International Society of the Learning Sciences

Fostering Computational Thinking Through Engineering Design Activities in a High School Biology Course

Ido Davidesco, Boston College, ido.davidesco@bc.edu
Bianca Montrosse-Moorhead, University of Connecticut, bianca@uconn.edu
Dylan Boczar, University of Connecticut, dylan.boczar@uconn.edu
Julia Oas, University of Connecticut, julia.oas@uconn.edu
Mary-Kate Coburn, University of Connecticut, marykate.coburn@uconn.edu
Aaron M. Dyke, Duke University, aaron.kyle@duke.edu
Leslie Bondaryk, Concord Consortium, lbondaryk@concord.org

Abstract: There is a critical need to incorporate Computational Thinking (CT) into a broad range of STEM courses to promote access to computing education. This study explored whether this could be achieved through engineering design activities embedded in a high school biology course. An interdisciplinary team of researchers and teachers developed a curricular unit where high school students design and program a simplified model of a bionic arm from the sensors up. Quantitative findings revealed significant pre-to-post improvements in some aspects of CT and in engineering design self-efficacy. Qualitative data highlighted students' initial apprehension with programming and engineering tasks, which evolved into engagement through scaffolded support and the use of a visual programming interface. The real-world context of the unit motivated students by linking CT and engineering to biological concepts. This study illustrates the potential of interdisciplinary approaches to the integration of CT and engineering design in STEM courses.

Objectives and significance

While Computational Thinking (CT) originated in computer science, CT practices like abstraction, decomposition, and algorithmic thinking are now embedded in virtually every Science, Technology, Engineering, and Mathematics (STEM) discipline (Malyn-Smith et al., 2018). Recognizing its interdisciplinary relevance, recent reforms in K-12 science education emphasize CT as a core science and engineering practice (NGSS Lead States, 2013). There is a critical need to integrate CT into non-computer science courses, such as biology, to demonstrate the interdisciplinary and fundamental nature of CT and to provide better access to all students (National Research Council, 2012). Biology is a promising context for CT development because computational tools are widely used in biology research (e.g., in vaccine development), yet CT is rarely incorporated into biology courses (Hsu, Chang, & Hung, 2021). Additionally, embedding CT in required science courses (e.g., biology) could expand the reach of CT education.

Even though the term CT was coined nearly 20 years ago by Wing (2006), building on earlier work by Papert (1980), an exact definition of CT remains elusive. In the current paper, we will use the following definition: "Computational Thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Wing, 2010; p. 1). Our conceptual framework builds on a highly cited CT taxonomy by Weintrop et al. (2016). This taxonomy consists of 22 distinct practices mapped to four broad categories: data practices, modeling and simulation practices, computational problem solving, and systems thinking.

Our conceptual framework also highlights the close alignment between CT practices and engineering design. Although the literature presents a variety of models for the design process, most models include (a) problem identification; (b) solution design generation; (c) assessment of solutions against problem requirements; and (d) iterative redesign and refinement (National Research Council, 2012). Each step in the engineering process can potentially engage multiple CT practices (Jacques, 2020). For example, problem identification could encompass "preparing problems for computational solutions" and "defining systems and managing complexity" (both are CT practices listed in the Weintrop et al. taxonomy). Importantly, shifting the focus from coding to engineering design could reduce barriers to entry for teachers and students.

The goal of the current study was to explore how CT can be incorporated into a non-computer science course through engineering design activities. The following research questions were explored: (1) how do students' CT thought processes change throughout their participation in a CT-intensive biology unit; (2) how does students' engineering design self-efficacy change throughout their participation in the unit; and (3) how did students experience CT and engineering design activities?

Methods

Participants

The study involved 158 high school students who participated in the CT unit as part of a biology course in the 2023-24 school year. The students were from five public schools in the northeastern United States. Most of them were in 11th (35%) or 12th (43%) grade. All students provided written assent and had permission from their caregivers to be in the study. All study protocols were reviewed and approved by our institutional ethics board.

