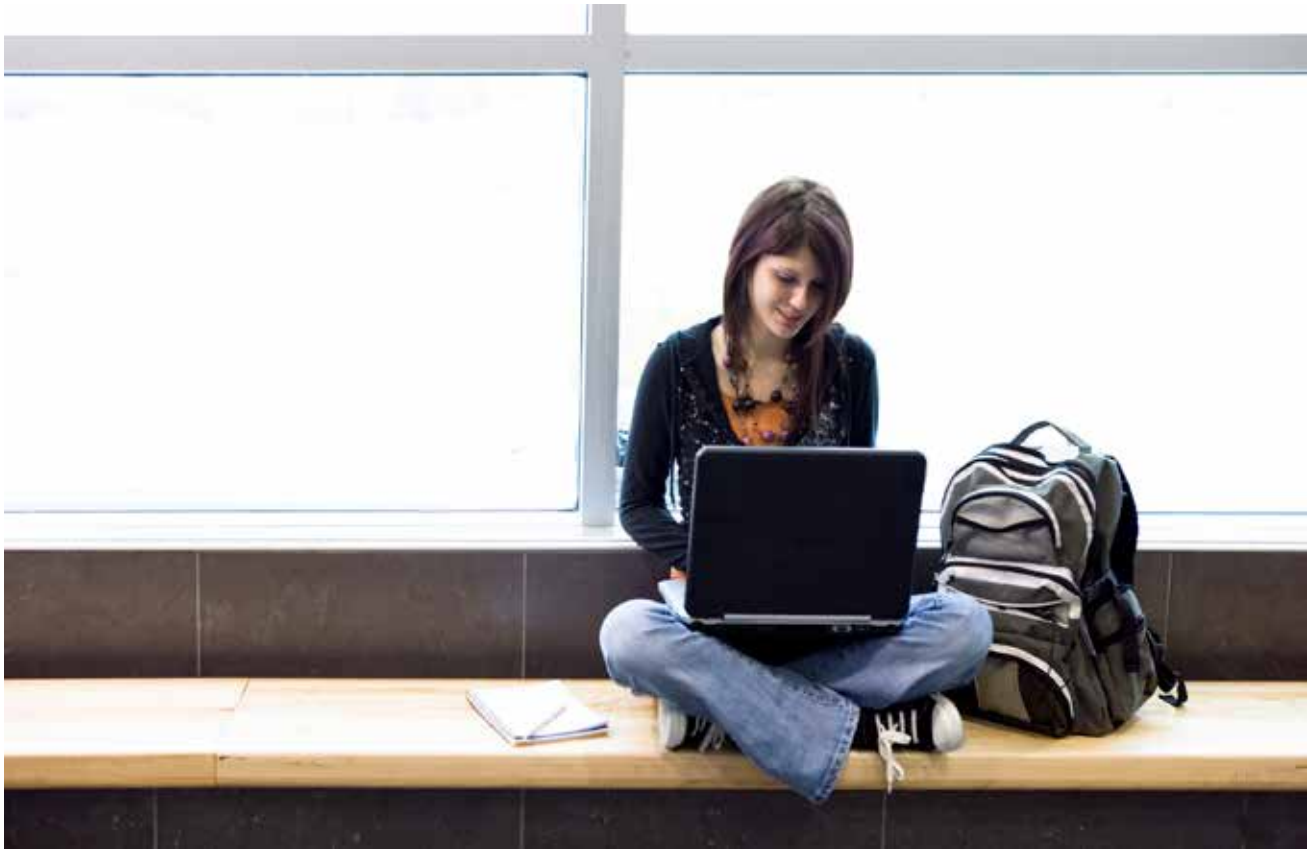
feature article



integrating computational thinking into technology and engineering education

There are several compelling reasons why CT should be integrated into T&E programs.

by Michael Hacker, DTE

Introduction

Computational Thinking (CT) is being promoted as "a fundamental skill used by everyone in the world by the middle of the 21st Century" (Wing, 2006). CT has been effectively integrated into history, ELA, mathematics, art, and science courses (Settle, et al., 2012). However, there has been no analogous effort to integrate CT into Technology and Engineering (T&E) education despite the vast opportunities it provides for engaging learners in CT practices in the context of authentic technological challenges.

