Crawl, Walk, Run: A Scaffolded Framework for AI Integration in Computing Education

Ramakrishnan Durairajan
University of Oregon and Link Oregon

Abstract

AI is rapidly transforming computing practice, creating new challenges for educators across computing, systems, and networking disciplines. While unrestricted AI use can undermine conceptual learning, prohibiting AI risks disconnecting coursework from modern professional practice.

This experience report presents a scaffolded framework for generative AI integration in computing education that balances foundational skill development with AI-assisted learning. With this framework, students progress through three stages: foundational skill development without AI assistance, structured AI-assisted pair programming, and AI-enabled project development where AI serves as a productivity multiplier. We describe the design rationale, implementation, and lessons learned from a pilot deployment of this framework. Our experience suggests that integrating AI into project-based coursework can help students develop critical awareness of AI's limitations and the professional habits needed to evaluate AI-generated code. However, students frequently reported tension between the speed AI affords and the depth of understanding that coursework is designed to build.

ACM Reference Format:
Ramakrishnan Durairajan. 2026. Crawl, Walk, Run: A Scaffolded Framework for AI Integration in Computing Education. In *SIGCOMM Education Workshop 2026: Networking Education for the AI Generation, August 17, 2026, Denver, CO, USA*. ACM, New York, NY, USA, 7 pages.

1 Introduction

AI has rapidly become part of everyday computing practice. In particular, generative AI tools (e.g., OpenAI Codex, Claude Code, Microsoft Copilot, etc. [5]), can generate code, explain APIs, assist with debugging, produce infrastructure configurations, and support development workflows at a level that was difficult to imagine only a couple of years ago. As these tools become increasingly common in industry, computing educators face a key question: how should students learn computing in an era when AI can produce substantial portions of software systems and infrastructure configurations?

The answer is not obvious. On the one hand, prohibiting AI use entirely risks creating a disconnect between classroom practice and the realities of contemporary computing. Students graduating into industry will almost certainly encounter AI-assisted workflows and will be expected to use them effectively [7, 9]. On the other hand, unrestricted AI use may allow students to bypass critical learning experiences, producing functioning artifacts without developing the conceptual understanding necessary to design, evaluate, debug, deploy, and maintain complex systems. Across computing disciplines (esp. in software engineering, systems, networking, etc.) learning the fundamentals and preserving deep understanding of core concepts remain key even as development tools evolve.

We believe this tension is particularly visible (and will aggravate more) in foundational computing courses. Students are already experimenting with AI tools in diverse and often unsupervised ways (e.g., Shadow AI [2, 4]). Some use AI as a productive collaborator, while others rely on it as a replacement for problem solving (e.g., leading to cognitive atrophy [3], cognitive debt [1]). Educators therefore face a pedagogical challenge that extends *beyond* questions of academic integrity: how can AI be integrated into coursework in ways that enhance learning rather than shortcut it? More broadly, how can educators prepare students to work effectively in AI-augmented environments (e.g., network management, AIOps settings, etc.) while ensuring that they still develop the foundational knowledge required to reason about systems, diagnose failures, and adapt to new technologies?

Through our experience redesigning and teaching undergraduate computing courses, we arrived at a simple observation: effective AI use should be learned, not assumed. The challenge is not whether students should use AI, but when and how should they use it? Said differently, students need opportunities to develop foundational competencies independently before learning to collaborate with AI and, ultimately, leverage it as a productivity multiplier. We therefore view AI integration as a pedagogical design challenge: determining when students should struggle independently, when AI can

This work is licensed under a Creative Commons Attribution-ShareAlike 4.0 International License.
ACM SIGCOMM A4NE '26, Denver, CO, USA
© 2026 Copyright held by the owner/author(s).

serve as a learning partner, and when it can be used to accelerate development without compromising understanding.

Motivated by this view, we propose a scaffolded framework for generative AI integration into computing education. At the core of the framework is a phased approach in which students: (1) complete course activities *without* AI assistance to establish foundational competencies in phase 1; (2) engage in *structured* AI-assisted activities that emphasized critical evaluation, reflection, and collaboration in phase 2; and (3) leverage AI as a *productivity multiplier* during a substantial systems-building project in phase 3.

This experience report presents the design rationale behind the framework, describes its implementation, and reflects on lessons learned from its initial deployment in *CS 322* (Introduction to Software Engineering) [8] at *University of Oregon*. While the framework was evaluated in a single course, we believe its underlying principles provide a starting point for AI integration across computing education. Furthermore, the framework is particularly relevant to systems and networking courses where deep conceptual understanding remains essential despite increasingly capable AI tools (e.g., AIOps for multi-cloud [10] and enterprise [6] network management) in the domain.

