Secondary Computer Science Teachers' Pedagogical Needs

Olgun Sadik[1]
Anne Todd Ottenbreit-Leftwich[2]
Thomas Andrew Brush[3]

[1]Inonu University
[2]Indiana University
[3]Indiana University

DOI: 10.21585/ijcses.v4i1.79

Abstract

The purpose of this study is to identify secondary computer science (CS) teachers' pedagogical needs in the United States. Participants were selected from secondary teachers who were teaching CS courses or content in a school setting (public, private, or charter) or an after-school program during the time of data collection. This is a qualitative study using CS teachers' discussions in the Computer Science Teachers Association's (CSTA) email listserv, responses to open-ended questions in a questionnaire, and discussions in follow-up interviews. Content analysis, thematic analysis and constant comparative method of qualitative data analysis were used to analyze the data. The most common pedagogical need expressed was learning student-centered strategies for teaching CS and guiding students' understanding with the use of scaffolding and team-management strategies in CS classes. Furthermore, addressing students' beliefs in CS and their preconceptions in math and reading were important factors influencing teaching CS effectively in secondary schools.

Keywords: computer science education, computing education, teachers' needs, teachers' challenges, computer science pedagogy, learning computer science, teaching computer science, teacher education

1 Introduction

Within the United States, more stakeholders have pushed for increased opportunities for computer science (CS) for K-12 students (Code.org Advocacy Coalitian, 2019; Smith, 2016). To meet this need, state departments of education and legislation have implemented numerous policies and resources to broaden K-12 student access to CS (Code.org, 2019). As of 2019, 15 states have made it a requirement that all high schools need to offer at least one computer science class each year (e.g., Indiana, Texas, West Virginia, Arkansas) (Code.org, 2019). Some have even pushed further, requiring all high school students to complete at least one CS course (e.g., Arkansas and West Virginia). However, integrating CS in K-12 schools is a systemic change (Barr & Stephenson, 2011; DeLyser &

www.manaraa.com

Wright, 2019) and this change requires well-prepared teachers (Lang et al., 2013). With the push for more CS classes, many states have been trying to identify how to meet this need and improve CS teachers' content, curriculum and pedagogical knowledge through teacher education and professional development programs. According to the Code.org Advocacy Coalition's State of Computer Science Education (2019) report, the number of teacher certification programs has been increasing in the United States from 27 in 2017, 33 in 2018 to 38 in 2019. Another way this need for more CS teachers has been met is by training teachers from other content areas (e.g., math, science, foreign languages, etc.) through professional development programs (Menekse, 2015). Other programs have created CS teachers through supporting industry. For example, the Technology Education and Literacy in Schools Program supports teachers by partnering industry computer scientists with practicing teachers (DeLyser & Preston, 2015). However, studies have shown that many secondary CS teachers express needing more support beyond the certification programs and professional development programs provided (Giannakos, Doukakis, Pappas, Adamopoulos, & Giannopoulou, 2015; Qian, Hambrusch, Yadav, & Gretter, 2018). This leads to the problem of what CS teachers' challenges and needs are to help them improve their craft of teaching CS.

1.1 Teachers' Challenges and Needs in CS Education

In spite of current efforts to offer high quality CS classes in K-12 schools, many teachers have challenges and needs to better teach CS in their classes. These challenges can be categorized as knowledge and skills needs, curricular needs, contextual needs and pedagogical needs (Sadik, 2017). In terms of knowledge and skills, CS teachers share their limited CS content knowledge and skills in various studies (Angeli et al., 2016; Yadav, Gretter, Hambrusch, & Sands, 2016). Yadav et al. (2016) expressed that some CS teachers came from different backgrounds and had to learn CS alone from books and online resources. When these teachers were asked about their knowledge and skills needs, they reported the need for better understanding of programming constructs and more experience on computer programming (Yadav et al., 2016). Furthermore, understanding the principles of coputational thinking emerged as another important knowledge and skills need. CS teachers shared their needs for professional development programs to learn and implement the principles of computational thinking in their CS classes (Fessakis & Prantsoudi, 2019). In terms of curriculum, CS teachers reported needs for more resources for content delivery and assessment (Sentance & Csizmadia, 2016). Most CS teachers reported creating their own curricular resources alone (e.g. books, materials, presentationss) (Brown & Kölling, 2013; Yadav et al., 2016). Finding or creating quality assessment is especially difficult due to problem based and collaborative nature of CS projects (Wilson & Guzdial, 2010). Therefore, teachers expressed the need for assessment materials to guide and grade students' work in CS classes (Vivian et al., 2020). In terms of contextual needs, one of the important barriers was no to limited collaboration opportunities between CS teachers in schools (Tenenberg & Fincher, 2007). Most teachers expressed the feeling of loneliness and asked for a colleague to share ideas and resources (Yadav et al., 2016). Even though there are online communities for CS teachers to collaborate (Sadik, 2017), CS teachers need other CS teacher colleagues to regularly discuss and share ideas and resources in their subject (Cutts, Robertson, Donaldson, & O'Donnell, 2017). Furthermore, CS teachers need more support from their schools in terms of accessing up-to-date computer hardware and software. This show the need for technical support staff who can take care of software and hardware updates and maintanance in schools (Sadik, 2017).

In order to address and help CS teachers' needs for content and curriculum, The Association for Computing Machinery (ACM), Code.org, the Computer Science Teachers Association (CSTA), the Cyber Innovation Center, National Math and the Science Initiative developed the K-12 Computer Science Framework. However, recent research emphasized CS pedagogy as the most significant aspect of teaching CS effectively in K-12 schools

www.manaraa.com

(Giannakos et al., 2015; Sentance & Humphreys, 2018; Yadav, et al., 2016). Due to the project and/or problem-based nature of CS education courses (Yadav et al., 2016), planning the lessons, keeping the students active and guiding them in their learning process can be difficult (Davenport, 2000). Furthermore, diverse student needs and interests make it more complicated to teach CS in their classes (Schulte & Knobelsdorf, 2007). Therefore, recent studies reported CS teachers' limited experience with student centered practices and teaching students with diverse needs (Che, Kraemer, & Sitaraman, 2019) and suggested conducting more research on CS teachers' pedagogical needs. In order to understand CS teachers' pedagogical needs, an interested reader needs to know what effective teaching and learning means in CS education as well as the successful instructional strategies in CS classrooms.

