Enhancing Computer Science Education through AI-Driven Scaffolding

Bogdan Denny Czejdo
Fayetteville State University
Fayetteville, NC, USA
bczejdo@uncfsu.edu

Abstract

This paper presents a methodology for utilizing Large Language Models (LLMs) to create AI-driven scaffolding materials in Computer Science. As enrollment in online and hybrid courses grows, the capacity for instructors to provide individual guidance to learners diminishes. This study introduces the "A-H Pedagogical Framework," grounded in cognitive load theory and scaffolding principles. The methodology utilizes a Human-in-the-Loop AI workflow to analyze legacy educational materials (e.g., Jupyter Notebooks, PDFs), identify content and delivery method, and then propose improvements based on the "A-H Pedagogical Framework." Through a case study of GIS Data-Processing assignments, this framework demonstrates how AI can enhance student learning by providing step-by-step, scaffolded guidance while reducing instructors' workload. The AI-driven approach enables scalable education and fosters learning experiences that are otherwise unattainable in large online classes.

CCS Concepts

• **Social and professional topics** → **Computing education**;
• **Computing methodologies** → ***Artificial intelligence.***

Keywords

AI in Education, Scaffolding, Large Language Models, GIS Education, Cognitive Load Theory

ACM Reference Format:
Bogdan Denny Czejdo. 2026. Enhancing Computer Science Education through AI-Driven Scaffolding. In ADMI 2026: Symposium on Computing at Minority Institutions, March 26–29, 2026, Orangeburg, SC, USA. ACM, New York, NY, USA, [4] pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

The shift to online and hybrid learning environments has led to a dilution of personal guidance in educational settings. Students often feel isolated and overwhelmed when tasked with complex assignments without the immediate support they would typically receive in face-to-face settings. For instance, a student learning GIS Programming might struggle with how to effectively apply theoretical concepts to data manipulation tasks. It is evident, especially for non-CS majors taking such courses. Without direct instructor intervention, the learner is left to navigate these complexities alone, thereby increasing cognitive load and leading to frustration.

Cognitive load theory [10] explains that learners have a limited capacity for processing information. When students are presented with tasks that require them to manage both the theoretical knowledge and the technical skills required, their cognitive load can become overwhelming. The gradual increase in complexity without proper guidance leads to cognitive overload, hampering effective learning and retention. In GIS Programming courses, students often encounter a steep learning curve when processing spatial data. They must simultaneously grasp complex coding techniques and domain-specific knowledge, such as coordinate systems and map projections. Traditional instructional methods may not be sufficient in large-scale online courses, where individual attention is minimal. This paper proposes an AI-driven approach that scaffolds students' learning experiences, guiding them step by step through both the theoretical and practical aspects of the domain.

2 Pedagogical Scaffolding Framework

A key feature of scaffolding is *fading*, a process where support is gradually withdrawn as learners gain independence. This concept, as described by Wood et al. [12], emphasizes that learning is most effective when assistance is initially provided but then gradually reduced as learners develop mastery. For example, in the case of file path construction, students may initially receive explicit guidance with code examples to understand the syntax. However, as they gain confidence, the AI shifts to prompting them to solve similar problems on their own. This step-by-step reduction in support aligns with cognitive load theory by avoiding early overload and allowing students to build competence gradually.

While dual coding theory [5] typically focuses on integrating text with images, our approach emphasizes *text-to-code* integration. Initially, concepts are introduced through detailed textual explanations, ensuring that students have a robust understanding of the underlying idea. For instance, before introducing any code, it is explained what a "data frame" is conceptually and why it's important in the context of GIS Programming. Only after this explanation does the introduction of code snippets take place, followed by code with comments. This method helps students internalize the conceptual foundations before delving into the technical aspects of coding. The idea is that students first comprehend the problem in plain language and only later see how it is solved through code, bridging the gap between theory and practice. This approach supports *dual coding* by integrating two forms of representation (text and code) to facilitate deeper learning [5].

