Book Review

Caldwell, J. (2018). *Creative coding: Lessons and strategies to teach computer science across the 6–8 curriculum*. Portland, OR: International Society for Technology in Education. ISBN: 9781564846761 (Paperback), $ 37.95, 138 pages.

Integrating computer science (CS) instruction into K–12 classrooms is now part of education reform in many school systems throughout the United States (Tate, 2018). Currently, forty-four states have adopted at least one of nine policies for providing students access to CS education (Tate, 2018). Moreover, this trend has also created suggestions and efforts to include CS into technology and engineering (T&E) education curriculum (Buckler, Koperski, & Loveland, 2018; Hacker, 2018; Love & Strimel, 2016). To help educators with making this transition, in this first edition, Caldwell provides the K–8 educational community assistance in developing foundational knowledge of both CS and coding as well as tips for collaboration. As teachers build their knowledge of CS, the book also provides them with lessons, strategies, and technology tools that connect computational thinking, coding, and aspects of CS to English language arts, mathematics, science, social studies, and other content and subject areas.

At first glance, Caldwell appears to write primarily to content-area teachers because of the emphasize of aligning CS to ELA, math, science, and social studies. However, he clarifies his position in the introduction of the book by highlighting a section titled "For the Technology or CS Teacher." In this section addressing technology teachers about the importance of teachers working together to make curricular connections for students, he states:

> For those already teaching computer science or coding in some form, you'll find this book to be a useful foot in the door to build cross-curricular opportunities for your students. While all of these projects could be used directly without modification in your technology class, I would urge you to use the content area contextual supports to work your peers teaching other courses. The arguments for teaching CS in each content area and the standards are there to help you make your case for initiating this collaboration. (p. 4)

Additionally, Caldwell expresses that by collaborating with colleagues in other content areas, technology teachers would demonstrate the value of their course or courses while strengthening the learning of CS and coding for students in their schools. This is very important because T&E education course content helps teach and reinforce key concepts and practices found in the standards of the four core disciplines through reading and writing, ratio conversion,

engineering design, and the impacts of technology on society and the environment (just to name a few connections). Therefore, using T&E teachers to aide in the teaching of CS would aide with cross-curricular efforts in schools and in connecting learning for more students.

Throughout the book, collaboration is not just highlighted for educators. As a tool for furthering the learning of CS and coding, the author also provides methods for developing positive group dynamics amongst students. Regarding the practice of students collaborating around computing, he states:

> Any time you're asking students to work collaboratively, whether pair programming or in larger groups, this can be a great practice to help students improve their general collaboration and communication skills. In this practice students should be working towards the following goals:
>
> 1. Cultivate working relationships with individuals possessing diverse perspectives, skills, and personalities.
> 2. Create team norms, expectations, and equitable workloads to increase efficiency and effectiveness.
> 3. Solicit and incorporate feedback from, and provide constructive feedback to, team members and other stakeholders.
> 4. Evaluate and select technological tools that can be used to collaborate on a project. (p. 101)

These collaboration goals described by the author are critical for students to achieve when completing activities in T&E projects because they will learn how diverse people solve computing problems in tandem with technology.

Relevance to Technology and Engineering Education

The author challenges educators to teach students programming skills in authentic and creative ways. Caldwell writes:

> Programming sometimes gets a bad rap for being boring, uncreative, and isolating. None of that's actually true, but sometimes perception dictates reality. Break that stereotype by showing students authentic uses for Computer Science and giving them ownership over how they get to engage with those authentic applications. (p. 32)

Many T&E teachers already infuse programming into classroom projects through the use of various technology tools such as robotics and microcontrollers. However, suggestions in the book provide activities that use a variety of technology (e.g., tutorials and app design). T&E teachers can use these activities to help their students further enhance their programming and coding skills by applying their knowledge through multiple scenarios and contexts. Moreover, students enrolled in T&E classes already participate in real-

world design challenges that are open-ended and academically rigorous. More concentrated efforts to infuse CS, programming, and coding into design challenges may propel some T&E students to pursue critical computing jobs while still achieving competency in technological literacy. According to the K–12 Computer Science Framework Steering Committee (2016), CS is the discipline that enables the use of computers and is the driving force of innovation in all industries and fields of study. Therefore, it is critical that T&E classes promote preparation for computing jobs.

