TechTrends (2020) 64:17–28
https://doi.org/10.1007/s11528-019-00423-0

ORIGINAL PAPER

# Hopscotch into Coding: Introducing Pre-Service Teachers Computational Thinking

Shenghua Zha[1] · Yi Jin[2] · Pamela Moore[1] · Joe Gaston[1]

Published online: 8 August 2019

**Abstract**
Researchers and educators have advocated computational thinking (CT) should be integrated into K-12 settings as early as elementary schools. However, there has been a lack of knowledge of how pre-service K-8 teachers would be engaged in the learning of CT and its integration in different subject areas. In this study, we taught a flipped learning module in an undergraduate Educational Technology course. Pre-service teachers learned and practiced CT knowledge and skills using a block programming app called Hopscotch. Results of this first iteration of design-based research showed that the orchestration of the technology and instructional methods, such as team-based learning, flipped classroom, and pair programming, supported students' transformative learning experience. It improved their understanding and application of CT concepts. Meanwhile, the mixed-method analysis found some instructional issues that needed to be addressed in future iterations. Suggestions were provided at the end of the paper.

**Keywords** Elementary teacher education · Computational thinking · Coding · Collaborative learning

## Introduction

Computer Science (CS) education in K-12 settings has been proliferating in the past 30 years. In 1980, Seymour Papert took the initiative in examining children's learning of LOGO programming (Papert 1980). This interest was resumed in 2006 when Wing (2006) coined the term Computational Thinking (CT). It referred to a set of skills that computer scientists frequently used in solving computing problems, such as problem formulations, abstraction, algorithmic thinking, and pattern recognition and generalization. Researchers and educators argued that these skills were applicable in every subject and hence every student should learn and master (Buitrago Flórez et al. 2017; Grover and Pea 2013; Wing 2006).

To meet the increasing needs of CS/CT education at K-12 settings, professional development activities for in-service teachers have been growing (Guzdial 2011; Menekse 2015). They were offered as independent CS courses or short-term and long-term training sessions. However, there has been a lack of knowledge of how pre-service teachers would engage in CS/CT learning and teaching (Yadav et al. 2017b). In a systematic review, Menekse (2015) found 21 studies of in-service teachers' CS training from journal and conference papers published between 2004 and 2014. However, only seven papers were found to be related to pre-service teachers.

In fact, it is probably more challenging to teach CS/CT to pre-service teachers than in-service teachers. At present, in-service teacher training was voluntary-based, and only teachers who had interests would participate in the training (Bean et al. 2015; Peterson and Scharber 2017; Rich et al. 2018). However, pre-service teacher training is set for all students, regardless of their initial interest or prior CS experiences. Therefore, the course instructors may expect a broad spectrum of interest and computing experience in a CS/CT teacher education course (Amiri 2000). Although challenging, with more states in the U.S. starting to integrate coding and

✉ Shenghua Zha
shzha@southalabama.edu

Yi Jin
yjin8@kennesaw.edu

Pamela Moore
prmoore@southalabama.edu

Joe Gaston
jgaston@southalabama.edu

[1] University of South Alabama, University Commons 3800, 75 N. University Blvd, Mobile, AL 36688-0002, USA

[2] Kennesaw State University, 585 Cobb Ave NW, Room 2334, MD 0127, Kennesaw, GA 30144, USA

CT standards into K-12 curriculum, teacher educators need to provide coding and CT preparation to pre-service teachers.

In this study, we implemented a flipped learning module teaching pre-service teachers (students, in brief) CT concepts and skills. The first purpose of this study was to examine the impact of the learning module on students' CT knowledge and attitude. The second purpose was to identify areas for improvements in the future iterations. The following section depicted a summary of the literature review regarding pre-service teachers' CS/CT education. The rest of the sections follow the DBR cycle. The planning and acting stages of DBR were described in the section of Learning Module Design. The Method section focused on the observation and analysis of students' learning in the flipped module. The Results and Discussions sections reflected this first iteration and offered suggestions for improvements in future iterations.

## Literature Review

### CS/CT Teacher Education in the U.S

CS has been offered as independent courses in teacher education programs in many countries, such as South Africa and Turkey (Balanskat and Engelhardt 2015; Cetin 2016; Mentz et al. 2008). However, in the United States, CS education in K-12 settings was different from other countries. At present, 15 states gave high-school students access to CS education, but only six states expanded it to lower grades (Promote Computer Science 2019). In addition, CS was not offered as an independent discipline in K-8 settings (Mouza et al. 2017). It was integrated into other subject courses, mostly the subjects of Science, Technology, Engineering, and Mathematics (STEM) (Lye and Koh 2014).

The lack of CS as an independent discipline in K-8 affected CS course offering in Teacher Education programs in the U.S. According to a recent report from code.org, 33 states plus Washington, D. C. had CS teacher certification programs (2018 state of Computer Science education: Policy and implementation 2019). Most of these programs targeted pre-service secondary education teachers. At present, there has not been an established framework or model for pre-service K-8 teachers even though researchers and educators advocated that CS/CT should be introduced early at elementary schools to leverage students' development of CT skill before college (Buitrago Flórez et al. 2017; Qualls and Sherrell 2010). One rule of thumb that has been well accepted is that CS education for pre-service K-8 teachers should be integrated with pedagogical content knowledge (Yadav et al. 2014). For instance, pre-service teachers may learn to develop collaborative learning instructions to teach students to draw different geometric shapes in 7th-grade math class while working in pairs using Scratch. In doing this, pre-service teachers not only learn CT and coding but also solve geometry problems. Thus, CS/CT is not a standalone subject. It becomes a set of skills/tool to solve problems in other subjects. Moreover, pre-service teachers' experience of appropriate pedagogy, such as pair work, as learners may set a model for their future teaching.

### Efforts to Introduce K-8 Pre-Service Teachers CS/CT

Due to the unique situations of CS/CT education in K-8 settings, researchers have been exploring innovative ways to introduce CS/CT to pre-service K-8 teachers. One practice is to embed introductory CS/CT learning modules in a teacher education course that every pre-service teacher takes. Yadav and his colleagues implemented a one-week CS/CT module in an Educational Psychology course (Yadav et al. 2011). Two lectures introduced basic CT concepts and its importance in K-12 education. Examples of CT application in science and humanities were given at the end of the lecture. During the lectures, students used an interactive response system called clicker to answer questions. Their results showed that this module helped students to develop a comprehensive understanding of CT and a positive attitude toward its integration in K-12 settings. Furthermore, the lectures emphasized CT concepts instead of computing technology. This approach helped the pre-service teachers to develop an understanding that CT was applicable in subject areas other than CS.

