Table of Contents

Volume XLIV, Number 1, Spring 2018

Infusing Computer Science in Engineering and Technology Education: An Integrated STEM Perspective

By Paul A. Asunda

ABSTRACT

This study examined how four engineering and technology education teachers infused Computer Science Principles (CSP) and Computational Thinking (CT) practices into their classrooms from an integrated STEM perspective. Two questions guiding this inquiry were: (1) How do engineering and technology education teachers infuse CSP and CT into engineering and technology education? (2) How do engineering and technology education teachers assess students CSP and CT projects that are integrated with engineering and technology education? Data were collected through class observations and semi-structured interviews. Using an instrumental case study approach this study identified key themes; pedagogy, programming, assessment, and problem solving as strategies K-12 teachers should consider when designing instruction that seeks to infuse computer science principles, and computational thinking in engineering and technology education and integrated STEM coursework.

Keywords: computer science, computational thinking, integrated STEM, engineering and technology education, assessment, and problem solving

INTRODUCTION

Skills in the 21st century center on the ability to analyze data, think critically, and solve problems both in teams and as individuals. Cultivating students with these types of skills requires an emphasis on STEM education paired with the breakthrough possibilities that facilitate creativity in ideas and exploration. Recent national reports emphasize the importance of Computer Science (CS) within K-12 curricula, and highlight concerns about national competitiveness and adequate workforce training in the global economy (National Science and Technology Council, 2013; The Office of Science and Technology Policy [OSTP], 2014; White House Fact Sheet, 2014). The teaching of CS at the K-12 level seeks to provide all students the opportunity to learn CSP and develop CT skills deemed necessary for success in the technological society (Yadav, Hong, & Stephenson, 2016). This attention may be in response to the growing demand for individuals with computer science-related skills and who are prepared to address critical issues such as cyber security attacks (Koch & Gorges, 2016). As such, there is need for a well-prepared workforce that can efficiently integrate and apply any or a combination of the CSP seven big ideas and CT skills in their work places (Mohaghegh & McCauley, 2016). The CSP big ideas are creativity, abstraction, data, algorithms, programming, Internet, and global impact. In addition to the seven big ideas underpinning computer science, six computational thinking practices typify the kinds of activities computer scientists engage in, and by extension, must typify the learning outcomes of a computer science course. These are companions to the seven big ideas. These six ideas include the following: analyzing the effects of computation, creating computational artifacts, using abstractions and models, analyzing problems and artifacts, communicating processes and results, and working effectively in teams (Synder, Astrachabm, Briggs, & Cuny, 2011).

The Framework for K-12 Science Education and the Next Generation Science Standards (NGSS) lists CT as one of the eight science and engineering practices. These standards emphasize the integration of science and engineering practices, crosscutting concepts, and core ideas in science disciplines in K-12 curricula. Such integration may refer to making meaningful connections between CSP and CT practices, and the core disciplinary practices of each STEM domain, with the goal of using this integrated knowledge to solve real-world problems. Research reveals that integrated learning also appeals to educators, because it projects real-world experiences, links subject areas, and fosters collaboration and networking among teachers (Hecht, Russo & Flugman, 2009; Siew, Nazir, & Chong, 2015). As such, Integrated STEM (i-STEM) education is a relatively recent phenomenon, particularly as it is still uncommon in K-12 classrooms (Chiu, Price, & Ovrahim, 2015). Integrated STEM has been viewed as an approach to teaching and learning in a manner such that the curriculum and content of the four individual STEM disciplines seamlessly merge into real-world experiences contextually consistent with authentic problems and applications in STEM careers (Mobley, 2015; Sanders, 2009). Synthesizing lessons from schools that integrate STEM practices with computer science principles

and computational thinking skills may begin to tell a story about how teachers plan, instruct, and assess CSP and CT in STEM programs.

Engineering and Technology education programs offer curricular flexibility that provides a variety of approaches to the infusion of computer science fundamentals into a K-12 curriculum. However, little information regarding K-12 computer science program development and integration into STEM areas is available in scholarly literature. To this end, two questions guiding this inquiry were: (1) How do engineering and technology education teachers infuse computer science principles (CSP) and CT into engineering and technology education? (2) How do engineering and technology education teachers assess students CSP and CT projects integrated with engineering and technology education?

