Integrating Scaffolding into Computing Education

What is *Scaffolding*?

Scaffolding is a teaching approach that involves providing students with a temporary support structure to assist them in achieving learning objectives that they would not be able to accomplish independently (Vygotsky, 1978). It is defined as the support mechanism that allows students to bridge the gap between their actual developmental level and their potential developmental level. This approach underscores the importance of interaction and dialogue in the learning process, where the instructor's role is to facilitate and gradually withdraw support as the learner gains competence (Erbil, 2020).

As learners gain competence, the support is gradually removed, enabling them to perform independently. This approach not only aids in cognitive development but also fosters metacognitive skills and self-regulation (Lodder et al., 2020).

In the context of *computing education*, scaffolding can help students grasp complex concepts and skills in programming, algorithm design, and software development, among other areas. Kim et al. (2023) demonstrated that analogical reasoning supported by scaffolding helps novice learners in computer science to better comprehend and debug programming tasks.

How to Implement Scaffolding

1. Initial Assessment and Introduction

- Evaluate Current Knowledge: Begin with pre-assessment to gauge students' understanding of the necessary prerequisites.

According to Shin et al. (2017), this assessment helps educators tailor the scaffolding to address specific learning needs, ensuring that the support provided is relevant and effective.

- Introduce Objectives: Clearly present the courses' learning objectives and their relevance in the larger context of computing education.

2. Break Down the Task with Detailed Instructions

- Segment Complex Tasks: Decompose the leaning contents into smaller, manageable steps, offering clear instructions for each phase.
- Provide Multiple Resources: Supply a range of materials, including written guides, video demonstrations, and example codes to cater to various learning preferences.

3. Promote Collaborative Learning

- Organize Peer Activities: Pair students to work on tasks together, facilitating knowledge exchange and support.

4. Provide Targeted Support and Challenges

- Offer Scaffolded Challenges: Present tasks that gradually increase in difficulty, providing just enough challenge to stimulate learning without overwhelming students.

Scaffolding can be categorized into "hard" scaffolds, which are pre-planned and embedded into the learning environment, and "soft" scaffolds, which are dynamic and provided by instructors in response to students' needs (Shin et al., 2017). Educators must decide on the appropriate mix of hard and soft scaffolds based on the learning objectives and the learners' needs.

- Be Available for Guidance: Maintain open channels for questions, whether through office hours, online forums, or direct communication tools.

Why Use Scaffolding in Computing Education?

This method involves providing students with temporary support to help them achieve learning objectives that they might not be able to accomplish independently.

De Souza et al. (2023) demonstrated that project-scale scaffolding helps students tackle large-scale, real-world problems by breaking them into manageable steps.

- Enhances Understanding: Simplifies complex concepts by breaking them down into more manageable parts. Enhances students' understanding by breaking down challenging content into manageable parts and providing continuous feedback.
- Boosts Confidence: Provides students with the tools and support they need to succeed, building their confidence.
- Promotes Independence: Gradually removes support as students become more proficient, encouraging independent problem-solving.

National Science Foundation

This material is based upon work supported by the National Science Foundation under Grant No. 2111097

Employ a variety of scaffolding methods, such as modeling, questioning, providing hints, and collaborative activities, to cater to different learning styles and needs (Brooke, 2015).

5. Encourage Reflective Practice

- Implement Reflective Assignments: Ask students to reflect on their learning process, the challenges they faced, and how they overcame them.
- Discuss as a Group: Facilitate group discussions where students can share their reflections, fostering a sense of community and shared learning.

6. Assess and Adapt Based on Feedback

Effective scaffolding requires continuous monitoring of learners' progress. Educators must observe how learners respond to the scaffolds and make adjustments as needed. This dynamic process ensures that the support remains aligned with the learners' evolving needs.

- Collect Feedback on Scaffolding: At the course conclusion, gather students' feedback on the scaffolding provided and their learning experience.
- Adjust Strategies Accordingly: Use this feedback to refine future scaffolding approaches, ensuring they remain responsive to students' needs and preferences.

7. Gradually Remove Scaffolds

- As learners become more proficient, the scaffolds should be gradually removed to promote independence.

Sample Scaffolding Implementation: Enhanced Guide for a C++ Sorting Algorithm Project

1. Initial Assessment and Introduction

- Evaluate Current Knowledge:
  - Conduct a pre-assessment quiz to gauge students' understanding of C++ fundamentals and basic sorting concepts.
  - Use results to identify knowledge gaps and tailor the scaffolding accordingly.
- Introduce Objectives:
  - Clearly articulate the learning objectives of the project:
  - Understand different sorting algorithms (e.g., Bubble Sort, Quick Sort, Merge Sort).
  - Implement these algorithms in C++.
  - Analyze and compare the efficiency of these algorithms.
  - Highlight the relevance of sorting algorithms in real-world computing applications.