Interest in computational thinking is not new. In the 1950s, it was referred to as "algorithmic thinking" (Denning, 2009). It can also be traced to Papert's interest in children working with computers to develop procedural thinking skills (Papert, 1980). A U.S. workforce well versed in CT was advocated by a presidential advisory committee over a decade ago (PITAC 2005).

Many definitions of CT have been proposed (NAS, 2010). The International Society for Technology in Education and the Computer Science Teachers Association (CSTA) have operationally defined computational thinking as:

> A problem-solving process that includes: formulating problems to enable us to use a computer to solve them; logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking; identifying, analyzing, and implementing solutions to achieve the most efficient and effective combination of steps and resources; and generalizing this problem-solving process to a wide variety of problems (Barr, Harrison, & Conery, 2011, p. 21).

CSTA suggests that students should apply CT strategies and tools in virtual and real-world

contexts to be better able to conceptualize, analyze, and solve complex problems (CSTA, 2011). Weintrop et al. (2016) remind us that:

> From a pedagogical perspective, providing meaningful contexts within which CT can be applied differs markedly from teaching CT as part of a stand-alone course in which the assignments students are given tend to be divorced from real-world problems. The sense of authenticity and real-world applicability is important to motivate diverse and meaningful participation in computational activities (p. 128).

Internationally, efforts to include CS in K-12 education are being made in Australia, China, Israel, Singapore, and South Korea (Wing, 2016). The UK Department for Education has provided statuory guidance for CS in the national curriculum. The purpose is to "implement high-quality computing education that equips pupils to use computational thinking and creativity to understand and change the world" (Gov.UK, 2013, p. 1). The curriculum focuses on helping 5- to 16-year-olds:

- Understand and apply the fundamental principles and concepts of computer science, including abstraction, logic, algorithms, and data representation.
- Analyze problems in computational terms and have repeated practical experience writing computer programs in order to solve such problems.
- Evaluate and apply information technology, including new or unfamiliar technologies, analytically to solve problems.
- Act as responsible, competent, confident, and creative users of information and communication technology.

In the U.S., teachers with varied backgrounds teach CS at the K-12 level, and many states do not require computer science certification (Teaching-Certification.com, 2011-2016). Guzdial (2012) has written that, "in most states, CS is classified in the business department, as a vocational education subject." Love and Strimel (2016) identified relationships between ITEEA's *Standards for Technological Literacy* (*STL*) and the K-12 CS Framework. Loveland (2012) discussed how teaching G&M code aligns with *STL* and the NCTM standards. *Next Generation Science Standards* lists "using mathematics and computational thinking" as one of eight "Science and Engineering Practices" (NGSS Lead States, 2013). However, no school discipline has yet to take on CT as a central focus. Since the school day is a zero-sum game, adding stand-alone new courses is a challenge. Integrating CS/CT into existing coursework might be considered more feasible. T&E can take the lead in addressing what is a growing need.

Rationale for Integrating CT into T&E Programs

There are compelling reasons why CT should be integrated into T&E programs. These relate to aligning T&E curriculum and instruction with societal and workforce needs; broadening participation in CT; the feasibility of implementation within T&E programs; and staunching the decline of T&E teachers. The need is real—T&E can help to fill that need, but transformational changes in professional mission, curriculum, and professional preparation are required.

Informed Design Pedagogy

The instructional model for integrating CT into T&E draws upon the informed design pedagogy that has been developed and validated through several large-scale NSF-funded projects (Hofstra, 2004, 2008). Using informed design (Burghardt & Hacker, 2004) students complete a series of just-in-time tasks called knowledge and skill builders (KSBs) that build their knowledge and skill base before they begin designing.

These pre-design KSBs help students gain the CS and CT competence needed to approach designing from a more informed perspective (rather than merely through trial-and-error). The subsequent design challenges call upon students to apply their new knowledge and skills to the modeling of prototypical solutions.