Bean and his colleagues conducted a 2-h intervention among a group of pre-service teachers (Bean et al. 2015). Their study focused on a different perspective of CS/CT: the integration of coding in three subject areas. The intervention began with an overview and demonstration of three coding exercises. Those exercises were tied to the areas of music, language arts, and math. Instructors addressed questions posed by students during the exercises. In the end, the class reflected and discussed their learning experiences and the applicability in K-12 classrooms. Results of the pre- and post-survey showed that pre-service teachers' CT and teaching self-efficacy had significant improvement as a result of the intervention. It may be that the three applications helped students to see the possibility of CS integration in subjects other than CS. Meanwhile, technical difficulties were observed when the class was requested to switch from laptops to iPads. They had to look for a new programming language as Scratch ran on Adobe Flash and was not natively supported by iOS.

Educational Technology courses in Teacher Education programs have served as a primary gateway for pre-service teachers to learn cutting-edge technologies and their pedagogical application in different subject areas (Wetzel et al. 2008). Often, it is the only technology course offered in most teacher preparation programs in the U.S. Hence, this course carries many expectations that other Teacher Education courses do not offer, such as improving pre-service teachers' technology beliefs and self-efficacy (Funkhouser and Mouza 2013). This course could be an ideal outlet to introduce CS/CT to pre-service teachers (Yadav et al. 2017a).

In an educational technology course, Chang and Peterson (2018) assigned two groups of pre-service teachers a CS/CT learning module, which started with a thirty-minute lecture on CT concepts. Then students were given one hour to explore robots and other CT resources. After the class, students had individual self-reflection on their learning experiences. Results of this case study showed an overall positive impact of the short-term exposure on pre-service teachers' CS/CT understanding. However, some students still held a stereotype that they were not trained as a CS major, and thus, CT should not be their field of teaching. This belief possibly prevented them from developing a positive learning attitude toward CS/CT.

In a nutshell, exploratory studies embedding CS/CT modules in educational methods or technology courses showed that it was promising to change students' CS/CT understanding and sometimes self-efficacy. These modules seemed to confirm the positive impact of the following design: First, CT concepts should be introduced to students before programming tools (Chang and Peterson 2018; Yadav et al. 2011). This method helped to foster students' understanding of CT application as problem-solving skills. Second, exercises should be designed to explicitly convey the possibility of CS/CT integration outside CS or even STEM areas (Bean et al. 2015). Reflective discussion should be offered to strengthen this perspective. Last, but not the least, visual-based block programming language or tools such as Scratch, should be used in the introduction module (Cetin 2016). These tools helped students to understand the CT application better than text programming language such as C, where students usually spent a significant amount of time debugging the syntax errors.

However, these studies also posed questions for further investigations: first, how much guidance and scaffolding should be provided in the module exercise? Bean et al. (2015) offered full guidance to teach students the coding in three exercises. In Chang and Peterson's (2018) study, students were given the freedom to explore a variety of computing tools. No conclusive suggestions have been made. Second, how much time did students need to develop a good understanding of CT concepts and foster a positive self-efficacy? This question was also tied with the re-design of Educational Technology courses as they carried many expectations but little room for the inclusion of CS/CT module.

## Research Questions

In the fall of 2018, we embedded a CS/CT flipped learning module in an educational technology course at a four-year university in the southern U.S. When designing the module, we followed the suggestions from the literature: 1. CT concepts were introduced first using online readings, videos, tutorials, and other materials; 2. Exercises were designed for the literacy content area; 3. Discussion forums were designed for students to reflect; and 4. A visual block programming app, Hopscotch, was the primary coding tool for the exercise. One purpose of this first iteration of design-based research was to identify areas for future improvement. The other purpose was to seek the impact of this flipped learning module on students' CS/CT knowledge and attitude that will help us understand how much guidance and time students need to develop a good understanding of CT concepts and foster a positive self-efficacy. Therefore, our research question was: How did the flipped learning module contribute to students' learning experience, knowledge of, and attitude toward CS/CT in K-8 education?

## Class Structure

This study took place in an undergraduate educational technology course. It was the only technology course that pre-service teachers took during their four years of study. Students learn technology-integrated learning theories and technologies in 15 weeks. It was offered in a blended mode. When online modules were not offered, students were expected to meet twice per week in class, and each class session lasted about 75 min. When online modules were offered, students were required to participate in asynchronous and/or synchronous activities posted on the learning management system. Each online module was usually available for about one week, and students could adjust their own learning pace in asynchronous activities.

The course instructor, also one of the researchers in this study, has had over ten years of experiences in instructional design, blended and online course teaching, and professional development. She had programming skills and knowledge in visual-based block programming languages, such as Scratch and Hopscotch, as well as text programming languages, such as HTML, JavaScript, and Visual Basic.

## Participants

Participating students were fifteen pre-service teachers, majoring in elementary education (87%, $n = 13$) or secondary education (13%, $n = 2$). Ninety-three percent ($n = 14$) were female, and 7 % ($n = 1$) were male. Twenty-seven percent of students were African American ($n = 4$), 7 % were Native Indian ($n = 1$) and Asian ($n = 1$) respectively, and 60 % were Caucasian ($n = 9$). Fourteen of them had no computer programming experiences before. Only one student mentioned that she had a little coding experience with LEGO robots when she was in middle school.

## Learning Module Design

The programming language used in this module was Hopscotch, a visual-based block programming language. It was chosen for a couple of reasons. First, its embedded video tutorials allowed learners to practice coding along with the tutorials without switching the screens. Second, the tutorials explicitly focused on the basic coding concepts, such as sequence, conditional logic, and algorithms. This feature saved the instructor's time either to look for relevant video tutorials or to introduce beginners the application of CT concepts in a programming language in class. Last, but not the least, students were able to complete an animation project following each tutorial, which made the learning engaging and fun. However, this program had its weaknesses. First, it was free, but users need to pay if using extra features, such as importing images or sounds. Those features were not essential learning component in this module. Second, it was only available on the iOS platform. Students in this study were able to check out iPads from the college. Therefore, the instructor decided to choose this program after weighing its strengths and weaknesses.

