2019-12-09

Teaching K-6 Computer Science: Teacher and Student Attitudes and Self-Efficacy

Stacie Lee Mason
Brigham Young University

Teaching K–6 Computer Science: Teacher and Student Attitudes and Self-Efficacy

Stacie Lee Mason

A dissertation submitted to the faculty of
Brigham Young University
in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Peter J. Rich, Chair
Royce Kimmons
Ross Larsen
Heather Leary
Erin Whiting

Department of Instructional Psychology and Technology

Brigham Young University

ABSTRACT

Teaching K-6 Computer Science: Teacher and Student Attitudes and Self-Efficacy

Stacie Lee Mason
Department of Instructional Psychology and Technology, BYU
Doctor of Philosophy

This article-format dissertation addresses elementary student and teacher attitudes and self-efficacy for computer science. The first article (Mason & Rich, in press) describes what the literature says about preservice and inservice training to help K-6 teachers increase knowledge and self-efficacy to teach computer science. The second article (Mason, West, & Leary, under review) describes an effort to provide training for local elementary school teachers to teach computational thinking with robots. The third article (Mason & Rich, under review) describes how we developed and validated an instrument to assess K-8 students' coding attitudes and beliefs, including perceived self-efficacy, interest, utility value, gender stereotypes, cultural stereotypes, and social value.

Keywords: coding, computer science, elementary students, primary students, self-efficacy, attitudes

ACKNOWLEDGEMENTS

Numerous people helped me complete this research. I am grateful to the elementary students who took our coding attitudes survey, the teachers who administered it, and the parents and administrators who allowed it. I also appreciate the teachers who voluntarily learned to use robots and answered my questions about the training. Their willingness to try new things benefits their students and the wider community.

My committee members and other faculty provided instruction, guidance, and feedback instrumental to my completing this dissertation. Thanks to Rick West for starting me on this project, guiding me through early drafts of the literature review, and co-authoring the second article along with Heather Leary. Thank you to Randy Davies for providing expert review of survey items. Thank you to Erin Whiting, Heather Leary, and Royce Kimmons for contributing your time, expertise, feedback, and offspring (as survey takers). Thanks to Ross Larsen for your expert instruction, without which I could not have begun to perform psychometric analysis of the coding attitudes survey. And thank you to Peter Rich for meeting with me often, co-authoring two of the articles, and patiently mentoring me throughout the project.

Finally, thanks to Kit and Len for exemplifying hard work, inquiry, and faith; LeAnn for showing interest and celebrating accomplishments; Sam, Anna, Michael, and Ellen, for inspiring and loving me; and Nick for doing all the above and providing endless empathic support.

TABLE OF CONTENTS

LIST OF TABLES

Article 1

Article 2

Article 3

LIST OF FIGURES

Article 1

Article 2

Article 3

DESCRIPTION OF RESEARCH AGENDA AND STRUCTURE OF DISSERTATION

This dissertation is in an article format that combines traditional dissertation requirements with journal publication formats. The preliminary pages of the dissertation reflect requirements for submission to the university. The dissertation report is presented as three journal articles and conforms to length and style requirements for submitting research reports to education journals.

The first article, "Preparing Elementary School Teachers to Teach Computing, Coding, and Computational Thinking" (Mason & Rich, in press), is an extended literature review that examines what published research indicates about training K-6 teachers to teach computer science and helping them to overcome their knowledge and efficacy barriers. It has been accepted by the journal *Contemporary Issues in Technology and Teacher Education*.

The second article, "Designing Teacher Professional Development for Teaching Computational Thinking" (Mason, West, & Leary, under review), describes an effort to provide training for local elementary school teachers to teach computational thinking with robots. This article is under review by the *Journal of Formative Design in Learning*.

The third article, "Development and Analysis of the Elementary Student Coding Attitudes Survey" (Mason & Rich, under review), describes how we developed and validated an instrument to assess elementary students' coding attitudes and beliefs, including perceived self-efficacy, interest, utility value, gender stereotypes, cultural stereotypes, and social value. It is formatted according to the guidelines for *Computers & Education*, where it is under review.

These articles are formatted for journal submission, with the references used for each article included at the end of that article.

ARTICLE 1

Preparing Elementary School Teachers to Teach Computing, Coding, and Computational Thinking

Stacie L. Mason

Peter J. Rich

Brigham Young University

Abstract

This literature review synthesizes current research on preservice and inservice programs that improve K–6 teachers' attitudes, self-efficacy, or knowledge to teach computing, coding, or computational thinking. A review of current computing training for elementary teachers revealed 21 studies: 12 involving preservice teachers and nine involving inservice teachers. The findings suggest that training that includes active participation can improve teachers' computing self-efficacy, attitudes, and knowledge. Because most of these studies were fairly short-term and content-focused, research is especially needed about long-term outcomes; pedagogical knowledge and beliefs; and relationships among teacher training, contexts, and outcomes.

Keywords: professional development (PD), computational thinking (CT), self-efficacy, teacher technology self-efficacy

Preparing Elementary School Teachers to Teach Computing, Coding, and Computational Thinking

Learning computer science (CS) skills can benefit students economically and academically. In the United States, job opportunities in computer and information technology are projected to increase 13% in 10 years, compared to 7% overall projected job growth (Bureau of Labor Statistics, 2018a, 2018b). Numerous studies have indicated a host of benefits from learning CS, including improvement in student engagement, motivation, confidence, problem-solving, communication, and science, technology, engineering, and math (STEM) learning and performance (Clements & Gullo, 1984; Kim et al., 2015; Rich, Leatham, & Wright, 2013; Schanzer, Fisler, & Krishnamurti, 2018; Scherer, Siddiq, & Sánchez Viveros, 2018). Recognizing these benefits, school districts and state governments are increasingly adopting policies that require CS instruction (Rich & Hodges, 2017). According to Stanton et al.'s (2017) report, seven U.S. states had adopted K–12 CS standards, all between 2014 and 2017, and as of 2017, an additional eight states were in the process of doing so. By May 2019, 31 states had adopted CS standards and six other states were in the process of doing so (Code.org, 2019).

In response to increased policies, there have been recent increases and revisions in standards for computational thinking (CT) and CS. For example, in 2013, England established standards for a national computing program of study (Department for Education, 2013). These are broken into four key stages, outlining ways in which pupils should be taught to think logically, understand networking, use technology safely, and design and develop information processing programs in at least two different languages. Other governments have followed suit. Based on UNESCO's 2011 Information and Communication Technologies (ICT) standards for teachers, Australia has implemented standards for digital literacy that direct teachers to begin

teaching computing as early as second grade (New South Wales Education Standards Authority, 2017). In Finland, the standards dictate that teachers must integrate CT into existing curricula beginning in first grade. In the United States, a CT framework has been developed to help local governments create standards that will suit their own needs (*K–12 Computer Science Framework*, 2016). The framework covers five key concepts: (a) computing systems, (b) networks and the Internet, (c) data and analysis, (d) algorithms and programming, and (e) impacts of computing. These concepts cross all grade bands (K–2, 3–5, 6–8, and 9–12), but include guidance for increasing complexity as students progress from elementary to secondary education. Most recently, the Computer Science Teachers Association and International Society for Technology in Education (ISTE) have re-written their CT standards to indicate how CT should be integrated across a variety of subject areas rather than in CS education alone (ISTE, 2017). These standards indicate how teachers ought to fill the roles of computational learners, leaders, collaborators, designers, and facilitators.

As the demand for CS instruction increases, there is a shortage of K–12 teachers who are trained to teach CS or CT. In a report by Google, Inc. and Gallup, Inc. (2016), 63% of surveyed K–12 principals in schools that did not offer CS instruction said that they lacked qualified teachers. As Kundukulam (2018) noted, CS graduates are more likely to take technology jobs than teaching jobs. Instead of hiring specialists to teach coding and computing, school principals are relying on existing teachers to teach CS (Rich, Browning, Perkins, Shoop, & Yoshikawa, 2018). Unfortunately, most elementary school teachers have not been trained in the content and pedagogy of CS (Ng, 2017; Rich, Jones, Belikov, Yoshikawa, & Perkins, 2017; Stanton et al., 2017). Lacking training, teachers face emotional and knowledge barriers. To gain the

competence and confidence to teach computing, elementary schools need effective preservice and inservice training.

Although extensive research has been published on the topics of teacher professional development (PD) and technology integration, relatively little has been published about preparing elementary teachers to teach CS skills. There now exist several related literature reviews that might inform how to effectively prepare computing teachers. For example, there are multiple literature reviews about teacher PD (Desimone, Porter, Garet, Yoon, & Birman, 2002; Guskey & Yoon, 2009, Institute for the Advancement of Research in Education at AEL, 2004), technology integration (e.g., Ertmer & Ottenbreit-Leftwich, 2010; Hew & Brush, 2007; Lawless & Pellegrino, 2007), computing education (Crick, 2017; Garneli, Giannakos, & Chorianopoulos, 2015; Kallia, 2017; K. Rich, Strickland, & Franklin, 2017; Waite, 2017), and CT in education (Grover & Pea, 2013; Heintz, Mannila, & Färngvist, 2016; Ilic, Haseski, & Tugtekin, 2018; Lockwood & Mooney, 2018; Lye & Koh, 2014). But there is a lack of reviews about teacher training for teaching computing, especially in elementary (K–6) education. The purpose of this literature review is to examine what research has found about training K–6 teachers to teach CS and helping them to overcome their knowledge and self-efficacy barriers to teach elementary computing.

Definitions and Frameworks

The articles in this review represent the intersection of three concepts whose interplay forms the foundation for successfully preparing elementary computing teachers: (a) CS, (b) barriers to computing instruction, and (c) teacher preparation and development. Before examining that interplay, we first define and discuss each concept separately.

CS

In this literature review, we include studies that relate to various aspects of elementary CS instruction. Certain authors have focused on computing skills, whereas others have focused on coding, robotics, or CT. The *K–12 Computer Science Framework* (2016), which was developed in partnership with states and districts by the Association for Computing Machinery, Code.org, the Computer Science Teachers Association, the Cyber Innovation Center, and the National Math and Science Initiative, uses Tucker et al.'s (2003) definition of CS: "The study of computers and algorithmic processes, including their principles, their hardware and software designs, their applications, and their impact on society" (p. 6). Accordingly, CS curricula includes all of the following:

> programming, hardware design, networks, graphics, databases and information retrieval, computer security, software design, programming languages, logic, programming paradigms, translation between levels of abstraction, artificial intelligence, the limits of computation (what computers *can't* do), applications in information technology and information systems, and social issues (Internet security, privacy, intellectual property, etc.). (Tucker et al., 2003, p. 6)

CS, then, is the broad discipline that encompasses computing, CT, coding, and other branches dealing with computing connectivity and hardware.

The K–12 Computer Science Framework (n.d.) defines *computing* as "any goal-oriented activity requiring, benefiting from, or creating algorithmic processes" and *code* as "any set of instructions expressed in a programming language." We define *coding* as the act of writing code.

According to Grover and Pea (2013), CT is "viewed as at the core of all STEM disciplines" (p. 38). Put simply, CT is "thinking like a computer scientist" (Wing, 2006, p. 34).

Despite the simple definition, CT is a broad term with multiple definitions (Barr & Stephenson, 2011; Jaipal-Jamani & Angeli, 2017; Sadik, Ottenbreit-Leftwich, & Nadiruzzaman, 2017; Shute, Sun, & Asbell-Clarke, 2017). Wing (2006) further explained, CT is "thinking recursively," "using abstraction and decomposition," and "reformulating a seemingly difficult problem into one we know how to solve," and "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (p. 33). The K–12 Computer Science Framework (n.d.) uses Lee's (2016) definition: "The human ability to formulate problems so that their solutions can be represented as computational steps or algorithms to be executed by a computer" (Lee, 2016, p. 3). For Lee, CT is using a computer to solve problems, whereas other definitions allow practitioners to apply principles of CT (e.g., algorithms or pattern-finding) without using computers or solving problems. For example, for the purposes of their literature review, Shute et al. (2017) defined CT as "the conceptual foundation required to solve problems effectively and efficiently (i.e., algorithmically, with or without the assistance of computers) with solutions that are reusable in different contexts" (p. 142). The articles in this literature review reflect these varied definitions of CT, as well as studies involving other aspects of CS, coding, and programming. Throughout the paper, we treat *CS* as encompassing computing, coding, programming, robotics, and CT.

Barriers

Primary teachers face a variety of barriers to teaching CS. Teachers may face physical barriers, such as a lack of computers or reliable Internet access; institutional barriers in the form of unsupportive administrators or legislators; and emotional barriers, including beliefs, attitudes, and dispositions that hinder technology use. The articles in this review focus largely on intrinsic barriers including knowledge, attitude, and efficacy barriers, which Ertmer, Ottenbreit-Leftwich,

and York (2006) suggested may be more important than external, first-order barriers such as access to technology and resources.

Teacher knowledge significantly affects teacher practice (Borko & Putnam, 1995; Ertmer & Ottenbreit-Leftwich, 2010), and effective technology instruction or integration requires multiple types of knowledge. For teachers to teach with technology, they must understand the content they are teaching, the technology they are using, and pedagogy related to the content, technology, and students (Mishra and Koehler, 2006). A lack of knowledge in any of these areas—content, technology, or pedagogy—could be a knowledge barrier for teachers of coding, computing, or robotics.

Teachers with low technology self-efficacy are less likely to use technology than are confident teachers (Holden & Rada, 2011; Vannatta & Fordham, 2004). Perceived self-efficacy is a judgement of one's ability to perform (Bandura, 1977); technology self-efficacy is a person's belief that "he/she will be successful in using the technology" (Holden & Rada, 2011, p. 347). Self-efficacy is distinct from what Bandura (1977) called outcomes expectations, the belief that specific behavior will lead to specific outcomes. Both efficacy beliefs and outcome expectations influence a person's likelihood to act. For example, if a teacher thinks students should learn to code, but lacks confidence in her ability to teach coding, she might choose not to teach coding. Similarly, a teacher who is confident in her abilities to use computers and teach effectively might choose not to teach coding if she does not expect that students need to learn coding.

In summary, there are several barriers that may prevent educators from successfully teaching elementary computing. Although first-order barriers such as access and resources are important, it is intrinsic, second-order barriers such as low self-efficacy or lack of technological,

pedagogical, or content knowledge that may enable or prevent them from successfully teaching computing (Ertmer, Ottenbreit-Leftwich, Sadik, Sendurer, & Sendurer, 2012).

Teacher Preparation and Development

This review includes studies involving both preservice and inservice teacher training to prepare elementary school teachers to teach computing, CT, or coding. Although preservice and inservice training may be similar, the needs of preservice teachers differ from those of inservice teachers.