CT unit and programming interface

Over a period of two years, an interdisciplinary team of researchers and teachers has iteratively designed a CT-intensive biology unit consisting of 12 45-minute lessons (Table 1). The anchoring phenomenon of the unit is a teenager named Tilly, who lost both arms at a young age and became the first teenager in the United Kingdom to have bionic arms. In the first module of the unit, through a sequence of design and programming challenges at increasing complexity, students learn how to control a robotic gripper using their own electrical muscle activity (i.e., Electromyography, or EMG). In the second module, they revise their bionic arm design to incorporate tactile feedback, allowing the user to detect how much pressure is applied to objects. In the unit's culminating activity, students propose and evaluate engineering solutions that can help improve the lives of different individuals. The curriculum and programming interface (see below) are embedded within the Collaborative Learner User Environment (CLUE), which was specifically designed to promote collaborative learning (Bondaryk, 2020). For example, when students log-in to CLUE they are assigned to a group of up to four students and can share their workspace with group members.

Table 1
Unit Plan

Lesson	Driving question	Learning objectives
1.1	What is a bionic arm?	• Explain the anchoring phenomenon • Develop an initial model of a bionic arm
1.2	How does electricity help us move?	• Program a light bulb to turn on and off • Develop a step-by-step solution to a problem
1.3	How does the brain control body movement?	• Measure your reaction time • Program a virtual gripper to open and close
1.4	How do muscles work?	• Measure and visualize muscle activity
1.5	Can you control a robot with your muscles?	• Design a muscle-controlled gripper
2.1	How does touch impact movement?	• Measure your reaction time with and without tactile feedback
2.2	How do we perceive touch?	• Measure your touch sensitivity • Simulate touch receptors
2.3	Can robots sense objects?	• Revise the gripper to detect pressure
2.4	Can you design a human-machine interface?	• Revisit your initial bionic arm model • Propose and evaluate engineering solutions

Throughout the unit, students use a flow-based programming interface called *Dataflow*, which has been expanded and customized to support the unit (Figure 1). This interface allows students to measure muscle activity and learn to manipulate it into control programs, returning the output to physical devices (e.g., a robotic gripper). This interface eases the burden of writing and 'flashing' text-based programs onto hardware, and it helps reveal the path of signals to students and teachers, which has been shown to be a stumbling block in more opaque systems with fewer modeling cues (Bondaryk, Hsi, & Van Doren, 2021).