ADMI '26, March 26–29, 2026, Orangeburg, SC, USA

ACM ISBN 978-x-xxxx-xxxx-x/26/03
https://doi.org/10.1145/nnnnnnn.nnnnnnn

In addition, recent meta-analyses of AI in programming education suggest that while AI assistants significantly reduce task completion time (SMD = -0.69), they do not automatically improve conceptual understanding [4]. This discrepancy highlights the necessity of our A-H Framework: we must artificially slow the learner down to ensure germane cognitive load is maintained.

Based on such pedagogical foundations, we build our A-H Framework with activities from A to H:

- **Activity A: General Concepts Introduction.** Novices often lack the domain-specific vocabulary required to even understand a technical question. Research supports the use of contextualization by connecting new technical terms to prior non-technical knowledge (e.g., explaining a "data frame" broadly before defining it technically). Other scaffolding techniques can elaborate on properties, functions, and other external relationships of concepts. For complex concepts, Activity A can be repeated multiple times (i.e., for their categories or components).
- **Activity B: Concept Examples Provided (Preparation for Code Creation).** This is well described in the educational literature: the recommended path from general to specific. The path may include as intermediate steps models (data and programming models) commonly used in CS. If the path is longer, Activity B can be repeated until we reach the level of examples. The concept examples are critical to the smooth transition to the code (i.e., they could be specific enough to describe the code samples). In our GIS programming, we primarily work with code snippets; therefore, a textual description of the code, with named functions, was sufficient. The distinguishing feature of this activity B was that it remained text-based.
- **Activity C: Worked Code Examples Provided (Transition to Coding).** For coding implementation, research warns against "unscaffolded" problem-solving, which causes frustration. Valid code scaffolding relies on *Worked Examples*. Providing working code allows students to observe correct syntax and execution before being asked to write it themselves, building a mental model of success without the immediate punishment of syntax errors. These and the next activities are probably the most critical failure points in CS education: the disconnection between the conceptual intent (text) and the syntactic implementation (code).
- **Activity D: Student Executes the Code.** It is often treated as trivial, but executing code properly in an environment with many components requires acquiring some skills.
- **Activity E: Student Explains the Output.** Asking a question about the output.
- **Activity F: Student Explains the Code/Tool.** Explaining the syntax and semantics of the code and action steps and their meaning used here (e.g., "Import Pandas as..."). The relationship between the code and the output could be investigated.
- **Activity G: Student Repeats the Pattern.** Code modification of worked examples is a common technique in CSC education. It requires a creative activity from the student and should be properly scaffolded, i.e., the level of code modification should be consistent with other activities. If there is a requirement for different types of modifications (e.g., different files, parameters, or functions), then Activity G can be repeated several times for each type of modification.
- **Activity H: Review and Deeper Understanding of Concepts.** Asking a question about previous activities related to a general concept.

3 Workflow for A-H Pedagogical Framework

Having established the A-H Framework as the target standard, we now present the methodology for upgrading "Legacy Materials" (e.g., existing Jupyter Notebooks, PDFs, or assignment sheets) to meet this standard. This "Human-in-the-Loop" workflow uses AI to analyze and restructure the teacher's original content.