2 Generative AI in Computing Education

The rapid adoption of generative AI has created a fundamental tension for computing educators. On one hand, generative AI development tools (e.g., [5]) have become increasingly common in professional practice. Software engineers, systems engineers, site reliability engineers, and platform teams routinely use AI tools to generate code, explain unfamiliar technologies, troubleshoot errors, and accelerate development workflows. Consequently, it is neither practical nor desirable for educators to ignore them. Students entering the workforce *will* be expected not only to understand computing concepts but also to use AI tools effectively.

On the other hand, the educational goals of computing courses extend beyond producing working artifacts. Students must develop the conceptual understanding needed to reason about systems, diagnose failures, evaluate design decisions, and adapt to new technologies throughout their careers. Generative AI complicates this process because it can often produce plausible solutions with minimal effort from the learner, leading to cognitive atrophy [3]. A student may successfully complete an assignment using AI-generated code, configurations, or explanations without fully understanding the underlying concepts. While the resulting artifact may function correctly, the learning objectives will not be met.

These concerns have led educators to question how AI should be incorporated into computing curricula. Early classroom experiences suggest that students can become overly reliant on AI tools, treating them as authoritative sources rather than critically evaluating their outputs. This reliance may reduce opportunities for productive struggle, experimentation, and debugging; these are the kind of activities that have traditionally played an important role in developing problem-solving skills in computing. At the same time, instructors face increasing challenges in assessing student understanding when AI systems can contribute substantially to assignment solutions.

Yet the appropriate response is not simply to prohibit AI use. Such restrictions may be difficult to enforce and risk leaving students unprepared for the realities of industry. Instead, educators must help students develop AI literacy: the ability to understand the capabilities and limitations of AI systems, critically evaluate generated outputs, recognize when AI assistance is appropriate, and effectively integrate AI into their workflows. The central challenge, therefore, is not whether AI belongs in computing education, but how it can be incorporated in ways that support learning objectives.

3 A Scaffolded Framework for AI Integration

The challenges described above suggest that the question facing computing educators is not whether AI should be used in the classroom, but how its use should be structured. Completely prohibiting AI may leave students unprepared for modern computing practice, while unrestricted use risks undermining the very learning outcomes that computing courses are designed to achieve. We therefore propose a scaffolded framework for AI integration that treats AI not as a binary choice but as a capability that students learn to use progressively over time.

The framework is motivated by a simple observation: effective AI use should be learned, not assumed. Just as students are taught how to debug software, evaluate system designs, or reason about performance trade-offs, they must also learn how to interact with AI systems productively and critically. Consistent with the scaffolding framework of Wood, Bruner, and Ross [11], the framework therefore introduces AI in stages, beginning with foundational skill development, followed by structured AI-assisted learning, and culminating in more autonomous AI-supported work. The goal is to preserve conceptual understanding while helping students develop the skills necessary to operate effectively in AI-augmented environments.

3.1 Design Principles

The framework is guided by four principles that emerged from our experience teaching computing courses and observing how students interact with generative AI tools.

Foundations before automation. Students should first develop a conceptual understanding of core computing principles before relying on AI assistance. Whether the topic is programming, debugging, distributed systems, networking, or platform engineering, students benefit from building mental models that allow them to reason about behavior, identify failures, and evaluate solutions independently. AI can accelerate implementation, but it cannot – and should not – replace first-principles thinking.

AI as a collaborator, not a replacement. The purpose of AI is to augment human learning rather than substitute for it. Students should engage with AI as a collaborator that provides suggestions, explanations, and alternative perspectives while retaining responsibility for understanding and evaluating the resulting work. This shifts AI from an answer-generating tool to a learning partner.

Progressive autonomy. Students should gain increasing freedom to use AI as their competence develops. Early learning activities emphasize independent problem solving, while later activities allow students to incorporate AI into increasingly complex tasks. This progression mirrors how expertise develops in *any* professional practice: autonomy is earned through demonstrated competence over a period of time.

Reflection and verification. Students should be encouraged to critically evaluate AI-generated outputs rather than accepting them at face value. Verification, comparison, testing, and reflection are essential skills for effective AI use. Learning occurs not only through generating solutions but also through understanding their correctness, limitations, and trade-offs.

