woven throughout the curriculum, not a standalone add-on. This vision focuses on upgrading existing specializations, such as human-centered computing, security, and computer architecture, by adding AI-focused components, resulting in "AI-enabled" or even "AI-native" versions of these tracks. Detailed examples included:

- *The new software engineering role in AI:* This addresses the practical skills needed for the evolving software engineering profession. It advocates for AI-ready assignments and experiential learning, proposing that students work on large, real-world, open-source projects that utilize AI tools. Such assignments could also incorporate verification-oriented workflows, where students use AI to generate code and then apply static analysis, property-based testing, or formal verification tools to validate the output—mirroring the quality assurance challenges that He et al. [18] have shown to be a major bottleneck in real-world AI-assisted development. Students will also need to learn how to incorporate AI components into software systems and evaluate their effectiveness [30].

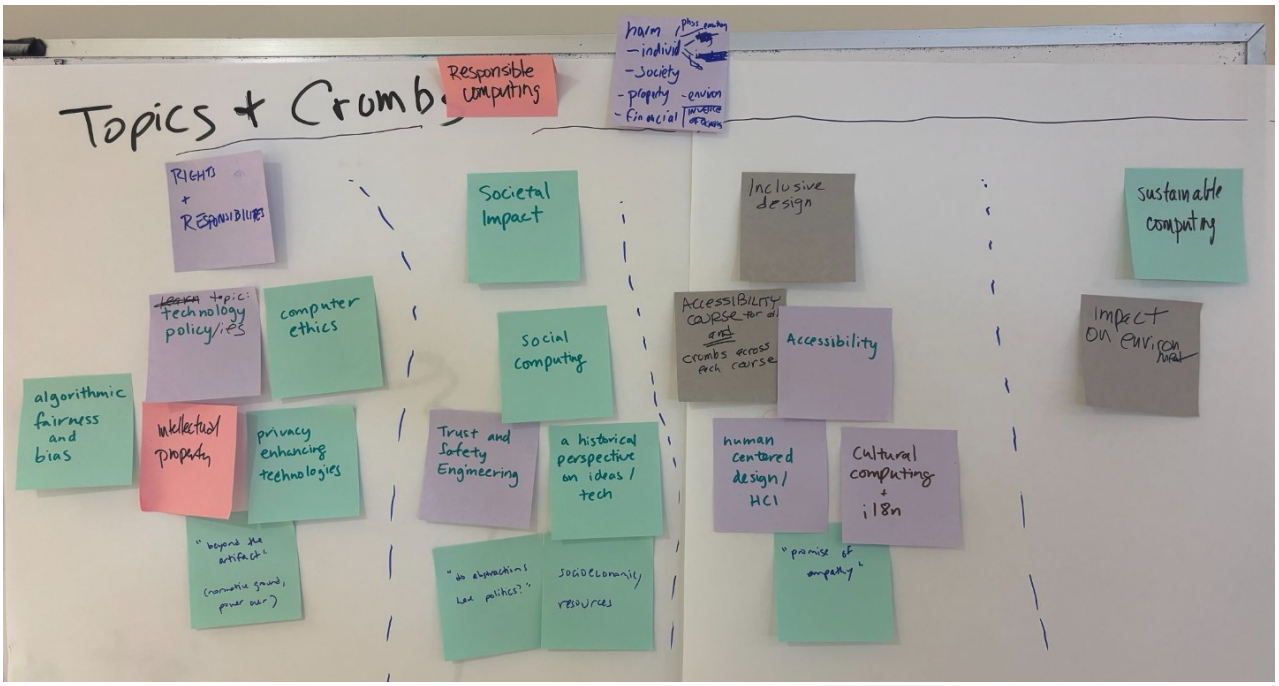


Figure 3: Faculty participants proposed the Breadcrumb strategy as a lightweight solution to foster students' AI-native skills. Small breadcrumbs (i.e., skills or topics) can be integrated progressively throughout the curriculum using small practical exercises, prompts, or nudges. This figure shows a part of the AI-native skill set brainstormed in the workshop.

- *Mathematical foundations and Responsible AI:* Strengthening connections between mathematics and AI ensures students understand the technical basis of responsible AI. Topics such as probability, linear algebra, and discrete mathematics, often taught as core courses, could be explicitly linked to AI applications rather than taught in isolation. At the same time, educators can emphasize ethical awareness and frame mathematical training to support responsible AI practices while recognizing that technical solutions alone cannot address all ethical challenges.
- *Evaluation of AI tool usage:* Attendees highlighted the importance of assessing the process of how students interact with AI for coding and the problem-solving process, shifting away from merely evaluating the final product. The end goal is to produce employable graduates with skills that differentiate them in the AI-supported workforce.

Embedding Skills through the Breadcrumb Strategy. Skills such as ethics, social & ecological impact, and responsible AI can be integrated throughout the four-year curriculum. Workshop attendees proposed the "Breadcrumb" strategy, which integrates small, practical exercises, nudges, or prompts into multiple courses — through assignments, exams, or projects. It is aimed at building students' ability to evaluate impacts, follow accessibility and legal standards, and make responsible design choices. In practical implementation, Breadcrumbs can be sequenced to build on previous knowledge, reinforcing learning year by year, and designed to fit diverse courses (theory to software engineering).

3.1.3 Challenges with Implementing Curricula Change

Supporting Faculty and Institutional Action Plans. The key to encouraging adoption is faculty engagement. Strategies such as opt-in participation and incentives (e.g., recognition, small funding) may help mitigate resistance. It is crucial to address instructor skepticism by framing AI adoption as low-risk experimentation. Effective adoption requires significant institutional support and resourcing. This includes securing budget for AI tools, providing dedicated staff and lab technicians to assist faculty, and ensuring management buy-in by aligning AI integration with core institutional priorities, such as student outcomes and reputation.

To maintain relevance, curriculum adaptation and teaching strategies must evolve. Faculty should be encouraged to rethink course content as AI automates certain skills, to engage in practical experimentation with AI-assisted assignments, and to utilize resource repositories to share best practices. In addition, industry collaboration and resource sharing were deemed vital, advocating for a consortium-based approach to share computational resources and emerging technologies. Encouraging companies to support open access for education can help universities bridge the gap between resource constraints and the fast pace of AI development. Finally, the impact on student experience and institutional outcomes requires careful management. This involves striking a balance between teaching innovation and evaluating metrics, while ensuring that AI integration maintains high educational quality and prepares students for a rapidly transforming job market.

3.2 Student Workshop

The Evolving Role of AI and Student Concerns. Student participants viewed AI as a multifaceted tool, functioning as an accessible 24/7 co-pilot, partner, and mentor for tasks such as academic writing, prototyping, simplifying complex topics, and accelerating workflows. It enables personalized and on-demand learning. However, the groups expressed significant concern about the erosion of critical thinking, overreliance, and student laziness. Concerns also centered on academic dishonesty (plagiarism, using AI to produce “fake answers”) and the need to teach students how to verify and reflect on potentially unreliable AI outputs (e.g., hallucinations). The consensus was that AI’s role is *fluid*: a shortcut for tasks with clear, predictable outcomes (e.g., information search, simple automation) and a collaborator for uncertain, creative tasks where students must inject their own critical thinking and opinions.