Aligning T&E Curriculum and Instruction with Societal and Workforce Needs

The public strongly supports the need for students to assimilate digital literacy. Silicon Valley executives are funding school-based CS programs (Singer, 2017), but since schools are not moving quickly enough into this space, coding bootcamps are proliferating to bridge the gap. T&E can play a role in helping students learn to become computational thinkers and thus become more highly regarded as part of the educational mainstream. This can be done without compromising the discipline's core mission of teaching students about the human-made world by integrating CS & CT principles, practices, and vocabulary with core T&E concepts—design, systems thinking, modeling, resources, and human values (Rossouw, Hacker, & de Vries, 2011).

Broadening Participation in CT

Integrating CT within project-based T&E contexts has the potential to significantly broaden participation for a large cohort of students (and their teachers) who might not be specifically interested in taking stand-alone CS courses but are interested in designing solutions to technological and engineering problems. Object-oriented programming environments like Scratch and Snap! have been used successfully to engage students (including those from underrepresented groups) in CS (Maloney, Peppler, Kafai, Resnick, & Rusk, 2008).

Figure 1. Supplies for use in physical-world contexts (e.g., robotics and computer control).

With the active advocacy of ITEEA, the 30,000 T&E teachers in the U.S. (Barbato, 2017) can serve as a significant new professional constituency that can support students to learn CS and CT skills.

Feasibility of Implementation

In a survey conducted by Hofstra University and ITEEA in December 2016, data was collected from a sample of T&E educators (n=202) about their interest in adding a CS/CT component to their programs. On a seven-point Likert scale, T&E teachers were highly supportive of adding a CS/CT component to their curriculum (median = 6.3); 76% of HS teachers responding would devote 18 or more weeks to such a program; and notably, 61% would attend a multi-week intensive summer PD program to learn how to teach CS/CT principles.

Design pedagogy is at the core of T&E instruction. Students frame the challenge, clarify criteria and constraints, engage in related research, iteratively generate ideas, make tradeoffs in choosing optimal solutions, develop and test prototypes, iteratively improve designs, and reflect upon and share their thinking with others.

Design thinking is also at the heart of CS and CT in that problems are framed; research is synthesized and informs the design process; problems are addressed through logical and systematic approaches; prototypes are tested for usability with target users; and designs are validated and iteratively improved through feedback.

Facilitating understanding of CS and CT requires conceptual understanding over and above coding skills. Preparing students to become computational thinkers requires a focus on the "big ideas" of computing: creativity, abstraction, programming, algorithms, data/information, the internet, and global impacts of computing (Snyder, Astrachan, Briggs, & Cuny, 2010).

T&E educators have a great deal of autonomy in making curricular choices, as they are normally not constrained by high-stakes testing. It is feasible to teach CT and computer science skills by incorporating real-world computing problems into T&E design challenges.

Stanching the Decline of T&E Teachers

The number of universities granting T&E undergraduate degrees in the U.S. has plummeted from 81 in 1988 (Moye, 2017) to 29 in 2016 (Rogers, 2017) (a 64% decrease); and the number of T&E BS/BA degrees awarded in the U.S. has fallen from 815 in 1995-96 to 206 in 2015-16 (Moye, 2017), a startling drop of 75%. T&E faces an existential challenge. Addressing CT not only will align curriculum and instruction with societal and workforce needs, but has the potential to expand the breadth of our teaching cohort, an issue critical to the survival of T&E education. Young people interested in CS, programming, and data science could serve as a new T&E teaching constituency. This new cohort could add immeasurably to the origination of design problems based on actionable insights from data and the subsequent data-driven analysis and optimization of solutions.

Curriculum and Professional Development

As with the introduction of any new educational program, exemplary curriculum must be provided (with guidance for students and teachers), and associated professional development (PD) should be offered.

Newly Developed or Adapted Curriculum

New curricula can be developed, but to do so requires funding and time for materials development, classroom testing, evaluation, and revision. Alternatively, existing exemplary curricula can be adapted for use in T&E programs.

An example. The NSF-funded *Beauty and Joy of Computing* (*BJC*), an introductory CS curriculum developed at UC Berkeley, is recognized for its appeal to a wide range of students. It uses an easy-to-learn, object-oriented language to teach key CS and CT principles (MSPnet, 2016). *BJC* has been extensively tested by students and teachers, including many in high-minority districts (Price, Albert, Catete, & Barnes, 2015). T&E curricular adaptations would apply *BJC* CS and CT concepts and skills to the solution of design problems in contexts that resonate well with the T&E community.