1.2 CS Pedagogy

Teaching is a complex field that requires strong pedagogical knowledge for planning, leading and mentoring dynamic classroom environments and students' learning experience. The International Society for Technology in Education (ISTE) (2011) highlighted that effective teaching and learning in CS education requires knowledge of various instructional strategies and materials. Research on CS education pedagogy has been a topic of interest since early 1980. Four of the successful strategies used in CS education include problem-based learning (Kay et al., 2000; Yadav, Subedi, Lundeberg, & Bunting, 2011), project-based learning (Mills & Treagust, 2003), pair programming (McDowell, Werner, Bullock, & Fernald, 2006), and media computation (Guzdial, 2003). For instance, in problem-based learning (PBL), students work in groups to solve a complex problem using various types of scaffolds (Hmelo-Silver, 2003). In CS education PBL, students are given a complex CS problem in a computer lab environment with tutorials to facilitate their problem-solving process. Project-based learning has also been implemented in CS education. Tasks in project-based learning are designed similar to projects in real life and give learners the opportunity to apply their knowledge in product design and development (Mills & Treagust, 2003). Even though there are similarities with PBL, project-based learning is product focused and requires students to be careful about resources and time, while PBL gives more flexibility in this process. Another important practice, pair programming, is a technique in which two programmers work on the same programming task design and development using one computer simultaneously. This technique was derived from CS industry practices and has been used as an instructional method in both K–12 and higher education. Media computation was a recently developed instructional technique that emphasized learning computing concepts and skills in digital media design (Guzdial, 2003). Guzdial has argued that digital media (e.g. images, videos, audios) can be modified and redesigned using programming and this process can help students learn computation in a more meaningful way. In addition to the instructional methods discussed here, previous studies documented the success of using various other tools and environments in CS education such as computer games (Papastergiou, 2009), virtual learning environments (Esteves, Fonseca, Morgado, & Martins, 2011) and robotics kits (Bers, Ponte, Juelich, Viera, & Schenker, 2002).

Even though CS education research has provided evidence of learning gains with all these strategies, tools and contexts, recent research expressed the need for understanding in-service teachers' explicit challenges in CS pedagogy, especially in student centered learning environments. Due to limited population in earlier levels, this study only targets secondary CS teachers and aims to understand their needs in effective teaching in CS education. The present study proposes that K-12 teachers may have crucial needs in pedagogy (Hazzan, Lapidot, & Ragonis, 2015; Yadav et al., 2016) and any efforts aiming to prepare CS teachers, initially, need to identify CS teachers' pedagogical needs and answer the following research question:

www.manaraa.com

1. What pedagogical needs do U.S. secondary school teachers share to better teach computer science classes or content in their classes?

The study aims to reach teachers who are teaching CS content in formal and informal learning environments and explore secondary education teachers' pedagogical needs in teaching CS in the US. It is argued that by identifying needs in this grade range, this study will help:

1. address the specific pedagogical needs of secondary CS teachers,
2. inform administrators and scholars as they develop data-driven professional development programs and resources and

inform teacher education programs about in-service secondary CS teachers' pedagogical needs and assist them in preparing courses that address those needs.

2 Method

Recent literature emphasized the need for improving CS teachers' knowledge and skills in CS pedagogy; however, there is limited exploratory research that explains what that need entails. Using multiple data collection methods for data complementarity and triangulation with rich data (Creswell & Clark, 2017), this study employs general qualitative research design to explore and explain CS teachers' pedagogical needs in detail.

2.1 Participants and Setting

Although previous research has recommended beginning CS education as early as kindergarten (Fessakis, Gouli, & Mavroudi, 2013; Kelleher, Pausch, Pausch, & Kiesler, 2007), due to the limited number of CS teachers and courses at the K-5 level at the time of the present study, the researchers focused only on secondary school level. Participants were selected from secondary teachers who were teaching CS courses or content in a school setting (public, private, or charter) or an after-school program during the time of data collection. In this study, secondary education refers to both middle and high school between grades 6-12. Even though there was a discussion in the literature about the title of the field of study as computing education versus computer science education (Guzdial, 2015), the second was selected because of its broader use in K-12 education and public society. "Model Curriculum for K-12 Computer Science" defined CS as "an academic discipline that encompasses the study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society" (Tucker, 1996, p. 6). With this description, teachers who mentioned teaching the following and related areas were identified and considered as CS teachers for the purposes of this study:

> *programming, hardware design, networks, graphics, databases and information retrieval, computer security, software design, programming languages and paradigms, logic, translation between levels of abstraction, artificial intelligence, the limits of computations (what computers cannot do), applications in information technology and information systems, and social issues (Internet security, privacy, intellectual property (Barr & Stephenson, 2011, p. 113).*

2.2 Data Collection and Analysis

This research was completed in three phases:

www.manaraa.com

1. The first phase included analysis of CS teachers' publicly available email communications in Computer Science Teachers Association (CSTA) listserv. The preliminary step of phase 1 was organizing all the emails in the listserv, which included creating an Excel spreadsheet that included the email subjects, email content and the dates of each email sent. As members responded at different days and times to various topics, all the communications in the spreadsheet were grouped by subject and then sorted by date and time for each subject. After organizing the data, the spreadsheet was imported to Nvivo qualitative data analysis software.
2. The second phase included conducting a questionnaire with open ended questions to all the teachers in the CSTA membership database.
3. In the third stage, interviews with eight purposefully selected teachers were conducted to understand CS teachers' pedagogical needs in more detail.

2.2.1 Email Listserv Analysis

In the first phase of the research, due to extensive data in the email listserv, the teachers' communications were analyzed in two steps. In the first step, the researcher used inductive content analysis to identify the conversations related to pedagogy using the teacher communications (3 years, N=1706) in the email listserv (Weber, 1990). Following the content analysis, in the second step of the analysis in this phase, the email conversations related to pedagogy were analyzed using the thematic analysis technique (Braun & Clarke, 2006) to identify CS teachers' specific pedagogical needs with evidence. Every email identified as pedagogical needs was coded and each of these was categorized into one of seven themes. Table 1 shows examples of codes emerged from the email conversations and depicts how the researchers categorized the codes into the themes. Peer debriefing technique was used in the analysis stage to manage subjectivity, challenge researcher assumptions and discuss alternative interpretations. Initially, one researcher analyzed the email data and other researchers reviewed and provided alternative views to his interpretations.

Table 1. Examples of codes and the themes emerged

CODE EXAMPLES	THEMES EMERGED
• How to introduce programming • Help students learn programming • Guiding students' learning	• Instructional Strategies for Teaching CS
• Connections between programming languages • Pedagogical parallels between programming languages • Transfer learning to new contexts	• Transferring Skills between Programming Languages and Platforms
• Debugging students code • Efficient strategies to answer students' coding questions • Strategies that guide students solve their own problems	• Answering Students' Questions

www.manaraa.com

• Effective teamwork strategies • Creating environments for students' collaboration • Allowing students to work together	• Facilitating Student Interaction and Collaboration
• Looking for resources that increase interest in CS • Increasing interest • Kids give up easily	• Teaching Students with Low Interest in CS
• Limited fundamental skills • Defining set of skills necessary to become a programmer Students different development levels in high school	• Teaching Students Who Lack Fundamental Skills

2.2.2 Questionnaire Analysis

In this phase, the questionnaire was disseminated to CS teacher members of CSTA through their email addresses. 121 members responded to the open-ended questions with rich comments and examples. The open-ended responses were included and analyzed using the constant comparative method of qualitative data analysis (Glaser, 1965). Glaser defines the purpose of the constant comparative method as providing an alternative to analysis, comparing themes and looking for agreements and conflicts ("negative cases or a consideration of alternative hypotheses"), and increasing credibility in the study results. The researcher imported the teachers' responses to Nvivo software and conducted comparison of the email listserv analysis results with the qualitative questionnaire responses. At the end of the questionnaire, the participants were asked to contribute voluntarily in a follow-up, semi-structured interview (3rd phase), in order to elaborate upon their pedagogical needs with examples from their practices.