The learning module was offered in a flipped learning mode to ensure students have enough time to explore CT readings and tutorials in Hopscotch before applying their knowledge in coding projects in the in-class session (Abeysekera and Dawson 2014; Enfield 2013). The online component of the module (online module, in brief) was

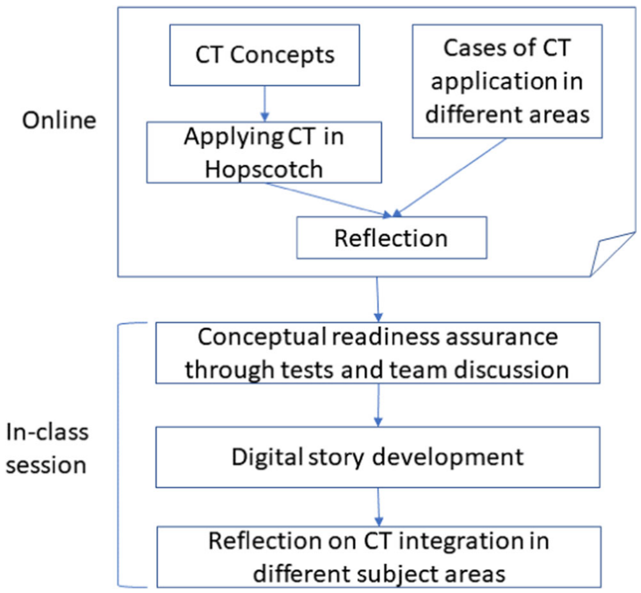

Online
CT Concepts
Cases of CT application in different areas
Applying CT in Hopscotch
Reflection
In-class session
Conceptual readiness assurance through tests and team discussion
Digital story development
Reflection on CT integration in different subject areas

**Fig. 1** Design of the flipped coding module

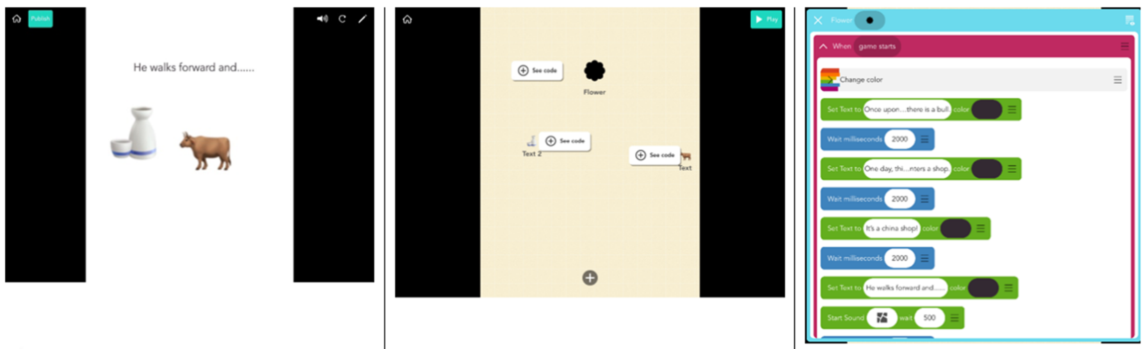
available to students in a week prior to the in-class session. As shown in Fig. 1, students were instructed to read the articles that introduced CT concepts and cases of its integration in K-8 subject areas. Then they were guided to set up accounts in Hopscotch and practice with the embedded video tutorials. Students were requested to participate in two online discussion forums. One discussion forum was for Hopscotch Q&A, and the other one was used for students' reflections of their learning experience with Hopscotch. In the second forum, students were required to answer the question: What was your experience at the beginning, middle, and end of this learning journey?

To ensure students' readiness for the in-class session, Team-Based Learning was adapted in the learning module, and students were informed of this at the beginning of the online module (Michaelsen et al. 2002). In the following Monday after the online module, students met in class for 75 min. First, they took an individual quiz testing their understanding of CT and Hopscotch programming. Thereafter, they discussed the same set of questions and submitted the quiz in teams. The purpose of this team quiz and discussion was to clarify students' conceptual understanding of CT and Hopscotch coding. Students had been working in the same teams in multiple course projects before. So they were quite familiar with each other. After their quiz submission, the instructor explained the answers to a question if a team did it wrong. Their participation in the online forums and in-class quizzes were counted into the course grades.

Thereafter, the core of the class time was spent on the application activity. Students were instructed to develop a digital story in Hopscotch explaining an idiom. The reason why the application activity was focused on the Language Arts was to break students' stereotype that coding belonged to and benefit STEM only (Chang and Peterson 2018). The instructor demonstrated a completed story "a bull in a China shop" and explained how the algorithm was developed (Fig. 2). This demonstration was made via a free remote-connection app called TeamViewer. It projected the iPad screen to the computer and projector screens so that students saw clearly how the coding was developed in Hopscotch. Then students were assigned into pairs to design and develop a story (McDowell et al. 2006; Williams and Kessler 2001). They had the option of developing the same or a different idiom story. The instructor rotated among teams and offered suggestions and prompts to help students design, code, and debug in Hopscotch.

## Methods

Design-Based Research (DBR) was conducted (Barab and Squire 2004; Reimann 2013). Different from traditional empirical research, the purpose of this DBR was to establish and refine the design of the flipped learning module for future



```
Flower
When game starts
Change color
Set Text to Once upon....there is a bull. color
Wait milliseconds 2000
Set Text to One day, thi....nters a shop. color
Wait milliseconds 2000
Set Text to It's a china shop! color
Wait milliseconds 2000
Set Text to He walks forward and...... color
Start Sound wait 500
```

1 Program output screen 2 Main code editor 3 Object code editor

**Fig. 2** Hopscotch interface

iterations. The course instructor participated in not only module design but also research. A mixed-method analysis was taken to study the delivery and outcome of this module.

## Data Collection

A pre-module online survey was delivered at the beginning of the module. Students were given two days to complete and submit the survey. Questions in this survey included students' demographic information and their prior computer programming experience. A post-module survey was handed to students at the end of the in-class session. Students had about 10 min to complete the paper survey. Both pre- and post-module surveys asked students' attitude toward CT, which was adopted from instruments validated in prior studies (Friday Institute for Educational Innovation 2012; Rich et al. 2017). Answers to the seventeen attitude questions were coded in 4 points, with 1 meaning "Strongly Disagree" and 4 meaning "Strongly Agree". The post-module survey also asked students' learning experiences in the flipped Team-Based Learning. This set of questions was adapted from instruments validated in Mennenga's (2012) study. Answers to these learning experience questions were coded in 5 points, with 1 meaning "Strongly Disagree," 3 meaning "Neither Disagree Nor Agree," and 5 meaning "Strongly Agree".

A pair of quizzes was deployed to assess students' knowledge of CT and Hopscotch coding. Five multiple-choice questions were included in each quiz. The first quiz was delivered at the beginning of the in-class session. The same quiz was used in the following team discussion as a part of Team-Based Learning. The second quiz was delivered at the end of the in-class session.

Qualitative data included the instructor's notes and students' answers to the open-ended question in the post-module survey and online discussion forums. The course instructor took notes to record her observation of students' learning behavior. In addition, the last question in the post-module survey asked students' comments about this learning experience.

## Data Analysis

A repeated measures ANOVA was conducted to identify changes in students' understanding of CT and Hopscotch coding in the first, team, and second quizzes. A set of *t*-tests was used to compare the changes in students' attitude toward CT in the pre- and post-module survey. In this test, a Bonferroni correction was used to minimize the Type I error (Miller 1981). The corrected alpha value was set at .00294, which was the value of .05 divided by 17.