Re-prioritizing Foundational and Soft Skills. The discussions highlighted that CS fundamentals—including strong basic math, data structures and algorithms, debugging concepts, and knowledge of operating systems and networking—have become more essential than ever as a core base for understanding and applying AI-generated code. Alongside these technical foundations, student groups stressed the importance of new cognitive and soft skills. Cognitive skills include learning how to learn, how to ask, and, in a provocative turn, “how to unlearn.” Essential soft skills, such as responsibility, ethical thinking, business awareness, humility, and caution, were identified as qualities that truly make a CS graduate stand out, even as AI can handle average-level and advanced technical problems. Students identified a combination of emerging and established technical compe-



Figure 4: Example brainstorming work from the student workshop. Students voiced opportunities, excitement, and concerns about the advancement of AI with regard to CS education.

tencies that are crucial for modern IT roles. Among the new skills highlighted is prompt engineering, although participants noted that its long-term relevance may be uncertain as AI systems continue to evolve. Just as important are well-established software engineering practices that gain renewed significance in the AI era. These practices include specification writing (defining the desired system behavior that will be later transferred into code), critical code review (assessing AI-generated output for correctness, security, and maintainability), and troubleshooting edge cases (distinguishing failure modes that AI tools often overlook). This distinction is important: while these latter skills are not new, their importance in a graduate's profile has increased significantly, especially when code is generated by AI rather than written from scratch.

Changes to Curriculum and Assessment. Participants advocated for major shifts in CS education away from traditional models focused on memorizing syntax. Recommendations for a curriculum shift include moving towards project-based learning to apply concepts in real-world, "in the wild" environments, introducing algorithms earlier in the curriculum, and incorporating more explicit instruction on AI ethics, legal compliance, and the social science aspects of technology. For assessment reform, the focus can shift from evaluating only the final answer to assessing the problem-solving process, understanding of errors, and debugging. Proposed assignment formats include presentations, "paper-like" work where students take authorship, and, notably, submissions of the history of their interactions with generative AI to ensure genuine understanding.

4 Consensus and Divergences between Faculty and Students

Both workshops demonstrated a shared understanding that AI is fundamentally changing the landscape of CS education, necessitating a comprehensive shift in pedagogy and curriculum focus. In this section, we discuss areas of consensus between the two workshops and the nuanced differences between them.

4.1 Areas of Consensus

- **Shifting from low-level to high-level.** There was a clear consensus that instruction must move away from memorizing syntax and mechanical coding and pivot toward mastering high-level concepts, abstraction, and problem decomposition.
- **Theoretical foundations to verify AI outputs.** Both workshops affirmed the enduring importance of core CS theory, such as mathematics, data structures, and algorithms. These foundational skills are deemed essential for a human to critically verify and debug the output generated by AI. Notably, neither workshop drew a clear distinction between informal verification (e.g., code review, testing) and formal or semi-formal approaches (e.g., type checking, static analysis, specification-based reasoning), suggesting that the precise meaning of “verification” as a graduate competency remains underspecified and warrants further clarification in curriculum design.
- **Fostering meta skills.** All agreed that new "meta-skills" are now paramount, though they emphasized different aspects: faculty focused on professional, human-centered competencies (i.e., human-computer interaction, communication, ethics, professional liability), while students highlighted personal intellectual skills (for example, critical thinking, self-regulation, and metacognition).
- **AI-native assessment.** Both workshops recognised a strong shared conviction that assessment practices must evolve to measure authentic understanding in an AI-enabled learning environment, with both proposing a dual approach of controlled (AI-free) testing and open (AI-allowed) assignments.
- **Project-based and experiential learning.** Both workshops converged on the value of engaging students with real-life, larger-scale projects rather than small, isolated programming exercises. Faculty participants proposed “reverse-engineering” methods, in which students extract specifications and requirements from existing compiled solutions. This approach serves as a pedagogical process for sense-making, conceptual learning, and verification learning, grounded in real-world codebases. Students advocated for project-based learning applied “in the wild.” Though the pedagogical framing differed, the shared conviction that authentic, large-scale projects should anchor the curriculum was a clear area of consensus.

4.2 Differing Areas of Concern

While unified on the broad direction, the workshops differed in their primary concerns and proposed solutions.

- **Junior-Senior competency gap.** Faculty primarily focused on professional competency gaps between entry-level and senior roles. They mentioned the "Junior Engineer problem," which requires new graduates to immediately possess "Senior Engineer" architectural skills. Conversely, the student workshop approached this problem more from an employability standpoint, seeking ways to stand out in an age of AI coding agents.
- **Professional liability of AI.** The faculty workshop voiced institutional challenges like the lack of professional liability for AI-generated code or challenges with assessment and teaching, when AI can perform assessment. The professional liability concern is particularly acute in safety-critical domains, where AI-generated code is introduced into systems without the formal sign-off processes that exist in disciplines such as civil or electrical engineering.
- **Pedagogical mechanisms.** While both workshops valued project-based and experiential learning, their views were different in how to structure it. The faculty workshop proposed specific instructor-driven curricular formats such as "prompt-first" courses and structured "reverse-engineering" modules with defined learning scaffolds. The student workshop, by contrast, favored greater student autonomy in selecting real-world problems and learning pathways, with less emphasis on prescribed pedagogical structures. The divergence is thus not about *whether* to use real-world projects, but about the degree of structure and instructor direction in how those projects are framed.

5 Discussion

5.1 What Are AI Disruptions in CS Education?

The transformative influence of AI presents challenges to established norms in CS professional progression, the integrity of assessment, and the foundational understanding of professional competence. Faculty participants agreed that the advancement of AI fundamentally disrupts the traditional learning path of computer science students. As defined in the prior section, “junior engineer” in this white paper refers both to the entry-level industry role centered on well-scoped, lower-complexity tasks and to the observation that AI tools now perform these same tasks with increasing competence. AI systems increasingly function as entry-level “junior engineers”: they can generate working code quickly, yet they routinely make poor architectural, security, or performance decisions. This disrupts the historical apprenticeship model in which students gradually build expertise through progressively complex tasks and evolve toward senior responsibility.

Large-scale evidence supports this concern: Brynjolfsson et al. [6] analyzed ADP payroll data covering millions of U.S. workers and found that employment for software developers aged 22–25 declined approximately 16% relative to less AI-exposed occupations since late 2022. The authors suggest that while adjustments following the pandemic may play a role in earlier declines, the significant impact on entry-level positions exposed to AI persists even when accounting for company-specific factors. This indicates that the distinction between "junior" and "senior" responsibilities is indeed evolving. In this context, the importance of low-level syntax drills, memorization of libraries, or even the choice of programming language diminishes, while architectural reasoning, systems thinking, and critical evaluation of AI outputs become central learning outcomes. Evidence is already emerging that this shift is not merely theoretical: He et al. [18] found that Cursor adoption in open-source projects led to a persistent increase in static analysis warnings and code complexity, suggesting that without deliberate quality assurance practices, AI-assisted development accumulates technical debt faster than it generates value.

Even before the rise of AI-assisted development, many computer science students already perceived a gap between what they learned in university and the skills that industry demanded [10]. With the rapid emergence of powerful AI coding tools, this conceptual gap has widened further. Students increasingly observe developers using AI to generate code, debug programs, and explore design alternatives, which can make traditional coursework seem even more removed from real-world practice. At the same time, some companies have become more cautious about hiring junior engineers and, in certain cases, have reduced or even eliminated entry-level positions [28], relying more heavily on experienced developers augmented by AI tools. However, the metaphor of AI as an entry-level engineer warrants careful qualification. Evidence suggests that even experienced developers may systematically overestimate AI’s benefits: Becker et al. [5] found a striking gap between perceived and measured productivity in a randomized controlled trial. This perception gap itself becomes an educational concern, reinforcing the importance of teaching students to critically evaluate not only AI outputs but also their own AI-mediated workflows. This has further reinforced the perception that the entry-level pipeline is shrinking. When students see fewer opportunities for

junior engineers and feel that the skills emphasized in school may not match the evolving demands of industry, their motivation to invest deeply in traditional coursework can decline significantly.