METHOD

The interdisciplinary aspect of i-STEM provides a rich test bed to infuse computer science principles that enhance CT. Such an approach allows numerous ways students at the K-12 level exercise critical thinking as they explore the very many ways a design challenge may be solved (Curzon, Peckham, Taylor, Settle, & Roberts, 2009). As such, this study purposefully selected and utilized snowball technique (Patton, 2002) to examine how four engineering and technology education teachers who had attended and completed Project Lead the Way (PLTW) summer CS training infused CSP and CT practices into their classrooms. The epistemology for this research was constructionism, the focus being the construction of meaning from the perspectives of these four teachers' beliefs and practices within the context of CSP, CT, and i-STEM in engineering and technology education. Crotty (1998) stated that constructionism is the view that all knowledge, and therefore all meaningful reality as such, is contingent upon human practices, being constructed in and out of interaction between human beings and their world and developed and transmitted within an essentially social context. This inquiry was designed to be a multi-site collective case study with each participant being viewed as a unit of analysis. Participants for this study were four high school teachers who were studied individually as cases and jointly examined to better understand their experiences. The study was limited to high schools within a radius of 100 miles from the researcher. In addition to driving the long distances to conduct interviews and observe teachers' natural settings, a challenge to the study was the frequency with which classroom observations were scheduled in order to observe students' working on CS-related projects. These teachers taught in the Midwest region of the United States and worked with approximately 200 students (Grades 9-12). Two of the teachers (pseudonyms Alex and John) taught at ABC high school, which enrolled approximately 600 students, with the engineering and technology courses being electives and attracting around 80 students (freshmen through seniors). Alex had three years of teaching experience, and John who had recently graduated from college was a first-year high school teacher. Teacher Cory (pseudonym) taught at EFG high school, which enrolled approximately 700 students, where approximately 70 students were pursuing engineering and technology, and other career and technical education (CTE) courses as electives. He had taught high school for a total of 20 years and was a Master PLTW teacher who had trained many teachers in PLTW curricula in the Midwest region. Teacher Brown (pseudonym) taught at MNO high school, which enrolled approximately 650 students, and had around 90 students (9-12 grade students) enrolled in the engineering and technology education courses. Teacher Brown had taught middle school for 7 years before transitioning to the high school setting, where he had taught for the last 12 years. All four teachers had graduated with a bachelor's degree in technology education / engineering and technology education. Teacher John also had a computer science minor for his undergraduate degree in his teacher preparation program.

DATA COLLECTION

A classroom observation coding instrument was developed to examine teacher's practices with students during instruction of CSP and CT in an i-STEM environment. Specifically, the practices to be observed were selected from the Secondary Science Teacher Analysis Matrix (STAM) (Gallagher & Parker, 1995) to code specific behaviors and actions within the teacher's classroom. The rationale for selecting the STAM instrument items was that it addressed the research questions guiding this study: It guided the observation of CSP and CT practices related to engineering and technology education focused around the integration of STEM concepts. As such, the researcher observed and interviewed teachers based on the following items: *structure of content; examples and connections; methods; labs, demonstrations and hands-on involvement; kinds of assessments; students' questions; student-initiated activity; students' understanding of teachers expectations; resources available, and students' works.*

The observational data were collected from classroom visits, which included a total of 4-5 visits that lasted 50 minutes at all three schools. The total was 15 visits: teacher Cory was visited 6 times, Alex and John were visited 3 times each (i.e., 6 visits between the two teachers), and teacher Brown also was visited 3 times. Face-to-face interviews with each teacher lasted up to 50 minutes. As such, a total of 15 lessons were observed, and a semi-structured interview with open-ended questions was utilized to supplement classroom observation data. Berg (2001) stated that semi-structured interview guides allowed the interviewer to probe far beyond answers that might be generated by pre-prepared standardized questions. Likewise, Patton (2002) posited that open-ended interview questions enabled researchers to understand and capture participants' views. The four teachers were then invited to participate in a 50-minute interview, and all the interviews were audio recorded and transcribed verbatim.