2.2.3 Interview Data Collection and Analysis

A semi-structured interview protocol was used (Fraenkel, Wallen, & Hyun, 2011) and the interview questions were developed from the questionnaire responses to gather more information on secondary teachers' needs. For example, in the questionnaire, when a participant shared his or her need for learning more strategies on student centered learning in CS classes, s/he was asked to define the need and give examples from his/her classroom. Eight purposefully selected teachers (based on their responses to the questionnaire) participated in the follow-up interviews. The researchers selected participants who reported diverse needs and aimed to enrich the data with detailed explanations and examples. For example, teachers who mentioned scaffolding as a need or strong need in the questionnaire were purposefully selected to the interviews to enrich the data with better explanations and more examples.

The interview data collection ended when the researchers decided that the data was saturated, as suggested by Guba and Lincoln (1985): "exhaustion of sources, saturation of categories, emergence of regularities, and over-extension." The interviews were fully transcribed. The teacher names and all the identifying information were replaced with pseudonyms. The data was analyzed using the constant-comparative method in the same way as conducted in phase 2 to enhance the findings. This phase also helped the researchers provide more descriptive information about CS teachers' needs with examples from the participants' explanations of their practices.

www.manaraa.com

2.3 Issues of Reliability-Validity and Limitations of the Study

The researchers employed various techniques to ensure the standard of trustworthiness of this research (Guba & Lincoln, 1985). In the early stages of the research, multiple researcher meetings were held face to face to establish our research questions, identify our criteria for participant selection and develop and clarify the data collection methods. The data collection process was started using existing teacher discussions in the email listserv. This gave researchers access and opportunity to interpret teacher discussions in a real-life context. This data is triangulated using multiple forms of data sources (questionnaire and interviews) to answer the research question with rich data. This helped the researchers ensure that the data is credible. This reduced the risk for researcher bias. The interviews in the final phase were fully transcribed and analyzed with multiple researchers' input and agreement. Furthermore, the researchers sent the results to the participants and asked their confirmation of the researchers' interpretations. This technique is called member checking and have been used in qualitative research to ensure confirmability of the research results (Birt, Scott, Cavers, Campbell, & Walter, 2016). Nevertheless, the study has limitations. Even though all the possible efforts have been made for trustworthiness, the participants were coming from secondary teacher members of one target organization and did not represent all the secondary CS teachers in the US. Furthermore, there is always risk for researcher bias in qualitative research.

3 Findings

The pedagogical needs were identified from CS teachers' communications in both the listserv and the questionnaire and expanded on with the interview data. Within the findings, email quotations are marked with "E," questionnaire responses are marked with "Q," and the follow up interviews are marked with "I." Each quotation is also identified with a number indicating a unique participant. For the purposes of this study, teachers' perceived pedagogical needs included the following themes:

3.1 Need for Learning Student Centered Strategies for CS Education

Most secondary CS teachers stated that they need to learn new strategies to teach CS content and enhance student learning in their courses. For example, one teacher wanted to know how to teach Linux with new instructional strategies: "I think it's really important for my students to learn Linux but I have no idea how to teach it" (E-2). Pair programming was one of the student-centered learning strategies primarily discussed in the listserv. For example, one teacher shared her failure in using pair programming in a CS class and asked for advice from other CS teachers: "I have done pairing, but must not have done it correct, because it was not as productive as I'd liked. What ideas do you have" (E-4)? In the questionnaire, 17 teachers asked for help related to student-centered strategies in CS education. For example, one teacher stressed the need for facilitating students' learning: "I will appreciate further pedagogical help with teaching computer science and facilitating student knowledge, particularly helping students make better connections with the material, and more effectively debug without needing face time or one-on-one time from me" (Q-2). Another teacher emphasized his need for supporting students' learning via scaffolds in a computer-programming course: "I need better strategies for teaching computer programming to students who have never written computer programs before. What's best to teach first, second, etc. I want a scaffolded approach to teaching programming" (Q-3). When asked to explain this need in more detail and provide examples in the interviews, all eight participants described their current practices and explained why they want to learn new instructional strategies. For instance, I-2 described his current practice as "straight lecture"

www.manaraa.com

and explained it:

> *I do a lot of straight lectures in my classroom, I do lot of type; I'll pull the projector, I'll put code on the projector and have kids follow through on their own computers, basically they copy down what I'm doing. (I-2)*

Another teacher emphasized his need to learn new instructional strategies, problem-based and pair programming approaches for student-centered CS learning:

> *I want to learn new instructional strategies that I never got, because I didn't go to a teaching school, I never went to school to learn to be a teacher. I feel that it's personal that I needed to learn better how to do things like problem-based learning and pair programming. (I-7)*

3.2 Need for Strategies Guiding Students Transfer Skills Between Programming Platforms and Languages

In the listserv emails, the participants expressed the need for helping secondary students transfer their skills from visual programming platforms (e.g., block-coding) to text-based environments (e.g., Python), and between text-based environments (e.g., from Python to Java). For example, one teacher emphasized her students' difficulty transferring their CS learning from visual programming environments to text-based programming environments: "In my teaching, it seems that majority of students have difficulty migrating concepts they learned from visual environments into text-based environments. Starting with Scratch/turtle I found that I essentially had to reteach concepts in Python/Java" (E-11). Another teacher shared the same concern between text-based programming languages: "I have found the same thing with students going from python to Java or python to C" (E-12). In the questionnaire responses, the participants exemplified the transfer issue. For instance, a CS teacher who had started teaching an advanced placement CS (AP-CS) course for the first time and complained about her students' lack of transfer from her previous classes to the new AP CS class: "…we are working on strings again and it seems they do not remember what we were supposed to learn previously. It makes me think there must be something else I can do to help the process" (Q-8). Similarly, the interviewees listed their students' lack of transfer between the CS courses when there were connections. For instance, I-4 shared his failure to help students see the connections between Scratch and other programming languages when moved from one set of content to another:

> *At the end of each semester I ask them what did they think of it, did they feel using Scratch was valuable? Were they able to transfer what they had learned from one language to another in them? The responses I get, for the most part, are that while they found it interesting, they didn't see the parallels. I know the parallel is there. They just don't make the connections. (I-4)*