A sound pedagogical approach would guide students to revisit the same CS and CT concepts in both *physical-world* contexts (e.g., robotics and computer control) and *virtual-world* contexts (e.g., game design). Since most students are familiar with robotics through toys, movies, and industry-based robotic systems and have played electronic games, these contexts are particularly promising for connecting student experiences to computing, technology, and engineering. Curricular development and/or adaptation will most likely require collaboration between T&E and CS educators. Computational thinking can be taught using systems with which T&E teachers are comfortable and familiar, like robotics and game design. See Figures 1 and 2.

Figure 2. Gaming professional development workshop.

The curriculum model shown in Table 1 (page 12) is an illustrative example of how a one-semester course might be implemented to integrate CS and CT concepts and skills within T&E contexts. In this model *BJC* is used as an example of a curriculum to be adapted.

This approach is not intended to teach students to become programmers in languages like Python or JavaScript (this can come later); rather, it serves as an introduction to computer science principles where students will use a block-based, drag-and-drop programming language (Snap!, based on Scratch) to learn and apply key CS and CT ideas.

Professional Development

Inservice T&E teachers will need to learn how to integrate CT into their practice. Thus, development and conduct of PD programs to support implementation is essential. As noted earlier, when surveyed, T&E teachers expressed eagerness to attend intensive PD programs focused on CS and CT. Preservice teacher educators can advocate for programmatic reform, but that will require courage in confronting the realization that, in some cases, our own backgrounds may be insufficient to provide the instruction necessary. Engaging colleagues who have CS expertise could lead to mutually beneficial collaborations.

Research-Based Professional Development

In planning PD programs generally (and especially in areas of endeavor outside teachers' comfort zones), PD initiatives informed by research will have the highest likelihood of success. According to Darling-Hammond and McLaughlin (1995), teachers need a rationale for adopting new curricula. Traditional notions of inservice education need to be replaced by opportunities for "knowledge sharing." Teachers need to learn collaboratively, discuss what they know and want to learn, and engage in planning and evaluating (Darling-Hammond & McLaughlin, 1995). Loucks-Horsley and colleagues (2010) further suggest that programs be linked to school-wide efforts, that teachers help each other and choose their own goals and activities, that ongoing support be provided, and that the focus be on practices that result in improved student learning (Loucks-Horsley, Stiles, Mundry, Love, & Hewson, 2010). The PD plan shown in Table 2 (page 13) illustrates a sample agenda for the design of a two-week PD workshop based on the curriculum model shown in Table 1. The PD plan introduces teachers to the major CS and CT concepts and skills within T&E robotics and game design contexts.

Potential Research Opportunities

Integrating CS and CT into T&E programs offers a rich environment for scholarly research. Possible research questions might include:

RQ1. How should T&E courses be designed to help students learn core CS and CT ideas and capabilities?

RQ2. How can we help T&E educators become competent and comfortable with enacting CS and CT-related projects in their T&E courses and facilitating learning of CS and CT content and capabilities?

RQ3. In the context of T&E education that integrates CT, what does it take to get students to value CT knowledge and capabilities, have interest in continuing to engage in CT, and see themselves as people who engage in CS and CT?

A comparative case study approach might be used to answer these research questions. This methodology would compare and contrast data relative to teacher engagement and student learning in order to extract generalizable lessons learned. Data would help us understand how teachers gain confidence, competence,

Table 1. Robotics and Game Design Adaptations Using Existing *BJC* Curriculum as an Example: Sample KSBs and Design Challenges Within a One-Semester Technology and Engineering Course

This particular example of a T&E curricular adaptation uses informed design methodology (page 9) as the pedagogical backbone.