Moreover, participants expressed concerns about assessment practices, learning equity, and student self-regulation. Traditional take-home assignments no longer reliably demonstrate a student's own work, as AI assistance blurs the boundary between independent effort and tool-supported production [23]. At the same time, invigilated examinations increasingly reveal a widening gap: stronger students tend to use AI as a learning accelerator, whereas weaker students may over-rely on it, developing an illusion of competence that fails under time-pressured conditions. This dynamic highlights a serious deficit in metacognitive skills, with many students unable to accurately assess their own understanding and limitations. Although oral examinations could be a promising alternative for assessing authentic understanding [34], the faculty workshop suggested that such approaches are rarely scalable given existing constraints on faculty time and resources. However, recent innovations in AI-assisted oral examination administration, including systems that generate personalized viva questions and conduct AI-mediated oral defenses at low per-student cost, suggest that this scalability barrier may be diminishing [19]. These assessment tensions are further exacerbated by issues of trust, as inconsistent faculty practices around AI use risk undermining the educational relationship between students and instructors.

Beyond coursework, participants identified challenges that extend beyond undergraduate education into professional work and research. As AI-generated code increasingly enters safety-critical systems, the absence of clear professional accreditation and liability frameworks in computer science arguably becomes more salient. Unlike fields such as civil engineering, there is no widely accepted model in which a human professional is explicitly accountable for their work, let alone for AI-assisted outputs. Formal verification offers one pathway forward: if a system's correctness can be established through machine-checkable proofs rather than human judgment alone, the question of professional sign-off shifts from "did someone review this code?" to "does this code meet its specification?" — a far more tractable standard for AI-generated artifacts. These trends point to an urgent need for curricula that emphasize verification, security, performance evaluation, and ethical responsibility, as well as communication and interpersonal skills that remain *uniquely human* and essential to responsible system design.

5.2 How Do Students Learn CS in the Era of AI?

The student workshop highlighted a significant shift in how students learn CS in the era of AI. The consensus is that AI is a powerful tool, but its integration requires a re-evaluation of educational priorities to safeguard and enhance human-centric skills.

Student participants reported incorporating AI into their academic journey. For undergraduate students, AI primarily serves as a 24/7 partner for assignment help. AI is valued for its ability to explain complex topics, provide alternative perspectives, and generate ideas for cross-referencing, effectively lowering the barrier to entry for coding. However, this ease of use sparked significant concern about the erosion of critical thinking, original input, and students' intrinsic motivation, as well as the risk of malicious use, such as plagiarism. These observations align with experimental evidence showing that

the design of AI tools significantly mediates their educational impact: guardrailed AI tutors that provide scaffolded hints rather than direct answers can preserve learning outcomes, whereas unrestricted access creates a “crutch” effect that undermines skill acquisition [3].

The role of AI is seen as dynamic, changing based on the specific task. The groups distinguished between two main roles. AI could be rather treated as a shortcut for tasks with a clearly defined or desired outcome, such as routine information searches, simple automation, or data collection and consolidation. Conversely, students can treat AI as a collaborator for tasks with uncertain or creative outcomes, such as design thinking or creative writing, where students are required to actively inject their own unique thoughts, opinions, and agency into the final product. Ultimately, the groups concluded that the educational system must adapt assessment methods to account for AI’s capabilities, for example, by requiring the submission of AI-use history alongside assignments or by placing greater emphasis on presentations and public peer review to ensure individual responsibility for the work.

5.3 Nuanced Implementation of Pedagogical Changes

The differing perspectives of students and faculty highlight the need to involve a range of stakeholders in such discussions. Changes to curricula must address students’ emotional and personal concerns, rather than simply meeting institutional objectives such as accurate assessment or graduates’ employability. Given that news about "PhD-level AI" [4] and fully-agentic workflows [1] continues to dominate headlines, current and prospective CS students rightly worry about their relevance and employability in the job market. Acknowledging these concerns is central to ensuring student buy-in and, therefore, active participation in any proposed curricular changes. Framing verification and specification skills not as an additional burden but as tools that give graduates a durable competitive advantage over AI—since machines generate code but cannot yet prove its correctness—may help address students’ anxieties about relevance in an AI-dominated job market.

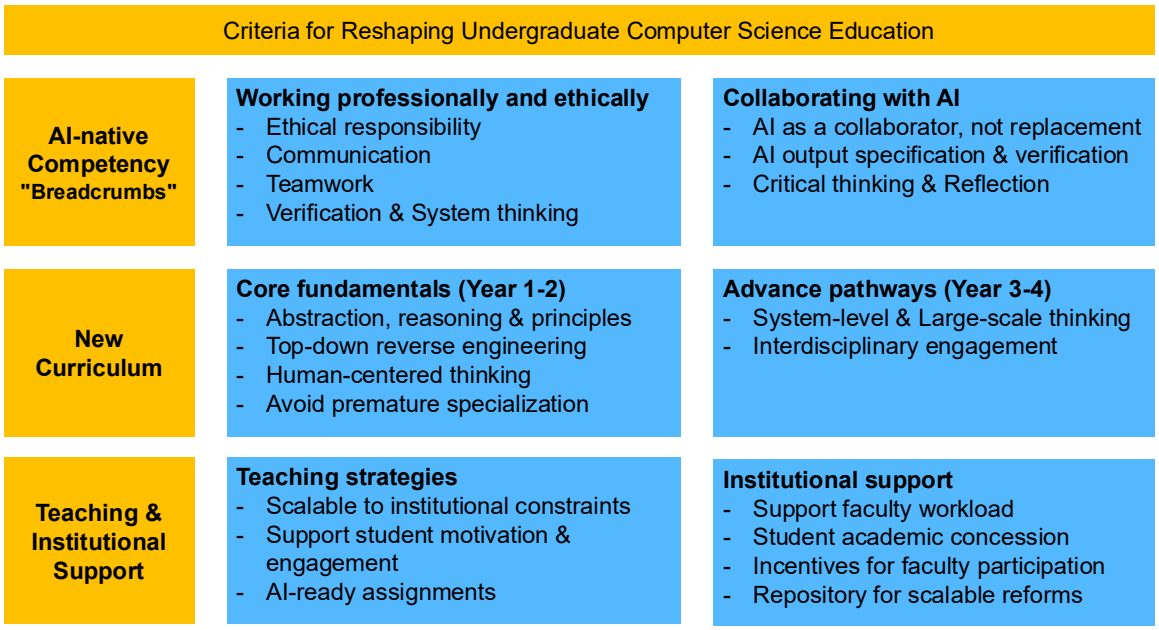


Figure 5: Criteria for CS Education Reforming Solutions

6 Criteria that Any Solution Must Meet

Participants voiced a diverse set of solutions to address the AI disruption in CS education. During the workshop discussion, they noted the requirements that these solutions must meet. We categorized them into four themes: fostering AI-native competencies; shaping strong fundamentals (Undergraduate years 1-2); enhancing advanced pathways (Undergraduate years 3-4); and embracing new pedagogical strategies and shifting institutional support.

6.1 Fostering AI-Native Competencies

Workshop participants reached consensus that CS education must train students to work with AI proficiently, responsibly, and ethically. While AI can perform low-level tasks (such as programming) quickly, it may not excel at high-level tasks, which encompass metacognitive thinking, abstraction, and critical thinking. Current AI systems also cannot verify their own outputs against formal specifications or provide guarantees of correctness, making human competence in specification and verification an AI-native skill in itself—not merely a legacy practice.

Particularly, the issue of ethics was raised in the discussion. In the CS context, ethical considerations go beyond abstract discussions of values. It also includes concrete technical aspects, such as recognizing and mitigating bias in training data, evaluating privacy risks in data collection and logging, assessing security vulnerabilities before deployment, understanding when system failures could cause real-world harm, and deciding whether an AI-generated solution is sufficiently reliable for use

in safety-critical or high-stakes settings. To ensure that downstream applications do not trigger and amplify societal issues, **it is important that CS graduates can decide beforehand whether using AI is the right approach and verify if the AI output is appropriate.** The future of CS education needs to equip students with the "soft skills" or competencies to work together with AI. The faculty workshop proposed the Breadcrumb strategy, which embeds these skills throughout the four-year curriculum. Breadcrumbs can be operationalized as small prompts and practical exercises in lectures, assignments, exams, or projects, reinforcing students' AI-native skills year by year in different courses. Under this theme, we discuss skills for working with AI professionally and ethically, as well as for collaborating with AI efficiently.