Building on these principles, the scaffolded framework entails the following three phases.

3.2 Phase 1: Crawl – Learning Without AI

The first phase focuses on foundational learning without AI assistance. The objective is to help students develop the mental models and problem-solving skills necessary to reason about computing systems independently.

During this phase, students engage directly with core concepts and experience the *productive struggle* that often accompanies learning. They write code, debug errors, design software components, and analyze system behavior using their own reasoning and the instructional resources provided by the course. Although this process may be slower than AI-assisted development, it creates opportunities for students to develop intuition about how systems work and why particular solutions succeed or fail.

3.3 Phase 2: Walk – Structured AI-Assisted Learning

Once students have demonstrated foundational competence, AI is introduced in a structured manner. The goal of this phase is not simply to increase productivity, but to help students learn how to work effectively with AI systems.

In this stage, students treat *AI as a collaborator* rather than an automated solution generator. Students may engage in AI-assisted pair programming, use AI tools to review code, request explanations of unfamiliar concepts, or critique alternative implementations. These activities expose students to the strengths and limitations of AI while preserving their role as active participants in the learning process.

A critical component of this phase is accountability. Students remain responsible for validating AI-generated outputs, explaining design decisions, and comparing AI-generated solutions with their own approaches. For example, students may be asked to justify why a particular AI-generated implementation is correct, identify weaknesses in a generated solution, or discuss trade-offs between human-produced and AI-produced designs. These activities encourage critical thinking and reinforce the idea that AI outputs should be evaluated rather than accepted uncritically.

Another critical component of this phase is rethinking assessment. Traditional assessments often focus on the correctness of a final artifact, making it difficult to distinguish between student understanding and AI-generated work. As AI becomes a routine part of the learning process, instructors must increasingly evaluate how students reason about problems rather than simply what they produce. Structured reflection and metacognitive prompts can provide insight into students' decision-making, use of prior knowledge, evaluation of AI-generated suggestions, and overall learning process. For example, students may be asked to explain why they turned to AI, describe instances where they modified or rejected AI-generated outputs, or reflect on how AI influenced their understanding of a particular concept. Such activities help shift assessment from artifact-focused evaluation toward evidence of reasoning, judgment, and conceptual understanding.

3.4 Phase 3: Run – AI as a Productivity Multiplier

The final phase allows students to use AI more freely as they work on larger and more complex projects. By this point, students have already demonstrated foundational understanding and gained experience critically evaluating AI-generated outputs. The role of AI therefore shifts from a learning scaffold to a productivity multiplier.

Students can leverage AI to accelerate tasks such as implementation, testing, documentation, integration, and infrastructure generation. For example, AI may assist in generating boilerplate code, producing test cases, creating deployment configurations, or drafting project documentation. Students remain responsible for the correctness and quality of the resulting artifacts, but AI can substantially reduce the time required to complete routine tasks.

Importantly, this phase does not represent a departure from the educational goals of the earlier stages. Instead, it reflects a transition toward authentic computing practice. Professional engineers routinely use tools that automate portions of their workflow while relying on their expertise to guide decisions, validate outputs, and resolve unexpected failures. By introducing AI only after students have established foundational competence and developed habits of critical evaluation, the framework enables students to experience AI as a multiplier rather than a substitute for understanding.

4 Pilot Deployment

We piloted the scaffolded framework in *CS 322*, a sophomore- and junior-level undergraduate course at the *University of Oregon*. Although formally designated as a software engineering course, *CS 322* spans a broad range of computing topics, including web architectures, containerization, testing, databases, APIs, security, and software integration. Through a sequence of hands-on projects, the 54 students (enrolled) in the course designed, implemented, tested, and deployed complete applications while gaining exposure to concepts commonly associated with systems, networking, and platform engineering.

CS 322 provides a useful setting for evaluating AI integration for three reasons. First, the course sits at the intersection of software, systems, and platform-engineering education, making it representative of the interdisciplinary skills increasingly required in modern computing practice. Second, it serves sophomore- and junior-level students who have completed introductory programming coursework but are still developing the conceptual foundations needed for more advanced computing topics. Third, the course is organized around substantial project-based work that requires students to integrate multiple technologies and reason about complex systems over an extended period of time. These characteristics create opportunities for both productive AI use and potential overreliance on AI-generated solutions.