***BJC* CS and CT Concepts and Skills**	**Robotics/Computer Control**	**Game Design Context**
Building a simple app. Draw, move, and turn sprites.	***KSBs:*** Create a new program to turn LEDs on and off. ***Challenge:*** Design and program a traffic control system for an emergency medical services (EMS) station on a busy highway.	***KSBs:*** Controlling sprites; keyboard input, mouse input. ***Challenge:*** Program keyboard controls to move a small animal to safety.
Building Your Own Blocks (BYOB); using loops.	***KSBs:*** Create custom blocks that use a loop to gradually fade an LED and to change colors of a tri-color LED; move servos and motors; use loops to blink lights and move servos between two positions; use nested loops to create complex combinations of lights and motion. ***Challenge:*** Continue EMS station work.	***KSBs:*** Sprite cloning as object creation; create blocks to draw simple shapes; use loops to make complex patterns. ***Challenge:*** Control a white blood cell that eats replicating bacteria cells, thus protecting the human body from infectious bacteria.
Building grids for games; students use mathematics expressions to draw grids.	*Game design only*	***KSBs:*** Use list and matrix design to simulate probable spread of a forest fire. ***Challenge:*** Create a forest fire and hero to locate and put out the fire.
Conditional blocks; if-else and if statements and predicates (such as < or =) to control a program's behavior.	***KSBs:*** Write conditionals to control a robot using sensors; use sensor inputs to control the output of motors. ***Challenge:*** Design and program a vehicle to move a sample away from a dangerous biological environment.	***KSBs:*** Artificial Intelligence (AI) in games. Watson (IBM's supercomputer) plays Tic-Tac-Toe. ***Challenge:*** Explore programming routines for Watson so it never loses a Tic-Tac-Toe game to a human player.
Script variables; tools and techniques; Boolean operators (and/or/not). Using local variables to control systems.	***KSBs:*** Create a block to find and return the threshold for a sensor; use sensors in compound Boolean statements. ***Challenge:*** Design a physical whack-a-mole game that has a mole appearing randomly using servos and sensors.	***KSBs:*** Script variables, using programming tools. ***Challenge:*** Create an on-screen version of the whack-a-mole game that has a mole appearing randomly.
Using abstraction to write clear, debuggable programs.	***KSBs:*** Abstraction, reducing complexity, increasing efficiency. ***Challenge:*** Sensor-controlled design project (e.g., a vehicle that follows a white-line guide track to enter a radioactive environment).	***KSBs:*** Abstraction, reducing complexity, increasing efficiency. ***Challenge:*** Design a number guessing game that uses abstractions to write clear, debuggable, improvable programs.
Introduction to lists to store data; programs that access and manipulate list contents.	***KSBs:*** Use lists to store sensor data. ***Challenge:*** Design and model an environment to monitor and adjust temperature and light to protect museum artwork.	***KSBs:*** Using lists to store sequence data. ***Challenge:*** Design a Simon game where a list can store a sequence of lights that the user must repeat.
Nesting lists.	***KSBs:*** List matrices used to record ordered pairs to represent sensor measurements at various times. ***Challenge:*** Continue with above.	***KSBs:*** List matrices, placing lists within lists. ***Challenge:*** Continue Simon game development.
Combining list operations; higher-order list-processing functions.	***KSBs:*** Use the map function to scale data before graphing. Use combine to find the mean of data. ***Challenge:*** Continue with above.	***KSBs:*** Combining lists, linked to writing a script for the "add item" button. ***Challenge:*** Continue with above.
Algorithms and data; graphing; timers; reporters.	***KSBs:*** Acting on input data algorithmically. Controlling multiple outputs. ***Challenge:*** Continue with above.	***KSBs:*** Timers, reporters, modeling a graphing app. ***Challenge:*** Design a simulation relating population size to the rate of disease spread.

and understanding and what instructional practices might be implemented as teachers hone their CT skills.

Summary

Integrating CS principles and CT within T&E can expand the role the discipline plays in all students' fundamental education, can broaden participation in computing education, and can increase T&E's status within the educational system.

Presently, no discipline has taken upon itself the responsibility of being the primary instructional vehicle to teach CT in the nation's schools. T&E can take great advantage of this opportunity—without compromising the discipline's core mission of teaching students about the human-made world—by integrating CS & CT principles, practices, and vocabulary with core T&E concepts. It is feasible to teach CT and computer science skills by incorporating real-world computing problems into T&E design challenges.