6.1.1 Working Professionally and Ethically

Ethics and societal responsibility as core CS competencies. Workshop participants emphasized that ethics, societal impact, and professional responsibility must be treated as foundational skills rather than peripheral considerations. Teaching approaches could, therefore, ensure that students routinely consider the needs of users (including direct users and those otherwise affected or impacted by the systems), accessibility, legal and regulatory constraints, environmental impact, and broader societal consequences in everyday system design and implementation. Ethical and risk-related concerns could be introduced early and revisited throughout the curriculum, reinforcing that responsible judgment about trade-offs, risks, and accountability is inseparable from technical competence, and that both practical and collaborative approachesare often central to addressing such issues.

Communication, specification, and justification. Clear communication was identified as central to AI-era computing. As AI increasingly handles implementation, the human bottleneck shifts from writing code to articulating intent: deciding *what* to build, *why* it should be built, *for whom*, and whether the result is correct. This change transforms communication from a secondary skill into a primary method of engineering control. In practical terms, this involves clearly defining problems, intentions, and constraints. This can be done through natural language prompts for AI tools, formal requirements for human collaborators, or explanations for stakeholders. Solutions involve developing students' ability to specify problems precisely (for example, by identifying and resolving potential ambiguities in problem statements [12]), articulating system requirements, justifing design decisions, and explaining systems and outcomes in accessible language. These skills support collaboration with both humans and AI systems and are essential for ensuring that the "right problem" is being solved, not merely that a system functions.

Teamwork and professional practice. Participants stressed that collaboration, teamwork, and stakeholder engagement are core CS practices. Teaching solutions could make these skills explicit and assessable, rather than assuming they emerge implicitly through group projects. Professional skills shall be intentionally cultivated.

Interdisciplinary engagement and diverse perspectives. As AI systems increasingly affect domains beyond computing, CS graduates must be prepared to collaborate with experts from other fields. This requires not only communication skills but also intellectual humility: the ability to recognize the limits of one's own disciplinary perspective and to engage meaningfully with non-CS stakeholders

who bring essential domain knowledge, ethical frameworks, and lived experience. Teaching solutions can rather create structured opportunities for such cross-disciplinary collaboration, particularly in project-based and capstone settings.

Resilience, responsibility, and systems thinking. Graduates should be prepared to reason about failure modes, energy use, and externalities of AI systems. In safety-critical contexts, this responsibility extends to determining the appropriate level of assurance: whether informal testing suffices or formal guarantees are required, as well as selecting and applying verification methods accordingly. Solutions should help students understand how systems behave under stress or uncertainty and encourage a sense of responsibility for long-term consequences, not just immediate performance.

6.1.2 Collaborating with AI

AI as collaborator, not replacement. Teaching solutions must prepare students to work productively with AI tools in realistic workflows, including code generation, review, debugging, and design assistance. The focus could shift away from merely using AI to produce results to forming a collaborative relationship with AI tools. This means students not only express their intent and control AI outputs but also are aware of, and make the best use of, AI's capabilities and limitations [32].

Specification and verification of AI outputs. A recurring criterion was the importance of students being able to specify desired behavior clearly and to verify and evaluate AI-generated artifacts. This includes understanding when AI outputs are incorrect, incomplete, or misaligned with requirements, and being able to correct or contextualize them. Specification languages, type systems, and contract-based design provide structured vocabularies for precisely expressing desired behaviour, making the gap between intent and AI output explicit and inspectable rather than implicit and ad hoc. As AI increasingly handles low-level syntax, students must focus on abstraction, problem decomposition, and precise reasoning. Teaching solutions could emphasize clarity of thought across representations—natural language, code, and formal models—while maintaining sufficient awareness of lower-level details to reason about failures when abstractions break down. Crucially, students must also consider the *purpose* of verification and evaluation in *context*, including the potential risks and consequences associated with the AI outputs. The convergence of AI code generation and formal verification tools further reinforces this emphasis. As AI-generated code proliferates, formal verification offers a scalable alternative to manual code review, and LLMs are themselves beginning to assist in proof generation [20]. Teaching students to work at this intersection positions them for an emerging AI-assisted verification workflow that may define the next generation of software quality assurance.

Critical thinking and reflection. Participants highlighted the need to strengthen students' ability to question AI outputs, reflect on their own AI use, and maintain attention in AI-rich environments. Reflection on process—not just outcomes—was seen as essential to meaningful learning.

6.2 Re-centering Fundamental Education

Participants expressed that the priority for undergraduate education in Years 1-2 is to equip students with durable intellectual skills that will remain relevant despite rapid technological change. Students must develop strong foundations in algorithms, data structures, and systems thinking; learning emphasis should shift from tool-specific knowledge toward abstraction, reasoning, and underlying concepts and principles. Equally important is cultivating computational thinking and the capacity to "learn how to learn," enabling students to decompose unfamiliar problems, reason algorithmically, and adapt to new domains and technologies over time. Ethical and human-centered thinking could be integrated from the outset, helping students reflect on societal impact, user needs, and responsibility before uncritical automation habits take hold. Finally, early curricula could avoid premature specialization—particularly in fast-moving areas such as machine learning—and instead foster broad conceptual understanding of AI and computational systems, ensuring that students are prepared to make informed choices about specialization in later years rather than being narrowly trained too early.

Emphasizing abstraction, reasoning, and principles. The discussion argued early undergraduate education should retain core CS foundations—algorithms, data structures, systems thinking—but shift emphasis toward abstraction, reasoning, and principles that remain stable as tools evolve. Solutions could avoid superficial coverage of many new topics in favor of depth in broadly applicable foundations. In particular, computational thinking—abstraction, decomposition, and algorithmic reasoning—should serve as a foundation for adaptability. Educators can explicitly support "learning how to learn," preparing junior students for problems and domains not explicitly taught.

Top-down reverse engineering approaches. Existing fundamental education generally starts with small-scale programming tasks, but real-life, we could integrate larger-scale projects into fundamental education, and analysed via "reverse-engineering" methods in CS101 and equivalents. This method would offer assignments that require students to extract specifications and requirements from a completed compiled solution. By working the pedagogy top-down, this will provide opportunities to teach principles and core concepts based on existing examples and customized to the expectations of incoming new generations, who are already relying on AI for problem-solving.

Integrating ethics, legal frameworks, and human-centered thinking. Faculty could introduce human-centered perspectives, including ethics, societal impact, and legal frameworks, in Year 1 rather than postponing them. Offline or discussion-based learning formats were seen as particularly valuable for cultivating reflection, attention, and ethical reasoning, before heavy reliance on AI tools.

Providing breadth whilst avoiding premature specialization. Workshop participants cautioned against forcing all students into deep machine learning early. Instead, Year 1–2 education shall ensure a broad understanding of AI principles and system behavior, including its wide applicability across fields such as social science and law, without requiring mastery of rapidly changing techniques. This may include incorporating modules about the core foundations of language models, statistics, and optimization techniques, with examples of applications in a variety of areas. This should allow students to choose additional specializations based on their interests and desired future career paths.