To align with the scaffolded framework, AI access was introduced progressively throughout the term (as shown in Table 1). Students first completed a series of mini-projects without AI assistance, followed by structured AI-assisted activities, before ultimately applying AI more freely during a larger team-based project.

Course Activity	AI Integration Phase
Projects 1–4	P1: Learning Without AI
Projects 5–6	P2: Structured AI-Assisted Learning
Final Project	P3: AI as a Productivity Multiplier

Table 1: Mapping course activities to the scaffolded AI integration framework.

The transition points between phases were chosen to align AI access with students' developing competence. Projects 1–4 focused on foundational concepts and core development skills, providing students with opportunities to build mental models and problem-solving strategies without AI assistance. After students demonstrated baseline proficiency, Projects 5–6 introduced structured AI-supported activities (e.g., via AI as a pair programmer) designed to help students learn how to critically evaluate, refine, and verify AI-generated outputs. Finally, the larger team-based project allowed students to use AI more freely as a productivity multiplier while remaining responsible for system design, implementation, testing, and deployment decisions. This progression was intended to balance foundational learning with authentic exposure to AI-assisted development practices.

Finally, while the specific content of *CS 322* differs from dedicated systems or networking courses, the underlying instructional challenges are similar. Students must develop strong conceptual foundations while learning to work effectively with increasingly capable AI tools. As a result, we view *CS 322* not merely as a software engineering course, but as a representative pilot environment for exploring AI integration across computing education more broadly.

5 Evaluation

5.1 Course Survey

The survey is composed of four questions:

(1) **Structured Reflection.** (i) What specific problem were you trying to solve when you turned to AI? (ii) What prior knowledge, course concepts, documentation, or other resources did you rely on before prompting the AI? (iii) Describe at least one instance where you modified, refined, or rejected an AI-generated suggestion. Why did you make that decision?
(2) **Metacognitive Analysis.** (i) Did using AI change how you thought about the problem? If so, how? (ii) Did AI accelerate your understanding, bypass part of your learning process, or both? Explain. (iii) At what point did you feel most cognitively engaged with the task—before, during, or after interacting with AI? Why?

(3) **Critical Evaluation.** (i) Did the AI-generated output raise any concerns related to security, privacy, correctness, maintainability, or software quality? If so, describe them. (ii) How did you verify the accuracy and reliability of the AI's suggestions or code?
(4) **Professional Practice.** Imagine that the AI-generated code or solution would be used in a production environment. (i) What additional testing, review, validation, or documentation would you require before trusting it? (ii) What risks would need to be addressed before deployment? (iii) Who else (e.g., teammates, reviewers, security engineers) should be involved in evaluating the solution?

Of the 54 students, 33 students responded to our reflection survey. We analyzed a corpus of 33 student reflection responses to these questions. We performed exploratory qualitative analysis consisting of thematic coding, keyword frequency analysis, and LLM-assisted sentiment categorization on this corpus. Each response was tagged against nine predefined themes (such as AI code generation, security, debugging, etc.) based on topics covered in the course, with both binary presence and raw mention counts recorded. Sentiment was scored per response and then bucketed into three categories: positive/learning-forward, mixed/neutral, and concern/friction-forward. Keyword and bigram frequencies were aggregated across the full corpus to surface the language students actually reached for when describing their experience. The per-response summaries were then read alongside the quantitative results to ground the numbers in specific student voices, which is how individual names and quotes were tied back to aggregate patterns. Note that because our evaluation only analyzes student reflections to these questions, we present perceptions and attitudes, not learning outcomes. Measuring the learning outcomes is part of our ongoing work.

5.2 Preliminary Results

Figure 1 breaks down sentiment two ways: the donut (left) shows the three-category split (nearly two-thirds of files lean toward concern or friction), while the scatter plot (right) shows each response's individual sentiment score. The distribution skews clearly negative, with most scores between -0.3 and -0.85 and only a handful of genuinely positive outliers above +0.3.

Students used AI extensively, but rarely trusted it blindly. All 33 students described using AI for code generation (428 mentions total), and all 33 also discussed security, testing, and the need for human review. Security alone accounted for 174 mentions, while 27 students (82%) explicitly wrote about prompting, iterating on, rejecting, and verifying AI-generated suggestions. The bigram "make sure" appeared 38 times and "verify accuracy" appeared 10 times across the corpus. These suggest that students approached AI with a posture of cautious verification rather than confident delegation.