Key Points of Agreement

The author points out that coding is ubiquitous and stresses that all students should learn to code as a foundational skill regardless of what career or studies they plan to pursue. According to Caldwell,

> We teach students about the digestive and circulatory systems not because we expect all students to become doctors, but because we expect engaged citizens to have a fundamental understanding of the world around them. I would argue that how the internet or a smartphone works is at least as essential as the basic biology that we teach all students. How do we expect students to engage thoughtfully in discussions around internet regulation, information privacy, or the role of artificial intelligence without at least a baseline understanding of Computer Science? (p. 9)

These are critical suggestions that can be readily facilitated in T&E classrooms because these are items that pertain to technological literacy. Additionally, the author provides tips and resources in Chapters 4–7 for helping students build the most basic and foundational coding skills through unplugged activities that do not require technology. Unplugged activities are a great scaffold because they allow teachers to introduce CS in ways that assist learners with understanding foundational CS skills. Some of the skills referenced in the book include how to store data, communicate, break down problems into smaller components, and create algorithms for solving problems logically. As the comfort levels and problem-solving skills of students improve, they can move on to more complex and plugged-in scenarios.

Recommendations for Future Editions

In this book, Caldwell made a rigorous effort to provide all teachers interested in integrating CS and coding across sixth- through eighth-grade curriculum with the foundational tools and activities needed for getting started. Although software development is mentioned, many readers would be interested in knowing what software development looks like in the workplace so that they can mimic the process with students. In T&E classrooms, the hierarchy of the software engineering profession can be a useful visual for this purpose (The

Genius Blog, 2017), and an adaptation of it could be incorporated in a future edition. Because both coders and programmers provide the detail work in computer programs and because it pertains to the software development life cycle (SDLC), T&E teachers should inform students that both software developers and software engineers utilize the SDLC for organizing and solving more extensive and complex problems (Hussung, 2016). Also, software engineers act as project managers and do the work of engineers by designing the specs and documentation required by the coders and programmers. Future editions of the book could benefit from such additions.

Conclusion

Caldwell has written an important book filled with tools and resources for T&E and content area teachers looking for ways to integrate CS and coding into their classes. This review has focused on the topics of developing foundational knowledge of CS and coding and increasing collaboration in T&E classrooms. Additionally, the book provides recommendations for content modification, specifically for T&E teachers, with suggestions for the development of an app or apps that can make a myriad of the calculation's students use while designing. Examples of calculations that T&E educators could have students develop an app for include the circumference of a circle and the length of a triangle's sides. Due to these and other logical connections to T&E, it is hoped that readers will begin to infuse CS core concepts and practices into their coursework. The author of *Creative Coding* compels us to make some additions to our curriculum and provides adequate resources for getting started.

References

Buckler, C., Koperski, K., & Loveland, T. R. (2018). Is computer science compatible with technological literacy? *Technology and Engineering Teacher*, *77*(4), 15–20.

The Genius Blog. (2017, February 7). Difference between a software engineer, an application developer and a computer programmer [Blog post]. Retrieved from https://www.kttpro.com/2017/02/07/difference-between-a-software-engineer-an-application-developer-and-a-computer-programmer/#comments

Hacker, M. (2018). Integrating computational thinking into technology and engineering education. *Technology and Engineering Teacher*, *77*(4), 8–14.

Hussung, T. (2016, March 10). What is the software development life cycle? [Blog post]. Retrieved from https://online.husson.edu/software-development-cycle/

K–12 Computer Science Framework Steering Committee. (2016). *K–12 computer science framework*. Retrieved from https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf

Love, T. S., & Strimel, G. J. (2016). Computer science and technology and engineering education: A content analysis of standards and curricular resources. *The Journal of Technology Studies*, *42*(2), 76–86. doi:10.21061/jots.v42i2.a.2

Tate, E. (2018, September 27). States are adopting more computer science policies. Are high schools keeping up? *EdSurge*. Retrieved from https://www.edsurge.com/news/2018-09-27-states-are-adopting-more-computer-science-policies-are-high-schools-keeping-up

Jorge Valenzuela (jvalenzu@odu.edu) is adjunct professor at Old Dominion University.