3.3 Need for Strategies Guiding Students' Errors while Coding

The email listserv members stressed the need for strategies addressing students' questions and problems in programming activities in classroom and explained it as the need for analyzing code quickly and guiding the students' CS learning process while coding. One teacher stressed the importance of analyzing code quickly: "When helping students with a project, the teacher needs to quickly analyze what is wrong with the frustrated students' program and then give some advice on how to fix it" (E-14). In another email, one teacher mentioned the need to provide one-to-one guidance to students "We don't provide one-on-one mentoring to students while programming"

www.manaraa.com

(E-3). In the questionnaire responses, the participants explained this need as assessing code for correctness, giving students appropriate feedback, guiding students through solving code errors, and creating scaffolding that does not need face-to-face guidance. For instance, one teacher explicitly stated correcting students' code as a need for better CS instruction: "I need help with fixing students code, etc. -sample debugging questions in C++, Java C - and pseudocode" (Q-11). Another teacher emphasized her need for strategies to assess students' code for correctness, structure, and efficiency: "I need to streamline the process of assessing my students' code for correctness, use of comments, ease of use, and grammar and spelling in output statements and comments" (Q10). When asked to explain this need in more detail and provide examples in the interviews, four teachers shared their own strategies and asked for more strategies that can lead their students to finding errors in their code. The interviewees asked for strategies that can lead students to find the errors themselves by exploration. One of these teachers used questioning as scaffolding and asked for more strategies that could guide his students:

> *The easy answer that is the wrong one is to point the student to the bug and say: Well, here's what you did wrong. The harder answer in mind, but the one that I prefer is to sort of ask the student: What do you mean by this chunk of code? And have them explain to me what they are trying to say and then Socratic method or using questioning to get them to see what they have said, where what they have said doesn't add up with what they intended to say. I would like to learn more strategies other than just questioning and flat out giving them the answer to help them with that. (I-1)*

3.4 Need for Strategies to Facilitate Student Interaction and Collaboration

The email listserv members stressed the need for strategies creating a collaborative classroom environment and guiding student discussions. For instance, when teachers discussed problems in their classroom, several teachers emphasized the importance of creating an environment for learning: "How do kids learn and how do we create an environment to allow that learning to best take place" (E-13). In the questionnaire responses, the participants explained this need as creating a collaborative environment where all the students help each other and attend learning activities: "How to facilitate student independence and helping each other and strategies for when they really are stuck and need individual help and I can't be there for everyone" (Q-12). Another teacher commented about a collaborative environment as a need: "Creating collaborative environment and excitement around problems and solutions" (Q-13). When asked to explain this need in more detail and provide examples in the interviews, four teachers described their own teaching practices, emphasizing the importance of student interaction and broadening the scope of the need to ensure all the students' active participation in collaborative work. One interviewee explained that collaboration is essential in a CS a class:

> *[Programming projects] forces collaboration in the classroom between me and the students but especially between the students. They quickly learn everybody's got questions and there's only one teacher. If you want your question answered, you've got to go to someone else in the class. The other people in the class don't have to be experts, they just have to be one step ahead of you, because they can help you with the thing you were trying to do." (I-5)*

This collaboration requires a flexible environment where students help each other, become creative and learn from each other. Another teacher explained the need for a less structured CS classroom for effective collaboration and need for ensuring student learn in teamwork:

www.manaraa.com

> *It's really trying to map out your lesson and your unit plans to allow that time, balance that time for them to be able to do some of that collaboration together, or working together around problem solving, or evaluating each other, like a game or an animation or something that they created. All of this requires a little bit less structured environment than what the students are used to, so it's about allowing them to be able to be creative, and have that time so they can just think and kick the tires, but also making sure that that time is effective, and they're learning (I-3)*

3.5 Need for Strategies Teaching Students with Low Interest in CS

Secondary CS teachers reported that they need more strategies to teach students with low interest in CS, which includes strategies to inform students about the challenges and benefits of CS. Increasing student interest especially became an issue for the participants when a CS class was required or students enrolled in a class when they did not have another option. They believe when a student does know what CS is and understands both the challenges and benefits of CS, their interest and motivation increase. Therefore, some teachers were willing to select students before they were allowed to register. The listserv discussions, questionnaire responses and interviews support these findings. The teachers in the email listserv stressed the need to increase their students' interest and motivation and defined those as preconditions, especially when learning programming. The following example demonstrates this need:

> *I have discovered as an in-service teacher that there are a small number of factors that have a disproportionate impact on learning. They are centered around student motivation and interest. The primary question here is how do you get the student engaged and actively seeking knowledge in CS. (E-12)*

While some teachers discussed that being an elective class reduced the enrollment rates in CS classes, some teachers in the listserv stressed this as a positive factor. One of the teachers described this as: "all it really needs is a desire to learn the stuff, and the fact that it is an elective really helps on that front" (E-12). In the questionnaire responses, 10 teachers commented about student interest in CS classes. In these responses, the teachers stated that increasing interest is a need in the following conditions:

- When the course is required, there were students in the class with no to little interest in learning CS (N=5).
- Students came with low interest from various backgrounds (N= 7).
- It was hard to sustain student interest and motivation in programming classes (N=7).

Therefore, those teachers solicited strategies for increasing student interest in CS, as in the following example with underrepresented student populations:

> *Students are now required to take a computer programming class before graduation - so I have many underrepresented populations and girls enrolled in my course. Students who have little interest in the class. I would be greatly beneficial to share strategies on how to motivate students who have little interest in the class. (Q-33)*

Sustaining interest in CS classes was also a need mentioned in the questionnaire responses. Most teachers stressed that many students come in with interest in CS, but as the class moves forward to difficult concepts in coding, they lose that interest. They are looking for strategies to sustain interest throughout the course. For instance, one teacher

www.manaraa.com

approached this issue and asked for help:

> *As a math teacher, I often encounter students not interested in a content. In that class, they have to accept it because it's a core requirement for graduation. In computer science, all students enter with an interest, but some lose that interest as the year progresses. How can I keep them motivated to finish the year even when interest wanes? (Q-35)*

The interviews validated the listserv and questionnaire responses. Seven out of eight interviewees expressed this as a need. When asked to explain this need in more detail and provide examples, the interviewees stated that there were students in their classes who lost interest in the subject and considered it both unrelated to their needs or too challenging. Therefore, most of the interviewees solicited strategies to motivate these students with low interest into learning CS. For instance, one of the teachers stated that there were not many electives for students to choose from in his school and students with low interest were forced to take his class. He expressed the need to be able to help those students to learn CS concepts and skills:

> *I do get a lot of kids in my classes because we're a small school, we don't have a ton of electives. Sometimes kids just get dropped into classes they don't necessarily want, but I still feel that there's value for those kids to understand. (I-2)*