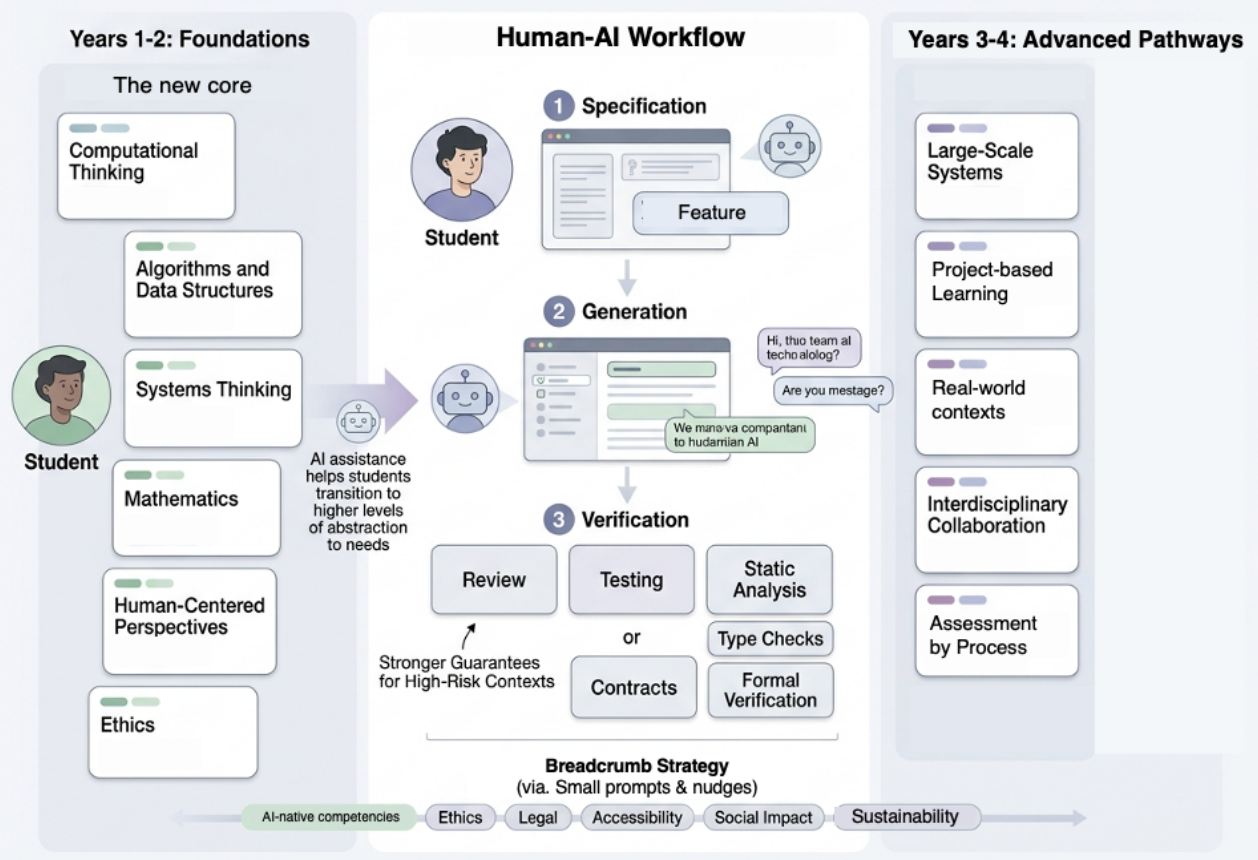


Figure 6: Undergraduate CS curriculum changes must involve recentering fundamental education (Years 1-2) by emphasising principles & human-centered thinking, and enhancing advanced pathways (Years 3-4) through large-scale, practical, and interdisciplinary project and process-based assessments. Meanwhile, AI-native skills can be integrated seamlessly as Breadcrumbs.

6.3 Enhancing Advanced Pathways

In the later years of undergraduate computer science education, workshop participants believed the emphasis should shift from mastering individual techniques to exercising judgment in complex, real-world contexts. Participants highlighted the importance of system-level thinking, with students gaining hands-on experience working with large-scale, distributed, and open-source systems that mirror contemporary software and AI deployments. Project-based and experiential learning—through capstone projects, internships, and applied coursework—was seen as essential for integrating technical depth with ethical reasoning and professional practice. At this stage, students should also be encouraged to engage across disciplinary boundaries, applying computational and AI methods in collaboration with domain experts to address problems beyond traditional computing contexts. Correspondingly, assessment practices must evolve to value process as much as product, evaluating how students reason through design trade-offs, iterate on solutions, reflect on their use of AI tools, and justify their decisions, rather than focusing solely on the final artifact.

System-level and large-scale thinking. In teaching late undergraduate years, faculty members can emphasize working with large-scale, distributed, and open-source systems that reflect real-world software and AI deployment. Solutions should help students understand how components interact across the computing stack and how AI systems behave in production contexts, including the increasingly service-based nature of modern computing systems and the additional complexities that arise from that. This can include experience with the quality-assurance challenges specific to AI-generated code, such as detecting subtle increases in code complexity and technical debt, which empirical studies have shown accompany AI tool adoption [18].

Project-based and experiential learning. Criteria strongly supported project-based learning, internships, and applied experiences in Years 3–4. These formats allow students to integrate technical, ethical, and professional skills while working on complex, realistic problems.

Interdisciplinary engagement. Advanced students should be encouraged to apply AI and computational thinking across domains, collaborating with experts from other fields. This supports transferable skills and reflects how AI is used in practice. Furthermore, computing departments could be positioned as interdisciplinary hubs, allowing cross-domain collaboration. This effort has been piloted in the forms of "AI+X" or "Computing+X" in various institutions (e.g., NUS[9]).

Assessment of process and judgment. Teaching solutions should assess not only final outputs but also students' processes, including iteration, prompting strategies, reflection on AI use, and decision-making. This helps distinguish genuine learning from superficial AI-assisted production.

6.4 Embracing New Pedagogies and Teaching Strategies

Taken together, the shifts toward AI-native competencies and the reform of undergraduate education in both the early and later years make clear that incremental content updates alone are insufficient. If the goals of modern computer science education are to cultivate judgment, adaptability, and responsible human–AI collaboration, then the strategies by which we teach, engage, and assess students must evolve in parallel. The workshop discussions, therefore, converged on the need for a fundamental rethinking of pedagogical approaches—one that aligns institutional realities with these new educational objectives. The following teaching strategies articulate how such changes can be enacted in practice, focusing on scalable, sustainable, and instructor-adoptable approaches that support meaningful learning in the era of AI.

Scaling across faculty and institutional constraints. Effective teaching solutions may be revised scale across courses, institutions, and class sizes. Specifically, solutions must be implementable without a complete curriculum redesign. They should be modular, adoptable by individual instructors, and feasible within resource, bureaucratic, and staffing constraints. A core barrier to pedagogical reform, however, is the overwhelming time constraint on faculty, with the current resource model unable to support labor-intensive, process-based assessments. To address this, institutions must make a strategic investment in human resources, beginning with a proactive stance to advocate for increased funding and dedicated personnel. This is essential to sustain diverse and personalized assessment methods. Crucially, administrations must ensure the maintenance of an appropriate

[9] https://ai.nus.edu.sg/research/ai-x/

instructor-to-student ratio, particularly in foundational courses, to guarantee quality instruction and support the increased grading load associated with new assessment types.