Preparing preservice teachers. Preservice teachers may be more comfortable with technology than many practicing teachers (Hargreaves & Fullan, 2000), but they lack experience, a particular teaching context, and advanced pedagogical knowledge (Ertmer & Ottenbreit-Leftwich, 2010). Numerous entities have described what preservice teachers should learn. In their *Framework for Understanding Teaching and Learning*, Bransford, Darling-Hammond, and LePage (2005) emphasize three areas of knowledge that preservice teachers must gain: (a) "knowledge of learners," (b) "knowledge of subject matter and curriculum goals," and (c) "knowledge of teaching," including pedagogy, differentiation, assessment, and classroom management (p. 11). The technological pedagogical content knowledge (TPACK) framework identifies three key, overlapping forms of knowledge used in technology integration: (a) content knowledge, (b) technological knowledge, and (c) pedagogical knowledge (Mishra & Koehler, 2006). The ISTE Standards for Computer Science Educators (ISTE, 2011) state that teachers must demonstrate content knowledge, effective teaching and learning strategies, and professional knowledge and skills, and create effective learning environments. Yadav, Stephenson, and Hong (2017) have argued that to prepare to teach CT, preservice K–12 teachers need deep understanding of both their content area and of CT, and that they should "learn to integrate

computational thinking into the context of particular subject areas" (p. 61). What these various frameworks share is a concern with both content and pedagogical knowledge.

Ertmer and Ottenbreit-Leftwich's (2010) recommendations, which we apply in this review, suggest that teacher preparation and development programs should promote change in practice by helping teachers gain relevant knowledge, increase self-efficacy, address pedagogical beliefs, and respond to school culture. Teacher educators can do so by providing instruction, models, practice, and opportunities for reflection (Ertmer & Ottenbreit-Leftwich, 2010).

Developing inservice teachers. Practicing teachers tend to have more advanced knowledge and entrenched beliefs than preservice teachers and operate within specific contexts and cultures. To promote teacher change, Ertmer and Ottenbreit-Leftwich (2010) have argued that PD programs should build on teachers' pedagogical content knowledge and "include information about how they can use these tools in very specific ways, within specific content domains, to increase student content learning outcomes" (p. 272). Studies suggest that effective PD is (a) school-based, (b) active, (c) of sufficient duration, (d) coherent, (e) collaborative, and (f) content-focused (Avalos, 2011; Darling-Hammond, Hyler, & Gardner, 2017; Desimone, 2009; Guskey & Yoon, 2009; Odden & Picus, 2014). Desimone's (2009) core conceptual framework suggests that PD studies be evaluated based on the following elements:

1. PD that adheres to principles of effective PD,
2. Changes in teacher knowledge, beliefs, and attitudes,
3. Improved instruction,
4. Improved student learning, and,
5. Context, including curriculum, policy, leadership, and teacher and student characteristics.

Kang, Cha, and Ha (2013) have provided substantial theoretical and empirical support for Desimone's (2009) framework, which has been widely applied and cited. In this paper we will apply Desimone's (2009) core conceptual framework as a lens to interpret inservice teacher studies.

Methods

The purpose of this literature review is to examine what research reveals about training K–6 teachers to teach CS and helping them to overcome their knowledge, attitude, and self-efficacy barriers. We completed a basic literature review in which we "summarize[d] and evaluate[d] the existing knowledge on a particular topic" (Machi & McEvoy, 2016, p. xiv). The lead author completed identification and analysis of studies in consultation with the second author. In studies of interventions to prepare elementary school teachers to teach CS, we asked:

1. How were preservice interventions aligned with elements of effective preservice preparation?
2. Were the PD interventions aligned with elements of effective PD?
3. What changes in knowledge, attitude, and beliefs were reported as a result of the training?

Using ERIC, ProQuest PAIS Index, Scopus, ACM Digital Library, Academic Search Premier, and Google Scholar, we searched for relevant studies using combinations of the following keywords and phrases:

- ("computational thinking" OR "robotics" OR "computing" OR "computer science") AND
- ("teacher training" OR "teacher education" OR "professional development" or "coaching" OR "mentoring") AND

- (“elementary” OR “primary” OR “k–5” OR “k–6” OR “k–3” or “4–6”) AND
- (“self-efficacy” OR “attitudes” OR “beliefs”).

From the initial 293 results, we included for further analysis empirical studies published in English-language academic journals, books, and conference papers, that included an intervention for training elementary school teachers to teach computational concepts or skills, and described changes in teachers’ knowledge of, attitudes toward, or self-efficacy for teaching CS. We also chose to focus on the 10-year period of 2008–2018, as approaches to preparing CS teachers more than 10 years ago may differ significantly from current needs and technologies. We eliminated articles that pertained only to (a) secondary school teachers, (b) administrator attitudes, (c) student perceptions, (d) student outcomes, or (e) teacher attitudes toward something other than teaching computing (e.g., gender). By scanning titles and abstracts, we narrowed the pool to 46 potential articles. In reading the 46 articles, we identified 15 articles that fit these criteria for further analysis. During the process of writing the paper, we identified six additional newly-published studies that matched our inclusion criteria.

After identifying relevant articles, we categorized articles based on whether they involved preservice or inservice teachers, then analyzed each study according to one of two frameworks: studies involving preservice teacher preparation were analyzed using the preservice teacher preparation framework in Figure 1; studies involving PD were analyzed using Desimone’s (2009) core conceptual framework (Figure 2).

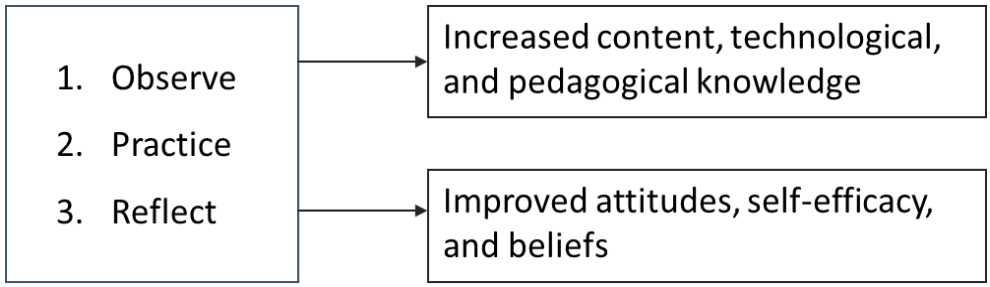

Figure 1. Preservice teacher preparation framework, based on Ertmer and Ottenbreit-Leftwich's (2010) recommendations for facilitating teacher change.

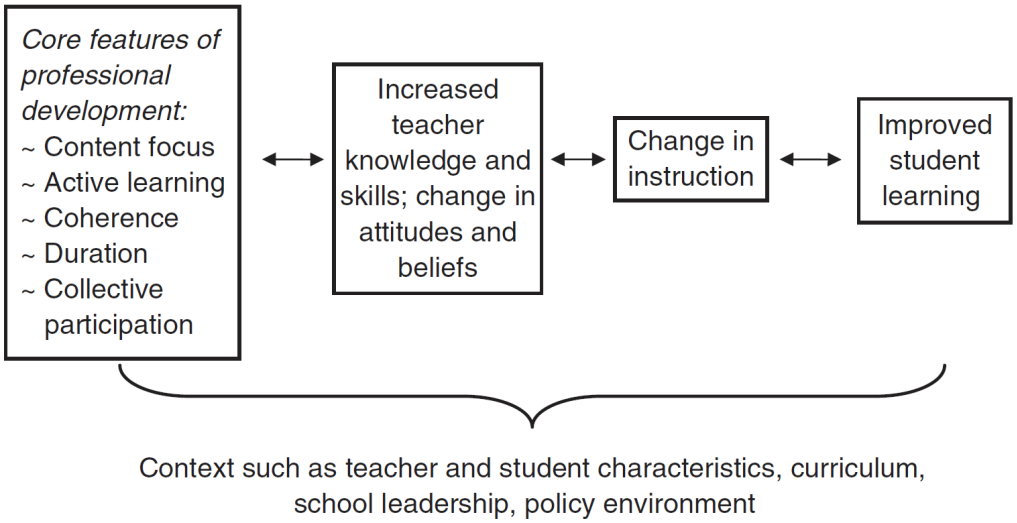

Figure 2. Core conceptual framework. From "Improving Impact Studies of Teachers' Professional Development: Toward Better Conceptualizations and Measures," by L. M. Desimone, 2009, Educational Researcher, 38, p. 185. Copyright 2009 by Sage Publications. Reprinted with permission.

The two frameworks are similar, but the preservice framework is simpler than Desimone's (2009) framework. Although Ertmer and Ottenbreit-Leftwich's (2010) recommendations may be applied to both preservice and inservice training, Desimone's (2009) framework allows for more detailed analysis by including additional elements relevant to PD that may not exist in preservice teacher preparation.

Findings

As noted, our search was narrowed to 21 studies about training K–6 teachers to teach CS and to overcome their knowledge, attitude, and self-efficacy barriers. This section includes

summaries of each study in terms of participants, context, focus, learning activities, duration and format, assessment instruments and data, learning outcomes, and attitude/belief outcomes (Tables 1 and 3), followed by analysis of the studies aligned with the research questions and frameworks described previously (Tables 2 and 4). We first discuss the 12 preservice teacher studies, followed by the nine inservice teacher studies.

Preservice Elementary CS Teacher Training

We provide a summary of the 12 preservice studies we identified in Table 1. Following the summaries is our analysis of the studies, centered around three main elements:

1. Instruction that includes models, practice, and reflection;
2. Increased content, technological, and pedagogical knowledge; and,
3. Improved attitudes, self-efficacy, and beliefs.

Table 1

Summaries of Studies Related to K–6 Preservice Teacher Training for Computing, Coding, and CT

Authors, year	Partici-pants	Context	Duration, format	Focus	Learning Activities	Assessment/data	Learning Outcomes	Attitude Outcomes
CS as Content								
Cetin, 2016	56 preservice, K–12 teachers	Program-ming languages course	30-hour, 6-week unit	Program-ming	Experimental group used Scratch to create games and animations; control group used C to solve programming problems	Achievement test; practice test; Computer Programming Attitude Scale (Cetin & Ozden, 2015); interviews	Students using Scratch performed significantly higher on parallel achievement and practice posttests than participants who used C	No significant difference between the two groups in attitude toward computer programming
Cetin & Andrews-Larson, 2016	58 preservice, K–12 teachers	Programmi ng languages course	10-hour unit	Computer program ming, sorting algorithm	Used Flash to construct visualizations, or animations, of sorting algorithms	Achievement test; Computer Programming Attitude Scale (Cetin & Ozden, 2015)	The experimental group scored significantly higher in achievement than the control group	No significant difference between the two groups in attitude toward computer programming
Jeon & Kim, 2017	110 preservice teachers	CT-based programmi ng course vs. ICT skill-based course	15-week, 45-hour course	Computer program ming, CT	Instruction; problem-based learning; constructing a responsive website	Pre/post computer learning attitude assessments (Lee, 2010)		Students in CT course had significantly higher gains in self-efficacy and attitude toward CS education
Ng, 2017	10 preservice, early	Early childhood education	Three workshop s	Coding, Bee-bots	Lecture, practice, modeling, lesson planning,	Learning package designed by students	Increased coding skills and ability to design coding	

Authors, year	Partici-pants	Context	Duration, format	Focus	Learning Activities	Assessment/data	Learning Outcomes	Attitude Outcomes
	childhood teachers	program, Hong Kong			microteaching, discussion, collaboration		activities aligned to learning theory	
Sadik et al., 2017	12 preservice teachers (11 K–6; 1 secondary)	Advanced computer education course	Several-week course unit	CT, coding, robotics	Collaboratively developed and taught a two-hour instructional activity	Student proposals, blog posts, video discussions, papers, and reflections	Increased understanding of CT, but misconceptions persisted	
Integrated CS								
Chang & Peterson, 2018	59 preservice, K–6 & special ed. teachers	Educational technology course	2-hour activity	CT, robotics, coding	Lecture, exploration, & sharing or reflection	Written reflection	Increased understanding of CT; teaching applications	Improved attitudes, perceptions of relevance
Jaipal-Jamani, 2018	36 preservice, K–8 teachers	Science methods course	Two 3-hour classes	Robotics, programming	Constructing and programming gears and LEGO WeDo robots	Pre/post assessments of interest, self-efficacy, and science content knowledge	Increased understanding of gears	Increased interest in robots & self-efficacy for teaching robotics
Jaipal-Jamani & Angeli, 2017	21 preservice, K–6 teachers	Science methods course	Two 3-hour classes	CT, robotics, programming	Modeling; constructing and programming LEGO WeDo robots	Pre/post assessments of interest, self-efficacy, and science content knowledge; programming activities	Increased knowledge of CT and gears	Increased interest in robots & self-efficacy for teaching robotics

Authors, year	Partici-pants	Context	Duration, format	Focus	Learning Activities	Assessment/data	Learning Outcomes	Attitude Outcomes
Kaya, Yesilyurt, Newley, & Deniz, 2018	35 preservice teachers	Undergrad science teaching methods course	Six 90-minute classes	CT, robotics, coding	Instruction; robotics challenge with Lego Mindstorms; programmed in code.org; solved Zoombinis video game puzzles	Modified Science Teaching Efficacy Belief Instrument (STEBI-B; Enochs & Riggs, 1990; STEM Learning and Research Center, n.d.)		Self-efficacy increased significantly; outcome expectancy did not
Kim et al., 2015	16 preservice, K–6 teachers	STEM instruction course	3-week course unit	Robot assembly, program ming	Assembled and programmed robots using My Robot Time and RoboRobo kits; developed lesson plans	Pre/post tests, surveys, interviews	No significant differences in pre and posttest scores for science, technology, engineering, or mathematical knowledge	Improved motivation, enjoyment, interest in science, and interest in engineering
Ma, Lai, Williams, Prejean, & Ford, 2008	32 preservice, K–6 teachers	Technolog y integration course	12-hour training	Robotics	Instruction, programming activities; practice facilitating activities with children; collaborative reflection	Reflective journal entries and interviews	Increased knowledge and skills to facilitate learning	
Yadav, Mayfield, Zhou, Hambrusc h, & Korb, 2014	357 preservice, K–12 teachers	Introductor y psychology course	Two 50-minute classes	CT	Instruction, problem-solving, examples	CT quiz, Computing Attitude Questionnaire	Greater understanding of CT and CT pedagogy than control group.	No statistically significant difference in comfort or interest.

Table 2

Reported Inputs and Outcomes in Preservice Studies

Authors, year	Inputs						Outputs		
	Content models	**Teaching models**	**Content practice**	**Lesson planning**	**Teaching practice**	**Reflection**	**Increased content knowledge**	**Increased pedagogical knowledge**	**Changed attitudes or beliefs**
CS as Content									
Cetin, 2016	X		X				X		
Cetin & Andrews-Larson, 2016	X		X				X		
Jeon & Kim, 2017			X						X
Ng, 2017		X	X	X	X		X	X	
Sadik et al., 2017				X	X	X	X		
Integrated CS									
Chang & Peterson, 2018		X	X			X	X	X	X
Jaipal-Jamani, 2018	X		X				X		X
Jaipal-Jamani & Angeli, 2017	X		X				X		X
Kaya et al., 2018			X						X
Kim et al., 2015			X	X					X
Ma et al., 2008		X			X	X	X	X	
Yadav et al., 2014	X	X	X	X		X	X	X	X

Elements of instruction. Although there are numerous elements of effective teacher preparation, Ertmer and Ottenbreit-Leftwich (2010) highlighted the value of observation, hands-on practice, and reflection. Table 2 summarizes which preservice studies reported each element.