Table 2. Sample Two-Week Workshop Schedule

Week 1	Monday	Tuesday	Wednesday	Thursday	Friday
AM	Introduction to integrating CS and CT within T&E. First look at Snap! Developing a phone app.	Build your own block! Blocks and scripts. Intro to control technology. Use blocks to control an LED.	Intro to lists and matrix design tools. Intro to abstraction. Complete white-blood-cell challenge.	More complex lists (e.g., nested) and programs. Use conditionals to control motor and servo outputs.	More complex algorithms. AI and its role in game design. Complete Tic-Tac-Toe game challenge.
PM	Experiment with basic commands. Write a program to move a sprite. On-screen drawing; make sprite follow the mouse.	Use scripts to create a light sequence. Complete the EMS station challenge. Discuss societal aspects, e.g., games & violence; discuss *Blown to Bits* book.	Game design using conditionals and script variables. Design a forest fire game: be a hero and put out a fire! Discuss privacy issues.	Design, make, and program a simple vehicle that reverses when it touches an object. Discuss copyrights and how they work in the computer world.	Use script variables. Linking game screen activity to HB inputs. Complete on-screen "Where in the World" challenge (to determine the best places to live in the future) using four digital inputs.
Week 2	**Monday**	**Tuesday**	**Wednesday**	**Thursday**	**Friday**
AM	Graphing in the real world: using analog inputs to display data on screen. Data tables and storing information.	Timers and timing. Design an "against the clock" number guessing game and improve it to store results of multiple tries.	Using the design journal as a scaffold, design, make, and program a four-mole Whack-a-Mole on-screen game.	Using the design journal, design, make, and program a vehicle that follows a white line guide track to enter a radioactive environment.	Managing the integration of CS and CT in the T&E classroom.
PM	Design and make a monitoring system for grow room temperature. Discuss encryption of data.	Linking digital and analog inputs to outputs. Simulate a one-axis prosthetic wrist. Discuss the workplace impact of robotics.	Extend the onscreen game to the real world using servos and switches. Present your work to group.	Improve the design to grasp the radioactive flask and remove it.	Present your work to group. Group discussion; implementation planning; evaluation, and feedback.

Doing so will require changes in curriculum and in the way teachers at preservice and inservice levels are prepared. This is well within the capability of those in our profession who are willing to be courageous enough to learn the necessary skills to lead what could be a transformative reform effort, well aligned with the transition from technology education to technology and engineering education.

The integration of CS and CT within T&E provides a rich area of inquiry for researchers to investigate how educators within and beyond T&E might optimize curriculum and pedagogy focused on broadening CS and CT participation.

References

Barbato, S. (2017, January 3). Personal email correspondence.

Barr, D., Harrision, J., & Conery, L. (2011). *Computational thinking: A digital age skill for everyone.* International Society for Technology in Education (ISTE), p. 21. Retrieved from www.iste.org/docs/learning-and-leading-docs/march-2011-computational-thinking-ll386.pdf

Burghardt, M. D. & Hacker, M. (2004). Informed design: A contemporary approach to design pedagogy as the core process in technology. *The Technology Teacher, 64*(1), 6-8.

Computer Science Teachers Association (CSTA) Standards Task Force. (2011). In *CSTA K-12 computer science standards*, p. 9. Retrieved from http://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/Docs/Standards/CSTA_K-12_CSS.pdf

Darling-Hammond, L. & McLaughlin, M. (1995). Policies that support professional development in an era of reform. *Phi Delta Kappan, 76*, 597-604.

Denning, P. J. (2009). The profession of IT: Beyond computational thinking. *Communications of the ACM, 52*(6), 28-30. doi: 10.1145/1516046.1516054.

Gov.UK. (2013, September 11). *National curriculum in England: computing programmes of study,* p. 1. London, England: Department for Education.

Guzdial, M. (2012, October 22). *Why isn't there more computer science in U.S. high schools?* Blog@CACM. Retrieved from https://cacm.acm.org/blogs/blog-cacm/156531-why-isnt-there-more-computer-science-in-u-s-high-schools/fulltext

Hofstra University. (2004). *The New York State curriculum for advanced technological education* (*NYSCATE*). Retrieved from www.hofstra.edu/academics/colleges/seas/ctl/nyscate/index.html

Hofstra University. (2008). The Mathematics, Science, and Technology Partnership Project (MSTP). Retrieved from www.hofstra.edu/academics/colleges/seas/ctl/mstp/index.html

Loveland, T. (2012). Understanding and writing G & M code for CNC machines. *Technology and Engineering Teacher, 71*(4): 24-29.