Students did not ask any questions on the Hopscotch Q&A forum. Therefore, it was not included in the data analysis. Students' responses in the reflective discussion forum were imported into QSR NVivo 11 Plus, a qualitative analysis software. Posts to the question "What was your experience at the beginning, middle, and end of this learning journey?" were coded to distinguish their reflections at the beginning, middle, and end stage of the online module. Term-frequency analysis was used to find patterns of students' perception (O'Neill et al. 2018). First, a word frequency query was run to detect the top words that were frequently presented in students' posts. Second, words that did not contribute to understanding students' perception, such as "some" or "this," were filtered out. At the end of the query, the top 20 words that had the highest frequency in students' posts were presented.

Instructor's observation was used firstly as a formative evaluation to adjust the learning pace and content and secondly as a summative evaluation to reflect the strengths and weaknesses of this module. These notes were used to supplement the quantitative and qualitative findings from students' performance and perceptions.

## Results

Statistics results from quizzes showed a significant change between the first, team, and second quizzes; Wilk's $\lambda = .17$, $F(2, 12) = 28.8$, $p < .01$ (Fig. 3). The post-hoc comparisons showed a significant difference in the scores for the first

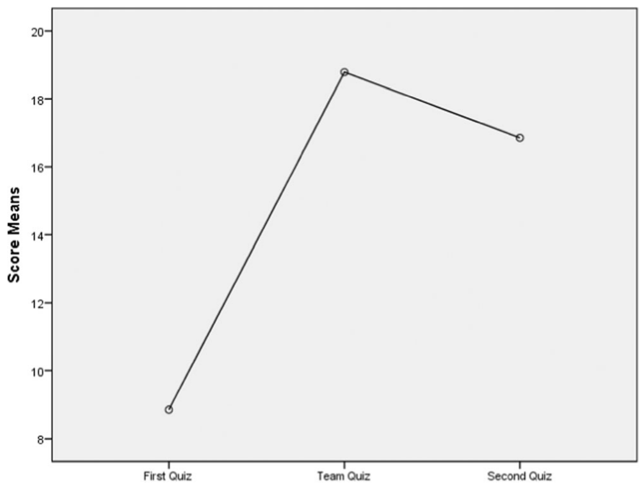


**Fig. 3** Students' scores in the first, team, and second quizzes

($M = 8.8$, $SD = 5.1$) and second ($M = 16.8$, $SD = 3.8$) quizzes; $t(14) = -5.3$, $p < .01$. Likewise, there was a significant difference in the scores for the first ($M = 8.8$, $SD = 5.1$) and team ($M = 18.79$, $SD = 2.08$) quizzes; $t(13) = -7.88$, $p < .01$. However, no statistical significance was found between the team and second quizzes; $t(13) = 1.60$, $p = .13$.

Items measuring students' attitude towards CT did not show significant changes except one. The item "I doubt that I can solve problems by using computer applications." showed a borderline significant change (Table 1). This item was reversely coded. An increase in the mean value meant that students showed a stronger disagreement with the item description. Hence, students showed less doubt in using computer applications for problem-solving after this flipped module.

Students' reflection in the discussion forum showed that they had a transformative learning experience. We listed the twenty most-frequent words that students used to describe their experience at the beginning, middle, and end of the learning module (Table 2). We then presented this result in three word-cloud images. In those word clouds, the stronger and larger a word was, the higher the frequency it showed up in students' posts. As shown in the left word-cloud of Fig. 4, students' initial impression when assigned the module was confused and worried mainly because they did not know what coding or programming looked like and how it could be applicable in education. One student acknowledged that she even did not want to do the project. Yet, two students expressed their interest and curiosity at the beginning. After students were instructed to install and open Hopscotch and learn with the embedded tutorials, their attitudes changed, as shown in the middle word-cloud of Fig. 4. Students commented that the program "really broke everything down in the tutorials and made everything very easy to understand". When they had no difficulty in following the tutorials and understanding what the coding did, they felt comfortable, and the confidence of coding increased. At the end of the online module, coding became an enjoyable and exciting learning experience, as shown in the right word-cloud of

**Table 1** Results of t-test and descriptive statistics of students' attitudes toward CT

| | Pre-Survey | | Post-Survey | | *t* | *df* | *p* |
|---|---|---|---|---|---|---|---|
| | *M* | *SD* | *M* | *SD* | | | |
| Knowledge of computing will allow me to secure a better job. | 3.33 | 1.05 | 3.47 | .64 | −.46 | 14 | .65 |
| My career goals do not require that I learn computing skills. | 2.93 | .59 | 2.93 | .70 | 0.00 | 14 | 1.00 |
| I doubt that I can solve problems by using computer applications. (R) | 2.73 | .80 | 3.53 | .52 | −3.60 | 14 | .00293 |
| I expect to use software in my future educational and teaching work. | 3.13 | 1.19 | 3.67 | .49 | −1.74 | 14 | .10 |
| I can achieve good grades (C or better) in computing projects or courses. | 3.33 | .82 | 3.47 | .52 | −.49 | 14 | .63 |
| The challenge of solving problems using computer science appeals to me. (R) | 2.80 | .86 | 2.87 | .92 | −.21 | 14 | .84 |
| I expect to use computer applications for future projects involving teamwork. | 3.13 | .99 | 3.60 | .51 | −1.61 | 14 | .13 |
| I can learn to understand computing concepts. | 3.20 | 1.01 | 3.60 | .51 | −1.47 | 14 | .16 |
| I am not comfortable with learning computing concepts. (R) | 3.00 | .76 | 3.07 | .70 | −.25 | 14 | .81 |
| I expect to use computing skills in my daily life. | 3.13 | .99 | 3.07 | .70 | .19 | 14 | .85 |
| I hope that my future career will require the use of computing concepts. | 3.13 | .74 | 2.80 | .78 | 1.32 | 14 | .21 |
| I think that computer science is interesting. | 3.00 | .93 | 3.00 | .66 | .00 | 14 | 1.00 |
| I will voluntarily teach kids coding if I were given the opportunity. | 3.20 | .86 | 2.93 | .70 | .94 | 14 | .36 |
| Computational thinking can be integrated into classroom education in other disciplines. | 3.20 | 1.01 | 3.47 | .52 | −1.00 | 14 | .33 |
| Computational thinking should be integrated into classroom education for other disciplines. | 3.13 | .99 | 3.33 | .62 | −.68 | 14 | .51 |
| Having background knowledge and understanding of computer science is valuable in and of itself. | 3.27 | 1.10 | 3.53 | .52 | −.81 | 14 | .43 |
| Knowledge of coding can be helpful to improve most careers. | 3.13 | .99 | 3.27 | .59 | −.41 | 14 | .69 |