DATA ANALYSIS

One of the researcher's challenges was to obtain and verify the true meaning of each participant's responses to the questions asked (Gall, Gall, & Borg, 2003). To begin making meaning of collected data (i.e., the interview data from the four teachers and classroom observations), the four interviews were analyzed separately as described by Miles and Huberman (1994) during data reduction, data display, conclusion drawing, and verification phases. The data analysis process helped the researcher approach the data without preconceptions about teacher's beliefs and practices. During this process the researcher reflected on the purpose of the study and the guiding research questions as they noted phrases and words that revealed each participant's CS teaching practices integrated in engineering and technology education with a focus in STEM experiences. The researcher then identified text segments that contained the same meaning and sought to derive *in vivo* codes from transcripts by identifying repetitive, descriptive, and interpretive phrases of participants' experiences, which were then developed into categories such as programming and pedagogy. Boeije (2009) stated that *in vivo* codes are not just catchy words; rather, they pinpoint the meaning of a certain experience or event. The researcher identified 14 initial *in vivo* codes in order (i.e., *computer science, evaluation, information science, criticism, data, pedagogy, computer programming, assessment, technology, problem solving, design, teaching, coding, open ended).* These codes were then compared with classroom observation data as a triangulation measure to further affirm the initial codes that had emerged from the classroom interviews into categories and subcategories. Afterward the researcher wrote memos describing identified categories to further reduce the data. Participants' explanations and ideas that had similar meanings were then collapsed into key categories informed by subcategories identified by reviewing the initial categories and participants' transcripts again. However, it should be noted that emergent categories had text descriptors in identified subcategories that overlapped. At this juncture, the researcher then embarked on establishing reliability of emerging themes by sharing these initial codes and descriptors with study participants for member- checking purposes through email correspondence for more than a month (Mays & Pope, 1995). Mays and Pope (1995) use the term "reliability" and claim that it is a significant criterion for assessing the value of a piece of qualitative research. During this process study participants crystallized their meanings and reduced the initial codes to eight, again in no order of priority to *computer science, programming, teaching, coding, pedagogy, open-ended, evaluation, and assessment.* The researcher then grouped these codes with accompanying text as suggested by participants into relevant categories. After member checks and reliability testing, the researcher proceeded to use Microsoft excel to display and organize data for cross-case analysis. Miles and Huberman (1994) defined cross-case analysis as searching for patterns, similarities, and differences across cases with similar variables and similar outcome measures. The researcher took note of both units to be eliminated and those that would be retained. Related terms and data with similar expressions (e.g., terms like "programming" "coding") that study participants had pointed out during member checking to express similar meanings were further grouped together into identified categories. The researcher then embarked on developing themes by grouping identified categories that had similar meaning into core themes. Table 1 provides the themes identified during this stage of analysis; they are termed as categories, along with subcategory labels, and descriptions as per the participants of this study.

Table1: Categories That Informed Themes Generated by Researcher

Categories	Subcategories	Descriptors
Pedagogy	Interactive	Working together with, and having an influence on learning process of students
	Learning curve	Students' progress in gaining new knowledge and skills.
	Remember	Recall an event or an experience from the past.
	Work sheets	Paper that teacher shares with student to help them learn similar concepts and skills as they progress through a given unit.
	Culminating projects	Series of related projects that give students an opportunity to demonstrate what they have learned at end of a given unit.
	Tutorials	An interactive method of learning that demonstrates concepts/ skills/knowledge you want students to attain.
	Standards	Documented specifications that recommend what students should know and be able to do at a given grade level.
	Unit of learning	Coherent set of concepts that teachers will instruct over a period of four to five weeks.
	Backward design	A method of designing a unit of learning by setting an end goal you want students to attain before choosing instructional methods and forms of assessment.
	Scaffolding	Using a variety of instructional techniques to help students learn progressively toward attainment of a given end goal.
Programming	Coding / Programming	Writing a set of instructions to execute a desired end goal in a computer program.
	Scripting languages	A form of communication, as such instructions that computers utilize to execute give task to attain a desired goal (e.g., HTML, JAVA).
	Syntax	Relates to the spelling and grammar of a programming language and hence good clean code.
	Tool kits	Companion wizard like program that helps students learn a given programming language (e.g., Tkinter in Python).
Assessment	Criticism	Offering value statements to make students' work better.
	Correlation	Connecting end result of students work with process.
	Understand	Students being able to demonstrate the desired end goal of unit of leaning set by teacher.
	Rubrics	Coherent set of criteria that reflect given standards and includes descriptions of expected levels of student's performance.
	Evaluation	Assigning value statements to students finished assignments.
	Documentation	Student's written thoughts reflecting how they arrived at their final solution.
Problem solving	Open ended	No specified conditions that hinder the adoption of multiple solutions to a design challenge.
	Backward design	No specified conditions that hinder the adoption of multiple solutions to a design challenge.
	Scaffold	Using a variety of strategies to help provide a solution to a design challenge.
	Planning	Process of thinking about, and organizing a strategy and key activities required to solve a given design challenge.
	Iterative	Repeating a given procedure in an effort to optimize possible solutions to a given design challenge.