3.6 Need for Strategies Teaching Students Who Lack Literacy and Math Skills

For all the participants of this research, math background was reported as an issue for the students in CS classes. Furthermore, reading comprehension tended to be a problem in some contexts where there were economically disadvantaged and minority students with limited content knowledge in general. Secondary CS teachers stated that they need more strategies to teach students with low mathematics and reading comprehension skills. The email listserv participants stated that some of their students lacked fundamental skills, which influenced their understanding and ability to apply the concepts and skills in CS classes. One teacher described the problem: "Somehow some of my students have core (base) knowledge missing or so confused that it makes it hard for them to progress" (E-37). Math, especially algebra, emerged as the most important skill that students needed to be successful in CS classes: "I have many instances in which I have to divert my curriculum to teach them Algebra concepts that should have known" (E-39). Reading comprehension was another factor identified in the listserv, especially for understanding instructions in CS classes. One teacher highlighted her concern: "Reading comprehension is a struggle. Some of them can't follow specific instructions and don't understand the importance of flawless execution, error free and clear thinking when writing programs" (E-39). In the questionnaire responses, 25 teachers shared opinions about fundamental skills they consider important for learning CS and their need for strategies in dealing with those students who lack them. For instance, one teacher emphasized this as an evolving need with CS becoming more available in schools:

> *Excellent question! As schools adopt CS for everyone, it will add the challenge of working with students with lower reading and math skills. This issue could be the game changer in CS education since up until now the students in the CS program have typically been quite high academic achievers. (Q-26)*

Seven out of eight final-phase interviewees expressed learning strategies to teach students who lack fundamental

www.manaraa.com

skills as a need. The interviewees provided examples of scenarios where their students had difficulty on simple calculations and reading directions. One of the interviewees explained lack of math background as a problem with an example in his CS classes:

> *Some students could think algebraically or abstractly without a transcript credit, but those students who have a limited background in math will struggle when solving complex problems. Even the very simple assignment like I'm going to prompt the user for the number of gallons of gasoline burned and the distance traveled and calculate the miles per gallon, I've seen students in 9th and 10th grade really struggle to figure out how to do that. (I-1)*

Reading comprehension was not explicitly mentioned in overall interview discussions; however, some teachers connected this issue to students' disadvantages and limited skills in core content areas in general. For instance, one of the teachers mentioned English as a second language students in his CS classes:

> *I do have a large number of students who are classified as English language learners. I would say the majority of my students, English is not their first language. The vast majority of their parents have very limited English and speak Spanish at home. (I-8)*

One of the interviewees explained this further, and argued that learning CS requires a background in different content areas:

> *I think it's a combination. I think the content knowledge is obviously extremely important because it is what's out there. I think it goes hand in hand with trying to figure out how to best present them to students. It involves all the related to other things they are learning about in school. It obviously ties in with a lot of math, science, English, a lot of other subjects want to make sure there's some development for students. (I-7)*

4 Discussion

The overall findings suggest that secondary CS teachers need community help from other teachers to meet their needs in teaching CS (Ni & Guzdial, 2012). This study focused on the pedagogical needs and found the participants' primary need as learning and using student-centered learning strategies in CS classes. Specifically, secondary CS teachers stressed the need for scaffolding strategies that can guide students in solving computing problems in various levels at the individual level and in teams. The sections below discuss the participants' pedagogical needs in detail.

4.1 Learning and Applying Scaffolding Strategies in CS Classes

The primary discussion was evolved from the need for student-centered learning strategies in solving CS problems in coding activities. In order to be successful in learning CS, students need guidance while solving problem based activities (Hmelo-Silver, 2004; Mayer, 2003). This guidance is conceptualized as scaffolding in the literature. Scaffolding helps students to complete complex learning tasks that are difficult to achieve without assistance (Belland, 2014; Wood, Bruner, & Ross, 1976). Scaffolding makes the learning process more manageable

www.manaraa.com

(Hmelo-Silver & Bromme, 2007) and is of primary importance in supporting students to reflect on their own learning and develop higher order thinking skills (Azevedo, Cromley, Winters, Moos, & Greene, 2005; Saye & Brush, 2002). Brush and Saye (2002) defined two types of scaffolding: hard and soft. Hard scaffolds are planned in advance such as multimedia resources (e.g. helpful websites). Soft scaffolds are more dynamic and happen simultaneously. Examples of soft scaffolding include teachers' using questioning strategy to guide their problem-solving processes.

Guzdial (2015) also stressed scaffolding as one of those primary conditions for effective CS education. In the present study, secondary CS teachers identified similar issues. In CS, when students are assigned a problem, they often use trial-and-error when they are not given adequate facilitation (Shute, Sun, & Asbell-Clarke, 2017). Trial-and-error is not an efficient problem-solving strategy (Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013), it takes time and often yields no results. The findings suggest a more purposeful design of computational problem solving (Shute et al., 2017), with teachers' scaffolding embedded in the process and facilitating students' coding practices. Secondary CS teachers want to learn more strategies to answer students' questions during coding practices, particularly when students are debugging their own codes, finding and fixing errors, and increasing the efficiency of their code. These are important components of computational thinking in programming activities (Grover & Pea, 2013). Students tend to expect teachers to point out their mistakes, but this is not an effective teaching method. Providing feedback in the form of guiding questions helps students to assess and reflect on their own learning (Nicol & Macfarlane-Dick, 2006).

4.2 Creating a Collaborative Teamwork Environment in CS Classes

The results suggest that teachers need tools and strategies to make sure all students actively participate and equally contribute during teamwork and recommend further research to identify successful team building and management strategies, especially in pair programming activities. Teamwork is a crucial part of CS learning and benefits students' learning experience through sharing information and receiving feedback within a social community of peers (Sancho-Thomas, Fuentes-Fernández, & Fernández-Manjón, 2009). In pair programming, students work collaboratively in teams. Pair programming involves strong pair collaboration within teams, with less teacher involvement (Nagappan et al., 2003). It is not surprising that teachers who requested feedback in the email listserv about pair programming were also looking for team building and management skills to create successful collaborative environments. The findings suggest that CS teachers want to make sure that all their students actively participate in teamwork. Poor teams often involve one expert student taking all the responsibility, while other members become passive participants who may not benefit from the learning opportunities (Shimazoe & Aldrich, 2010). Cooperative team work is the primary condition of successful pair programming practices in computer programming courses (Umapathy & Ritzhaupt, 2017). Teachers in the study also solicited strategies for creating a collaborative learning environment where students search for answers from their partners rather than the teacher. It is important to note that CS teachers have limited time during a class period to answer all the questions, and team work becomes an important opportunity to alleviate those problems (Sancho-Thomas et al., 2009). However, teachers need to be aware of the differences between team members in terms of knowledge/skills and personality (Ally, Darroch, & Toleman, 2005).

4.3 Transferring Knowledge and Skills between Programming Languages and Platforms

Even though limited examples were found, the findings suggest that secondary teachers need strategies to help

www.manaraa.com

students transfer learning and build on their previous knowledge in subsequent CS classes. Transfer of learning is important, not only to make connections between concepts and skills, but also when developing new learning (Driscoll, 1994). Transfer of learning can make CS experiences more connected and meaningful. This need was emphasized in previous CS education studies related to programming. For example, learning the concepts and logic of programming (e.g., variables, loops) from visual programming tools (e.g., Scratch) in middle school helped students learn complex programming concepts more easily in text based languages, and also increased enrollment numbers in CS classes (Armoni, Meerbaum-Salant, & Ben-Ari, 2015). In another study, Franklin et al. (2016) conducted a study with high-school students' and reported this as a challenge. Franklin et al. reported this as an important concern and recommended the teachers help students see the connection between visual programming and text-based programming activities in their instruction.