Observation. One way that people learn is through observing examples or models (Bandura, 1977). To understand how to teach any content, preservice teachers need to see examples of effective teaching, and to understand CS it is helpful to see specific skills demonstrated (Ertmer & Ottenbreit-Leftwich, 2010). As shown in Table 2, of 12 preservice interventions in this review, only one included demonstrations of both CS concepts and teaching applications (Yadav et al., 2014); three other studies included examples of teaching with technology (Chang & Peterson, 2018; Ma et al., 2008; Ng, 2017); four included demonstration of coding, robotics, or CT but not examples of teaching children that content (Cetin, 2016; Cetin & Andrews, 2016; Jaipal-Jamani, 2018; Jaipal-Jamani & Angeli, 2017); four studies mentioned neither content nor pedagogical demonstrations (Jeon & Kim, 2017; Kaya et al., 2018; Kim et al., 2015; Sadik et al., 2017). It is possible, and perhaps likely, that the latter interventions included demonstrations of content or pedagogical skills but did not describe them in those terms. Additionally, one could argue that all the interventions that included instruction in coding, robotics, or CT inherently included models of teaching CS. However, because elementary students are very different developmentally from college students, we did not count college-level instruction as a model for primary-level instruction.

Practice. Preservice teachers need practice not only mastering new technological skills, but also teaching skills and concepts to others—ideally to children (e.g., Ertmer & Ottenbreit-Leftwich, 2010; Hammerness, Darling-Hammond, & Bransford, 2005). All preservice studies in this review included some form of practice. As shown in Table 2, nine interventions included

practice with coding, CT, or robotics, but no practice teaching the skills and concepts (Cetin, 2016; Cetin & Andrews, 2016; Chang & Peterson, 2018; Jaipal-Jamani, 2018; Jaipal-Jamani & Angeli, 2017; Jeon & Kim, 2017; Kaya et al., 2018; Kim et al., 2015; Yadav et al., 2014). One study included teaching practice without separate content practice (Sadik et al., 2017). Only two of twelve studies reported providing practice both in mastering CS skills or concepts and teaching those skills or concepts (Ma et al., 2008; Ng, 2017). In Ma et al.'s (2008) study, preservice teachers facilitated an activity with children; in Ng's (2017) study, preservice teachers taught each other. Overall, the 12 studies were much stronger in providing content practice than teaching practice.

Reflection. Reflecting on experiences and new information helps preservice teachers examine their pedagogical beliefs, make sense of their experiences, and assimilate or accommodate new knowledge and beliefs (Smagorinsky, Shelton, & Moore, 2015). Four preservice studies reported including reflection as part of their interventions (Chang & Peterson, 2018; Ma et al., 2008; Sadik et al., 2017; Yadav et al., 2014). In all four cases, written reflections were used as assessment data. It is possible that additional studies incorporated reflection but did not describe it as such.

Overall, the preservice studies in this review were moderately aligned with Ertmer and Ottenbreit-Leftwich's (2010) recommended practices for effectively preparing preservice teachers. Although almost all interventions included practice doing CS, only three studies included practice teaching CS. Only a third of interventions included demonstrations of how to teach such concepts to children, and only a third included reflection. Reported outcomes emphasized content over pedagogy.

Increased knowledge. A prime goal of teacher preparation is to increase teacher knowledge. As described previously, there are various ways to categorize teacher knowledge. Here we will use the three key types described by Mishra and Koehler (2006): content knowledge, technological knowledge, and pedagogical knowledge. However, in teaching CS content and skills, content and technology are often intertwined.

Content and technological knowledge. CS can be taught as its own content area or integrated into other subjects. The preservice studies in this review include both approaches. As shown in Tables 1 and 2, five studies treated CS as subject matter (Cetin, 2016; Cetin & Andrews-Larson, 2016; Jeon & Kim, 2017; Ng, 2017; Sadik et al., 2017). Although these interventions varied in characteristics, we highlight two trends. First, four of five included both primary and secondary teachers, suggesting that there is a fundamental set of knowledge that is essential to understanding CS content regardless of what level one will be teaching it at. Secondary teachers need to have a depth of CS knowledge beyond elementary teachers, but these studies all ensure that both elementary and secondary understanding fundamental concepts (e.g., sequences, commands, loops, conditionals). Second, four of five studies reported changes in content knowledge, but only one reported changes in pedagogical knowledge, and one reported changes in attitudes or beliefs. Thus, it would appear the main focus of many preservice studies is to build teachers' content knowledge first and foremost.

Seven studies focused on integrating computing or CT into other subjects (Chang & Peterson, 2018; Jaipal-Jamani, 2018; Jaipal-Jamani & Angeli, 2017; Kaya et al., 2018; Kim et al., 2015; Ma et al., 2008; Yadav et al., 2014). Four of these interventions were taught within the context of science or STEM methods courses (Jaipal-Jamani, 2018; Jaipal-Jamani & Angeli, 2017; Kaya et al., 2018; Kim et al., 2015). Other contexts included two technology integration

courses (Chang & Peterson, 2018; Ma et al., 2008) and one introductory psychology course (Yadav et al., 2014). Although there was a heavy focus on strengthening teachers' computing content knowledge, all but one integration study reported changes in attitudes or beliefs.

The studies in this review included training in coding, computing, CT, programming, robotics, or a combination thereof. As shown in Figure 3, 11 of 12 preservice interventions included coding or programming, eight included robotics, and six included CT. All six studies involving CT were published in 2017 or 2018, which suggests that studies on training preservice elementary CS teachers is a new and growing area of study. The significant overlaps among coding, CT, and robotics suggest that robots are frequently being used to teach coding or programming, and coding, programming, and robotics can effectively be used to teach CT.

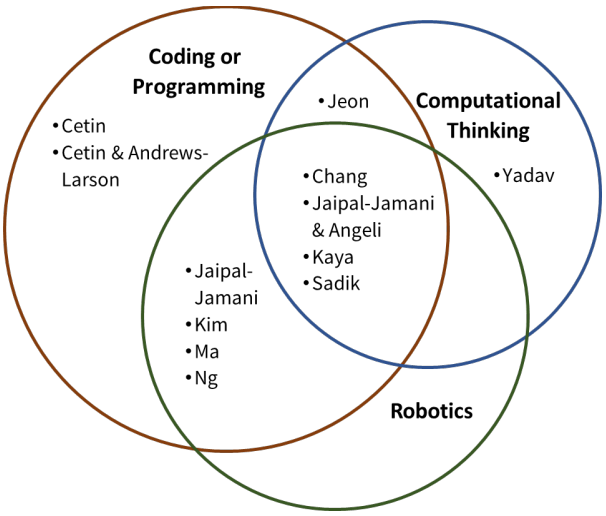

Figure 3. CS focus in preservice studies, listed by primary author. The relative size of circles reflects the relative number of related studies.

As shown in Table 2, nine of 12 preservice studies in this review reported increased knowledge of coding, robotics, or CT. Jeon and Kim (2017) and Kaya et al. (2018) did not assess changes in knowledge. Kim et al. (2015) assessed changes in STEM knowledge and found no

significant difference between pre and posttest scores following a three-week robotics unit in a STEM instruction course.

Pedagogical knowledge. Pedagogical knowledge is a key objective for preservice teacher development (Bransford et al., 2005; Ertmer & Ottenbreit-Leftwich, 2010; Mishra & Koehler, 2006). Pedagogical knowledge refers to knowledge of how to teach effectively, which incorporates knowledge of human development, learning and teaching theories, and classroom management. Pedagogical knowledge incorporates pedagogical content knowledge, or understanding how to teach specific subjects, and technological pedagogical knowledge, or knowing how to use technology effectively to teach. As shown in Figure 3, among 12 preservice studies in this review, only four reported increases in pedagogical knowledge following interventions (Chang et al., 2015; Ma et al., 2008; Ng, 2017; Yadav et al., 2014)—the same four studies that provided demonstrations of how to teach computing concepts and skills. This demonstrates once again an overall greater focus on developing preservice teachers' CS content knowledge than pedagogical knowledge.

Improved attitudes, self-efficacy, and beliefs. Besides increasing knowledge, preservice teacher preparation should improve self-efficacy and attitudes toward teaching coding, CT, or robotics. As shown in Table 2, seven studies reported changed attitudes or beliefs, including self-efficacy (Chang & Peterson, 2018; Jaipal-Jamani, 2018; Jaipal-Jamani & Angeli, 2017; Jeon & Kim, 2017; Kaya et al., 2018; Kim et al., 2015; Yadav et al., 2014). All but one of the integration studies reported changes in attitudes or beliefs, although only one non-integration study did so.

In all 12 studies, researchers observed improvements in teachers' knowledge, self-efficacy, or attitudes. In addition to the observed changes, several authors noted no significant

differences in certain assessed criteria. Cetin (2016) and Cetin and Andrews-Larson (2016) found that their programming training led to no significant change in preservice teachers' attitudes toward computer programming, although it did lead to an increase in knowledge. The lack of change could be explained by the fact that attitudes were very positive before the training. The authors also noted that the interventions were relatively brief (six weeks and three weeks). Similarly, Kaya et al. (2018) reported improved self-efficacy, but no change in outcome expectancy. Kim et al. (2015) found that among preservice teachers, training in robotics did not lead to increased interest in math, technology, and STEM careers, nor to significant increases in understanding of science, technology, engineering, or math. Furthermore, Yadav et al. (2014) found that preservice teachers given training in CT did not have higher comfort or interest in computing, compared with preservice teachers given alternate training in cognition, problem solving, and creativity. The intervention was two 50-minute sessions. In all five of these cases, the authors accounted for the lack of change in an area and observed changes in other areas. We would not expect brief interventions to motivate immediate or lasting changes to teachers' career plans nor significantly affect tests in broad STEM subjects. The problem with no observable difference in such studies may be more closely associated with the duration of the training than teachers' interest in CS.

In summary, research on preservice CS teacher training is emergent, with the majority of such studies having been published in just the past two years. The main focus of most of these studies has been to strengthen teachers' content knowledge and attitudes toward CS. The only studies that focused on developing teachers' pedagogical knowledge were those that followed Ertmer and Ottenbreti-Leftwich's (2010) recommendation of teacher observation. The duration of these studies has varied from as little as two 50-minutes sessions to an entire semester.

Despite these differences, results with preservice CS teacher training have been overwhelmingly positive, especially in regards to increasing teachers' knowledge of CS.

Inservice Training

Tables 3 and 4 summarize the nine studies involving inservice teacher PD. Following the summaries, we analyze the studies using Desimone's (2009) core conceptual framework for evaluating PD studies. The five main elements that we considered were:

1. effective professional development;
2. changes in knowledge, attitudes, and beliefs;
3. changes in instruction;
4. improved student learning; and,
5. context.

Table 3

Articles Related to K–6 Inservice Teacher Training for Computing, Coding, and CT

Authors, year	Participants	Context	Duration, format	Focus	Learning Activities	Assessment/data	Learning Outcomes	Attitude Outcomes
CS as Content								
Bers, Seddighin, & Sullivan, 2013	25 early childhood educators	Free institute held in Massachusetts; participants from 7 states	3-day workshop	Robotics & programming	Lecture; discussion; KIWI robotics sets and CHERP programming software; curriculum design	Pre/post questionnaires to assess attitudes, self-efficacy, and knowledge; interviews	Significant increases in technology, pedagogy, and content knowledge	Significant improvement in technology self-efficacy and attitudes toward technology
Leonard et al., 2018	45 K–9 teachers	Wyoming, online graduate course	8-week course	Robotics, game design, & culturally responsive pedagogy (CRP)	Readings & discussion; built & programmed LEGO MindStorms robots; designed games	Pre/post CT attitude survey (Yadav, Zhou, Mayfield, Hambrusch, & Korb, 2011); Dimensions of Success rating tool (Shah, Wylie, Gitomer, & Noam, 2018); games assessed with rating rubric; CRP survey	Small gains in CT understanding—higher improvements for teachers who did game design than for those who did robotics.	Small improvements in CT attitudes—greater effect for game design than robotics.
Marcelino, Pessoa, Vieira, Salvador, &	7 K–12 teachers, 1 other participant	University of Coimbra, Portugal;	54-hour course	Scratch programming, CT, pedagogy	Individual and collaborative programming	Activity & project evaluations; Dr. Scratch	Improved CT knowledge and programming skill, but learning depth varied	

Authors, year	Participants	Context	Duration, format	Focus	Learning Activities	Assessment/data	Learning Outcomes	Attitude Outcomes
Mendes, 2018		online course			activities and project		among participants.	
Roberts, Prottsman, & Gray, 2018	3,092 K–5 teachers and staff	University-driven workshops in Alabama and Indiana	1-day workshop	Computing, coding, CT	Teaching & observing using Code.org's Computer Science Fundamentals curriculum and additional CS content	Post-PD surveys provided by Code.org	Increased content and pedagogical knowledge	Improved self-efficacy, content knowledge, beliefs, and attitudes toward CS
Toikkanen & Leinonen, 2017	501 K–9 teachers	Finland, online course	2-month MOOC	Teaching programming, CT	Instruction; programming in ScratchJr, Scratch, or Racket; online discussion	Pedagogical ideas shared in Padlet	Increased knowledge and skills to teach programming	Teachers overcame reservations and preconceptions.
Integrated CS								
P. J. Rich et al., 2017	27 K–6 teachers	Title I school, western U.S.	1 year, weekly, embedded PD	Integrating computing and engineering	Engineering challenges, Engineering is Elementary curriculum, computing lessons and activities, Scratch programming	Survey of self-efficacy & beliefs; semi-structured interviews		Significantly more positive technology self-efficacy and beliefs toward computing than comparison group
Carter et al., 2014	53 fourth- and fifth-grade teachers	Southeastern U.S., large, urban school district	5-day workshop; embedded PD	Integrating computing	Instruction, modeling, and lesson plan creation	Survey of computing attitudes and anxiety		Amount of training correlated with decreases in anxiety and

Authors, year	Participants	Context	Duration, format	Focus	Learning Activities	Assessment/data	Learning Outcomes	Attitude Outcomes
								improvements in attitude.
Coleman, Gibson, Cotten, Howell-Moroney, & Stringer, 2016	54 fourth- and fifth-grade teachers	Southeastern U.S., large, urban school district	5-day workshop; embedded PD	Integrating computing	Instruction, modeling, lesson plan creation, practice teaching, supported classroom integration	Survey; in-class observation; rating scale for preparedness & execution (ability to teach lesson without assistance)	Summer institute participants scored higher in preparedness and execution than other teachers.	Attitude positively influenced execution. Attitude and anxiety showed no impact on preparation.
Hestness, Ketelhut, McGinnis, & Plane, 2018	13 Grades 3–5 mentor teachers	Mentor teachers from 3 Maryland public school districts	2 half-day workshops, 6 weeks apart	Integrating CT into classroom practice	Learned CT concepts; collaboratively completed robotics challenges with LEGO MindStorms, KIBO, & Think & Learn Code-a-Pillar; discussed integration	Drawings, written reflections, and focus group interviews	New content and pedagogical content knowledge were integrated with previous professional knowledge.	