Student experience was more stressful than exciting. 21 students (64%) wrote reflections that leaned toward concern, friction, or uncertainty, while only four (12%) expressed a clearly positive view. This does not necessarily indicate failure of the pilot study; meaningful learning is often accompanied by difficulty. However, it does suggest that working with AI introduced genuine cognitive load for most students rather than functioning as a simple shortcut.

Students also recognized a tension between efficiency and understanding. Learning and metacognition-related issues appeared in 32 of 33 reflections (97%, 175 mentions). In fact, several students acknowledged that AI's speed sometimes came at the expense of deeper comprehension. 5 students explicitly identified concepts or tasks they skipped, delegated, or failed to retain because AI handled them.

Students who engaged most deeply and critically came out most positive. The students who engaged and pushed back on AI reported the most positive experiences. The 4 positively scored reflections showed evidence of questioning AI-generated outputs, and attempting to understand AI-generated code rather than simply using it with multiple follow-up questions.

Overall, students demonstrated awareness of responsible AI use. 32 of 33 could clearly articulate why AI-generated code requires scrutiny, what risks it introduces, and the importance of human oversight. Yet many of those same students described the experience as difficult, uncertain, or friction-filled, and some identified gaps in their own understanding that resulted from relying on AI. Our results therefore point to a distinction between knowing the principles of responsible AI use (which was nearly universal) and developing the confidence and depth of understanding required to apply those principles effectively in practice.

5.3 Limitations

This work has several limitations. First, the pilot deployment was conducted in a single undergraduate computing course at one institution. Because of this, the generalizability of the findings to other courses, disciplines, and student populations are limited and more longitudinal work is needed on this front. Second, our evaluation relied primarily on student reflection essays and self-reported experiences, which provide insight into student perceptions and reasoning but do not directly measure learning outcomes, conceptual understanding, skill acquisition, or long-term retention. Third, no control or comparison group was used, making it difficult to

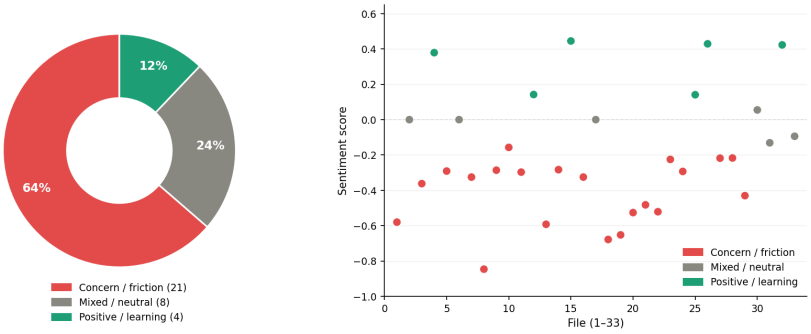

Figure 1: Student sentiment toward AI-assisted coding: distribution (left) and per-response scores (n=33) (right).

attribute observed outcomes solely to the scaffolded AI integration framework. Finally, while thematic coding, keyword analysis, and sentiment scoring provided useful descriptive insights, these methods cannot fully capture the complexity of student learning processes or establish causal relationships between AI use and educational outcomes. As a result, the preliminary results should be interpreted as exploratory evidence intended to inform future research rather than as definitive validation of the framework's effectiveness.

5.4 Discussion

CS area-specific considerations. While the proposed framework is broadly applicable across computing education, we note that the boundaries between the phases will likely differ across areas of computer science (e.g., networking, programming languages, and databases). This means that the foundational competencies students should develop without AI are discipline-specific. For example, programming languages emphasize algorithmic thinking and debugging; systems focus on processes, memory, and concurrency; networking requires reasoning about protocols, routing, and packet behavior; databases depend on relational modeling and query optimization; graphics builds on linear algebra and the rendering pipeline; and security demands an understanding of threat models and cryptographic principles. Once these conceptual foundations are established, AI can serve as a productivity multiplier by assisting with implementation-specific tasks such as boilerplate code, configuration generation, documentation, testing, and refactoring without replacing the underlying reasoning that computing education seeks to develop.

Rethinking learning outcomes in the age of AI. This work also raises several questions for future research. As AI becomes increasingly integrated into computing workflows, how should learning outcomes evolve? Traditional artifact-based assessments become less informative when AI can generate correct implementations, motivating assessment approaches that emphasize conceptual understanding, debugging, transfer of knowledge, oral explanation, and critical evaluation of AI-generated outputs. How can scaffolded AI integration be adapted to systems, networking, and other computing curricula with different learning objectives and technical constraints? Finally, what metrics should educators use to evaluate the success of AI integration beyond traditional measures of assignment completion and course performance? Addressing these questions will be essential as computing educators continue to redefine what it means to prepare students for an AI-enabled future.