Supporting student motivation, engagement, and autonomy. Pedagogy depends on student presence and attention [24]. There is a need to provide clear value for in-person participation and communicate this to students. This can be done by including engaging or enjoyable elements and fostering social and community aspects of learning to mitigate isolation and disengagement. Teaching strategies could allow multiple pathways for students to engage and demonstrate learning, supporting autonomy while maintaining clear standards and assessable outcomes. Furthermore, persistent and exclusive exposure to AI-generated outputs can quietly erode students' intrinsic motivation, their willingness to experiment, and their confidence in their own intellectual agency. To reach autonomy, students should feel comfortable to take intellectual risks, and understand that while they can offload tasks to AI, learning needs individual's effort [7]. Pedagogical designs can therefore create structured spaces where students have the potential to engage independently before turning to AI

Aligning assessment with AI use. New pedagogies could encourage responsible and constructive use of AI rather than banning it, and target emerging skills (for example, by having students craft natural language prompts for solving computational tasks, and providing explicit feedback on their quality [29]). The Prompt Programming platform [29] demonstrates one concrete implementation of specification-driven assessment: students must craft natural language specifications to generate code solving specified problems, requiring iterative refinement and critical evaluation of AI outputs. This approach directly instantiates the specification-and-verification workflow advocated in this paper and demonstrates scalability through its freely available implementation. Assessment strategies can evaluate learning processes, critical reflection, and effective AI use, ensuring that AI supports rather than undermines educational goals. Experimental evidence further demonstrates that guardrailed AI deployments, where AI provides structured hints rather than direct solutions, can maintain learning outcomes while preserving the benefits of AI assistance [3], supporting the principle that AI integration should be carefully designed rather than simply permitted or banned. In STEM education, learning can be operationalized as a deep engagement with artifacts and a development of intentionality (i.e., paths to approach an idea) [26]. Monitoring learning processes would be an effective way to assess whether students are learning. Tool-assisted assessment, where students submit specifications, tests, or formal properties alongside their code, can provide scalable, objective evidence of understanding while naturally encouraging verification-oriented workflows.

6.5 Shifting Institutional Support

The successful adoption of AI tools and contemporary teaching methods requires significant institutional support and a strategic reallocation of resources. The faculty workshop identified three critical areas for administrative intervention, focusing on resourcing, incentives, and the creation of shared tools to facilitate scalable and sustained change.

Supporting faculty workload. Institutions can support the use of technology explicitly to reduce mechanical faculty workloads in grading and office-hour duties, redirecting faculty time toward higher-value activities such as curriculum development and personalized student interaction. Furthermore,

the faculty or school could establish specialized support services, such as ethics consulting services, to assist both students and faculty in navigating the complex ethical challenges inherent in AI system design.

Considerations for Academic Concessions Traditionally, in many universities, students receive concessions (e.g., extra time to complete an assessment) for compassionate or unforeseen circumstances. Concessions are given under the assumption that students work on their own. However, access to AI tools may violate this assumption. Staff who determine academic concession solutions now need to understand how AI affects learning and exam participation.

Incentivizing faculty participation. For innovative teaching to become widespread, institutions must move beyond passive encouragement to actively incentivize faculty participation. To foster the development and sharing of new materials, a system of financial and professional incentives is recommended, including small awards or grants for contributing high-quality, "AI-ready" assignments and reports to shared repositories. This should be coupled with initiatives to enhance the prestige and visibility of such contributions through internal recognition or publication pathways.

Maximizing efficiency and scalability. Institutions may invest in establishing a centralized, moderated, open resource repository to host actionable assignments, projects, and reports that accelerate AI integration. A dedicated, small moderation team is necessary to oversee submissions and ensure quality control. To complement this repository, institutions would need to support the organization of an annual assignment-sharing conference to facilitate the global discussion, submission, and adoption of tested, "AI-ready assignments," thus creating a scalable learning ecosystem. This ensures that faculty have immediate access to a "smorgasbord" of practical, embeddable materials, preventing the need for individual course redesigns and fostering a culture of rapid, informed pedagogical adaptation.

7 Conclusion and Next Steps

The rise of generative AI represents a significant and likely lasting disruption to CS education and the broader IT industry, though the ultimate scope and trajectory of this transformation remain unknown. As AI tools increasingly automate tasks traditionally performed by junior software engineers, the traditional assumptions in CS education – where coding, debugging, and documentation are scarce – no longer hold. This shift calls an urgent and comprehensive reform of CS curricula, moving away from a focus on manual code generation toward higher-order skills such as system architecting, verification, and ethical responsibility. Early empirical evidence supports this urgency [18], confirming that the quality assurance gap is not hypothetical but present and measurable. To remain relevant, CS education must adapt with expediency to produce graduates who can operate immediately and steward complex, AI-generated systems.

In light of recent significant changes, the NUS-Google Workshops brought together a diverse group of faculty and students to gather insights from key stakeholders in CS education. These two workshops—one focusing on faculty and institutional challenges, and the other on student experiences and concerns—offer several valuable contributions that enhance existing initiatives. Specifically, the dual-workshop approach captures both faculty and student perspectives, highlighting areas of agreement and disagreement that single-stakeholder methods overlook. In addition, rather than focusing solely on course-level responses, this paper proposes criteria for restructuring programs across all four years of undergraduate study. Both groups affirmed the enduring value of the foundation knowledge while stressing the paramount importance of metacognitive skills such as computational thinking, critical verification, accountability, and ethical reasoning. We note that “verification” as a graduate competency spans a wide range of rigour—from code review and testing to type-system-guided development and formal proof—and that curriculum designers can be more explicit about which levels of this spectrum they aim to teach and assess.

This white paper outlines a strategic response and documents several concrete solutions proposed during the workshops, such as "prompt-first" or "reverse-engineering" courses, new project-based learning models, and the “breadcrumb“ strategy to embed and reinforce AI-native skills throughout the curriculum. However, the efficacy of these proposed reforms remains an open question. Future work must be rigorously dedicated to validating these solutions against the key criteria and requirements voiced by the workshop attendees. Furthermore, to fully align the curriculum with the rapidly evolving job market, future studies will require the critical input of a broader range of industry practitioners and business stakeholders to ensure that academic pathways genuinely prepare the next generation of engineers for long-term success. In particular, the Practical Solutions envisioned by the proposed criteria (Section 6) remain to be implemented; we invite the community to contribute concrete, implemented examples—such as specification-driven assignments, verification-oriented project workflows, and AI-integrated capstone designs—against which the criteria in this paper can be tested. The evolving capabilities of AI and how they are integrated in the industry also mean this will require constant re-evaluation of the CS curricula over time.

8 Whitepaper Contributors

- Yi-Chieh (EJ) Lee, National University of Singapore, yclee@nus.edu.sg [10]
- Nattapat Boonprakong, National University of Singapore, nattapat.boonprakong@nus.edu.sg
- Yugin Tan, National University of Singapore, tanyugin@nus.edu.sg
- Harold Soh, National University of Singapore, dcshssh@nus.edu.sg
- Alex Potanin, Australian National University, alex.potanin@anu.edu.au
- Viraj Kumar, Indian Institute of Science, viraj@iisc.ac.in
- Anoop K. Sinha, Google, Paradigms of Intelligence, anoopsinha@google.com
- Chen Qian, Shanghai Jiao Tong University, qianc@sjtu.edu.cn
- Paul Denny, University of Auckland, paul@cs.auckland.ac.nz
- Mennatallah El-Assady, ETH Zurich, melassady@ai.ethz.ch
- Ian Oakley, KAIST, ian.r.oakley@gmail.com
- Jake Renzella, University of New South Wales, jake.renzella@unsw.edu.au
- Amy Zhang, University of Washington, axz@cs.uw.edu
- Jat Singh, RC-Trust, UA-Ruhr/UDE & University of Cambridge, jatinder.singh@cst.cam.ac.uk
- Wee Sun Lee, National University of Singapore, dcsleews@nus.edu.sg
- Hsuan-Tien Lin, National Taiwan University, htlin@csie.ntu.edu.tw
- Jane L. E, National University of Singapore, ejane@nus.edu.sg
- Anthony Tang, Singapore Management University, tonyt@smu.edu.sg
- Margaret M. Burnett, Oregon State University, burnett@eecs.oregonstate.edu
- Sowmya Somanath, University of Victoria, sowmyasomanath@uvic.ca
- Renwen Zhang, Nanyang Technological University, renwen.zhang@ntu.edu.sg
- Vicky Charisi, Singapore-MIT Alliance for Research and Technology, vasiliki.charisi@smart.mit.edu
- Alexandra I. Cristea, University of Durham, alexandra.i.cristea@durham.ac.uk

[10] The author ordering reflects the chronological sequence in which authors entered and edited the manuscript, rather than relative contribution.