Items ending with R meant they were reversely coded in the analysis

**Table 2** 20 most frequent words representing students' beginning, middle, and end of the learning experiences

| Frequency Rank | Beginning | Middle | End |
|---|---|---|---|
| 1 | Confused | Able | Excited |
| 2 | Interesting | Confident | Enjoyed |
| 3 | Long | Interesting | Future |
| 4 | Slow | Better | Learned |
| 5 | Wanted | Different | Liked |
| 6 | Excited | Adjusting | Fun |
| 7 | Hard | Attention | Able |
| 8 | New | Code | Confident |
| 9 | Unfamiliar | Comfortable | Knew |
| 10 | Worried | Confused | Loved |
| 11 | Adventure | Difficult | New |
| 12 | Clueless | Easy | Playing |
| 13 | Complicated | Enjoy | Engaged |
| 14 | Curious | Frustrating | Interesting |
| 15 | Different | Fun | Understood |
| 16 | Eager | Helpful | Wanted |
| 17 | Intimidating | Intrigued | Accomplished |
| 18 | Lost | Obstacle | Appealing |
| 19 | Fun | Practice | Beneficial |
| 20 | Thrilled | Prefer | Curious |

Fig. 4. A couple of students mentioned that they benefit a lot from pausing and rewinding the video tutorials, which cannot be accomplished in in-class lectures. Although not prompted, ten students mentioned that they were interested in learning more about computer programming and teach it in their future classes. Three students thought coding would help K-12 students to learn other subjects such as math.

Results of the post-module survey showed that students' perception of this flipped learning experience was very positive on average (Table 3). This finding was well supported by their qualitative answers in the survey. Six students provided qualitative comments in the post-module survey. Among them, four described this learning experience as fun and interesting experiences. One of them said that this was one of her favorite modules in the course. One student shared a mixed feeling, saying that it was hard but very interesting and fun. Only one student provided negative comments, "I actually found it confusing myself, so I can imagine that for young students."

The course instructor's observation supported that students did not have much difficulty in using Hopscotch. One online discussion forum was created for students' questions. Nonetheless, no one asked questions or showed any confusions of Hopscotch programming. In the in-class session, when students were asked if they had any confusions or questions regarding the first quiz after the team discussion, all students shook their heads. During the application practice, no students asked questions on where to find a button or what block to use under a specific condition, meaning that they knew how to code in Hopscotch. Instead, all pairs were actively engaged in discussions and coding. Often, one student coded in Hopscotch while the other peer observed and offered suggestions. When they could not proceed, both students worked on their iPads and explored different solutions until they found one and shared with the other.

However, the course instructor noticed that although students had about 45 min to create a new idiom story or recreate the demonstrated story, no groups finished it when the class was over. As shown in Fig. 5, the demonstrated story had three objects: the texts, the bull, and the china. It meant that students need to develop coding for each object. Students were prompted to draw a flowchart for each object to help the coding. Nevertheless, most student pairs drew one flowchart in the form of a storyline and then started coding immediately without further referencing the flowchart (Fig. 6). In doing this, groups were involved in the trial and error coding process. Although they were able to complete the coding of one object: the bull, they did not code the actions of the other two objects: texts and china. When prompted, students began the coding of the other two objects in Hopscotch without planning it in the flowchart or any other visual forms. Then it became another trial and error loop. As a result, the reflective discussion had to be cancelled due to the time shortage.

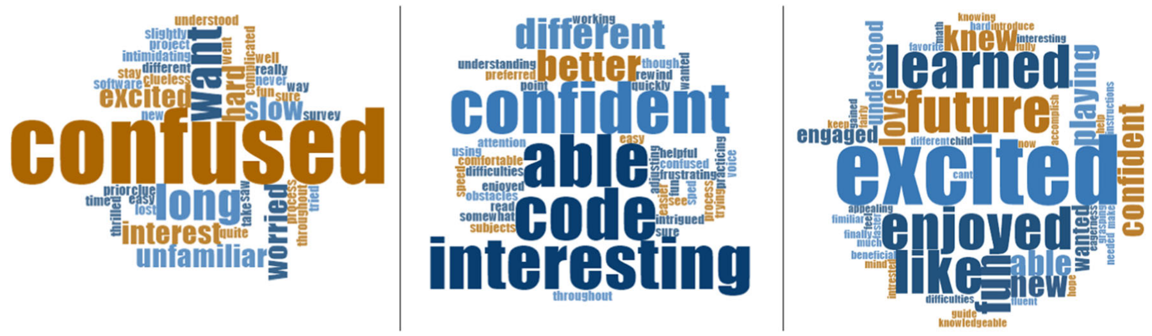


**Fig. 4** Students' attitude toward CT at the beginning (left), middle (middle), and end (right) of online learning

**Table 3** Descriptive statistics of students' perception of their flipped learning experience ($N$= 15)

| | Min | Max | Mean | SD |
|---|---|---|---|---|
| I enjoy the learning activities. | 4 | 5 | 4.73 | .46 |
| I learn better in a team setting. | 3 | 5 | 4.47 | .74 |
| I think the learning activities are an effective approach to learning. | 3 | 5 | 4.60 | .63 |
| The learning activities are fun. | 4 | 5 | 4.67 | .49 |
| I have a positive attitude towards this learning module. | 4 | 5 | 4.67 | .49 |
| I have had a good experience with this learning module. | 4 | 5 | 4.67 | .49 |

## Discussions and Implications

Results of this first iteration were inspiring. Pre-service teachers in this class were either females and/or African Americans, who have been identified as underrepresented groups in pursuing CS or other STEM careers (NCES 2018; Williams 2017). Their success in CS/CT teaching would set up role models that inspires underrepresented students in K-8 schools to study CS (Buschor et al. 2013). Most of the pre-service teachers in this class did not have any coding experiences except one. Thus, they expressed confusions, anxiety, and resistance, while a couple of them showed curiosity at the beginning stage. As they paced along with the step-by-step instructions in the online module, their resistance and anxiety were reduced. They became interested in learning and practicing the coding activities.