FINDINGS

This study sought to find out how engineering and technology education teachers infuse computer science principles and computational thinking into engineering and technology education, in addition to their assessment practices. Quotes from the four teachers have been used throughout this section to emphasize core themes that emerged with no observed priority or order. Four core themes (pedagogy, problem solving, programming, assessment) were identified from the reduced meanings of participants' verbatim transcripts. Verbatim quotes from participants were used throughout this section to emphasize core themes.

Core Theme Pedagogy

Responses supporting this theme offer insights into how teachers scaffold CSP and CT into engineering and technology education. Cory shared that integration of computer science into engineering and technology education courses had been facilitated by organizations like code.org, and the Computer Science Teachers' Association (CSTA) working closely with the College Board. He shared that many forms of curricula exist that teachers could utilize. For example, author please explain abbreviation (PLTW) of which teacher Cory was certified to teach, and had facilitated professional development sessions to prepare other teachers integrate CSP and CT in their engineering and technology courses. He was of the view that these curricula offered a framework that teachers could utilize to teach CSP and CT in engineering and technology education. In contrast, Alex defined scaffolding as a teaching and learning strategy by which he helps his students learn how to write code and develop programming skills through toolkits and worksheets. Brown shared that he used a mix of hands-on activities and projects to help his students with CSP and CT concepts, whereas John viewed the process of teaching CSP and CT through design problem solving challenges to be synonymous to engineering design practices. In visiting the classrooms, the researcher was able to observe students working on various projects. For example, a data visualization project that required students to design data collection tools and utilize the tool to collect data that they could eventually analyze; design of interactive graphical user interface (GUI); and raspberry pi projects that looked into designing alarm systems controlled by a sensor and a camera. These projects required that students have some coding background. Cory used reflecting practices and recall procedures to help students relate their current projects to previous learning experiences. For example, he stated " … other languages as one of the common widget is a canvas as I have mentioned before and we have all encountered that all the way back to our very first day in scratch right a big canvas that you use again." In contrast, teacher Alex mentioned that this being new to his school and teaching, he incorporated a survey to help him understand his students' needs and what kind of projects they would enjoy. Especially because engineering and technology courses were electives, he had to find a balance to grow the program. He also mentioned that he would give course material that looked into CSP and CT upfront so as to give his students background information. For example, he mentioned that cybersecurity was new to him and to engage his students with hands-on activities he shared that, "I went to a workshop this summer, I got a lot of materials on cybersecurity design challenges, I use these materials to guide students [to] make their own encryption device of some sort." Brown shared that there were worksheets that he accessed from the Internet to help his instruction, and John pointed out that he also used worksheets in his teaching; he was quick to mention NGSS as key factors in his teaching and planning. This was a sentiment shared by the other teachers, and they recognized the value of standards in the planning of their teaching of CS concepts in engineering and technology education. One key aspect all the teachers shared is that the projects they conducted were culminating in nature. As such, a given CS unit and designated project typically runs approximately four weeks with some buffer time built in. He pointed out that his teaching of CS including projects was modeled around the engineering design cycle.

Core Theme Programming

John noted that the good thing about computer science "is that there are a lot of different programming languages, and some of them are far easier to learn than others." John who had a minor in computer science stated that he helps his students learn how to write code by asking them to first verbalize what they want the code to do. He stated, "before you write any code or anything you have to think through the problem in your head. So I can give the students a problem and then have them explain to me or a partner how they would solve it and the write it

down step by step." All the teachers mentioned that they infused object- oriented programming (OOP) into their engineering and technology education courses that enhance STEM learning. According to Cory this is when "someone combined data and functionality and wrapped it inside something called an object." For example, in one class observation, Cory had his students program a sphere that could be programmed to change location on the canvas, through a toolkit called Tkinter in python, one of the many toolkits he shared with his students. He began by having the students generate code to create the canvas and a sphere (i.e., the object). As they did this exercise Cory reminded his students about classes and objects. As the students worked in the OOP environment, classes and objects became the two main aspects of object-oriented programming. A class creates a new *type* where objects are instances of the class (Corradi & Leonardi, 2001). Further, teachers, Brown, Cory, Alex, and John shared that that they all introduced their students to scratch software as an initial tool to generate CS and programming interest in their students. Scratch (https://scratch.mit.edu/) is a free programming language developed by the Massachusetts Institute of Technology (MIT) whereby individuals can program their own interactive stories, games, and animations and share their creations with others in the online community. Cory mentioned that he started out with Scratch, and then introduced his students to app inventor, an open-source web application originally provided by Google, and now maintained by MIT, and finally Python, a high-level programming language for general-purpose programming. In contrast, Alex and John mentioned that they looked into Python and JAVA, while Brown worked with Scratch and Python. John noted that Scratch gave students an opportunity to see drag and drop functionality from a programming perspective. He also added that Hypertext Markup Language (HTML) and other languages like Cascading Style Sheets (CSS) interested his students. According to Alex, he emphasized some key terms when teaching programming, and asked his students to take note of when learning programming included syntax, algorithms, good code, and control flow.