9 Workshop Organizers

9.1 Faculty Workshop Organizers

- Yi-Chieh (EJ) Lee, National University of Singapore
- Zicheng Zhu, National University of Singapore
- Michael Terry, Google
- Shruti Sheth, Google
- Anoop K. Sinha, Google
- Harold Soh, National University of Singapore

9.2 Student Workshop Organizers

- Nattapat Boonprakong, National University of Singapore
- Jingshu Li, National University of Singapore
- Yugin Tan, National University of Singapore
- Vishruti Ranjan, National University of Singapore
- Emran Poh, Singapore Management University
- Tianqi Song, National University of Singapore
- Peinuan Qin, National University of Singapore
- Yitian Yang, National University of Singapore
- Zhengtao Xu, National University of Singapore
- Han Meng, National University of Singapore
- Junti Zhang, National University of Singapore
- Xuehui Yu, National University of Singapore
- Pranav Dangi, National University of Singapore
- Weiyan Shi, Singapore University of Technology and Design

Our workshops were made possible through the generous support of Google.

10 Faculty Workshop Attendees

- Alexandra I. Cristea, Durham University
- Alex Potanin, Australian National University
- Amy X. Zhang, University of Washington
- Anoop K. Sinha, Google
- April Yi Wang, ETH Zürich
- Chen Qian, Shanghai Jiao Tong University
- Chris Piech, Stanford University
- Jane L. E, National University of Singapore
- Hsuan-Tien Lin, National Taiwan University
- Ian Oakley, KAIST
- Jake Renzella, University of New South Wales
- Li-Shiuan Peh, National University of Singapore
- Jat Singh, RC-Trust, UA-Ruhr/UDE & University of Cambridge
- Johannes Schöning, University of St. Gallen
- Khai N. Truong, University of Toronto
- Paul Denny, The University of Auckland
- Margaret M. Burnett, Oregon State University
- Mennatallah El-Assady, ETH Zürich
- Wei Tsang Ooi, National University of Singapore
- Sowmya Somanath, University of Victoria
- Xiaojuan Ma, Hong Kong University of Science and Technology
- Robert Xiao, University of British Columbia
- Roman Cunci, Avature
- Weinan E, Peking University
- Ye Wang, National University of Singapore

- Tony Tang, Singapore Management University
- Viraj Kumar, IISc Bangalore
- Wee Sun Lee, National University of Singapore
- Yi-Chieh Lee, National University of Singapore

11 Student Workshop Attendees

- Alex Chien, National University of Singapore
- Arpita Dutta, National University of Singapore
- Aryan Jain, National University of Singapore
- Botao 'Amber' Hu, University of Oxford
- Chan Lok Tung, National University of Singapore
- Chayapol Chaoveeraprasit, National University of Singapore
- Emily Aurelia, Singapore Management University
- G Dhanesh Sabaratnam, Singapore Institute of Management-London School of Economics
- Gangyi Zhang, The University of Melbourne
- Grant Lee, ST Engineering
- Gu Zhen Feng, National University of Singapore
- Gullapalli Chetan, National University of Singapore
- Hans, Singapore Polytechnic
- Ho Yu Hua, Singapore University of Technology and Design
- Irene Ho, University of Wisconsin-Madison
- Jason Pang, University of Pennsylvania
- Jessica Chen, National University of Singapore
- Jingzhu Chen, Tongji University
- King Jean Lynn, Nanyang Technological University
- Lim Jie Han, Nanyang Technological University
- Liu Yan, National University of Singapore

- Meng Yuxuan, National University of Singapore
- Michael Cenreng, National University of Singapore
- Ong Jia Cheng, National University of Singapore
- Pang Zi Haur, Kyoto University
- Pannapat Chanpaisaeng, Singapore University of Technology and Design
- Phua Yu Cheng, Universiti Teknologi Malaysia
- Rachel Yeo Hui Min, National University of Singapore
- Rimmon Saloman Bhosale, Singapore Management University
- Sarthak Kumar, National University of Singapore
- Shizun Wang, National University of Singapore
- Sparsh Rastogi, Thapar Institute of Engineering and Technology
- Tan Jin Hong, Universiti Tunku Abdul Rahman
- Tee Qian Hui, Universiti Tunku Abdul Rahman
- Ting Zhang, National University of Singapore
- Vignesh Sundararaj, Singapore Management University
- Wang Chuansheng, National University of Singapore
- Wang Qilin, University of Wollongong
- Weiyan Shi, Singapore University of Technology and Design
- Yang Chen Lin, National Tsing Hua University
- Yew Foong Yik, Asia Pacific University of Technology & Innovation
- Yifei Luo, RWTH Aachen
- Yiran Jiang, Dartmouth College
- Zhang Dingpu, Nanyang Technological University
- Zhang Jingze, National University of Singapore
- Zhicong Chen, National University of Singapore
- Zhizhou Cao, National University of Singapore

References

[1] P. AI. Introducing perplexity computer. URL https://www.perplexity.ai/hub/blog/introducing-perplexity-computer.

[2] G. Amasanti and J. Jahic. The impact of ai-generated solutions on software architecture and productivity: Results from a survey study, 2025. URL https://arxiv.org/abs/2506.17833.

[3] H. Bastani, O. Bastani, A. Sungu, H. Ge, Ö. Kabakcı, and R. Mariman. Generative ai without guardrails can harm learning: Evidence from high school mathematics. *Proceedings of the National Academy of Sciences*, 122(26):e2422633122, 2025.

[4] BBC. Openai claims gpt-5 model boosts chatgpt to 'phd level'. URL https://www.bbc.com/news/articles/cy5prvgw0r1o.

[5] J. Becker, N. Rush, E. Barnes, and D. Rein. Measuring the impact of early-2025 ai on experienced open-source developer productivity. *arXiv preprint arXiv:2507.09089*, 2025.

[6] E. Brynjolfsson, B. Chandar, and R. Chen. Canaries in the coal mine?: Six facts about the recent employment effects of artificial intelligence. 2025.

[7] V. Charisi. Artificial intelligence in classrooms: Pedagogical dimensions. Briefing PE 784.574, European Parliament, Policy Department for Citizens, Equality and Culture, March 2026. URL https://www.europarl.europa.eu/RegData/etudes/BRIE/2026/784574/IUST_BRI(2026)784574_EN.pdf.

[8] R. Choudhuri, A. Ramakrishnan, A. Chatterjee, B. Trinkenreich, I. Steinmacher, M. Gerosa, and A. Sarma. Insights from the Frontline: GenAI Utilization Among Software Engineering Students. In *2025 IEEE/ACM 37th International Conference on Software Engineering Education and Training (CSEE&T)*, pages 1–12. IEEE, 2025.

[9] R. Choudhuri, C. Sanchez, M. Burnett, and A. Sarma. Why johnny can't think: Genai's impacts on cognitive engagement. *arXiv preprint arXiv:2601.22430*, 2026.

[10] T. Clear, A. Cajander, A. Clear, R. McDermott, M. Daniels, M. Divitini, M. Forshaw, N. Humble, M. Kasinidou, S. Kleanthous, C. Kultur, G. Parvini, M. Polash, and T. Zhu. Ai integration in the it professional workplace: A scoping review and interview study with implications for education and professional competencies. In *2024 Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE 2024, page 34–67, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712081. doi: 10.1145/3689187.3709607. URL https://doi.org/10.1145/3689187.3709607.