4.4 Increasing Students' Interest in Learning CS

Most secondary teachers in this study reported that students in their classes do not perceive a connection between CS and their personal lives or their professional futures; thus, they do not value learning CS. Therefore, if teachers want to increase students' positive beliefs and interest in learning CS, they need to find ways to make CS relevant to their students' lives (Tew, Fowler, & Guzdial, 2005). For example, Umbleja (2016) stressed that code.org activities increase students' understanding and interest in CS if they are provided the concepts and skills in primary education or earlier secondary levels. Umbleja emphasized that as they age into their teenager years, it may be difficult to change their biases about CS. This statement is validated in studies with middle school girls. Outlay, Platt, and Conroy (2017) stressed that young ages are critical to develop positive beliefs of CS competency and knowledge and increase interest. To assist with CS teachers' needs regarding students' beliefs, the researchers suggest the following:

1. Introducing CS to students early in their education through short-term programs and local campaigns,
2. Developing curriculum that represents diverse student interests,
3. Defining short-term benefits and learning outcomes

Short-term programs and national promotions (e.g., code.org) have been reported as helpful in gaining interest in CS (Wilson, 2014); however, sustaining that interest in the classroom may be challenging. The findings suggest that teachers need to constantly define goals and benefits for students to retain their interest. Therefore, a place-based education approach may be a successful strategy to define goals for students to serve their local communities through service learning projects. Service learning projects in CS higher education provide long-term motivational benefits for learning (Sanderson, 2003) while creating an engaging and motivating learning environment in K-12 (Billig, 2000). However, K-12 CS education literature appears to be limited in this regard. Service learning approaches may be promoted and further research should explore effective strategies to employ service learning in K-12 CS education.

4.5 Teaching Students with Limited Math and Reading Background

The findings suggest that low abilities in Math and reading create challenging teaching environments for CS teachers trying to manage and teach students with widely varying needs and learning goals. For example, CS teachers reported that some students were unable to do simple calculations and read instructions to complete simple programming tasks. These findings align with previous research regarding the importance of math background in

www.manaraa.com

learning CS. In a study with 123 first-year introductory programming courses at a higher education institutions, Bergin and Reilly (2005) found that mathematics ability is one of the important predictors of students' success in programming classes. Grover, Pea, and Cooper (2016) reported correlation between middle school students' prior reading and Math abilities, and learning computer programming. Regarding students' math and reading comprehension, secondary CS teachers were looking for strategies to guide students with problems in those areas before they register for CS classes. However, this goes against the national goal known as "CS for All" (Smith), and would create bigger problems in the future. In fact, several teachers in the study mentioned that "CS for All" may create classes that are difficult to manage. Teachers need strategies to help them deal with wide range of skills in math and reading among their students. Further research may be helpful to develop student-centered practices for teaching students with different math and reading skills in secondary CS classes.

5 Conclusion

In the present study, the data in the email listserv demonstrated a wide range of teachers' self-reported needs in a natural community setting. Furthermore, rich discussions evolving from the listserv discussions allowed the researchers to enrich the findings with the questionnaire and interview data. Similar studies were also conducted to understand CS teachers' needs in smaller samples. This study is important to validate the findings of previous studies and start new discussions in the area with rich data. Similar to this study, previous studies recommended using student-centered strategies for effective CS education (Hazzan et al., 2015). For example, current literature discussed problem-based learning (Mills & Treagust, 2003) and pair programming (McDowell et al., 2006). Furthermore, scaffolding and facilitating students' learning in student centered environments were discussed in the CS education literature (e.g. (Caspersen & Bennedsen, 2007). However, this study provided evidence that secondary CS teachers try to use pair programming, problem-based learning and scaffolding in their classes but they were not satisfied with the outcomes of their efforts. Secondary teachers need feedback to improve their practices.

Although this study represents a large population of secondary CS teachers, the participants in this study are U.S. members of one international organization and do not represent all the CS teachers in the U.S. Furthermore, the findings are not suggested to be generalized. The data represents teachers who are members of the CSTA and were identified as secondary CS teachers in the email listserv based on available information. Although all the efforts have been made to exclude them, there is a small risk that some elementary teachers or higher education faculty might be included in the findings. This study suggests that future studies need to be conducted to explore the following topics in CS classrooms:

- Successful student-centered learning strategies (e.g. PBL and pair-programming) in secondary CS classes
- The development of strategies to increase student interest and motivation in learning CS.

Furthermore, it would be helpful to develop research studies that observe successful teachers' practices in their classrooms. Action research studies that allow teachers and researchers to work together (Lang et al., 2013) and aim to address secondary CS teachers' needs in practice would create helpful information for practicing teachers. Developing PD programs based on the suggestions of this study may promote and develop data-driven PD programs for teachers. With the goal of CS for All, this study may be replicated to explore the needs of elementary CS teachers.

www.manaraa.com

Disclosure statement

No potential conflict of interest was reported by the authors.

References

Ally, M., Darroch, F., & Toleman, M. (2005). A framework for understanding the factors influencing pair programming success. In *Proceedings of the International Conference on Extreme Programming and Agile Processes in Software Engineering* (pp. 82-91). New York, NY. https://doi.org/10.1007/11499053_10

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Journal of Educational Technology & Society, 19*(3), 47-57.

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From scratch to "real" programming. *Transactions on Computing Education, 14(*4), 25. https://doi.org/10.1145/2677087

Azevedo, R., Cromley, J. G., Winters, F. I., Moos, D. C., & Greene, J. A. (2005). Adaptive human scaffolding facilitates adolescents' self-regulated learning with hypermedia. *Instructional Science, 33*, 381-412. https://doi.org/10.1007/s11251-005-1273-8

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: what is involved and what is the role of the computer science education community? *Inroads, 2*(1), 48-54. https://doi.org/10.1145/1929887.1929905

Belland, B. R. (2014). Scaffolding: Definition, current debates, and future directions. In *Handbook of Research on Educational Communications and Technology* (pp. 505-518): Springer. https://doi.org/10.1007/978-1-4614-3185-5_39

Bergin, S., & Reilly, R. (2005). Programming: Factors that influence success. *ACM SIGCSE Bulletin, 37*(1), 411-415. https://doi.org/10.1145/1047124.1047480

Bers, M. U., Ponte, I., Juelich, C., Viera, A., & Schenker, J. (2002). Teachers as designers: Integrating robotics in early childhood education. *Information Technology in Childhood Education Annual, 2002*(1), 123-145. https://www.learntechlib.org/primary/p/8850/article_8850.pdf

Birt, L., Scott, S., Cavers, D., Campbell, C., & Walter, F. (2016). Member checking: A tool to enhance trustworthiness or merely a nod to validation?. *Qualitative Health Research, 26(*13), 1802-1811. https://doi.org/10.1177/1049732316654870