Core Theme Problem Solving

All the teachers attested that the essence of teaching CS to their students as an alternative way to equip them with problem solving skills. They all noted that solving a problem in CS resembled the process of solving a design challenge in engineering and technology education through the process of engineering design. Cory shared depending on the curriculum being used to introduce CS to high school students, sometimes the problems that students work on are not open ended. He also shared that he works on making the challenges his students work on as open-ended as possible. Likewise, John agreed and shared that the open-ended nature helped him explain the process of solving the problem to students using a backward design process. He stated, "I set it up in like a three step process, using backwards design a lot so I would take what objectives I want like what I want the students to learn and my goals for teaching and hopefully as I instruct, the student will be able to make the connection." Alex on the other hand, posited that it was imperative for students to learn how to write good code in order to successfully solve CS-related challenges. He mentioned that good code was clean, easy to follow, and would be easy to troubleshoot. Teacher Brown suggested that building an opened nature perspective into CS problems developed creativity in students, as such a key tenet of constructionist learning theories where students constructed mental representations of possible solutions using the engineering design process to understand the how possible solutions to a given challenge might look like. Constructionism advocates student-centered, discovery learning whereby students use information they already know to acquire more knowledge (Alesandrini & Larson, 2002). Students learn through participation in project-based learning where they make connections between different ideas and areas of knowledge facilitated by the teacher through coaching rather than using lectures or step-by-step guidance. Cory noted that toolkits (e.g., TKinster) provided through some of the programming languages made realization of a solution to a problem become a living object through a visual medium.

Core Theme Assessment

Participants shared that assessment was seen to be a challenging aspect of infusing CS into engineering and technology education. Alex mentioned having criteria was key and aligning these to the requirements shared with students beforehand. Cory, on the other hand, shared that he used rubrics. The PLTW curricula he utilized provided rubrics that he could use in evaluating students assessments, although he also considered other informal measures and asked that students

document all their work as they solved each presented challenge. For example, he mentioned that he considered the functionality of the final project and if students were able to adhere to the criteria shared. Cory also stated that, "I really really want them to learn. I keep on watching for the kids who are putting an effort and try to learn, the projects are cumulative in a sense and how can I give a failing grade if a kid shows me growth."

John shared that "grading" as such was a challenge, in the same vein Alex posited that he did not have paper and pencil tests in his CS classroom, rather projects. Like Cory, Brown also utilized PLTW rubrics to assess his students' completed assignments. For example, the PLTW rubric for App design, Scratch game or Story assignments had the same eight criteria elements and had a grading scale that ranged from 4 to 1, with 4 being the highest score and 1 being the lowest. The criteria elements include: solves problem, documentation, collaboration, presentation, appropriate algorithm, explanation of algorithm, explanation of problem solution, and planning. For example, under the criterion 'solves problem' to score a 4, a student's "artifact fully addresses personal, practical, or societal intent posed by problem statement," a score of 3 depicted that "artifact addresses the personal, practical, or societal intent posed by problem statement," a score of 2 meant that "artifact mostly addresses the personal, practical, or societal intent posed by problem statement", and a score of 1 meant "artifact does not adequately address the personal, practical, or societal intent posed by problem statement." As such, the PLTW rubric elements were similar to engineering design rubrics (e.g., Asunda & Hill 2007; Groves, Abts, & Goldberg, 2014; Robelen, 2013; Spurlin, Rajala, & Lavelle, 2008) that have been used to assess engineering design challenges with the exception of the criteria "appropriate algorithm" and "explanation of algorithm." These two criteria required that students show that the "code demonstrates use of appropriate algorithms" as well as provide "comments that clearly and thoroughly explain the algorithm(s)." The engineering design process asks for students to show the iteration progression they utilize to reach a viable solution and provide evidence of optimization of chosen solution, as such a similar process to the two criteria "appropriate algorithm" and "explanation of algorithm."