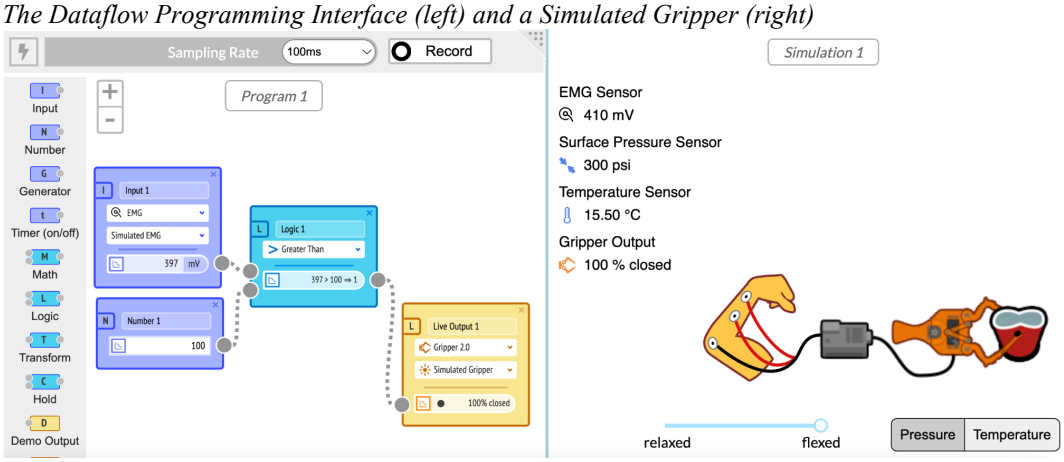
Data collection and analysis

To address RQ1 ("how do students' CT thought processes change throughout their participation in a CT-intensive biology unit"), we assessed students' CT thought processes both before and after the unit using several instruments. In this paper, we focus on data obtained using the Computational Thinking Scale (Tsai, Liang, & Hsu, 2021). This survey consists of 25 items along 5 dimensions (e.g., *I am used to thinking about a problem from a whole point of view, rather than looking at the details*), rated on a 5-point Likert scale ranging from 1 (do not

agree at all) to 7 (totally agree). In the current analysis, we focus on three commonly referenced dimensions of CT: abstraction, algorithmic thinking, and decomposition. The statistical significance of pre-to-post changes within each subscale was assessed using paired samples t-tests.

Figure 1

The Dataflow Programming Interface (left) and a Simulated Gripper (right)



To address RQ2 ("how does students' engineering design self-efficacy change throughout their participation in the unit"), we administered the confidence scale of the Engineering Design Self-Efficacy Instrument (Carberry, Lee, & Ohland, 2010). This survey asked students to rate their level of confidence in performing 9 engineering tasks (e.g., identify a design need) on a scale from 1 (very low) to 7 (very high). A paired-samples t-test was used to assess whether pre-to-post changes in engineering design self-efficacy were statistically significant.

To address RQ3 ("how did students experience CT and engineering design activities"), we conducted a semi-structured focus group post-implementation with three to five students from each class (a total of 7 groups). Students were randomly selected out of those who granted consent to participate in this study. The focus group was used to capture qualitative information on students' experience of the unit. For example, we asked students to reflect on the aspects of the unit that they found most challenging, interesting, or relevant. The focus group was recorded through an audio recorder and transcribed. The current paper consists of an initial thematic analysis of the data to discern learning experiences and challenges.

Results

Quantitative findings

As shown in Table 2, there was a significant pre-to-post increase in the decomposition subscale of the Computational Thinking Scale (d = .24) but not in the abstraction and algorithmic thinking subscales. We also observed a significant positive shift in students' engineering design self-efficacy (d = .27).

Table 2

Pre- and Post-Unit Scores (Mean ± Standard Error of the Mean)

Instrument	Subscale	Pre-unit	Post-unit	P-value	Effect size
Computational thinking scale	Abstraction	4.95 ± 0.08	5.04 ± 0.09	.267	.11
	Decomposition	4.59 ± 0.09	4.77 ± 0.10	.019	.24
	Algorithmic thinking	5.15 ± 0.08	5.14 ± 0.09	.726	.03
Engineering design self-efficacy		4.53 ± 0.08	4.82 ± 0.11	.006	.27

Qualitative findings

Several themes have emerged in the initial coding of the focus group data. First, rather than explicitly using the term "Computational Thinking," students focused on the process of solving a problem using a computer. Since

the unit was implemented in a biology rather than a computer science or engineering course, many students were initially unfamiliar with programming and engineering design. For example, one of the students shared that, "cause I know enough about science to come up with solutions, like from things we've learned. Yeah, but I don't know enough about coding and programming to be an engineer."

Second, the unfamiliarly with programming and engineering design initially triggered anxiety and confusion, but these feelings seemed to change over the course of the unit. For example, one of the students said, "At first I panicked because I thought that it was impossible for me to accomplish making a gripper function on a computer. And I was scared, but when [Teacher] walked us through it, it was actually pretty easy." Similarly, another student shared that, "… the first time we used the EMG I was scared, like connecting myself to the wires, the computer and all that. That was weird at first, but then it was cool." Specifically, the visual programming seemed to contribute to this positive change:

> Yeah, I feel people were at first really confused by block coding and stuff like that. But people started to get it and coding is hard, but it's useful. And I know a lot of people in the school would not wanna do computer science or coding. So it just seems really hard. But this showed that it wasn't that hard. I know a lot of people don't have resources to learn about it.