Table 4

Reported Inputs and Outcomes in Inservice Studies

Authors, year	Inputs				Outputs		
	CS Practice	Lesson planning	Teaching practice	20+ hours	Increased content knowledge	Increased pedagogical knowledge	Changed attitudes or beliefs
CS as Content							
Bers et al., 2013	X	X		X	X	X	X
Leonard et al., 2018	X			X			X
Marcelino et al., 2018	X			X	X		
Roberts et al., 2018			X		X	X	X
Toikkanen & Leinonen, 2017	X			X	X	X	X
Integrated CS							
P. J. Rich et al., 2017	X			X			X
Carter et al., 2014		X		X			X
Coleman et al., 2016		X	X	X	X	X	X
Hestness et al., 2018	X				X	X	

Principles of effective PD. According to Desimone (2009), the five key features of effective PD are content focus, active learning, coherence, duration, and collective participation.

Content focus. PD with a specific content focus has had better outcomes for teacher learning than PD that lacks subject matter focus (Desimone, 2009). CS can be both taught as its own content area or integrated into other subjects; the studies in this review include both approaches. As indicated in Tables 3 and 4, five studies treated CS as subject matter (Bers et al., 2013; Leonard et al., 2018; Marcelino et al., 2018; Roberts et al., 2018; Toikkanen & Leinonen, 2017); four studies focused on integrating computing or CT into other subjects (Carter et al., 2014; Coleman et al., 2016; Hestness et al., 2018; P. J. Rich et al., 2017). There is considerable variation among the studies in each group. For example, of the two briefest interventions, one approached coding as its own subject (Roberts et al., 2018); the other involved integration of CT (Hestness et al., 2018). However, there are also some trends within groups. All three of the online interventions approached CS as discrete subject matter (Leonard et al., 2018; Marcelino et al., 2018; Toikkanen & Leinonen, 2017). All three interventions that included embedded PD involved integrating computing or CT into other subjects (Carter et al., 2014; Coleman et al., 2016; P. J. Rich et al., 2017).

The studies in this review included training in coding, computing, CT, programming, robotics, or a combination thereof. As shown in Figure 4, six of the nine inservice interventions included coding or programming. The five studies that included a focus on CT also involved computing, robotics, coding, or programming, which suggests that hands-on experiences can facilitate changes in elementary teacher attitudes or knowledge about CT. Four of the inservice interventions included computing, which was not the focus of any preservice studies.

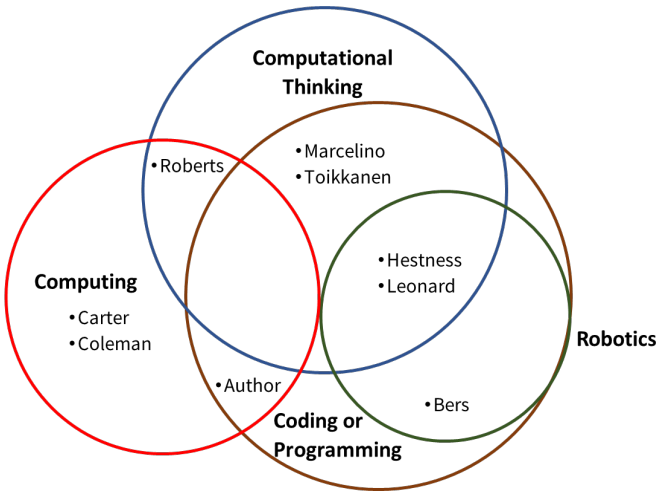

Figure 4. CS focus in inservice studies, listed by primary author. The relative size of circles reflects the relative number of related studies.

Active learning. Active learning has had better outcomes for teacher PD than passive learning (Desimone, 2009). Whereas listening to a lecture is passive, discussion, observation, and hands-on practice are active. As shown in Table 4, all nine PD interventions in this review included active learning. Six included programming activities or challenges (Bers et al., 2013; Hestness et al., 2018; Leonard et al., 2018; Marcelino et al., 2018; P. J. Rich et al., 2017; Toikkanen & Leinonen, 2017). Three interventions included lesson planning or curriculum design (Bers et al., 2013; Carter et al., 2014; Coleman et al., 2016). Despite this focus on an active approach, only two included practice teaching (Coleman et al., 2016; Roberts et al., 2018). As with the preservice interventions, the studies in this review focused more on developing teachers' CS content knowledge than providing pedagogical practice.

Coherence. Desimone (2009) suggested two key aspects to coherence: First, learning must cohere with teachers' knowledge and beliefs; second, PD content must be consistent with school, district, and state policy. Among the PD studies in this review, designing and evaluating the PD for coherence with teachers' beliefs and relevant policies did not seem to be a prominent

concern; authors did not explain their designs or methods in terms of coherence. Six of the nine studies addressed teacher beliefs of various types and to various extents (Bers et al., 2013; Coleman et al., 2016; Leonard et al., 2018; P. J. Rich et al., 2017; Roberts et al., 2018; Toikkanen & Leinonen, 2017). For example, Leonard et al. (2018) assessed teachers' beliefs regarding outcome expectancy and culturally responsive pedagogy, while Roberts et al. (2018) assessed teachers' self-efficacy and beliefs about CS. Only two studies addressed policy contexts (Hestness et al., 2018; P. J. Rich et al., 2017).

Duration. For PD to be effective, interventions must be of sufficient duration, over an appropriate span of time. Desimone (2009) suggested a general guideline of 20 hours or more of contact spread over a semester. Like the preservice interventions described previously, inservice training programs have varied in length (Table 3). Three studies included extensive, ongoing PD (Carter et al., 2014; Coleman et al., 2016; P. J. Rich et al., 2017). Three included summer workshops (Bers et al., 2013; Carter et al., 2014; Coleman et al., 2016). Two involved short-term inservice training (Hestness, et al., 2018; Roberts et al., 2018). Three studies involved distance inservice teacher training (Leonard et al., 2018; Marcelino et al., 2018; Toikkanen & Leinonen, 2017). All nine studies, whether brief or extended, reported positive changes in knowledge or attitude, suggesting that brief interventions may be sufficient to promote changes in knowledge, attitudes, or perceived self-efficacy.

Collective participation. Research has shown that PD tends to be more effective when teachers from the same school, grade, or department participate together (Desimone, 2009). Collective participation also suggests collaboration, and most or all interventions in this review incorporated collective participation in terms of collaboration or discussion. Most studies also included participants from the same school, grade, or department (Table 3). One intervention

involved all the teachers in a single school, working in grade-level teams (P. J. Rich et al., 2017). Four interventions targeted teachers from large geographic regions but a narrow band of grade levels (Bers et al., 2013; Carter et al., 2014; Coleman et al., 2016; Hestness et al., 2018). Four interventions, including three online, involved multiple schools and grade levels (Leonard et al., 2018; Marcelino et al., 2018; Roberts et al., 2018; Toikkanen & Leinonen, 2017). In the larger studies, there were enough participants for teachers to find colleagues teaching similar student populations. However, in the smallest study (Marcelino et al., 2018), only eight participants came from multiple countries and taught K–12, limiting the opportunity for collaborating with teachers from the same school or grade. The studies in this review suggest that collective participation has been a higher priority when integrating CS than when teaching CS as a subject. All four studies that involved integration included a narrow band of grade levels or a single school, whereas among the five interventions that treated CS as a discrete subject, four spanned several grade levels and large geographic regions.

Based on the features of PD described by Desimone (2009), the PD studies in this review were strong in providing active learning opportunities, but varied in their content focus, coherence, duration, and collective participation. Despite differences, all nine studies reported improvements.

Changes in knowledge, skills, attitudes, and beliefs. According to Desimone's (2009) framework, effective PD should promote "increased teacher knowledge and skills" and/or "changes in attitudes and beliefs" (p. 185). As shown in Table 4, seven studies reported improved attitudes or self-efficacy beliefs, six studies showed increased content knowledge, and five studies reported increased pedagogical knowledge. Only two studies assessed beliefs other than self-efficacy (P. J. Rich et al., 2017; Roberts et al., 2018). According to Pajares (1992),

beliefs are harder to change than knowledge. However, if we want to help teachers change their practice, then we need to either teach in ways that are consistent with their beliefs, or help teachers change their beliefs (Desimone, 2009; Ertmer & Ottenbreit-Leftwich, 2010; Pajares, 1992). As stated previously in the discussion of coherence, most of the studies in this review did not explicitly design their intervention to cohere to nor to change teachers' pedagogical beliefs.

In addition to the reported changes, two PD studies reported no significant differences for certain assessed factors. In the study by Coleman et al. (2016), teacher attitude and anxiety did not noticeably affect teacher preparedness to teach computing. It may have been that knowing they would be observed motivated teachers to prepare lessons that they may not have wanted to teach. Leonard et al. (2018) showed small gains in understanding and attitude, but those gains did not reach levels of significance, perhaps because the 45 participants were divided into multiple, small cohorts.

Changes in instruction and student learning. The third element in Desimone's (2009) path model is "change in instruction," followed by "improved student learning" (p. 185). Only one of nine PD studies in this review assessed classroom instruction. Coleman et al. (2016) found that participation in summer institutes positively influenced preparedness (having a plan with clear lesson objectives) and lesson execution (ability to teach lesson without assistance), attitude positively influenced execution, and neither attitude nor anxiety influenced preparation. Although P. J. Rich et al. (2017) and Toikkanen and Leinonen (2017) did not assess changes in instruction, they did note changes. Toikkanen and Leinonen (2017) observed that "the vast majority [of teachers] had no trouble with the tools and seeing their immediate benefits in their classrooms" (p. 246). Teachers in P. J. Rich et al.'s (2017) study were not required to implement their training, but researchers found that teachers who chose to do so had more positive attitudes

toward CS than teachers who did not. The remaining six studies did not address changes in practice, and none of the nine studies examined student learning. In Guskey's (2002) Model of Teacher Change, changes in teacher practice and student learning precede substantial changes in attitudes and beliefs: "The key element in significant change in teachers' attitudes and beliefs is clear evidence of improvement in the learning outcomes of their students" (pp. 383–384). If the ultimate purpose of PD is to improve student learning, then improving teacher knowledge and attitudes are a necessary, intermediate step on the path.

Context. Underlying the four elements of Desimone's (2009) model, context "operates . . . as an important mediator and moderator" (p. 185). Context includes "teacher and student characteristics, curriculum, school leadership, [and] policy environment" (p. 185). Numerous theorists and researchers have written on context and of the need for teacher educators to align PD to teachers' contexts (Desimone, 2009). Although each of the studies in this review includes a description of the research context, most interventions were provided to teachers who represented a wide range of contexts. As described previously, only one of the interventions (P. J. Rich et al., 2017) was provided to teachers from a single school. Tailoring training to context is difficult when teachers represent multiple grades, districts, and levels of experience. Policy context was at least a minor concern in two studies (Hestness et al., 2018; P. J. Rich et al., 2017). Leonard et al. (2018) addressed student characteristics by teaching robotics and game design within a framework of culturally responsive pedagogy. Teacher beliefs, primarily self-efficacy beliefs, were addressed in six studies (Bers et al., 2013; Coleman et al., 2016; Leonard et al.,

2018; Rich et al., 2017; Roberts et al., 2018; Toikkanen & Leinonen, 2017). Most of the studies addressed only one or two aspects of context.

In summary, the few published studies on training inservice elementary CS teachers have thus far yielded positive results. All studies reported increases in either content knowledge, teacher attitudes, or beliefs. Nearly all studies emphasized the development of content knowledge, but fewer than half reported on developing teachers' pedagogical content knowledge. The emphasis in these studies has been firmly fixed on teacher growth, with no studies reporting how students have changed or how changes in student knowledge have affected teacher growth in content knowledge, attitudes, or beliefs about CS.

Study Limitations

In this review we chose to focus on training for elementary school teachers, and therefore excluded sources related only to secondary teachers because the training and expertise of secondary school teachers differ substantially from the training and expertise of primary school teachers. However, several studies included in this review involved both primary and secondary teachers (Cetin, 2016; Cetin & Andrews-Larson, 2016; Jaipal-Jamani, 2018; Jeon & Kim, 2017; Kaya et al., 2018; Leonard et al., 2018; Marcelino et al., 2018; Sadik et al., 2017; Yadav et al., 2014). Therefore, the results from those studies may not be entirely relevant to elementary school teacher training. At the same time, we may have missed important contributions from studies involving only secondary teachers. Similarly, the studies in this review included interventions and outcomes related to overcoming barriers to teaching computing or coding. We are aware of papers related to teacher training for CT, computing, or coding, that did not fit our inclusion criteria because they lacked interventions or focused on outcomes other than changes in teacher attitude or content knowledge (e.g., Bower & Falkner, 2015; Francis et al., 2018; Israel et al.,

2018; Mannila, Nordén, & Pears, 2018; Rich et al., 2018; Yadav, Gretter, Good, & McLean, 2017). We also chose to focus on the 10-year period of 2008–2018, as approaches to teaching CS more than 10 years ago may not be relevant to current needs. Although a few relevant studies may have been published prior to 2008, only one of the studies in this review was published before 2013, and most were published between 2016–2018, demonstrating how nascent our understanding of elementary computing teacher training really is.

As with all research, we are also wary of the publication bias toward studies that show positive results. Because we chose not to search archives of unpublished studies, it is possible that there have been other studies on elementary teacher CS preparation that did not yield positive results. Finally, given the limited number of relevant studies, we did not apply quality exclusion criteria in our selection process; future reviewers may choose to be more selective. In particular, we included recent conference papers to provide the most up-to-date information; however, conference papers often lack the detail, and in some cases the rigor, of peer-reviewed journal articles.

Implications for Practice

At the start of this paper, we argued that elementary school teachers should teach their students CT and coding but lack the knowledge and confidence to do so. The studies we have reviewed indicate that training and PD can help elementary school teachers to overcome their knowledge, attitude, and efficacy barriers. They also suggest that effective interventions may vary but should provide opportunities to practice both doing CS and teaching CS.

All 21 studies in this review included active opportunities for practice, thus supporting a link between experience and improved self-efficacy (Ertmer & Ottenbreit-Leftwich, 2010; Mueller, Wood, Willoughby, Ross, & Specht, 2008; Somekh, 2008). However, although most

interventions included practice doing coding, robotics, or CT, few interventions included practice teaching. To prepare to teach CS concepts and skills, both preservice and inservice teachers need practice teaching those concepts in authentic contexts (Ertmer & Ottenbreit-Leftwich, 2010; Hammerness et al., 2005). Interventions that include CS but not teaching practice may be sufficient to influence changes in knowledge, attitudes, and self-efficacy but insufficient to support lasting changes in teacher practice and student learning (Desimone, 2009). Likewise, many of the studies in this review involved relatively brief interventions. Brief interventions may be appropriate for targeted skills training, but we recommend long-term, on-going, contextualized training for teachers who will be expected to teach coding, CT, computing, or robotics either as a discrete subject or integrated across the curriculum.