Billig, S. (2000). Research on K-12 school-based service-learning: The evidence builds. *Phi Delta Kappan*, *81*(9), 658-664. https://digitalcommons.unomaha.edu/slcek12/3

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3*(2), 77-101. https://doi.org/10.1191/1478088706qp063oa

Brown, N. C. C., & Kölling, M. (2013). A tale of three sites: Resource and knowledge sharing amongst computer science educators. In *Proceedings of the Ninth Annual Conference on International Computing Education Research* (pp. 27–34). La Jolla, CA: ACM. https://doi.org/10.1145/2493394.2493398

www.manaraa.com

Brush, T. A., & Saye, J. W. (2002). A summary of research exploring hard and soft scaffolding for teachers and students using a multimedia supported learning environment. *The Journal of Interactive Online Learning, 1*(2), 1-12. http://www.ncolr.org/jiol/issues/pdf/1.2.3.pdf

Caspersen, M. E., & Bennedsen, J. (2007). Instructional design of a programming course: A learning theoretic approach. In *Proceedings of the 3rd International Workshop on Computing Education Research* (pp. 111-122). Atlanta, Georgia: ACM. https://doi.org/10.1145/1288580.1288595

Che, S. M., Kraemer, E. T., & Sitaraman, M. (2019). Prospective high school computer science teachers' perceptions of inquiry pedagogy and equity. In *Proceedings of the AERA Online Paper Repository*. Toronto, Canada: AERA. https://doi.org/10.302/1444296

Code.org. (2019). State Tracking 9 Policies. Retrieved from https://docs.google.com/spreadsheets/d/1YtTVcpQXoZz0IchihwGOihaCNeqCz2HyLwaXYpyb2SQ/pubhtml

Code.org Advocacy Coalitian. (2019). 2019 State of Computer Science Education. Retrieved from https://advocacy.code.org/

Creswell, J. W., & Clark, V. L. P. (2017). *Designing and conducting mixed methods research*: Sage Publications.

Cutts, Q., Robertson, J., Donaldson, P., & O'Donnell, L. (2017). An evaluation of a professional learning network for computer science teachers. *Computer Science Education, 27*(1), 30-53.

Davenport, D. (2000). Experience using a project-based approach in an introductory programming course. *IEEE Transactions on Education, 43*(4), 443-448. https://doi.org/10.1109/13.883356.

DeLyser, L. A., & Wright, L. (2019). A Systems Change Approach to CS Education: Creating Rubrics for School System Implementation. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 492-498). Aberdeen, Scotland: ACM. https://doi.org/10.1145/3304221.3319733

DeLyser, L. A., & Preston, M. (2015). A public school model of cs education. In *Proceedings of the International Conference on Frontiers in Education: Computer Science and Computer Engineering* (pp. 233–238). Nevada, USA. https://pdfs.semanticscholar.org/90fc/817b131253e2092e712aa1c78fd0f7778d68.pdf

Driscoll, M. P. (1994). *Psychology of learning for instruction*. Washington, DC: Allyn & Bacon.

Esteves, M., Fonseca, B., Morgado, L., & Martins, P. (2011). Improving teaching and learning of computer programming through the use of the Second Life virtual world. *British Journal of Educational Technology, 42*(4), 624-637. https://doi.org/10.1111/j.1467-8535.2010.01056.x

Fessakis, G., & Prantsoudi, S. (2019). Computer science teachers' perceptions, beliefs and attitudes on computational thinking in Greece. *Informatics in Education, 18*(2), 227. https://doi.org/10.15388/infedu.2019.11

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87-97. https://doi.org/10.1016/j.compedu.2012.11.016

Fraenkel, J. R., Wallen, N. E., & Hyun, H. H. (2011). *How to design and evaluate research in education*. New York: McGraw-Hill

www.manaraa.com

Franklin, D., Hill, C., Dwyer, H. A., Hansen, A. K., Iveland, A., & Harlow, D. B. (2016). Initialization in Scratch: Seeking knowledge transfer. In *Proceedings of the 47th ACM Technical Symposium on Computer Science Education* (pp. 217–222). Memphis, Tennessee, USA: ACM. https://doi.org/10.1145/2839509.2844569

Giannakos, M. N., Doukakis, S., Pappas, I. O., Adamopoulos, N., & Giannopoulou, P. (2015). Investigating teachers' confidence on technological pedagogical and content knowledge: An initial validation of TPACK scales in K-12 computing education context. *Journal of Computers in Education, 2*(1), 43-59. https://doi.org/10.1007/s40692-014-0024-8

Glaser, B. G. (1965). The constant comparative method of qualitative analysis. *Social Problems, 12*(4), 436-445. https://dx.doi.org/10.4135/9781412950558.n101

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

Grover, S., Pea, R., & Cooper, S. (2016). Factors influencing computer science learning in middle school. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 552–557). Tennessee, USA: ACM. https://doi.org/10.1145/2839509.2844564

Guba, E. G., & Lincoln, Y. S. (1985). *Naturalistic inquiry*. New York, NY: Sage.

Guzdial, M. (2003). A media computation course for non-majors. *ACM SIGCSE Bulletin, 35*(3), 104-108. https://doi.org/10.1145/961511.961542.

Guzdial, M. (2015). Learner-centered design of computing education: Research on computing for everyone. *Synthesis Lectures on Human-Centered Informatics, 8*(6), 1-165. https://doi.org/10.2200/S00684ED1V01Y201511HCI033

Hazzan, O., Lapidot, T., & Ragonis, N. (2015). *Guide to teaching computer science: An activity-based approach*. New York, NY: Springer. https://doi.org/10.1007/978-1-4471-6630-6.

Hmelo-Silver, C. E. (2003). Problem-based learning. In J. W. Guthrie (Ed.), *Encyclopedia of Education* (Second ed., Vol. 4, pp. 1173-1175). New York: MacMillan Reference.

Hmelo-Silver, C. E. (2004). Problem-Based Learning: What and how do students learn? *Educational Psychology Review, 16*, 235-266. https://doi.org/10.1023/B:EDPR.0000034022.16470.f3

Hmelo-Silver, C. E., & Bromme, R. (2007). Coding discussions and discussing coding: Research on collaborative learning in computer-supported environments. *Learning and Instruction, 17*(4), 460-464. https://doi.org/doi:10.1016/j.learninstruc.2007.04.004

International Society for Technology in Education. (2011). NETS for Teachers. Retrieved from https://www.iste.org/standards/for-educators

Kay, J., Barg, M., Fekete, A., Greening, T., Hollands, O., Kingston, J. H., & Crawford, K. (2000). Problem-based learning for foundation computer science courses. *Computer Science Education, 10*(2), 109-128. https://doi.org/10.1076/0899-3408(200008)10:2;1-C;FT109

Kelleher, C., Pausch, R., Pausch, R., & Kiesler, S. (2007). *Storytelling alice motivates middle school girls to learn computer programming.* In the Proceedings of the SIGCHI Conference on Human Factors in Computing Systems (pp. 1455–1464). California, USA: ACM. https://doi.org/10.1145/1240624.1240844

www.manaraa.com

Lang, K., Galanos, R., Goode, J., Seehorn, D., Trees, F., Phillips, P., & Stephenson, C. (2013). *Bugs in the system: Computer science teacher certification in the US*. Retrieved from https://c.ymcdn.com/sites/www.csteachers.org/resource/resmgr/CSTA_BugsInTheSystem.pdf

Mayer, R. E. (2003). *Learning and instruction*. New Jersey, NY: Pearson.