DISCUSSION

Computer scientists, just like engineers, play a central role in our technological infrastructure. They develop hardware, software and other applications for use by the military, businesses, and average consumers (Singh, 2016). The findings of this study revealed problem solving as a key element in infusing CSP and CT into STEM-related coursework at the K-12 level. The use of problem solving as a strategy to develop and impart in students critical thinking skills in engineering and technology education programs has been reported by several authors (e.g., Eison, 2010; Pacific Policy Research Center, 2010; Ralston & Bays, 2015). Participants of this study posited that the procedure their students utilized to solve computer science design challenges was similar to the engineering design problem solving process utilized by engineers and technologists to solve everyday challenges that society faces. Today, students as young as six and seven are learning the logic behind computer programs and, in some cases, how to create simple programs of their own. Working with age-appropriate programming tools like Scratch, App inventor etc. and curricula, students can be innovative in their solving of given design challenges as they explore and experiment with crosscutting interdisciplinary skills and knowledge as detailed by NGSS (Bers, 2010; Bers & Horn, 2010; Grover & Pea, 2013; NGSS, 2013). Programming at the K-12 consists of two bodies of theoretical work: computational thinking, which discourses problem solving with computers; and technological literacy and fluency, which addresses expressivity with new technologies (Barr, Harrison, & Conery, 2011; Grover & Pea, 2013; Guzdial, 2008; Lee et al., 2011; Next Generation Science Standards [NGSS], 2013; Wing 2008). Utilizing problem solving strategies to innovatively design and program computational artifacts can facilitate students' engagement in high-level cognitive processes such as divergent thinking, and reflective practice (Resnick, 2007). As such, the findings of this study report that programming as a vehicle to develop computational thinking practices may lead to the realization of viable solutions to given design challenges.

With regards to core themes pedagogy and assessment, Magana, Brophy, and Bodner, (2012) investigated aspects of teaching and learning for integrating CS modeling and simulation practices in STEM coursework. Aspects of

teaching relate to the identification of intended learning outcomes instructors would like to accomplish when integrating computational tools into their disciplinary courses. On the other hand, learning aspects for integrating modeling and simulation practices have centered on the reasoning processes afforded by computational tools (Magana et al., 2017), as well as scaffolding strategies that can overcome possible cognitive overload (Vieira, Magana, Falk, & Garcia, 2017; Vieira, Magana, Roy, Falk, & Reese, 2016). Aspects of teaching and learning inform assessment practices, as well as pedagogical strategies that include hands-on, real-world projects. Such is a vehicle for the integration of CS practices and STEM coursework, as a result helping students develop useful skills and take what they learn in the classroom and apply it to everyday life. Thus, project- and problem-based challenges and learning through modeling and simulation practices engage students in rigorous and relevant learning experiences that may generate their enthusiasm as well as impart in them CSP and CT skills and knowledge. On the other hand, assessments that teachers utilize are processes used to examine students' assignments with the aspects of teaching and learning that the teacher has identified as appropriate for a given learning segment. In essence, assessment practices that these teachers utilized helped them gauge the development of CSP and CT skills by using design challenges as a vehicle to support the learning of crosscutting concepts in i-STEM environments.

CONCLUSION

In conclusion, the findings of this study suggest that i-STEM opens a range of possibilities by which teachers at the K-12 level may utilize to further develop in 21st century students skills that become future workforce requirements to be competitive at the workplace. Infusion of CSP and CT into STEM-related course work engages students in applied learning as they solve design challenges through a variety of crosscutting concepts. Such a process is similar to the engineering design process that exemplifies a process of steps that are developmental, structured, and iterative in solving design problems, building prototypes, and testing solutions. As such students are exposed to critical skills in problem solving, teamwork, time management, communication, and leadership strategies. Such an approach may ensure college and career readiness for the STEM-enabled 21st century careers.

LIMITATIONS AND RECOMMENDATIONS FOR FUTURE RESEARCH

As with all educational research, there are limitations to this study that must be addressed. The infusion of CS into STEM-related courses through engineering and technology education is an emerging area of work. As such, there is limited literature with regards to how teachers integrated their teaching as well as assessment process. NGSS standards are a recent introduction to the K-12 arena, and teachers are still learning how to incorporate them into already existing state standards for instructional planning purposes. Besides, states are still grappling with where CS fits in the K-12 curricula, and there is a need for qualified teachers. First, a small number of participants within a radius of 100 miles were purposefully chosen for this study, limiting the ability to utilize sophisticated statistical methodologies and examine how engineering and technology education teachers infuse CSP and CT into engineering and technology education. Second, assessment strategies were varied and it was difficult to comprehend and relate how participants of this study may have integrated NGSS suggestions into assessing students' CS integrated STEM assignments and projects. It was noted that participants' utilized rubrics and that choice may not clearly highlight the role of performance assessment as highlighted in the NGSS. Although there is much that remains to be done toward integration of CS into K-12 teaching, caution must be used in generalizing findings of this study to larger populations. Future research studies would benefit from the use of a larger sample. Third, investigate assessment practices that clearly articulate and align CS to performance assessment, and lastly computer science departments and STEM educators should continue to collaborate and develop CS coursework that can be integrated into teacher education course work concepts and pedagogical knowledge practices.