In summary, to help elementary school teachers overcome knowledge and efficacy barriers as they prepare to teach computing, both preservice and inservice teacher trainers should continue to provide elementary school teachers opportunities to practice CS. Teachers appear to appreciate more hands-on, practical coding experiences to develop their own content knowledge. In addition, preservice training should more often include modeling of CS and teaching skills; practice teaching CS to children; and reflection (Ertmer & Ottenbreit-Leftwich, 2010). Inservice teacher training has tended to have a strong CS focus and training, but should pay greater attention to coherence, pedagogical training, and context (Desimone, 2009; Ertmer & Ottenbreit-Leftwich, 2010; Guskey, 2002; P. J. Rich et al., 2017). Inservice training could be improved as teacher trainers address teacher beliefs, include lesson planning and implementation as components of training, and design and deliver instruction specific to school environment, culture, leadership, and teacher and student characteristics (Desimone, 2009; Ertmer & Ottenbreit-Leftwich, 2010).

Implications for Research

All 21 studies in the review focused on short-term outcomes, such as changes in self-efficacy and knowledge. Only three discussed how such changes might translate into improved teaching practice, and none assessed elementary student outcomes. This may be because measures of change in elementary students are also nascent and may not be widely known among professional developers. For example, there is only a single validated instrument that has been created to measure elementary students' CT (Román-González, Pérez-González, & Jiménez-Fernández, 2017; Román-González, Pérez-González, Moreno-León, & Robles, 2016). There are also only a few instruments aimed at assessing elementary student attitudes toward CS (Dorn & Tew, 2015; Gibbons, Hirsch, Kimmel, Rockland, & Bloom, 2004; Hoegh & Moskal, 2009). Although these instruments existed prior to the majority of studies reviewed in this paper, such instruments are not well known and have not been widely used by researchers to examine the effects of teaching elementary CS. As more researchers become familiar with these tools, we may see more studies that correlate teacher learning of CS with student growth or outcomes.

This review also suggests that there is little emphasis on how teacher training has affected their actual practice. Coleman et al. (2016) examined the influence of PD and attitude on preparedness and execution but did not show the influence of PD on attitude. The next step would be to show the influence of PD on both attitude and practice. Toikkanen and Leinonen (2017) included only a few findings but mentioned that "the vast majority [of teachers] had no trouble with the tools and seeing their immediate benefits in their classrooms" (p. 246). More data are needed to support their conclusion. P.J. Rich et al. (2017) found that self-efficacy varied widely by teacher background, willingness to experiment, and level of implementation, thus suggesting a relationship among experimentation, implementation, and self-efficacy. However,

P. J. Rich et al.'s findings could have been strengthened by including pre and post surveys, which might more accurately demonstrate teacher growth over time. More studies that examine the influence of PD on both teacher attitudes and practice are needed.

To understand effective preservice and inservice training for teaching CS, longitudinal studies are needed to show changes in attitudes, beliefs, knowledge, practice, and student outcomes over time. We recognize that word count limits and tenure and promotion requirements conspire to discourage such studies, yet unless we assess changes in instruction and student learning, we will not know whether interventions support lasting improvements. We recognize that at least six studies in this review were part of larger studies (Carter et al., 2014; Cetin, 2016; Cetin & Andrews, 2016; Coleman et al., 2016; Jaipal-Jamani, 2018; Jaipal-Jamani & Angeli, 2017), which may include longitudinal data collection and analysis. Thus, we are cautiously optimistic that we may begin to see reports of the effect of training on practice and student outcomes as we watch this research over the next several years.

Research that addresses teachers' contexts and pedagogical beliefs is recommended. Interventions should be designed to either cohere to or change teachers' pedagogical beliefs, and for that to happen, researchers must assess teachers' pedagogical beliefs. In addition, the contexts in which teachers work heavily influence teacher change and practice. We need intervention studies that examine how policy and culture influence teachers' CS attitudes, beliefs, and practices.

To summarize, the studies in this review show that training can improve elementary teachers' self-efficacy, attitudes, and knowledge for teaching CS. Insofar as this field is relatively new, more studies are needed, especially to understand how preservice and inservice training

influences elementary student outcomes; changes in teacher practice, both immediate and long-term; and interactions among context, training, and outcomes.

Conclusion

The purpose of this paper was to better understand the training of computing teachers in elementary education in the current era. In this review of literature, we found that (a) few studies have been published about training elementary school teachers to teach computing, coding, and programming, with slightly more studies on preservice teacher training than inservice PD; (b) interventions have focused more on developing elementary CS teachers' content knowledge than pedagogical knowledge; (c) studies overwhelmingly showed that training can improve teachers' self-efficacy, attitudes, and knowledge, even over relatively short interventions; and (d) the literature has said little about whether or to what extent changes in self-efficacy, attitudes, and knowledge lead to changes in actual practice or improved student learning.

Our research questions focused on examining the current state of training for both preservice and inservice computing teachers. In addition, we asked to what extent these studies reported on changes in teachers' knowledge, attitudes, and beliefs about computing. We analyzed preservice teacher preparation using Ertmer and Ottenbreit-Leftwich's (2010) framework for PD, which proposes that PD that includes modeling, practice, and reflection helps teachers to gain relevant knowledge and increase self-efficacy. We found that most preservice studies included opportunities to practice CS and reported increases in preservice teachers' content knowledge and computing self-efficacy. This is especially encouraging, considering that most reported interventions were relatively short.

To analyze inservice teacher PD, we used Desimone's (2009) framework for improving the impact of teachers' PD. This framework adds coherence, collective participation, and

sufficient duration as core features of effective PD, and indicates that there should be increased knowledge/skills and positive changes in attitudes and beliefs. As with preservice CS teacher development, we found that even short-term PD can increase teachers' self-efficacy to teach computing. Many efforts included hands-on learning, with teachers creating their own programs and experimenting with robots. Developing teachers' computing content knowledge was the focus of most of these studies. In the few studies where teachers were able to apply their lessons, they reported greater growth in both their knowledge/skills and their attitudes and beliefs about computing education.

Desimone's (2009) framework further suggests that effective PD produces changes in teacher practice that promote student learning. To that end, improving teacher knowledge and self-efficacy is essential but insufficient. Teachers may understand and feel confident to teach CS, but lack motivation to do so. Teachers are under pressure to align their teaching with state standards and assessments; therefore, unless students are being assessed in CS, coherent CS standards are developed, or parents and students are demanding CS instruction, teachers may not take time to teach CS. School districts and state governments are increasingly adopting policies that require CS instruction (Code.org, 2019; Stanton et al., 2017). Such policies are driving teacher change, are needed to support teacher change, and are themselves supported by teacher education in CS.

References

Avalos, B. (2011). Teacher professional development in *Teaching and Teacher Education* over ten years. *Teaching and Teacher Education*, *27*, 10–20. https://doi.org/10.1016/j.tate.2010.08.007

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, *84*, 191–215. https://doi.org/10.1037/0033-295X.84.2.191

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? *ACM Inroads*, *2*(1), 48–54. https://doi.org/10.1145/1929887.1929905

Bers, M. U., Seddighin, S., & Sullivan, A. (2013). Ready for robotics: Bringing together the T and E of STEM in early childhood teacher education. *Journal of Technology and Teacher Education*, *21*, 355–377.

Borko, H., & Putnam, R. T. (1995). Expanding a teacher's knowledge base: A cognitive psychological perspective on professional development. In T. R. Guskey & M. Huberman (Eds.), *Professional development in education: New paradigms and practices* (pp. 35–65). New York, NY: Teachers College Press.

Bower, M., & Falkner, K. (2015). Computational thinking, the notional machine, pre-service teachers, and research opportunities. In D. D'Souza & K. Falkner (Eds.), *Proceedings of the 17th Australasian Computing Education Conference* (pp. 37–46). Sydney, Australia: Australian Computer Society.

Bransford, J., Darling-Hammond, L., & LePage, P. (2005). Introduction. In L. Darling-Hammond & J. Bransford (Eds), *Preparing teachers for a changing world: What teachers should learn and be able to do* (pp. 1–39). San Francisco, CA: Jossey-Bass.

Bureau of Labor Statistics. (2018a). Computer and information technology occupations. Retrieved from https://www.bls.gov/ooh/computer-and-information-technology/home.htm

Bureau of Labor Statistics. (2018b). Occupations with the most job growth. Retrieved from https://www.bls.gov/emp/tables/occupations-most-job-growth.htm

Carter, A., Cotten, S. R., Gibson, P., O'Neal, L. J., Simoni, Z., Stringer, K., & Watkins, L. S. (2014). Integrating computing across the curriculum: Incorporating technology into STEM education. In Z. Yang, H. H. Yang, D. Wu, & S. Liu (Eds.), *Transforming K–12 classrooms with digital technology* (pp. 165–192). Hershey, PA: Information Science Reference. https://doi.org/10.4018/978-1-4666-4538-7.ch009

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

Cetin, I., & Andrews-Larson, C. (2016). Learning sorting algorithms through visualization construction. *Computer Science Education*, *26*, 27–43. https://doi.org/10.1080/08993408.2016.1160664

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

Clements, D. H., & Gullo, D. F. (1984). Effects of computer programming on young children's cognition. *Journal of Educational Psychology*, *76*, 1051–1058. https://doi.org/10.1037/0022-0663.76.6.1051

Code.org. (2019). K–12 computer science policy and implementation in states. Retrieved from Google Docs: https://docs.google.com/document/d/1J3TbEQt3SmIWuha7ooBPvlWpiK-pNVIV5uuQEzNzdkE

Coleman, L. O., Gibson, P., Cotten, S. R., Howell-Moroney, M., & Stringer, K. (2016). Integrating computing across the curriculum: The impact of internal barriers and training intensity on computer integration in the elementary school classroom. *Journal of Educational Computing Research*, *54*, 275–294. https://doi.org/10.1177/0735633115616645

Crick, T. (2017, April). *Computing education: An overview of research in the field*. Retrieved from Swansea University's Research Repository website: https://cronfa.swan.ac.uk/Record/cronfa43589

Darling-Hammond, L., Hyler, M. E., & Gardner, M. (2017, June). *Effective teacher professional development*. Retrieved from Learning Policy Institute website: https://learningpolicyinstitute.org/product/effective-teacher-professional-development-report

Department for Education. (2013, September 11). Statutory guidance: National curriculum in England: Computing programmes of study. Retrieved from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

Desimone, L. M. (2009). Improving impact studies of teachers' professional development: Toward better conceptualizations and measures. *Educational Researcher*, *38*, 181–199. https://doi.org/10.3102/0013189X08331140

Desimone, L. M., Porter, A. C., Garet, M. S., Yoon, K. S., & Birman, B. F. (2002). Effects of professional development on teachers' instruction: Results from a three-year longitudinal study. *Educational Evaluation and Policy Analysis*, *24*, 81–112. https://doi.org/10.3102/01623737024002081

Dorn, B., & Tew, A. E. (2015). Empirical validation and application of the Computing Attitudes Survey. *Computer Science Education*, *25*, 1–36. https://doi.org/10.1080/08993408.2015.1014142

Enochs, L. G., & Riggs, I. M. (1990). Further development of an elementary science teaching efficacy belief instrument: A preservice elementary scale. *School Science and Mathematics*, *90*, 694–706. https://doi.org/10.1111/j.1949-8594.1990.tb12048.x

Ertmer, P. A., & Ottenbreit-Leftwich, A. T. (2010). Teacher technology change: How knowledge, confidence, beliefs, and culture intersect. *Journal of Research on Technology in Education*, *42*, 255–284. https://doi.org/10.1080/15391523.2010.10782551

Ertmer, P. A., Ottenbreit-Leftwich, A. T., Sadik, O., Sendurur, E., & Sendurur, P. (2012). Teacher beliefs and technology integration practices: A critical relationship. *Computers & Education*, *59*, 423–435. https://doi.org/10.1016/j.compedu.2012.02.001

Ertmer, P. A., Ottenbreit-Leftwich, A. T., & York, C. S. (2012). Exemplary technology-using teachers: Perceptions of factors influencing success. *Journal of Computing in Teacher Education*, *23*, 55–61.

Francis, K., Yáñez, G. A., Chapman, O., Cherkowski, G., Dodsworth, D., Friesen, S., . . . & Turner, J. (2018). Forming and transforming STEM teacher education: A follow up to Pioneering STEM education. In *Proceedings of 2018 IEEE Global Engineering Education Conference* (pp. 686–693). Piscataway, NJ: IEEE.

Garneli, V., Giannakos, M. N., & Chorianopoulos, K. (2015). Computing education in K–12 schools: A review of the literature. In *Proceedings of 2015 IEEE Global Engineering Education Conference* (pp. 543–551). Piscataway, NJ: IEEE.

Gibbons, S. J., Hirsch, L. S., Kimmel, H., Rockland, R., & Bloom, J. (2004, August). *Middle school students' attitudes to and knowledge about engineering*. Paper presented at the International Conference on Engineering Education, Gainesville, FL.

Google, Inc., & Gallup, Inc. (2016). *Trends in the state of computer science in U.S. K–12 schools*. Retrieved from http://services.google.com/fh/files/misc/trends-in-the-state-of-computer-science-report.pdf

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

Guskey, T. R. (2002). Professional development and teacher change. *Teachers and Teaching: Theory and Practice*, *8*, 381–391. https://doi.org/10.1080/135406002100000512

Guskey, T. R., & Yoon, K. S. (2009). What works in professional development? *Phi Delta Kappan*, *90*, 495–500. https://doi.org/10.1177/003172170909000709

Hammerness, K., Darling-Hammond, L., & Bransford, J. (with Berliner, D., Cochran-Smith, M., McDonald, M., & Zeichner, K.). (2005). How teachers learn and develop. In L. Darling-Hammond & J. Bransford (Eds.), *Preparing teachers for a changing world: What teachers should learn and be able to do* (pp. 358–389). San Francisco, CA: Jossey-Bass.

Hargreaves, A., & Fullan, M. (2000). Mentoring in the new millennium. *Theory Into Practice*, *39*, 50–56. https://doi.org/10.1207/s15430421tip3901_8

Heintz, F., Mannila, L., & Färngvist, T. (2016). A review of models for introducing computational thinking, computer science and computing in K–12 education. In *2016 IEEE Frontiers in Education Conference Proceedings* (pp. 1–9). Piscataway, NJ: IEEE.

Hestness, E., Ketelhut, D. J., McGinnis, J. R., & Plane, J. (2018). Professional knowledge building within an elementary teacher professional development experience on computational thinking in science education. *Journal of Technology and Teacher Education*, *26*, 411–435.

Hew, K. F., & Brush, T. (2007). Integrating technology into K–12 teaching and learning: Current knowledge gaps and recommendations for future research. *Educational Technology Research and Development*, *55*, 223–252. https://doi.org/10.1007/s11423-006-9022-5

Hoegh, A., & Moskal, B. M. (2009). Examining science and engineering students' attitudes toward computer science. In *39th Annual IEEE Frontiers in Education Conference Proceedings* (pp. 1–6). Piscataway, NJ: IEEE.