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

Mills, J. E., & Treagust, D. F. (2003). Engineering education—Is problem-based or project-based learning the answer. *Australasian Journal of Engineering Education, 3*(2), 2-16.

Nagappan, N., Williams, L., Ferzli, M., Wiebe, E., Yang, K., Miller, C., & Balik, S. (2003). Improving the CS1 experience with pair programming. *ACM SIGCSE Bulletin, 35*(1), 359-362. https://doi.org/10.1145/792548.612006

Ni, L., & Guzdial, M. (2012). *Who am I? Understanding high school computer science teachers' professional identity.* In P*roceedings of the 43rd ACM Technical Symposium on Computer Science Education* (pp. 499–504). North Carolina, USA: ACM. https://doi.org/10.1145/2157136.2157283

Nicol, D. J., & Macfarlane-Dick, D. (2006). Formative assessment and self-regulated learning: A model and seven principles of good feedback practice. *Studies in Higher Education, 31*(2), 199-218. https://doi.org/10.1080/03075070600572090

Outlay, C. N., Platt, A. J., & Conroy, K. (2017). Getting IT together: A longitudinal look at linking girls' interest in IT careers to lessons taught in middle school camps. *ACM Transactions on Computing Education, 17*(4), 20. https://doi.org/10.1145/3068838

Papastergiou, M. (2009). Digital game-based learning in high school computer science education: Impact on educational effectiveness and student motivation. *Computers & Education, 52*(1), 1-12. https://doi.org/10.1016/j.compedu.2008.06.004

Qian, Y., Hambrusch, S., Yadav, A., & Gretter, S. (2018). Who needs what: Recommendations for designing effective online professional development for computer science teachers. *Journal of Research on Technology in Education, 50*(2), 164-181. doi:10.1080/15391523.2018.1433565

Sadik, O. (2017). *What do secondary computer science teachers need? Examining curriculum, pedagogy, and contextual support*[Doctoral dissertation, Indiana University]. ProQuest Dissertations Publishing.

Sancho-Thomas, P., Fuentes-Fernández, R., & Fernández-Manjón, B. (2009). Learning teamwork skills in university programming courses. *Computers & Education, 53*(2), 517-531. https://doi.org/10.1016/j.compedu.2009.03.010.

Sanderson, P. (2003). Where's (the) computer science in service-learning? *Journal of Computing Sciences in Colleges, 19*(1), 83-89

Saye, J. W., & Brush, T. (2002). Scaffolding critical reasoning about history and social issues in multimedia-supported learning environments. *Educational Technology Research and Development, 50*(3), 77-96. https://doi.org/10.1007/BF02505026

www.manaraa.com

Schulte, C., & Knobelsdorf, M. (2007). Attitudes towards computer science-computing experiences as a starting point and barrier to computer science. In *Proceedings of the Third International Workshop on Computing Education Research* (pp. 27–38). New York, NY: ACM. https://doi.org/10.1145/1288580.1288585

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351-380. https://doi.org/10.1007/s10639-012-9240-x

Sentance, S., & Humphreys, S. (2018). Understanding professional learning for computing teachers from the perspective of situated learning. *Computer Science Education, 2*8(4), 345-370. https://doi.org/10.1080/08993408.2018.1525233

Sentance, S., & Csizmadia, A. (2016). Computing in the curriculum: Challenges and strategies from a teacher's perspective. *Education and Information Technologies*, 1–27. https://doi:10.1007/s10639-016-9482-0

Shimazoe, J., & Aldrich, H. (2010). Group work can be gratifying: Understanding & overcoming resistance to cooperative learning. *College Teaching, 58*(2), 52-57. https://doi.org/10.1080/87567550903418594

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142-158. https://doi.org/10.1016/j.edurev.2017.09.003.

Smith, M. (2016). Computer science for all. Retrieved from https://obamawhitehouse.archives.gov/blog/2016/01/30/computer-science-all

Tenenberg, J., & Fincher, S. (2007). Opening the door of the computer science classroom: The disciplinary commons. *ACM SIGCSE Bulletin, 39*(1), 514-518. https://doi.org/10.1145/1227504.1227484

Tew, A. E., Fowler, C., & Guzdial, M. (2005). Tracking an innovation in introductory CS education from a research university to a two-year college. *ACM SIGCSE Bulletin, 37*(1), 416-420. https://doi.org/10.1145/1047124.1047481

Tucker, M. S. (1996). Skills, Standards, Qualification systems, and the american workforce. In L. B. Resnick & J. G. Wirt (Eds.), *Linking school and work: Role for standards and assessment* (pp. 23-51). San Francisco CA: Jossey-Bass.

Umapathy, K., & Ritzhaupt, A. D. (2017). A meta-analysis of pair-programming in computer programming courses: Implications for educational practice. *ACM Transactions on Computing Education, 17*(4), 16. https://doi.org/10.1145/2996201

Umbleja, K. (2016). *Can K-12 students learn how to program with just two hours?* In *Proceedings of the International Workshop on Learning Technology for Education Challenges* (pp. 250-264). New York, NY: Springer. https://doi.org/10.1007/978-3-319-42147-6_21

Vivian, R., Franklin, D., Frye, D., Peterfreund, A., Ravitz, J., Sullivan, F., ... & McGill, M. M. (2020). Evaluation and assessment needs of computing education in primary grades. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 124-130). Trondheim, Norway: ACM. https://doi.org/10.1145/3341525.3387371

Weber, R. P. (1990). *Basic content analysis*. Los Angeles, CA: Sage Publications.

Wilson, C. (2014). Hour of code: We can solve the diversity problem in computer science. *Inroads, 5*(4), 22. https://doi.org/10.1145/2684721.268472

www.manaraa.com

Wilson, C., & Guzdial, M. (2010). How to make progress in computing education. *Communications of the ACM, 53*(5), https://doi.org/35-37.10.1145/1735223.1735235

Wood, D., Bruner, J. S., & Ross, G. (1976). The role of tutoring in problem solving. *Journal of Child Psychology and Psychiatry, 17*(2), 89-100. https://doi.org/10.1111/j.1469-7610.1976.tb00381.x

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: Understanding teacher experiences and challenges. *Computer Science Education, 26*(4), 235-254. https://doi.org/10.1080/08993408.2016.1257418

Yadav, A., Subedi, D., Lundeberg, M. A., & Bunting, C. F. (2011). Problem-based learning: Influence on students' learning in an electrical engineering course. *Journal of Engineering Education, 100*(2), 253-280. https://doi.org/10.1002/j.2168-9830.2011.tb00013.x

www.manaraa.com