Miles et al. (2017) found that team quizzes scored higher than individual quizzes due to the collective intelligence involved in the team discussion of Team-based Learning. Hence, what reflected individual students' growth in our study was their differences in the first and third quizzes. Results of the knowledge quizzes showed a significant growth of students' CS/CT knowledge and skill before and at the end of the in-class session, which supported findings from other studies (Chang and Peterson 2018; Yadav et al. 2011). Students' overall perception of their flipped learning experience was positive. These findings suggested that different instructional methods were orchestrated successfully in this module. The flipped learning module enabled students to pace their initial conceptual learning in CT and coding. The online module provided explicit and step-by-step instructions to guide students to understand and practice the coding. The in-class session engaged students in collaborative learning activities. The instructor's observation showed that students were actively engaged in pair programming in the application activity. Team-Based Learning was an effective method to ensure students' readiness. The in-class readiness assurance activities, including tests and team discussion, offered a second

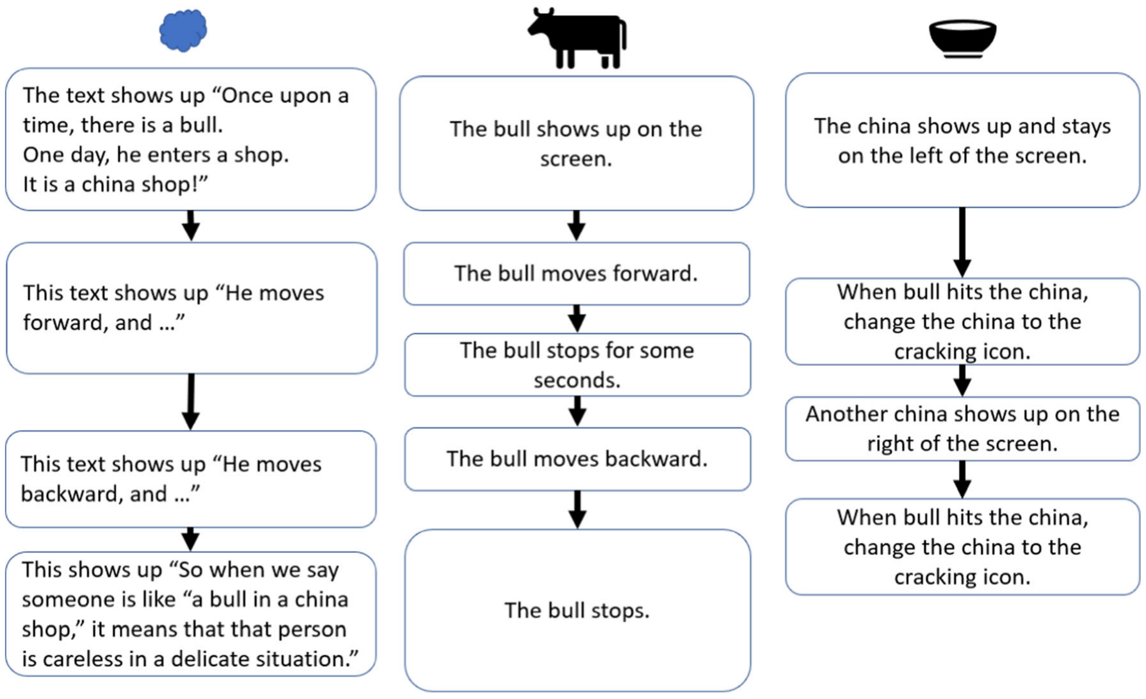

The text shows up "Once upon a time, there is a bull. One day, he enters a shop. It is a china shop!"
This text shows up "He moves forward, and …"
This text shows up "He moves backward, and …"
This shows up "So when we say someone is like "a bull in a china shop," it means that that person is careless in a delicate situation."
The bull shows up on the screen.
The bull moves forward.
The bull stops for some seconds.
The bull moves backward.
The bull stops.
The china shows up and stays on the left of the screen.
When bull hits the china, change the china to the cracking icon.
Another china shows up on the right of the screen.
When bull hits the china, change the china to the cracking icon.

**Fig. 5** A sample of the flowchart that helps students' coding

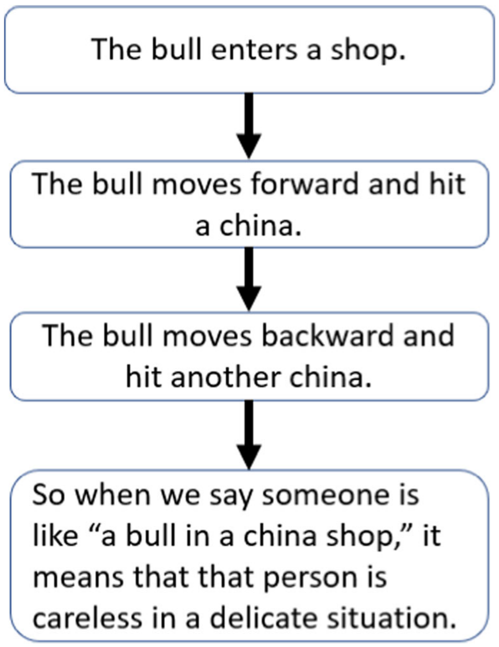

The bull enters a shop.
The bull moves forward and hit a china.
The bull moves backward and hit another china.
So when we say someone is like "a bull in a china shop," it means that that person is careless in a delicate situation.

**Fig. 6** A sample of students' initial flowchart

opportunity for students to strengthen their understanding of CT and Hopscotch coding. Thus, it prepared them for the upcoming learning activity at a higher cognitive level.

The embedded video tutorials in Hopscotch were helpful. They not only explained the basic CT concepts but also demonstrated their application in the development of digital stories or games. These tutorials were critical in taking students out of their comfort zone and working on a new subject area that they did not know or even resist to learn. However, a lot of K-12 schools had a Bring-Your-Own-Device policy. Therefore, in our next iteration, we may switch a program that is available on all major operating systems. Regardless of which program is chosen for the course, tutorials need to suffice the following requirements: 1) Each clip is short and focuses on one CT or coding concept; and 2) It is better to make each tutorial clip the production of a digital story or a game to engage learners.

A couple of changes for the next iteration were proposed based on researchers' observations and analyses results. First, a couple of coding practices will be added to the online module. These practices will be small scale and easy to finish. The purpose of these practices was to engage students in practicing their skills and fostering their positive attitude toward coding. Meanwhile, the practices will help students to be skillful in coding the in-class projects.

Second, students need a sustained scaffolding in their practice. Similar to Chang and Peterson's (1) study, students in our study felt positive towards the end of their online learning experience when given the freedom to explore Hopscotch in the online module. However, during the in-class application or problem-solving, students had difficulty in completing the demonstrated story on their own even after the instructor walked through the steps with the class. Therefore, we propose the following changes in future iterations: 1). A step-by-step scaffolding should be offered in the in-class practice. Although students learned CT concepts and played with Hopscotch in the online module, their knowledge and skills were segmented. The in-class practice was the first time when they pulled them together systematically. Thus, scaffolding would help them to build their first coding scheme. 2). Our observation showed that students went directly into coding without any planning. They were not used to utilizing the flowchart to design the coding. As a result, their coding process followed the pattern of trial and error, which was time-consuming. So in future iterations, students will be trained to design the coding from what they feel comfortable with: the storyboard. Then students will learn to convert storyboards to flowcharts. 3). Students' difficulty in the practice was probably due to the fact that the story involved more than one object, which was challenging for them to handle as their first hands-on project. It should start with a simple storytelling project, such as a story with only one object. When students successfully code a one-object project, the difficulty of the following projects may increase. In consideration of the time limit of an in-class session, the complicated project may occur at the beginning of the second session. Thereafter, students will join in a class discussion on the integration of CS/CT in the teaching of different subject areas. A take-home project will be the development of a CS/CT lesson plan. Therefore, two weeks is proposed for this module to be integrated into an educational technology course in future iterations (Fig. 7).