Holden, H., & Rada, R. (2011). Understanding the influence of perceived usability and technology self-efficacy on teachers' technology acceptance. *Journal of Research on Technology in Education*, *43*, 343–367. https://doi.org/10.1080/15391523.2011.10782576

Ilic, U., Haseski, H. I., & Tugtekin, U. (2018). Publication trends over 10 years of computational thinking research. *Contemporary Educational Technology*, *9*, 131–153. https://doi.org/10.30935/cet.414798

The Institute for the Advancement of Research in Education at AEL. (2004, April). *Review of the research: Nine components of effective professional development*. Retrieved from Texas Instruments' Educational and Productivity Solutions Division website: https://education.ti.com/sites/us/downloads/pdf/research_iare_ael.pdf

International Society for Technology in Education. (2011). *ISTE standards for computer science educators.* Retrieved from https://www.iste.org/standards/for-computer-science-educators

International Society for Technology in Education. (2017). *ISTE standards for educators.* Retrieved from https://www.iste.org/standards/for-educators

Israel, M., Ray, M. J., Maa, W. C., Jeong, G. K., Lee, C. e., Lash, T., & Do, V. (2018). School-embedded and district-wide instructional coaching in K–8 computer science: Implications for including students with disabilities. *Journal of Technology and Teacher Education*, *26*, 471–501.

Jaipal-Jamani, K. (2018). Developing pre-service teachers' self-efficacy and science knowledge through a scaffolded robotics intervention: A longitudinal study. In E. Langran & J. Borup (Eds.), *Proceedings of SITE 2018: Society for Information Technology & Teacher Education International Conference* (pp. 1913–1919). Waynesville, NC: Association for the Advancement of Computing in Education.

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. *Journal of Science Education and Technology*, *26*, 175–192. https://doi.org/10.1007/s10956-016-9663-z

Jeon, Y., & Kim, T. (2017). The effects of the computational thinking-based programming class on the computer learning attitude of non-major students in the teacher training college. *Journal of Theoretical and Applied Information Technology*, *95*, 4330–4339.

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

Kallia, M. (2017, November). *Assessment in computer science courses: A literature review*. Retrieved from King's College London website:

https://royalsociety.org/~/media/policy/projects/computing-education/assessment-literature-review.pdf

Kang, H. S., Cha, J., & Ha, B.-W. (2013). What should we consider in teachers' professional development impact studies? Based on the conceptual framework of Desimone. *Creative Education*, *4*(4A), 11–18. https://doi.org/10.4236/ce.2013.44A003

Kaya, E., Yesilyurt, M. E., Newley, A. D., & Deniz, H. (2018). *Investigating computational thinking self-efficacy beliefs of pre-service elementary teachers*. Poster session presented at the 2018 ASEE Annual Conference and Exposition, Salt Lake City, UT.

Kim, C., Kim, D., Yuan, J., Hill, R. B., Doshi, P., & Thai, C. N. (2015). Robotics to promote elementary education preservice teachers' STEM engagement, learning, and teaching. *Computers & Education*, *91*, 14–31. https://doi.org/10.1016/j.compedu.2015.08.005

Kundukulam, V. (2018, January 23). 4 ways to truly expand #CSforAll. *EdSurge*. Retrieved from https://www.edsurge.com/news/2018-01-23-4-ways-to-truly-expand-csforall

Lawless, K. A., & Pellegrino, J. W. (2007). Professional development in integrating technology into teaching and learning: Knowns, unknowns, and ways to pursue better questions and answers. *Review of Educational Research*, *77*, 575–614. https://doi.org/10.3102/0034654307309921

Lee, I. (2016). Reclaiming the roots of CT. *CSTA Voice: The Voice of K–12 Computer Science Education and Its Educators*, *12*(1), 3–4.

Lee, J., (2010). *The effects of girl-friendly teaching-learning program on software learning attitude and achievement* (Master's thesis). Korea National University of Education, Cheongju, South Korea.

Leonard, J., Mitchell, M., Barnes-Johnson, J., Unertl, A., Outka-Hill, J., Robinson, R., & Hester-Croff, C. (2018). Preparing teachers to engage rural students in computational thinking through robotics, game design, and culturally responsive teaching. *Journal of Teacher Education*, 69, 386–407. https://doi.org/10.1177/0022487117732317

Lockwood, J., & Mooney, A. (2018). Computational thinking in secondary education; Where does it fit? A systematic literary review. *International Journal of Computer Science Educational in Schools*, *2*(1), 41–60. https://doi.org/10.21585/ijcses.v2i1.26

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

Ma, Y., Lai, G., Williams, D., Prejean, L., & Ford, M. J. (2008). Exploring the effectiveness of a field experience program in a pedagogical laboratory: The experience of teacher candidates. *Journal of Technology and Teacher Education*, *16*, 411–432.

Machi, L. A., & McEvoy, B. T. (2016). *The literature review: Six steps to success* (3rd ed.). Thousand Oaks, CA: Corwin Press.

Mannila, L., Nordén, L.-Å., & Pears, A. (2018). Digital competence, teacher self-efficacy and training needs. In *Proceedings of the 49th ACM Conference on International Computing Education Research* (pp. 78–85). https://doi.org/10.1145/3230977.3230993

Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., & Mendes, A. J. (2018). Learning computational thinking and scratch at distance. *Computers in Human Behavior*, *80*, 470–477. https://doi.org/10.1016/j.chb.2017.09.025

Mishra, P., & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for teacher knowledge. *Teachers College Record*, *108*, 1017–1054.

Mueller, J., Wood, E., Willoughby, T., Ross, C., & Specht, J. (2008). Identifying discriminating variables between teachers who fully integrate computers and teachers with limited integration. *Computers & Education*, *51*, 1523–1537. https://doi.org/10.1016/j.compedu.2008.02.003

New South Wales Education Standards Authority. (2017). *Quality of initial teacher education in NSW: Digital literacy skills and learning report: A report on teaching information and communication technologies in initial teacher education in NSW.* Sydney, Australia: NSW Education Standards Authority.

Ng, W. S. (2017). Coding education for kids: What to learn? How to prepare teachers? In L. Morris & C. Tsolakidis (Eds.), *Proceedings of ICICTE 2017: The International Conference on Information Communication Technologies in Education* (pp. 195–205). Rhodes, Greece: Southampton Solent University.

Odden, A. R., & Picus, L. O. (2014). *School finance: A policy perspective* (5th ed.). New York, NY: McGraw Hill.

Pajares, M. F. (1992). Teachers' beliefs and educational research: Cleaning up a messy construct. *Review of Educational Research*, *62*, 307–332. https://doi.org/10.3102/00346543062003307

Rich, K., Strickland, C., & Franklin, D. (2017). A literature review through the lens of computer science learning goals theorized and explored in research. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 495–500). https://doi.org/10.1145/3017680.3017772

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

Rich, P. J., & Hodges, C. (Eds) (2017). *Emerging research, practice, and policy on computational thinking*. Cham, Switzerland: Springer. doi: 10.1007/978-3-319-52691-1

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), 1-20.

Rich, P. J., Leatham, K. R., & Wright, G. A. (2013). Convergent cognition. *Instructional Science*, *41*(2), 431-453. doi:10.1007/s11251-012-9240-7?LI=true#page-1

Roberts, M., Prottsman, K., & Gray, J. (2018). Priming the pump: Reflections on training K–5 teachers in computer science. In *Proceedings of the 49th ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 723–728). https://doi.org/10.1145/3159450.3159560

Román-González, M., Pérez-González, J.-C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior*, *72*, 678–691. https://doi.org/10.1016/j.chb.2016.08.047

Román-González, M., Pérez-González, J.-C., Moreno-León, J., & Robles, G. (2016). Does computational thinking correlate with personality? The non-cognitive side of computational thinking. In *Proceedings of the Fourth International Conference on*

Technological Ecosystems for Enhancing Multiculturality (pp. 51–58). https://doi.org/10.1145/3012430.3012496

Sadik, O., Ottenbreit-Leftwich, A., & Nadiruzzaman, H. (2017). Computational thinking conceptions and misconceptions: Progression of preservice teacher thinking during computer science lesson planning. In P. J. Rich & C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 221–238). https://doi.org/10.1007/978-3-319-52691-1_14

Schanzer, E., Fisler, K., & Krishnamurti, S. (2018). Assessing *Bootstrap:Algebra* students on scaffolded and unscaffolded word problems. In *Proceedings of the 49th ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 8–13). https://doi.org/10.1145/3159450.3159498

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2018). Technology and the mind: Does learning to code improve cognitive skills? In *Proceedings of the Technology, Mind, & Society 2018 Conference*. https://doi.org/10.1145/3183654.3183658

Shah, A. M., Wylie, C., Gitomer, D., & Noam, G. (2018). Improving STEM program quality in out-of-school-time: Tool development and validation. *Science Education*, *102*, 238–259. https://doi.org/10.1002/sce.21327

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

Smagorinsky, P., Shelton, S. A., & Moore, C. (2015). The role of reflection in developing eupraxis in learning to teach English. *Pedagogies: An International Journal*, *10*, 285–308. https://doi.org/10.1080/1554480X.2015.1067146

Somekh, B. (2008). Factors affecting teachers' pedagogical adoption of ICT. In J. Voogt & G. Knezek (Eds.), *International handbook of information technology in primary and secondary education* (pp. 449–460). https://doi.org/10.1007/978-0-387-73315-9_27

Stanton, J., Goldsmith, L., Adrion, W. R., Dunton, S., Hendrickson, K. A., Peterfreund, A., . . . Zinth, J. D. (2017). *State of the states landscape report: State-level policies supporting equitable K–12 computer science education*. Retrieved from Education Development Center website: https://www.edc.org/state-states-landscape-report-state-level-policies-supporting-equitable-k-12-computer-science

STEM Learning and Research Center. (n.d.). Instruments: Science Teaching Efficacy Belief Instrument (STEBI). Retrieved from Education Development Center website: http://stelar.edc.org/instruments/science-teaching-efficacy-belief-instrument-stebi

Toikkanen, T., & Leinonen, T. (2017). The Code ABC MOOC: Experiences from a coding and computational thinking MOOC for Finnish primary school teachers. In P. J. Rich & C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 239–248). https://doi.org/10.1007/978-3-319-52691-1_15

Tucker, A. (Ed.), Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2003, October). *A model curriculum for K–12 computer science: Final report of the ACM K–12 task force curriculum committee*. New York, NY: Association for Computing Machinery.

Vannatta, R. A., & Fordham, N. (2004). Teacher dispositions as predictors of classroom technology use. *Journal of Research on Technology in Education*, *36*, 253–271.

Waite, J. (2017). *Pedagogy in teaching computer science in schools: A literature review*. Retrieved from King's College London website:

https://royalsociety.org/~/media/policy/projects/computing-education/literature-review-pedagogy-in-teaching.pdf

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Computational thinking in teacher education. In P. J. Rich & C. B. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking* (pp. 205–220). https://doi.org/10.1007/978-3-319-52691-1_13

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

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011). Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 465–470). https://doi.org/10.1145/1953163.1953297

ARTICLE 2

Professional Development for Teaching Computational Thinking with Robots: An Exploratory Study

Stacie L. Mason

Richard E. West

Heather Leary

Brigham Young University

Abstract

This paper describes a professional development program for preparing K-6 teachers to teach computational thinking using robots, which was shown to increase teacher technology self-efficacy. Five K–6 teachers participated in an exploratory qualitative study examining learning objectives, barriers to implementation, teacher technology self-efficacy, teacher beliefs, and professional development preferences. The findings support theories linking experience to self-efficacy, and indicate that the short-term, targeted intervention provided teachers the skills and confidence to implement new technology.

Keywords: professional development (PD), computational thinking (CT), robotics, self-efficacy, teacher technology self-efficacy

Professional Development for Teaching Computational Thinking with Robots: An Exploratory Study

As technological literacy has become increasingly needed and valued, computing skills have been increasingly taught in school (Rich, Jones, Belikov, Yoshikawa, & Perkins, 2017). One aspect of computer science gaining traction in K-12 education is computational thinking. Put simply, computational thinking is "thinking like a computer scientist" (Wing, 2006, p. 35). It is an approach to problem solving that involves decomposition, pattern recognition, abstraction, algorithms, and analysis (Google, n.d). K–12 standards developed by the Computer Science Teachers Association (CSTA, 2017) and International Society for Technology in Education (ISTE, 2018) include computational thinking skills such as breaking problems into parts, creating algorithms, and developing models to solve problems.

Teaching computational thinking through computer science in K-12 environments has been an important topic for decades. Papert (1980) believed that computers could strongly influence how children learn to think, particularly when they are involved in programming the computers. In his work, he emphasized teaching children Logo programming as a constructionist method for tinkering and learning mathematical reasoning. In particular, Papert (1993) believed we should "rethink" schools so that computers were not an isolated subject to be taught, but a way of thinking and a set of tools used to learn various subjects.

Since Papert's work, there have been many studies on how best to achieve Papert's vision of integrating computational thinking instruction into educational settings. Lye and Koh (2014) argued there were three dimensions to computational thinking (computational concepts, practices, and perspectives) and analyzed 27 different intervention studies to determine how these dimensions had been incorporated into the curriculum. They found most of the studies to

be related to teaching computational concepts, not practices. In addition, they found very few to be focused on K-12 environments (only 9 of the 27 studies). These nine studies discussed interventions that focused primarily on English and mathematics (in addition to computer science specifically), and most of these interventions were through optional afterschool practices. Their conclusion was that there remained a large gap in research about K–12 computational thinking, particularly in developing practices and perspectives in addition to conceptual understanding.

One strategy that could be particularly promising for this demographic is to teach computational thinking via robots (Leonard et al., 2017). This strategy has been shown effective even with children as young as kindergarten (Bers, Flannery, Kazakoff, & Sullivan, 2014) and has jump started a whole industry of programmable robotic toys for children. With programmable robots and connected apps, students can develop the conceptual practices of computational thinking that Lye & Koh (2014) referenced, using a tinkering/constructionist strategy that would have been familiar to Papert. However, as is often the case with emerging technologies, the first challenge is in how to help develop the understanding and skills in the teachers, so that they can integrate the instruction into their curriculum for the students.

Literature Review

We developed our training program based on principles of effective teacher professional development, to overcome barriers to technology use by teachers.

Professional Development

Based on numerous professional development (PD) studies, experts have described effective PD as school-based, active, of sufficient duration, coherent, collaborative, and content-focused (Avalos, 2011; Darling-Hammond, Hyler, & Gardner, 2017; Desimone, 2009; Guskey & Yoon, 2009). In their discussion of the literature, Ertmer and Ottenbreit-Leftwich (2010)

suggested that to use technology as "meaningful pedagogical tools," teachers need to know how to use technology to facilitate student-centered instruction (p. 255). Professional development programs should build on teachers' pedagogical content knowledge (PCK) and "include information about how they can use these tools in very specific ways, within specific content domains, to increase student content learning outcomes" (p. 272).