[11] P. Denny, J. Prather, B. A. Becker, J. Finnie-Ansley, A. Hellas, J. Leinonen, A. Luxton-Reilly, B. N. Reeves, E. A. Santos, and S. Sarsa. Computing education in the era of generative ai. *Commun. ACM*, 67(2):56–67, Jan. 2024. ISSN 0001-0782. doi: 10.1145/3624720. URL https://doi.org/10.1145/3624720.

[12] P. Denny, V. Kumar, S. MacNeil, J. Prather, and J. Leinonen. Probing the unknown: Exploring student interactions with probeable problems at scale in introductory programming. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2025, page 618–624, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715679. doi: 10.1145/3724363.3729093. URL https://doi.org/10.1145/3724363.3729093.

[13] E. W. Dijkstra. On the foolishness of "natural language programming". In *Program Construction, International Summer Schoo*, page 51–53, Berlin, Heidelberg, 1978. Springer-Verlag. ISBN 354009251X.

[14] B. Dongol, C. Dubois, S. Hallerstede, E. Hehner, C. Morgan, P. Müller, L. Ribeiro, A. Silva, G. Smith, and E. de Vink. On formal methods thinking in computer science education. *Formal Aspects of Computing*, 37(1):1–23, 2024.

[15] E. Eaton and S. L. Epstein. Artificial intelligence in the cs2023 undergraduate computer science curriculum: Rationale and challenges. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 23078–23083, 2024.

[16] Y. Fan, L. Tang, H. Le, K. Shen, S. Tan, Y. Zhao, Y. Shen, X. Li, and D. Gašević. Beware of metacognitive laziness: Effects of generative artificial intelligence on learning motivation, processes, and performance. *British Journal of Educational Technology*, 56(2):489–530, 2025.

[17] D. Gries. What should we teach in an introductory programming course? In *Proceedings of the Fourth SIGCSE Technical Symposium on Computer Science Education*, SIGCSE '74, page 81–89, New York, NY, USA, 1974. Association for Computing Machinery. ISBN 9781450374835. doi: 10.1145/800183.810447. URL https://doi.org/10.1145/800183.810447.

[18] H. He, C. Miller, S. Agarwal, C. Kästner, and B. Vasilescu. Speed at the cost of quality: How Cursor AI increases short-term velocity and long-term complexity in open-source projects. In *Proceedings of the 23rd International Conference on Mining Software Repositories*, MSR '26, New York, NY, USA, 2026. ACM. doi: 10.1145/3793302.3793349. arXiv:2511.04427.

[19] P. Ipeirotis and K. Rizakos. Scalable and personalized oral assessments using voice ai, 2026. URL https://arxiv.org/abs/2603.18221.

[20] M. Kleppmann. Prediction: Ai will make formal verification go mainstream, 2025. URL https://martin.kleppmann.com/2025/12/08/ai-formal-verification.html.

[21] A. N. Kumar, R. K. Raj, S. G. Aly, M. D. Anderson, B. A. Becker, R. L. Blumenthal, E. Eaton, S. L. Epstein, M. Goldweber, P. Jalote, et al. Computer science curricula 2023, 2024.

[22] H.-P. Lee, A. Sarkar, L. Tankelevitch, I. Drosos, S. Rintel, R. Banks, and N. Wilson. The impact of generative ai on critical thinking: Self-reported reductions in cognitive effort and confidence effects from a survey of knowledge workers. In *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, pages 1–22, 2025.

[23] E. Mohammadkarimi. Teachers' reflections on academic dishonesty in efl students' writings in the era of artificial intelligence. *Journal of Applied Learning & Teaching*, 6(2):105–113, 2023.

[24] E. Moores, G. K. Birdi, and H. E. Higson. Determinants of university students' attendance. *Educational Research*, 61(4):371–387, 2019. doi: 10.1080/00131881.2019.1660587. URL https://doi.org/10.1080/00131881.2019.1660587.

[25] S. Peng, E. Kalliamvakou, P. Cihon, and M. Demirer. The impact of ai on developer productivity: Evidence from github copilot, 2023. URL https://arxiv.org/abs/2302.06590.

[26] M. Petrich, K. Wilkinson, and B. Bevan. It looks like fun, but are they learning? 2013. URL https://api.semanticscholar.org/CorpusID:111722217.

[27] J. Prather, P. Denny, J. Leinonen, B. A. Becker, I. Albluwi, M. Craig, H. Keuning, N. Kiesler, T. Kohn, A. Luxton-Reilly, S. MacNeil, A. Petersen, R. Pettit, B. N. Reeves, and J. Savelka. The robots are here: Navigating the generative ai revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE-WGR '23, page 108–159, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400704055. doi: 10.1145/3623762.3633499. URL https://doi.org/10.1145/3623762.3633499.

[28] J. Prather, J. Leinonen, N. Kiesler, J. Gorson Benario, S. Lau, S. MacNeil, N. Norouzi, S. Opel, V. Pettit, L. Porter, B. N. Reeves, J. Savelka, I. Smith, David H., S. Strickroth, and D. Zingaro. Beyond the hype: A comprehensive review of current trends in generative ai research, teaching practices, and tools. In *2024 Working Group Reports on Innovation and Technology in Computer Science Education*, ITiCSE 2024, page 300–338, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400712081. doi: 10.1145/3689187.3709614. URL https://doi.org/10.1145/3689187.3709614.

[29] V.-A. Pădurean, P. Denny, A. Gotovos, and A. Singla. Prompt programming: A platform for dialogue-based computational problem solving with generative ai models. In *Proceedings of the 30th ACM Conference on Innovation and Technology in Computer Science Education V. 1*, ITiCSE 2025, page 458–464, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400715679. doi: 10.1145/3724363.3729094. URL https://doi.org/10.1145/3724363.3729094.

[30] V. J. Reddi. *Introduction to Machine Learning Systems*. MIT Press, 2026. Available online at https://mlsysbook.ai/book/.

[31] M. I. H. Shihab, C. Hundhausen, A. Tariq, S. Haque, Y. Qiao, and B. W. Mulanda. The effects of github copilot on computing students' programming effectiveness, efficiency, and processes in brownfield coding tasks. In *Proceedings of the 2025 ACM Conference on International Computing Education Research V.1*, ICER '25, page 407–420, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713408. doi: 10.1145/3702652.3744219. URL https://doi.org/10.1145/3702652.3744219.

[32] S. Somanath, R. Wakkary, O. Ettehadi, H. W. J. Lin, A. Behzad, J. Eshpeter, and D. Oogjes. Exploring the composite intentionality of 3d printers and makers in digital fabrication. 2023. URL https://api.semanticscholar.org/CorpusID:261364233.

[33] L. Tankelevitch, V. Kewenig, A. Simkute, A. E. Scott, A. Sarkar, A. Sellen, and S. Rintel. The metacognitive demands and opportunities of generative AI. In *CHI Conference on Human Factors in Computing Systems*, pages 1–24, 2024.

[34] A. S. Theobold. Oral exams: A more meaningful assessment of students' understanding. *Journal of Statistics and Data Science Education*, 29(2):156–159, 2021. doi: 10.1080/26939169.2021.1914527. URL https://doi.org/10.1080/26939169.2021.1914527.

[35] H. Washizaki, editor. *Guide to the Software Engineering Body of Knowledge (SWEBOK Guide)*. IEEE Computer Society, 4.0 edition, 2024. URL https://www.swebok.org.

[36] L. Yu. Paradigm shift on coding productivity using genai. In *Proceedings of the 29th International Conference on Evaluation and Assessment in Software Engineering*, EASE '25, page 708–713, New York, NY, USA, 2025. Association for Computing Machinery. ISBN 9798400713859. doi: 10.1145/3756681.3757081. URL https://doi.org/10.1145/3756681.3757081.