2. Break Down the Task with Detailed Instructions

- Segment Complex Tasks:
  - Divide the project into manageable steps:

1. Introduction to sorting algorithms.
2. Detailed study and implementation of Bubble Sort.
3. Implementation of Quick Sort.
4. Implementation of Merge Sort.
5. Comparative analysis of algorithms.

- Provide step-by-step instructions for each phase.
- Provide Multiple Resources:
  - Written guides with detailed explanations of each algorithm.
  - Video tutorials demonstrating the implementation of each algorithm.
  - Sample codes for reference.

National Science Foundation

This material is based upon work supported by the National Science Foundation under Grant No. 2111097

3. Promote Collaborative Learning

- Organize Peer Activities:
  - Form study pairs or small groups to work on algorithm implementations together.
  - Encourage peer reviews and discussions to enhance understanding and problem-solving skills.

4. Provide Targeted Support and Challenges

- Offer Scaffolded Challenges:
  - Start with simpler tasks like implementing Bubble Sort, then progress to more complex algorithms like Quick Sort and Merge Sort.
  - Gradually increase the difficulty by introducing edge cases and optimization challenges.
- Be Available for Guidance:
  - Schedule regular office hours and set up online forums for questions.
  - Provide timely feedback through direct communication channels.
- Employ a Variety of Scaffolding Methods:
  - Use modeling, questioning, and hints to guide students through challenges.
  - Incorporate collaborative activities to address different learning styles.

5. Encourage Reflective Practice

- Implement Reflective Assignments:
  - Assign reflective writing tasks where students document their learning process, challenges faced, and solutions found.
  - Encourage students to reflect on the efficiency and applicability of each sorting algorithm.
- Discuss as a Group:
  - Facilitate group discussions where students share their reflections and learn from each other's experiences.

6. Assess and Adapt Based on Feedback

- Continuous Monitoring:
  - Regularly observe and assess students' progress to identify areas needing additional support.
  - Adjust scaffolding strategies based on these observations to keep support aligned with students' evolving needs.
- Collect Feedback on Scaffolding:
  - At the end of the project, gather students' feedback on the scaffolding provided.
  - Use this feedback to refine future scaffolding approaches.

7. Gradually Remove Scaffolds

- Promote Independence:
  - As students gain proficiency, gradually reduce the level of support.
  - Encourage independent problem-solving and application of learned concepts in new contexts.

Developed by: Jafar Tavakoli/ jtavakol@purdue.edu

National Science Foundation

This material is based upon work supported by the National Science Foundation under Grant No. 2111097

References

Brooke, M. (2015). Implementing the scaffolding interaction cycle to enable first year undergraduate students to write effective summary-reflections. *Malaysian Journal of ELT Research*, *11*(2).

De Souza, J. G., Flavell, M., Suleiman, A. D., Shepherd, D., DeWaters, J., Small, M. M., ... & Hou, D. (2023, October). The importance of project-scale scaffolding for retention and experience in computing courses. In *2023 IEEE Frontiers in Education Conference (FIE)* (pp. 01-09). IEEE.

Erbil, D. G. (2020). A review of flipped classroom and cooperative learning method within the context of Vygotsky theory. *Frontiers in Psychology*, *11*, 1157–1157. https://doi.org/10.3389/fpsyg.2020.01157

Kim, C., Dinç, E., Lee, E., Baabdullah, A., Zhang, A. Y., & Belland, B. R. (2023). Revisiting analogical reasoning in computing education: Use of similarities between robot programming tasks in debugging. *Journal of Educational Computing Research*, *61*(5), 1036–1063. https://doi.org/10.1177/07356331221142912

Lodder, J., Heeren, B., & Jeuring, J. (2020). Providing hints, next steps and feedback in a tutoring system for structural induction. *Electronic Proceedings in Theoretical Computer Science, EPTCS*, *313*, 17–34. https://doi.org/10.4204/EPTCS.313.2

Shin, S., Brush, T. A., & Glazewski, K. D. (2017). Designing and implementing web-based scaffolding tools for technology-enhanced socioscientific inquiry. *Educational Technology & Society*, *20*(1), 1–12.

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

National Science Foundation

This material is based upon work supported by the National Science Foundation under Grant No. 2111097