Finally, several students mentioned that the anchoring phenomenon (designing a bionic arm for Tilly) and the interdisciplinary nature of the unit had a profound impact on their learning experience. For example, one of the students shared that "… engineering is more like kinda mechanic, you're working with metal or electronics … while I feel like being a scientist is more like you're working with life to figure out how to either continue pushing it or help people or... But I feel like they go kind of go hand in hand". This is further illustrated in the following quote from another student:

> "but it [referring to a previous coding experience] wasn't really inspiring as opposed to coding bionic arms for people who need help. And especially as somebody who wants to go into a medical field and help somebody, it was a thing that said like, Hey, engineering ties into this. And I also think engineering is really cool. I'm a very mathy person, I'm a very science person. So to see both of those things connect together, like the EKG readings and all that connecting to the sign graphs that we saw where it went up and down, which is what I was learning in math at the time."

Discussion and conclusions

This study demonstrates the potential for integrating CT practices into STEM courses through engineering design activities. Our quantitate findings show pre-to-post unit improvements in some elements of CT (decomposition) but not others (abstraction and algorithmic thinking). While the unit provided opportunities for students to engage in abstraction and algorithmic thinking, it is possible that more direct instruction and scaffolding are required (a detailed analysis of instructional strategies used in the unit is forthcoming). Indeed, in focus groups, students reported unfamiliarity with programming. This may have led students to solve problems using trial and error instead of developing a systematic, step-by-step solution (i.e., engaging in algorithmic thinking).

Our findings align with prior research on contextualized and integrated STEM education (National Research Council, 2012). The convergence of CT and engineering design in a biology context underscores the need for broader integration of CT across STEM disciplines. Future research should investigate the scalability of this approach to other STEM domains and grade levels. Longitudinal studies are needed to explore the sustained impact of CT-infused curricula on students' STEM identities and career trajectories. Additionally, expanding the focus to other CT practices such as abstraction and algorithmic thinking could reveal strategies to more comprehensively develop these skills. These results provide a foundation for designing inclusive, interdisciplinary curricula that prepare students for the computational demands of the modern workforce.

References

Bondaryk, L. (2020). *CLUE* [Computer software]. The Concord Consortium. https://learn.concord.org/

Bondaryk, L. G., Hsi, S., & Van Doren, S. (2021). Probeware for the Modern Era: IoT Dataflow System Design for Secondary Classrooms. *IEEE Transactions on Learning Technologies*, *14*(2), 226–237.

Carberry, A. R., Lee, H., & Ohland, M. W. (2010). Measuring engineering design self-efficacy. *Journal of Engineering Education, 99*(1), 71–79.

Hsu, T. C., Chang, S. C., & Hung, Y. T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education, 126*, 296–310.

Jacques, L. (2020). Using the Engineering Design Process to Teach Computational Thinking. In *Proceedings of Society for Information Technology & Teacher Education International Conference* (pp. 41-45). Association for the Advancement of Computing in Education (AACE), Waynesville, NC.

Malyn-Smith, J., Lee, I. A., Martin, F., Grover, S., Evans, M. A., & Pillai, S. (2018). Developing a framework for computational thinking from a disciplinary perspective. *Proceedings of the International Conference on Computational Thinking Education*, Hong Kong, China.

National Research Council. (2012). *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. The National Academies Press.

NGSS Lead States. (2013). *Next generation science standards: For states, by states*. The National Academies Press.

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books.

Tsai, M. J., Liang, J. C., & Hsu, C. Y. (2021). The Computational Thinking Scale for computer literacy education. *Journal of Educational Computing Research, 59*(4), 579–602.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127–147.

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.

Wing, J. M. (2010). Research notebook: Computational thinking—What and why? *The Link: The Magazine of the Carnegie Mellon University School of Computer Science*. Retrieved from https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2101615.