3.1 Input and AI-Based Analysis

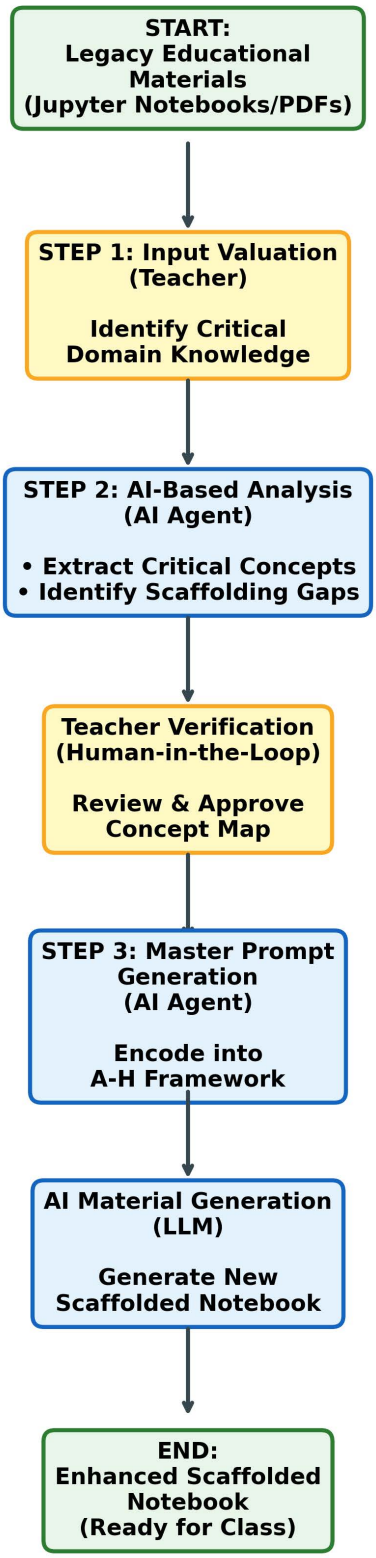
The AI system works by first extracting and analyzing the key pedagogical concepts from the legacy educational materials. Using NLP models and rule-based logic defined in the *Master Prompt*, the system identifies crucial concepts and assesses the existing scaffolding. For instance, if the legacy material asks students to load a dataset but does not provide sufficient context or examples of how to do so, the AI flags this as a gap. The system then provides specific suggestions for how to address these gaps, such as adding an introductory section on how to load datasets in Python.

This step ensures that the original learning objectives are preserved while improving the clarity and pedagogical rigor of the materials. The system's recommendations are designed to align with the *A-H Framework*, which guarantees that each learning task builds upon the previous one, with increasing levels of independence from the instructor. This iterative process of *conceptual refinement* mirrors the work of Chi et al. [2], who demonstrated that effective learning is supported by structured, self-explanatory prompts that guide students through problem-solving.

3.2 Master Prompt

The *Master Prompt* is a two-part specification. The first part provides the rules that ensure that the AI adheres to the A-H Pedagogical Framework. This step involves breaking down the materials into individual *learning concepts* and linking them to specific pedagogical strategies (e.g., using worked examples, providing active learning tasks). The second part involves analyzing legacy materials and mapping them to these predefined scaffolding steps. For example, a problem in the legacy material might involve filtering geospatial data. The Master Prompt will define the *learning concept* (filtering data based on geographic criteria) and determine which scaffolded steps are required (e.g., Step A: Introduce general concept of filtering, Step C: Show syntax with example code). The AI will generate a scaffolded notebook based on this mapping, ensuring that all concepts are fully supported and aligned with the A-H Framework.

Pedagogical Workflow: Enhancing Legacy Artifacts



4 Jupyter Workbook as Educational Delivery Tool

4.1 Cognitive Containment

One of the primary advantages of the Jupyter Notebook is its ability to keep students immersed in the learning process without having to switch between different tools. In contrast, Google Colab or LMS platforms often separate code execution from instructional text, leading to a *Split-Attention Effect* [10], where students' cognitive resources are divided between multiple contexts. By using Jupyter, we eliminate this fragmentation and allow students to work directly with the content in a seamless flow. This integration allows students to apply their learning immediately, minimizing unnecessary cognitive load. Jupyter Notebooks also support *dynamic interaction* with the learning material, enabling real-time exploration and experimentation. This engagement, described by Robins et al. [8], fosters active learning by allowing students to immediately apply what they have learned in a practical context.

4.2 Error Feedback

A major feature of this scaffolding approach is the *intentional use of errors*. When a student makes a mistake, they are presented with Python error messages such as KeyError or FileNotFoundError. These are not failures but learning signals, guiding students to correct their code. Unlike auto-graders that provide immediate feedback of "Correct" or "Incorrect," our system encourages students to debug their code, thus promoting critical thinking and problem-solving skills. While these errors may seem like obstacles, they provide valuable insights into the students' understanding and allow them to improve their coding proficiency. This process reflects principles of *error-driven learning*, which has been shown to enhance retention by encouraging deeper cognitive engagement [2].