Loucks-Horsley, S., Stiles, K., Mundry, S., Love, N., & Hewson, P. (2010). *Designing professional development for teachers of science and mathematics* (3rd ed.). Thousand Oaks, CA: Corwin Publishing.

Maloney, J. H., Peppler, K., Kafai, Y., Resnick, M., & Rusk, N. (2008). Programming by choice: Urban youth learning programming with Scratch. *ACM SIGCSE Bulletin, 40*(1): 367-371.

MSPnet. (2016, January 27). *The beauty and joy of computing website*. Retrieved from http://hub.mspnet.org/index.cfm/29055

National Academy of Sciences. (2010). *Report of a workshop on the scope and nature of computational thinking.* Washington, DC: National Academies Press.

NGSS Lead States. (2013). *Next generation science standards: For states, by states.* Appendix F–Science and Engineering Practices in the *NGSS*. Washington, DC: The National Academies Press. Retrieved from www.nextgenscience.org/sites/default/files/Appendix%20F%20%20Science%20and%20Engineering%20Practices%20in%20the%20NGSS%20-%20FINAL%20060513.pdf

National Science Teachers Association (NSTA). (n.d.). *Science and engineering practices. Using mathematics and computational thinking.* Retrieved from http://ngss.nsta.org/Practices.aspx?id=5

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

President's Information Technology Advisory Committee (PITAC). (2005). *Report to the President. Computational science: Ensuring America's competitiveness.* Arlington, VA: National Coordination Office for Information Technology Research and Development.

Price, T., Albert, J., Catete, V., & Barnes, T. (2015). *BJC* in action: Comparison of student perceptions of a computer science principles course. *Research in Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)*. Retrieved from http://ieeexplore.ieee.org/document/7296506/?reload=true

Rossouw, A., Hacker, M., & de Vries, M. J. (2011). Concepts and contexts in engineering and technology education: An international and interdisciplinary Delphi study. *International Journal of Technology and Design Education, 21*, 409. doi:10.1007/s10798-010-9129-1.

Settle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., & Wildeman, B. (2012). Infusing computational thinking into the middle- and high-school curriculum. In: *Proceedings of the 17th ACM Conference on Innovation and Technology in Computer Science Education.* Association for Computing Machinery, New York, pp 22-27.

Singer, N. (2017, June 6). The Silicon Valley billionaires remaking America's schools. *The New York Times.* Retrieved from https://mobile.nytimes.com/2017/06/06/technology/tech-billionaires-education-zuckerberg-facebook-hastings.html

Snyder, L., Astrachan, O., Briggs, A., & Cuny, J. (2010). *An open letter to the computer science community: AP computer science principles.* Retrieved from https://csprinciples.cs.washington.edu/thecase.html

Teaching-Certification.com. (2011-2016). *Computer science teacher certification.* Retrieved from www.teaching-certification.com/computer-science-teacher-certification.html

Weintrop, D., Beheshti, El., 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*, 125-147. DOI 1.1007/s10956-015-9581-5.

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

Wing, J. M. (2009). Computational thinking. J*ournal of Computing Sciences in Colleges 24*(6), 6-7.

Wing, J. M. (2016, March 24). Computational thinking, 10 years later. *Phys.org.* Retrieved from https://phys.org/news/2016-03-years.html

Acknowledgements

The author would like to acknowledge the contributions made by several colleagues to his understanding of the potential of computational thinking to support T&E instruction. Dr. Tiffany Barnes (North Carolina State University) and Tony Gordon (Hofstra University) helped the author to better understand how CT might be integrated within robotics and gaming contexts; and Dr. Janet Kolodner (The Concord Consortium) framed the research questions suggested herein as potential research opportunities.

Michael Hacker, Ph.D., DTE, *is Co-Director of the Hofstra University Center for STEM Research. He served as the EfA Project Principal Investigator. He can be reached at Michael.Hacker@hofstra.edu.*

This is a refereed article.