Our study demonstrated the potentials of integrating a CT module into an educational technology course as suggested by the literature (Chang and Peterson 2018; Yadav et al. 2011). This study also addressed another question emerged in the literature. The flipped module in our study took one week and an additional 75 min. Considering students' near-zero background in CS/CT, the time and format of flipped module seemed to be appropriate to improve their CT understanding as shown in prior studies (Chang and Peterson 2018; Yadav et al. 2011). However, this time length was not enough for students to work out a specific CT application in coding, needless to say, the development of lesson ideas. In addition, results of our study failed to provide strong and solid evidence that the short exposure improved students' attitudes or self-efficacy, which was inconsistent with findings from Bean et al.'s (2015) study.

We suggest that teacher educators consider integrating a CT module in the stand-alone education technology courses in their teacher education programs. This module should be both standards-based and content area specific. However, one module is not enough. Teacher educators should also collaborate to integrate standards-based content area specific CT projects into the methods courses so that pre-service teachers

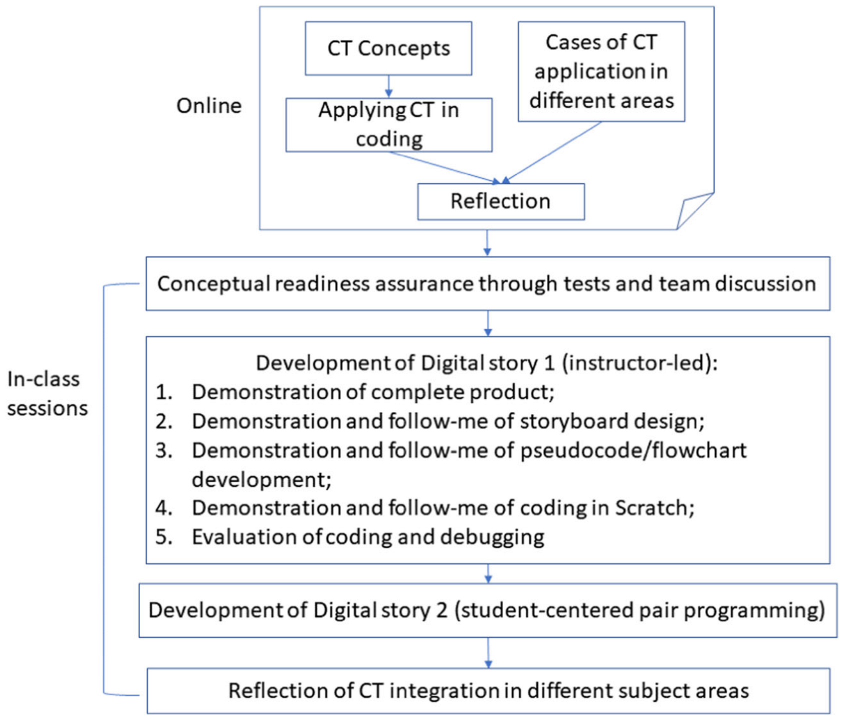

Online
CT Concepts
Applying CT in coding
Cases of CT application in different areas
Reflection
In-class sessions
Conceptual readiness assurance through tests and team discussion
Development of Digital story 1 (instructor-led):
1. Demonstration of complete product;
2. Demonstration and follow-me of storyboard design;
3. Demonstration and follow-me of pseudocode/flowchart development;
4. Demonstration and follow-me of coding in Scratch;
5. Evaluation of coding and debugging
Development of Digital story 2 (student-centered pair programming)
Reflection of CT integration in different subject areas

**Fig. 7** Proposed design for future iterations

will have more exposure and training on how to integrate CT into various content areas. A CT course especially designed for pre-service teachers that is both standards-based and content area specific is also a promising strategy. Future research should investigate the effectiveness of these approaches.

## Limitations

The sample size in this study was small, which restricted the generalizability of the results. A large sample is needed in future studies to verify pre-service teachers' growth of CS/CT conceptual understanding and skills as well as their attitudes.

Due to the time limit, we did not have an in-class reflection on CS/CT integration in subject areas. As a result, we were not able to compare students' reflections in the online module with those from the in-class session. Future studies shall include this comparison, which will help to detect the impact of the in-class session from students' qualitative perceptions.

## Conclusion

In this design-based research, we embedded a flipped learning module in an educational technology course to teach pre-service teachers CS/CT concepts and integration. Results of the mixed-method analysis showed that a CS/CT flipped module embedded in an educational technology course had the potential of improving pre-service teachers' CT understanding and probably some attitudes. In the in-class session, the readiness assurance activities improved students' understanding of CT and coding. Students enjoyed collaborative learning activities. However, the instructor's observation found students' learning challenge in completing the coding of the class project. We proposed a revised module framework to better scaffold students' CT learning for future iterations.

## Compliance with Ethical Standards

**Conflict of Interest** The authors declare that they have no conflict of interest.

**Ethical Approval** All procedures performed in studies involving human participants were in accordance with the ethical standards of the institutional and/or national research committee and with the 1964 Helsinki declaration and its later amendments or comparable ethical standards. This article does not contain any studies with animals performed by any of the authors.

## References

*2018 state of Computer Science education: Policy and implementation*. (2019). Code.org Advocacy Coalition & Computer Science Teachers Association. Retrieved from https://code.org/files/2018_state_of_cs.pdf. Accessed 10 Jan 2019.

Abeysekera, L., & Dawson, P. (2014). Motivation and cognitive load in the flipped classroom: Definition, rationale and a call for research. *Higher Education Research & Development, 34*(1), 1–14. https://doi.org/10.1080/07294360.2014.934336.

Amiri, F. (2000). IT-literacy for language teachers: Should it include computer programming? *System, 28*(1), 77–84.

Balanskat, A., & Engelhardt, K. (2015). *Computing our future: Computer programming and coding - priorities, school curricula and initiatives across Europe*. European Schoolnet: Brussels, Belgium.

Barab, S., & Squire, K. (2004). Design-based research: Putting a stake in the ground. *The Journal of the Learning Sciences, 13*(1), 1–14.