Paul A. Asunda *is an Assistant Professor of Engineering and Technology Teacher Education in the Department of Technology, Leadership, and Innovation at Purdue University, West Lafayette, IN. He is a member of the Gamma Rho Chapter of Epsilon Pi Tau.*

REFERENCES

Alesandrini, K., & Larson, L. (2002). *Teachers bridge to constructivism*. The Clearing House, 119-121.

Asunda, P. A., & Hill, B. R. (2007). Critical features of engineering design in technology education. *Journal of Industrial Technology Education. 44* (1), 25-48.

Barr, D., Harrison, J., & Conery, L. (2011) Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology, 38*(6), 20-23.

Berg, B. L. (2001). *Qualitative research methods for the social sciences* (4th ed.). Needham Heights, MA: Allyn & Bacon.

Bers, M. (2010) Beyond computer literacy: Supporting youth's positive development through technology. *New Directions for Youth Development, 128,* 13-23.

Bers, M. U., & Horn, M. S. (2010). Tangible programming in early childhood: revisiting developmental assumptions through new technologies. In I. R. Berson, & M. J. Berson (Eds.), *High-tech tots: Childhood in a digital world* (pp. 49–70). Greenwich, CT: Information Age Publishing.

Boeije, H. (2002). A purposeful approach to the constant comparative method in the analysis of qualitative interviews. *Quality & Quantity, 36,* 391–409. doi:10.1023/A:1020909529486.

Chiu, A., Price, A. C., Ovrahim, E. (2015, April). *Supporting elementary and middle school STEM education at the whole school level: A review of the literature.* Paper presented at the Annual International Conference of the National Association for Research in Science Teaching [NARST], Chicago, IL. Retrieved from http://www.msichicago.org/fileadmin/Education/pdf/MSI-SLI-Literature_Review_White_Paper.pdf

Corradi, A., & Leornadi, L. (2001). *Static vs. dynamic issues in object-oriented programming languages.* Retrieved from https://adtmag.com/Articles/2001/07/13/Static-vs-Dynamic-Issues-in-ObjectOriented-Programming-Languages.aspx?p=1

Crotty, M. (1998). *The foundations of social research: Meaning and perspective in the research process.* Thousand Oaks, CA: Sage.

Curzon, P., Peckham, J., Taylor, H., Settle, A., & Roberts, E. (2009). Computational Thinking (CT): On Weaving It In. *SIGCSE Bull.,* 41(3), 201-202. doi:10.1145/1595496.1562941.

Eison, J. (2010). *Using active learning instructional strategies to create excitement and enhance learning.* Retrieved from http://www.cte.cornell.edu/documents/presentations/ActiveLearningCreatingExcitement in the Classroom-Handout.pdf

Gallagher, J., & Parker, J. (1995). *Secondary teaching analysis matrix (STAM).* East Lansing, MI: Michigan State University.

Gall, M. D., Gall, J. P., & Borg, W. R. (2003). *Educational research: An introduction* (7th ed.). New York: Allyn and Bacon.

Grover, S., & Pea, R. (2013). Computational thinking in K-12: A review of the state of the field. *Educational Researcher 42*(1), 38-43.

Groves, F. J., Abts, R. L., & Goldberg, L.G. (2014). *Using an engineering design process portfolio scoring rubric to structure online high school engineering education.* Conference presentation at 2014 ASEE. Retrieved from http://www.asee.org/public/conferences/32/papers/ 10738/view

Guzdial, M. (2008). Paving the way for computational thinking. *Communications of the ACM, 51*(8), 25–27.

Hecht, D., Russo, M., & Flugman, B. (2009). *Infusing mathematics into science, technology, and engineering classes: Lessons learned from middle school teachers and students.* Retrieved from http://www.hofstra.edu/pdf/Academics/Colleges/SOEAHS/ctl/mstp/mstp_STEM_Symposium.pdf.