6 Summary

Generative AI is reshaping computing practice, but its integration into computing education cannot be reduced to a simple choice between prohibition and unrestricted use. As AI tools become commonplace in professional environments, educators must prepare students to use them effectively while preserving the foundational knowledge and problem-solving skills that remain central to computing disciplines. We argue that AI integration is fundamentally a pedagogical design problem: the key question is not whether students should use AI, but when and how they should use it throughout the learning process.

To address this challenge, we presented a scaffolded framework that progressively introduces AI through three phases: foundational learning without AI, structured AI-assisted learning, and AI-enabled project development. Piloted in an undergraduate course spanning software engineering, systems, and platform-engineering concepts, the framework seeks to balance conceptual learning with authentic exposure to modern AI-assisted workflows.

Our initial experience suggests that this approach provides a practical middle ground between the extremes of unrestricted adoption and outright prohibition. Students demonstrated awareness of responsible AI use. In particular, 32 of 33 students were able to articulate why AI-generated outputs require scrutiny, what risks AI introduces, and why human

oversight remains essential. At the same time, many students described AI-assisted development as difficult, uncertain, or cognitively demanding, and several identified gaps in their own understanding that emerged when relying too heavily on AI-generated solutions. These findings suggest an important distinction between understanding the principles of responsible AI use and developing the confidence, judgment, and technical depth required to apply those principles effectively in practice.

Acknowledgements

We thank the anonymous reviewers for their constructive feedback. This work is supported by University of Oregon's Williams Instructional Grant. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of UO or Williams Instructional Grant.

Private generative AI tools were used for enhancing the readability of paper as well as for processing the course survey data.

References

[1] Nataliya Kosmyna, Eugene Hauptmann, Ye Tong Yuan, Jessica Situ, Xian-Hao Liao, Ashly Vivian Beresnitzky, Iris Braunstein, and Pattie Maes. 2025. Your brain on ChatGPT: Accumulation of cognitive debt when using an AI assistant for essay writing task. *arXiv preprint arXiv:2506.08872* 4 (2025).

[2] Tom Krantz, Alexandra Jonker, and Amanda McGrath. 2026. *What is Shadow AI?* IBM. https://www.ibm.com/think/topics/shadow-ai

[3] Hao-Ping Lee, Advait Sarkar, Lev Tankelevitch, Ian Drosos, Sean Rintel, Richard Banks, and Nicholas Wilson. 2025. 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*. Association for Computing Machinery. doi:10.1145/3706598.3713778

[4] Adeleh Mazaheriyan and Erfan Nourbakhsh. 2025. Beyond the Hype: Critical Analysis of Student Motivations and Ethical Boundaries in Educational AI Use in Higher Education. *arXiv preprint arXiv:2511.11369* (2025).

[5] Olivia Moore. 2026. *The Top 100 Gen AI Consumer Apps — 6th Edition*. Andreessen Horowitz (a16z). https://a16z.com/100-gen-ai-apps-6/

[6] Riya Ponraj, Yu Wang, and Ramakrishnan Durairajan. 2026. Bootstrapping Transparency in AI-Powered Network Operations: A Network Explanation Framework. In *Proceedings of the ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. Jeju, South Korea.

[7] Diana Maria Popa, Simona-Vasilica Oprea, and Adela Bâra. 2026. Generative-AI and the Transformation of Workforce: A Job Postings-Driven Analysis. *arXiv preprint arXiv:2605.00843* (2026).

[8] Ramakrishnan Durairajan. 2026. *CS 322: Introduction to Software Engineering*. https://classes.cs.uoregon.edu/26S/cs322/

[9] Michael Reicho, Kathrin Otrel-Cass, and Martin Ebner. 2026. Generative AI Literacy Across Education and Business: Competencies, Obstacles, and Benefits—A Systematic Literature Review. *International Journal of Educational Technology in Higher Education* 23 (2026).

[10] UnitedLayer. 2025. *Multicloud AIOps: Smarter IT, Seamless Operations*. UnityOne AI. https://unityone.ai/multicloud-aiops/

[11] David Wood, Jerome S Bruner, and Gail Ross. 1976. The Role of Tutoring in Problem Solving. *Journal of child psychology and psychiatry* 17, 2 (1976), 89–100.