Relatively few studies have focused on preparing elementary school teachers to teach computing. In a review of 21 studies related to teacher training for teaching elementary computing (Mason & Rich, in press), 14 studies showed improvements in self-efficacy or attitudes toward teaching computer science, and 15 studies showed evidence of increased knowledge, understanding, or performance. Both limited and extensive trainings yielded results. For example, Jaipal-Jamani and Angeli (2017) found that after completing only six hours of robotics instruction and activities, preservice elementary school teachers' STEM engagement increased significantly. And in a study by Rich et al. (2017), elementary teachers who participated in yearlong professional development reported significantly higher self-efficacy and more positive beliefs toward computing and engineering compared to teachers who had not participated in PD, though self-efficacy varied widely by teacher background, level of implementation, and willingness to experiment. Thus, there is considerable promise that professional development focused specifically on developing computational thinking awareness and skill could improve the ability of teachers to teach these skills in the classroom.

Barriers

Despite the potential to develop computational thinking skills in teachers, they often experience barriers to implementing technology. West and Davies (2014) explained that the first major barrier facing teachers integrating technology was one of access to the tools, both for

themselves and their students. In our project we addressed this first barrier by not only providing robots, but also delivering them to teachers. Our training sought to overcome the second and third barriers described by West and Davies by helping teachers to integrate the technology into their teaching and help them do so effectively through addressing knowledge and emotional barriers.

To succeed, this technology training needs to reflect the complexity of teacher knowledge. This complexity has been represented by the TPCK (or TPACK) framework: Technological Pedagogical and Content Knowledge (Mishra & Koehler, 2006). Mishra and Koehler argued that to integrate technology into instruction, teachers must understand the content they are teaching; the technology they are using; and pedagogy related to the content, technology, and students. A lack of knowledge in any of these areas—content, technology, or pedagogy—could be a knowledge barrier for teachers of coding, computing, or robotics. In addition to having knowledge barriers, teachers may lack the confidence or self-efficacy to teach with technology.

Bandura (1977) defined perceived self-efficacy as a judgement of one's ability to perform. Teacher self-efficacy is a teacher's judgment of her ability to teach. Technology self-efficacy (TSE) is a person's belief that "he/she will be successful in using the technology" (Holden & Rada, 2011, p. 347). Teacher technology self-efficacy, then, is a teacher's judgment of her ability to use technology to teach. Self-efficacy is distinct from what Bandura (1977) called outcomes expectations, the belief that specific behavior will lead to specific outcomes. Both efficacy beliefs and outcome expectations influence a person's likelihood to do things. For example, a teacher who is confident in their abilities to use computers and teach effectively might choose not to use robots in instruction if they don't expect that using robots will help

students to learn what they need to learn. Similarly, if the teacher thinks that using robots will help students gain important skills, but she lacks confidence in her ability to use robots to teach, she again might choose not to teach using robots.

According to Ertmer and Ottenbreit-Leftwich (2010), "self-efficacy may be *more important* than skills and knowledge among teachers who implement technology in their classrooms" (p. 261). Teachers with low technology self-efficacy are less likely to use technology than are confident teachers (Holden & Rada, 2011; Vannatta & Fordham, 2004). Experience is key to developing technology self-efficacy. Teacher training can help teachers gain self-efficacy by providing time to play (Somekh, 2008) and facilitating positive experiences with technology in the classroom (Mueller, Wood, Willoughby, Ross, & Specht, 2008).

In conclusion, the research literature shows that teachers can learn to use robots, teach students to use robots, and understand and apply computational thinking effectively. The challenge, then, remains in how to provide the effective training that can help them in developing the self-efficacy and skills to integrate computational thinking into their curriculum by teaching students to program robots. While there have been many studies on how to provide this training in the area of teacher technology integration in general, there are few studies considering how to develop teacher self-efficacy and skill in computational thinking particularly.

The purposes of this project, then, were to design and implement an intervention to prepare K–6 teachers to teach computational thinking using robots. The research questions guiding this study were as follows:

For a sample of K–6 teachers preparing to teach with robots,

1. What computational thinking learning objectives did teachers use the robots to teach?
2. What obstacles and challenges to implementation did teachers face?

3. What influence did training have on teachers' technology self-efficacy?
4. Before and after training, what did teachers believe were the benefits of using robots in the classroom?
5. What types of robot-related professional development did teachers value?

Research Context

In 2017, we received donor funding to purchase robots for training preservice and in-service teachers in how to teach computational thinking and computer science in their classrooms. One aim of the project was to provide technology-enhanced learning opportunities to as many teachers as possible; therefore, in addition to using the technologies for our university courses, we made the robots available to local school teachers. However, we anticipated two main barriers to adoption by these teachers: (1) the logistics involved in borrowing the robots; and (2) the lack of knowledge, skills, and self-efficacy for the teachers to integrate robots into their instruction. To help overcome these barriers, we offered delivery and training to teachers in two schools, Hillview and Riverside, within one school district (names of teachers and schools have been changed).

Hillview is a large elementary school with 754 students and 33 K-6 classroom teachers. Over the past three school years, enrollment has remained steady, while test scores have increased. In 2017, 40% of students were economically disadvantaged and 18% were English language learners (Table 1). Not long before the study, Principal Davis purchased four Dash robots for school use. He hoped that by participating in this project, his teachers would be prepared and motivated to use the robots with their students.

Riverside Elementary is a Title I school with 475 students, and 20 K-6 classroom teachers. As shown in Table 1, in 2017 most Riverside students were economically

disadvantaged, and 44% were English Language Learners (ELL). Despite the school's challenges, student test scores have increased steadily over the past four school years, and the school was awarded the 2017 National Title I Distinguished School Award. All Riverside students engage in STEAM (Science, Technology, Engineering, Arts, and Math) activities for an hour each Friday.

Table 1

School Demographics, 2016-2017

	Hillview	Riverside
October Enrollment	754	475
Ethnic Minority	29%	64%
Economically Disadvantaged	40%	80%
Special Education	12%	15%
English Language Learner	18%	44%
Chronic Absenteeism	13%	18%
Mobility	<10%	26%

(PSD, 2017, PSD School Data Profile, Hillview and Riverside Elementary)

The demographics of each school may affect teachers' experience implementing their training. Students learning English may have a harder time than other students following instructions or learning the computational thinking vocabulary. Teachers may need to adapt instruction for ELLs.

Methodology

The current study is an exploratory study to help us design effective interventions to prepare teachers to teach computational thinking with robots. Data were collected in March 2018 using qualitative methods, including questionnaires and informal interviews.

Participants

Two teachers at Riverside Elementary School and three teachers at Hillview Elementary School participated in the study (Table 2). The teachers taught grades K-4 and used at least one of three types of robots with their students for 1-3 weeks.

Table 2

Participants by Grade Level, Technology, and Period of Implementation

Teacher Pseudonym	Grade level	Robot	Period of Implementation
Annie	3	Ozobots	3 weeks
Becca	4	Spheros & Ozobots	2 weeks
Cathy	1	Dash	1 week
Deb	2	Ozobots	10 days
Evelyn	K	Dash	1 week

Development of Training

For this study, the lead author developed 30-60 minute in-school training modules for individual teachers and grade-level groups. The training was brief, providing just enough information and resources to get started, with the idea that teachers would continue learning on their own through experience and personal research.

In keeping with expert recommendations cited above, this training was school-based, active, and focused on developing content and practical knowledge. However, the training was short-term and in four of five cases, one-on-one. Several studies of PD for elementary technology education have shown improved self-efficacy or knowledge with 12 hours or fewer

of instruction (Cetin & Andrews-Larson, 2016; Jaipal-Jamani, Angeli, 2017; Kim et al., 2015; Ma et al., 2008; Yadav et al, 2014; Ng, 2017). Yadav et al (2014) observed increased understanding of CT after only 100 minutes of instruction. In a survey of 313 primary teachers who taught computing (Rich et al., 2018), 55% of respondents had little or no training in computing or coding before teaching it. When asked to make recommendations about how to teach computing or coding, a third of respondents said to just do it. Twelve percent indicated that teachers should learn with and from their students. Based on this advice, the training was a targeted, minimal intervention aimed at helping teachers gain confidence in their ability to teach using technology.

The two main goals of training were to help teachers (a) feel comfortable using robots and (b) plan ideas for how to teach with robots. Somekh (2008) and Mueller et al. (2008) asserted that teachers gain self-efficacy through play and positive experiences with technology. Therefore, the training incorporated play and experience. Ertmer and Ottenbreit-Leftwich (2010) suggested that professional development programs provide teachers with specific guidance for using technology; therefore, the training included lesson planning. Thus, the training helped teachers learn to use and teach with robots by having them actually use and design lessons for the robots. The basic format for training was as follows:

1. *Modeling*: The trainer modeled the use of the robots (e.g., how to turn them on).
2. *Practice*: Teachers played with and practiced programming the robots.
3. *Instruction:* The trainer explained computational thinking and shared relevant state, ISTE, and CSTA learning standards.
4. *Planning*: The trainer and teachers discussed lesson plans to teach students to use the robots.

Following the 30-60 minute training session, teachers had 1-3 weeks to use the robots with their students. This period of implementation was an essential part of the training. Although all training sessions followed the same basic format, the content was based on grade level and technology.

We had three types of robots available for classroom use: Ozobot bit, Sphero SPRK+, and Dash robots. All three robots have guides and lessons available online, and all three have both basic and advanced programming options. All three robots were used in this study and were available as self-contained kits for the teachers to reserve and use for several weeks on loan from the university. The teachers in this study used the robots with their students for one to three weeks, to practice basic programming.

Data Collection and Analysis

Teachers who chose to participate were asked to complete a self-efficacy and attitude survey before training (pre-survey) and after implementation (post-survey). The lead author also engaged participants in informal interviews when she delivered and picked up the robot kits. We analyzed data from the surveys using descriptive statistics. To code and categorize open-ended survey responses and interview notes, we used "constant comparative analysis" (Glaser & Strauss, 1967), allowing themes to emerge from the data, and looking for common patterns across cases. Nine months after initial data collection, we informally emailed participants to ask follow-up questions.

Survey and Interview Instruments

The research questions that guided the study likewise guided the development of the questionnaires and informal interview questions. The purpose of the surveys was to help us understand teachers' learning objectives, obstacles and challenges to implementation, teacher

technology self-efficacy, perceived benefits of using robots, professional development preferences. The pre-survey, administered before training, included one multiple-choice question, two open-ended questions, and seven five-point Likert items (Table 3). The post survey, administered after implementation, included one ordering question, three open-ended questions, and eight five-point Likert items. Five items on pre and posttests were identical and were used to indicate changes in attitude or self-efficacy. Four items were almost identical on pre and posttests, but differed in tense (e.g., "What do you expect to happen" vs. "What actually happened?"). The pre-survey included one question about objectives that was most relevant before training and implementation, and therefore not included on in the post survey. The post survey included two questions about professional development preferences and one question about future plans that would have been less relevant before implementation, and therefore were not included on the pre-survey. Table 3 reports the survey items and their alignment to the research topics.

Table 3

Survey Items by Topic

Topic	Pre-survey Items	Post survey Items
Objectives	What is your reason for borrowing the robots?	--
	What objectives do you hope to achieve by using robots in your classroom?	What learning objectives did you achieve by using robots in your classroom?
Challenges	What obstacles are there to using robots in your classroom?	What challenges did you experience using robots in your classroom?
Teacher self-efficacy	I am an effective teacher.	Same

Technology self-efficacy	I know how to operate _____ (Dash robots, Ozobot s, or Spheros).	Same
Teacher technology self-efficacy	I am confident in my ability to use robots in the classroom to meet specific learning objectives.	Same
Beliefs about the benefits of teaching with robots	It is important for students to learn computational thinking skills (e.g., decomposition, pattern-finding, algorithms).	Same
	Using robots in the classroom will help students learn computational thinking skills.	Using robots in the classroom helped students learn computational thinking skills.
	Helping K-6 students gain digital literacy skills is important.	Same
	Using robots in the classroom will help students gain digital literacy skills.	Using robots in the classroom helped students gain digital literacy skills.
Professional development preferences	--	What sort of future training or support would best help you to use robots to meet specific learning objectives?
	--	Which activities have best helped you prepare to use robots to meet specific learning objectives?
Other	--	I am planning to use the robot kits again to teach.

Informal interviews were conducted during training sessions and after implementation, when the lead author retrieved the technology sets from teachers. The purpose of initial informal interviews was to gain an understanding of what the teachers needed and wanted from the training session. Informal interview questions asked during the training session included questions such as:

1. What are your plans for using the robots in your classroom?
2. Have you used robots before?
3. How can I help?

The purpose of informal interviews conducted after implementation was to gain an understanding of what about the training was successful and what could be improved. Participants were asked the following and similar questions:

1. What went well?
2. What would you do differently the next time you use robots with students?
3. What additional training do you wish you had had before using robots with students?
4. What additional training would you like to have before using robots again?

Any question asked during training or after implementation was considered part of the informal interview.

In follow-up emails sent in December 2018, teachers were asked whether they had used robots with students or had plans to do so during the 2018-2019 school year, and why they had chosen to use or not use robots with their students.

Five Cases

The lead author met with each of the five teachers to provide training. Before training each teacher was asked to sign an informed consent form and complete the pre-survey. Below we describe the five training sessions.

Annie

Annie, a third-grade teacher in her third year at Hillview, illustrates how a teacher could develop confidence and skill in using these robots after a brief one-on-one training. In the 30-minute training session, the trainer first demonstrated the basics of Ozobots and guided Annie

through a series of inquiry-based activities. Annie tried the Ozobot on various surfaces, then on various lines. Next, Annie drew codes and noticed how they made the robot act differently. The trainer summarized the principles of computational thinking and asked Annie for examples of CT principles in other parts of the curriculum. We discussed algorithms as sets of instructions, and the trainer suggested that to help students understand algorithms, the class could write a "getting started" algorithm describing what to do to get ready to use Ozobots.

We then planned activities for Annie to teach her students to program Ozobots, incorporating our practice activities and lesson plans available at ozobot.com. After drawing a few more programs of her own Annie commented that she felt comfortable using the robots and had enough ideas for her teaching. Over three weeks, her students spent about two hours programming the bots. In the end, Annie would have liked to do more of the online lessons but lacked the time.

Becca

Becca's experience shows persistence despite technical difficulties. Becca, an experienced fourth grade teacher at Riverside, planned to use robots for one hour with each of three classes of fourth graders in her school. Two months prior, Becca had attended a two-hour district-sponsored training session in which she learned about computational thinking and used Dash and Sphero robots for 15 minutes each. She requested Dash robots for her classroom, but since they were not available she agreed to use Sphero robots. When the trainer delivered the robots, Becca admitted that she had struggled with them at the district training session and was a little worried.