Koch, M., & Gorges, T. (2016). Curricular influences on female afterschool facilitators' computer science interests and career choices. *Journal of Science Education & Technology,* 25(5), 782-794. doi:10.1007/s10956-016-9636-2

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. & Werner, L. (2011). Computational Thinking for Youth in Practice. *ACM Inroads,* 2(1), 32-37. doi:10.1145/1929887.1929902

Magana, A. J., Brophy, S. P., & Bodner, G. M. (2012). Instructors' intended learning outcomes for using computational simulations as learning tools. *Journal of Engineering Education, 101*(2), 220-243.

Mays, N., & Pope, C. (1995). Rigor and qualitative research. *British Medical Journal, 311*(6997), 109-112.

Miles, M. B., & Huberman, A. M. (1994). *Qualitative data analysis: A sourcebook of new methods.* Newbury Park, CA: Sage.

Mobley, C. M. (2015). *Development of the SETIS instrument to measure teachers' self-efficacy to teach science in an integrated STEM framework* (Unpublished doctoral dissertation). University of Tennessee, Knoxville.

Mohaghegh, M., & McCauley, M. (2016). Computational thinking: The skill set of the 21st century. *International Journal of Computer Science and Information Technologies* (IJCSIT), 7(3), 1524-1530.

Next Generation Science Standards [NGSS]. (2013). *The next generation science standards.* Retrieved from: http://www.nextgenscience.org/next-generation-science-standards

National Science and Technology Council. (2013). *A report by committee on STEM education national science and technology council. Federal science, technology, engineering, and mathematics (STEM) education 5-Year strategic Plan.* Retrieved from https://www.whitehouse.gov/sites/default/files/microsites/ostp/stem_stratplan_2013.pdf

Pacific Policy Research Center. (2010). *21st century skills for students and teachers.* Honolulu: Kamehameha Schools, Research & Evaluation Division.

Patton, M. Q. (2002). *Qualitative research & evaluation methods* (3rd ed.). Thousand Oaks, CA:

Office of Science and Technology Policy [OSTP] (2014). *Progress report on coordinating federal science, technology, engineering, and mathematics (STEM) education.* Retrieved from https://www.whitehouse.gov/sites/default/files/microsites/ostp/stem_ed_budget_supplement_fy16-march-2015.pdf

Ralston, P., & Bays, C. (2015). Critical thinking development in undergraduate engineering students from freshman through senior year: A 3-cohort longitudinal study. *American Journal of Engineering Education, 6*(2), 85-98.

Resnick, M. (2007). *All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten.* ACM Creativity & Cognition Conference, Washington DC, June 2007. Retrieved from http://web.media.mit.edu/~mres/papers.html.

Robelen, E. (2013). *AP engineering may be on the horizon. Education Week,* March 29, 2013. Retrieved from http://blogs.edweek.org/edweek/curriculum/2013/03 /ap_engineering_may_be_on_the_horizon.html.

Sanders, M. (2009). STEM, STEM education, STEM mania. *Technology Teacher, 68*(4), 20-26.

Siew, N. M., Nazir, A., & Chong, C. L., (2015). *The perceptions of pre-service and in-service teachers regarding a project-based STEM approach to teaching science.* doi: 10.1186/2193-1801-4-8. eCollection 2015.

Singh, H. (2016). *What is the best area of study to get into with computer coding focus,* Retrieved from https://www.careervillage.org/questions/43776/

Spurlin, E. J. Rajala, A. S., Lavelle, P. J., Eds. (2008). *Designing better engineering education through assessment: A practical resource for faculty and department chairs on using assessment and ABET criteria to improve student learning.* Stylus Publishing, Sterling, Virginia

Synder L., Astrachabm, O., Briggs, A., & Cuny, J. (2011). *AP computer science principles: Six computational thinking practices - AP Computer Science Principles*, Retrieved from https://csprinciples.cs.washington.edu/sixpractices.html

Vieira, C., Magana, A. J., Falk, M. L., & Garcia, R. E. (2017). Writing in-code comments to self-explain in computational science and engineering education. *ACM Transactions on Computing Education (TOCE).*

Vieira, C., Magana, A. J., Roy, A., Falk, L. M., & Reese, J. M. (2016). Exploring undergraduate students' computational literacy in the context of problem solving. *Computers in Education Journal., 7*(1), 100-112.

White House, Fact Sheet (2014). *New commitments to support computer science education.* Retrieved from http://www.whitehouse.gov/the-press-office/2014/12/08/fact-sheet-new-commit-ments-support-computer-science-education

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of The Royal Society A*, 366, 3717-3725. doi:10.1098/rsta.2008.0118

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 Classrooms. *TechTrends, 60*(6), 565–568. doi:10.1007/s11528-016-0087