5 Application of the A-H Framework in GIS Data Processing Course

In the case study of GIS Data processing, the AI system analyzed legacy materials and suggested pedagogical improvements. For example, the AI identified a gap where students were asked to "load the data" but did not provide an explicit example of how to identify the file path. The AI flagged this as a pedagogical gap and suggested adding an introductory step that explains how to navigate file paths in Python. This addition makes the task more accessible, ensuring that students understand the prerequisites before attempting the assignment. This adjustment mirrors Vygotsky's [11] concept of the *Zone of Proximal Development (ZPD)*, where learners are provided with scaffolding to bridge the gap between their current capabilities and what they can achieve with support.

6 Future Work

Currently, we are working on how this framework could incorporate *adaptive scaffolding*, where the AI dynamically adjusts the support provided to students based on their learning pace. This would allow the system to be more responsive, tailoring the level of assistance based on real-time data about a student's performance. For example, if a student is struggling with basic data loading tasks, the AI might provide more explicit guidance and resources before gradually shifting towards more complex tasks. This vision aligns with recent studies on *personalized learning* that highlight the benefits of tailored, real-time adjustments to instructional content [1].

We also aim to transition from static notebooks to an 'AI-Agent' model. Using an Evidence-Decision-Feedback (EDF) approach, the system will dynamically withhold Step D (Code) if the student has not sufficiently answered Step A (Concepts), effectively enforcing a 'mastery gate' that static PDFs cannot provide [9].

7 Conclusion

This study introduces an innovative approach to improving legacy Computer Science educational materials using AI-driven scaffolding. By utilizing the A-H Pedagogical Framework, we ensure that learning materials are rigorously structured and pedagogically sound. This method allows instructors to provide high-quality, personalized scaffolding at scale, reducing their workload while improving student outcomes. Our approach not only enhances the learning experience but also enables the scalable delivery of high-quality instruction in large online courses.

References

[1] Beck, I. L., McKeown, M. G., & Kucan, L. (2013). *Bringing words to life: Robust vocabulary instruction.* Guilford Press.

[2] Chi, M. T., Bassok, M., Lewis, M. W., Reimann, P., & Glaser, R. (1989). "Self-explanations: How students study and use examples in learning to solve problems." *Cognitive Science*, 13(2), 145–182.

[3] Collins, A., Brown, J. S., & Newman, S. E. (1989). "Cognitive apprenticeship: Teaching the crafts of reading, writing, and mathematics." In *Knowing, learning, and instruction: Essays in honor of Robert Glaser.*

[4] Kazemitabar, M., et al. (2023). "Studying the efficacy of AI-generated learning resources in computer science education: A systematic review." *Journal of Educational Computing Research*, 61(1), 45-78.

[5] Mayer, R. E., & Anderson, R. B. (1992). "The instructive animation: Helping students build connections between words and pictures in multimedia learning." *Journal of Educational Psychology*, 84(4), 444.

[6] Mayer, R. E., & Moreno, R. (2003). "Nine ways to reduce cognitive load in multimedia learning." *Educational Psychologist*, 38(1), 43–52.

[7] Pearson, P. D., Moje, E., & Greenleaf, C. (2010). "Literacy and science: Each in the service of the other." *Science*, 328(5977), 459–463.

[8] Robins, A., Rountree, J., & Rountree, N. (2003). "Learning and teaching programming: A review and discussion." *Computer Science Education*, 13(2), 137–172.

[9] Shute, V. J. (2011). "Stealth assessment in computer-based games to support learning." *Computer Games and Instruction*, 55(2), 503-524.

[10] Sweller, J. (1988). "Cognitive load during problem solving: Effects on learning." *Cognitive Science*, 12(2), 257–285.

[11] Vygotsky, L. S. (1978). *Mind in society: The development of higher psychological processes.* Harvard University Press.

[12] Wood, D., Bruner, J. S., & Ross, G. (1976). "The role of tutoring in problem solving." *Journal of Child Psychology and Psychiatry*, 17(2), 89–100.