Bean, N., Weese, J., Feldhausen, R., & Bell, R. S. (2015). *Starting from scratch: Developing a pre-service teacher training program in computational thinking.* Paper presented at the 2015 IEEE Frontiers in Education Conference (FIE).

Buitrago Flórez, F., Casallas, R., Hernández, M., Reyes, A., Restrepo, S., & Danies, G. (2017). Changing a generation's way of thinking: Teaching computational thinking through programming. *Review of Educational Research, 87*(4), 834–860. https://doi.org/10.3102/0034654317710096.

Buschor, C. B., Berweger, S., Frei, A. K., & Kappler, C. (2013). Majoring in STEM-what accounts for women's career decision making? A mixed methods study. *The Journal of Educational Research, 107*(3), 167–176. https://doi.org/10.1080/00220671.2013.788989.

Cetin, I. (2016). Preservice teachers' introduction to computing: Exploring utilization of scratch. *Journal of Educational Computing Research, 54*(7), 997–1021. https://doi.org/10.1177/0735633116642774.

Chang, Y. H., & Peterson, L. (2018). Pre-service teachers' perceptions of computational thinking. *Journal of Technology and Teacher Education, 26*(3), 353–374.

Enfield, J. (2013). Looking at the impact of the flipped classroom model of instruction on undergraduate multimedia students at CSUN. *TechTrends, 57*(6), 14–27.

Friday Institute for Educational Innovation. (2012). *Teacher efficacy and attitudes toward STEM survey.* NC: Raleigh.

Funkhouser, B. J., & Mouza, C. (2013). Drawing on technology: An investigation of preservice teacher beliefs in the context of an introductory educational technology course. *Computers & Education, 62*, 271–285. https://doi.org/10.1016/j.compedu.2012.11.005.

Grover, S., & Pea, R. (2013). Computational thinking in K–12. *Educational Researcher, 42*(1), 38–43. https://doi.org/10.3102/0013189X12463051.

Guzdial, M. (2011). Learning how to prepare computer science high school teachers. *Computer, 44*(10), 95–97. https://doi.org/10.1109/MC.2011.316.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior, 41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012.

McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2006). Pair programming improves student retention, confidence, and program quality. *Communications of the ACM, 49*(8), 90–95. https://doi.org/10.1145/1145287.1145293.

Menekse, M. (2015). Computer science teacher professional development in the United States: A review of studies published between 2004 and 2014. *Computer Science Education, 25*(4), 325–350. https://doi.org/10.1080/08993408.2015.1111645.

Mennenga, H. A. (2012). Development and psychometric testing of the team-based learning student assessment instrument. *Nurse Educator, 37*(4), 168–172. https://doi.org/10.1097/NNE.0b013e31825a87cc.

Mentz, E., van der Walt, J. L., & Goosen, L. (2008). The effect of incorporating cooperative learning principles in pair programming for student teachers. *Computer Science Education, 18*(4), 247–260. https://doi.org/10.1080/08993400802461396.

Michaelsen, L. K., Knight, A. B., & Fink, L. D. (2002). *Team-based learning: A transformative use of small groups in college teaching.* Westport: Praeger.

Miles, J. M., Larson, K. L., & Swanson, M. (2017). Team-based learning in a community health nursing course: Improving academic outcomes. *Journal of Nursing Education, 56*(7), 425–429. https://doi.org/10.3928/01484834-20170619-07.

Miller, R. G. (1981). *Simultaneous statistical inference*. New York: Springer-Verlag.

Mouza, C., Yang, H., Pan, Y.-C., Yilmaz Ozden, S., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology, 33*(3), 61–76. https://doi.org/10.14742/ajet.3521.

NCES. (2018). *Number and percentage distribution of science, technology, engineering, and mathematics (STEM) degrees/certificates conferred by postsecondary institutions, by race/ethnicity, level of degree/certificate, and sex of student: 2008–09 through 2016–17.* National Center for Education Statistics. Retrieved from https://nces.ed.gov/programs/digest/d18/tables/dt18_318.45.asp?current=yes

O'Neill, M. M., Booth, S. R., & Lamb, J. T. (2018). Using NVivo™ for literature reviews: The eight step pedagogy (N7+1). *The Qualitative Report, 23*(13), 21–39.

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

Peterson, L., & Scharber, C. (2017). Learning about makerspaces: Professional development with K-12 inservice educators. *Journal of Digital Learning in Teacher Education, 34*(1), 43–52. https://doi.org/10.1080/21532974.2017.1387833.

Promote Computer Science. (2019). Retrieved from https://code.org/promote

Qualls, J. A., & Sherrell, L. (2010). Why computational thinking should be integrated into the curriculum. *Journal of Computing Sciences in Colleges, 25*, 66–71.

Reimann, P. (2013). Design-based research. In R. Luckin, S. Puntambekar, P. Goodyear, B. L. Grabowski, J. Underwood, & N. Winters (Eds.), *Handbook of Design in Educational Technology* (pp. 37–50). New York: NY: Routledge.

Rich, P. J., Jones, B., Belikov, O., Yoshikawa, E., & Perkins, M. (2017). Computing and engineering in elementary school: The effect of year-long training on elementary teacher self-efficacy and beliefs about teaching computing and engineering. *International Journal of Computer Science Education in Schools, 1*(1). https://doi.org/10.21585/ijcses.v1i1.6.

Rich, P. J., Browning, S. F., Perkins, M., Shoop, T., Yoshikawa, E., & Belikov, O. M. (2018). Coding in K-8: International trends in teaching elementary/primary computing. *TechTrends*, 1–19. https://doi.org/10.1007/s11528-018-0295-4.

Wetzel, K., Foulger, T. S., & Williams, M. K. (2008). The evolution of the required educational technology course. *Journal of Computing in Teacher Education, 25*(2), 67–71.

Williams, T. (2017). What happened to women in Computer Science? Retrieved from https://www.goodcall.com/news/women-in-computer-science-09821. Accessed 10 Jan 2019.

Williams, L. A., & Kessler, R. R. (2001). Experiments with industry's "pair-programming" model in the computer Science classroom. *Computer Science Education, 11*(1), 7–20. https://doi.org/10.1076/csed.11.1.7.3846.

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

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing computational thinking in education courses. Paper presented at the 42nd ACM technical symposium on computer science education, Dallas.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education, 14*(1), 1–16. https://doi.org/10.1145/2576872.

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017a). Computational thinking in teacher education. In P. Rich & C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 205–220). Cham: Springer.

Yadav, A., Stephenson, C., & Hong, H. (2017b). Computational thinking for teacher education. *Communications of the ACM, 60*(4), 55–62. https://doi.org/10.1145/2994591.