Becca had scheduled our meeting for 8:00 am on a school day. The students arrived at 8:35, and Becca was planning to use the robots at 9:00. Sphero requires more set up than the

other robots in this study, and we spent much of our time deciding between university iPads and school Chromebooks and setting up accounts. Becca started an online Sphero lesson ("Blocks 1") and arranged with another teacher to take her students until 9:00. Becca then logged into Sphero.edu site to see what her students would experience, and we talked through Becca's plan for teaching the students. Becca planned to do the Blocks 1 lesson using a paired programming strategy where students take turns giving and following instructions.

Becca later reported that all but one of the robots had run out of batteries almost immediately. The next week, again, several quit working. After two unsuccessful attempts with the Spheros, Becca agreed to try Ozobots. We met for 10 minutes on a Wednesday to practice Ozobot activities, and Becca used the Ozobots for one hour with students the subsequent Friday. Despite the setbacks, Becca remained positive, enjoyed the Ozobots, and said she would like to use them again.

Cathy

Cathy, an experienced first grade teacher, planned to use Dash robots with her students during their invention unit. Although her primary purpose for using robots was to illustrate new inventions, through the training, she also saw their value for teaching computational thinking. She had invited two other first grade teachers to our 40-minute training session, though Cathy was the only one planning to use the robots. After I gave each teacher a robot, an iPad, and instruction for finding the "Path" app, they played with the robots, with minimal help provided as needed. As they played, one teacher commented that the Dash robots were noisy. Cathy had planned to use eight robots in the classroom but was rethinking her plan—maybe she would have her 17 students work in groups instead of pairs. After about 15 minutes of practice, we discussed the teachers' goals and objectives for using the robots. Cathy said she liked that robots teach

persistence—the students would try and fail and try again. Cathy and her students had fun learning to program the robots for a week and would have enjoyed more time with them. Cathy is an example of an experienced teacher who was willing to try new things, had a plan, and incorporated new principles from training.

Deb

Deb, a second-grade teacher in her first year of teaching, was excited to use Ozobots, but lacked a plan for teaching with them. When she first tried the Ozobot, she noticed it stopped on the white paper and that it always faced the same direction, remarking, "That's the front of it." After testing the bot on various lines, Deb noted that the lines needed to be thick. Next, Deb attempted to draw a line with code using some provided coding sheets. When asked what sorts of lesson objectives she might use the Ozobots to teach, she said she noticed a lot of patterns. After instruction about computational thinking, the trainer shared Ozobot website resources and lesson plans for the activities we had done.

In their 10 days with the bots, Deb's students had a positive experience experimenting with lines and codes, making a racetrack, and playing a space game found on ozobot.com. Coming into the training, Deb needed a little guidance and practice to know how to use the robots, but after the training, Deb kept learning by exploring the resources on her own.

Evelyn

The fifth teacher was Evelyn, an experienced kindergarten teacher at Hillview. She had demonstrated Dash robots for other teachers at her school and for her students. Despite her familiarity with Dash, she seemed unfamiliar with the Path app. She turned on a robot and opened the app but then asked what to do next. She practiced for several minutes to get used to the app.

After Evelyn practiced with the robot, the trainer shared the ISTE and CSTA standards and explained computational thinking. Evelyn seemed to like the idea of creating "getting started" and "putting robots away" algorithms. The trainer described a couple other unplugged activities: programming a partner and train conducting. We discussed Evelyn's plans for teaching students to use Path and tried a more basic app, Go, in which users drive the robot as they would a remote-control car. After using the robots for a week, Evelyn reported that her students had had a lot of fun programming robots.

Follow Up

Of the five teachers, three responded to follow-up emails nine months after training. None of the three had used robots with their current students. One planned to do so because the kids enjoyed them and practiced reasoning skills. One teacher did not plan to use robots this year due to time constraints. One teacher said she would consider using them but would need to thoroughly test the robots beforehand, to avoid technical difficulties.

Results

In this section, we share the findings from surveys and interviews in relation to our research questions. Responses are organized into the following themes and categories: teaching objectives related to using the robots, challenges, teacher technology self-efficacy, beliefs about technology, and professional development preferences.

Teaching Objectives

In pre-surveys and interviews, all five teachers said they borrowed the robots for the purpose of teaching students; four of five teachers said they borrowed the robots to learn to use them personally; and only two teachers said they wanted to share the robots with other teachers. When asked what learning objectives they hoped to achieve, only one response was not directly

related to computational thinking: "To give students a hands-on opportunity to learn digital literacy skills." Considering that the survey mentioned “computational thinking” and “digital literacy,” the two teachers who mentioned those terms in their responses were likely influenced by the survey and may not have had specific learning objectives in mind. If we do not consider those two responses, the remaining responses include recognizing patterns, critical thinking, organizing information, trial and error, and exploring, all of which relate to computational thinking, and none of which was mentioned in the survey. However, the sample size is too small to draw any lasting conclusions.

When asked in post surveys and interviews what learning objectives students had achieved, four of the teachers mentioned computational thinking concepts or approaches:

- “We talked about how inventors use observations, trial and error, and debugging to make their inventions better.”
- “Following directions; being careful and specific when programming”
- "Making patterns; Analyzing problems & creating solutions"
- “We learned about Algorithms, procedures”

The response by the fifth teacher was related to computing and coding but was less specific than the other four: "Talked about computers, robots, coding." The teacher who mentioned algorithms also mentioned ordinal numbers, taking turns, and interactive writing. These responses suggest that (a) simple robots were used for teaching a variety of concepts; (b) robots were used to teach computational thinking; (c) the teachers in the study recalled the computational thinking concepts that were shared with them weeks earlier.

Challenges

When asked in pre-surveys and interviews what obstacles they faced to using robots in their classrooms, teachers noted multiple barriers to use. Three teachers cited “time,” two mentioned classroom management, one mentioned alignment to standards, and one said, "not knowing how to use them." When asked in post surveys and interviews what challenges they had faced while using robots, two teachers noted technical difficulties (robots not working), two mentioned logistical difficulties (giving the robots a clear path; remembering to charge robots and iPads), and one noted user difficulty (students had a hard time drawing lines thick enough for the robot to read). Nine months after training, time and technical difficulties remained barriers to ongoing implementation.

Teacher Technology Self-Efficacy

In pre and post surveys, teachers were asked to indicate their level of agreement with statements about their teacher self-efficacy, technology self-efficacy, and teacher technology self-efficacy. On both surveys, all five teachers agreed or strongly agreed with the statement, “I am an effective teacher,” which indicates all five teachers viewed themselves as effective teachers and trusted in their ability to teach well (see Figure 1).

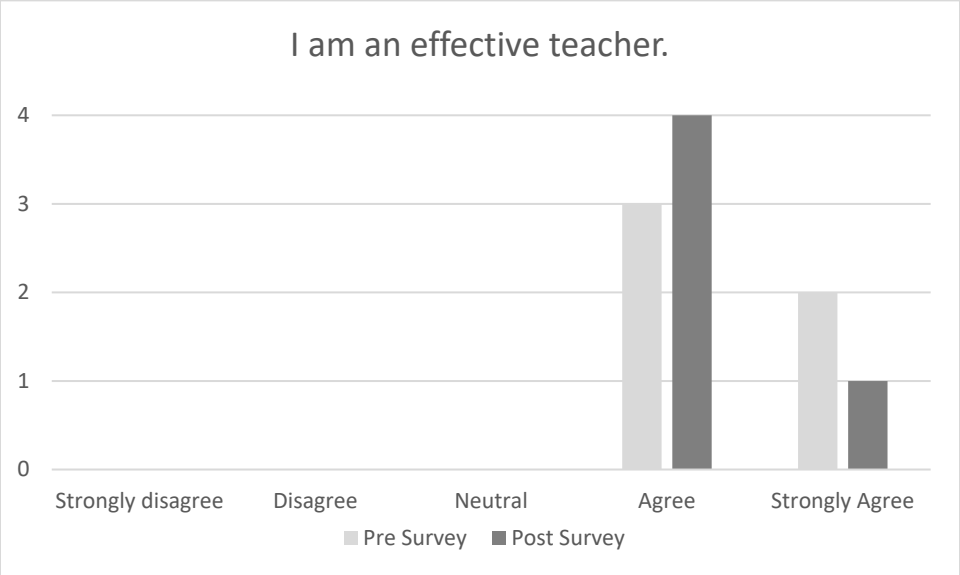

Figure 1. Teacher self-efficacy. This figure illustrates the level of agreement to the statement, "I am an effective teacher" on pre and post surveys.

In pre-survey responses, three teachers disagreed with the statement, "I know how to operate _____ (Dash robots, Ozobots, or Spheros)"; one teacher was neutral, and one teacher agreed with the statement (see Figure 2). In post survey responses, all five teachers agreed with the statement, which indicates positive change for four of five teachers in their technology self-efficacy.

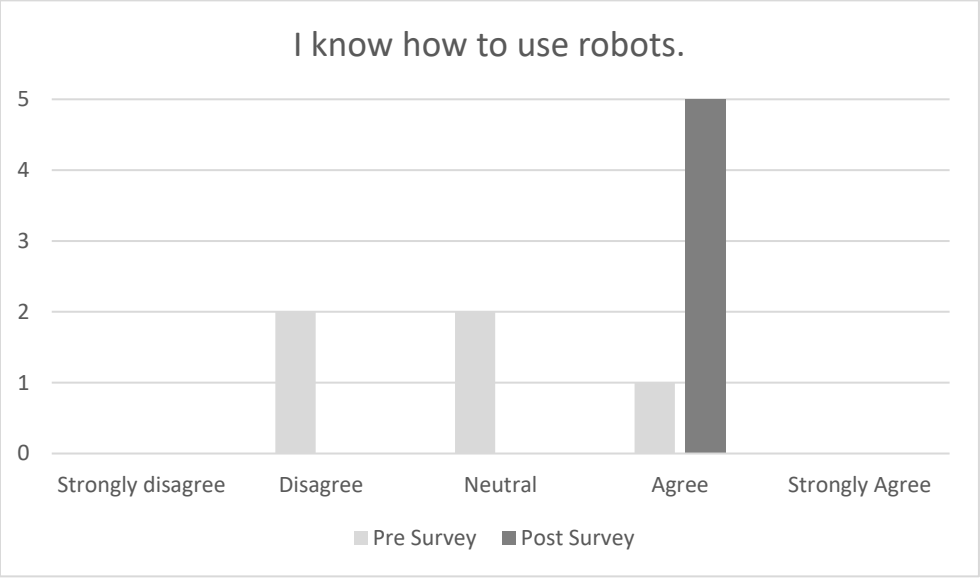

Figure 2. Technology Self-Efficacy. This figure illustrates the level of agreement to the statement, "I know how to operate _____ (Dash robots, Ozobots, or Spheros)" on pre and post surveys.

Responses also indicated positive change for three of the teachers in teacher technology self-efficacy. In pre-survey responses, two teachers disagreed with the statement, "I am confident in my ability to use robots in the classroom to meet specific learning objectives"; one teacher was neutral, and two agreed with the statement. After the intervention, all five teachers agreed with the statement (see Figure 3).

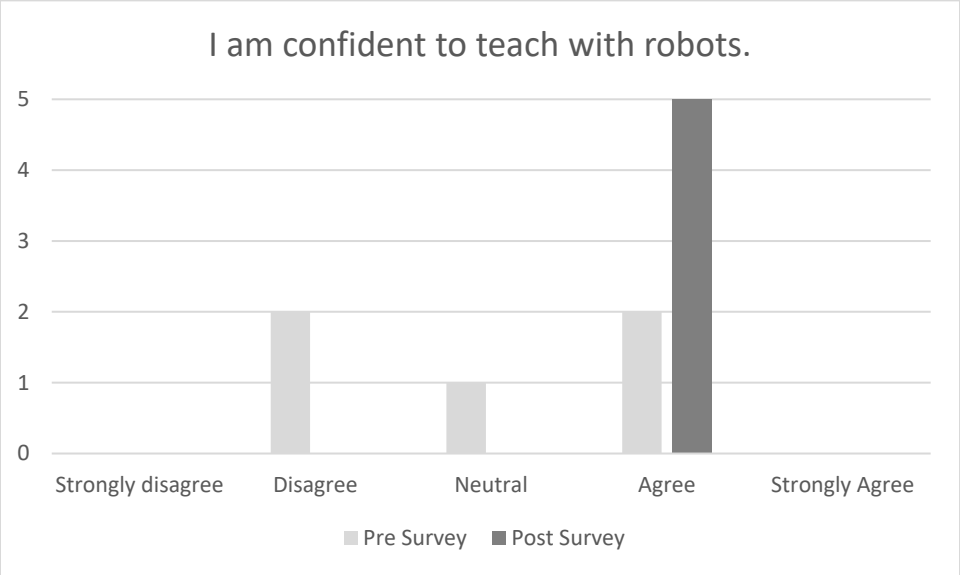

Figure 3. Teacher technology self-efficacy. This figure illustrates the level of agreement to the statement, "I am confident in my ability to use robots in the classroom to meet specific learning objectives" on pre and post surveys.

In post-implementation interviews, all five participants said that they had positive experiences using the robots and were likely to use them again.

Beliefs about Technology

In their responses to both pre and post surveys, all respondents agreed or strongly agreed with the statements that computational thinking and digital literacy are important, and that using robots would help students gain computational thinking skills and digital literacy. For the five teachers, there was little or no change between pre-training and post-implementation responses.

Professional Development Preferences

In the post survey, we asked teachers to rank training activities in order from most useful to least useful. The activities that ranked highest were

1. in-school training,
2. teaching students using robots,

3. using the robots on their own, and
4. personal research and planning.

Four of the five teachers ranked in-school training as the most helpful activity. The activities that ranked lowest were

5. PLC collaboration and planning,
6. other school-sponsored training, and
7. district training.

For most teachers in the study, these three low-ranked activities were not applicable to their use of robots. Only one of the five teachers had invited her PLC group to the in-school training session, and only one of the five teachers had attended a recent, relevant district training session. When asked what sort of future training or support would be most useful, two teachers said more time to explore with the robots, two requested lesson plan ideas, and the fifth had no suggestions for further training. The in-school training session consisted largely of time to use the robots and lesson plan ideas, and responses suggest that teachers valued these aspects of training.

Limitations

This study is limited primarily by the size of the study and scope of the intervention. Because the study was small, the results may not be generalizable and should be interpreted with caution. Outcomes were also limited by the scope of the intervention. More research is needed to understand the long-term outcomes of training. We do not know how the experience with robots influenced student attitudes or achievement.

Additionally, our data were limited by our approach to data collection. Because we were examining beliefs and attitudes, we relied on self-reported data, which may not be entirely reliable. To better understand how teachers applied their training, classroom observation may

have provided additional data. However, we determined that surveys and interviews were sufficient and appropriate for answering our research questions. Since we could not find a validated survey aligned to our particular intervention and research questions, we developed our own survey and interview questions. These questions could have been asked and answered entirely in interviews; however, we determined that most of our research questions could be answered efficiently in a survey format. Informal interviews allowed for follow-up and triangulation of data.

Implications and Conclusion

Although small in scope, the intervention seems to have been a success, by multiple measures. First, participants had positive experiences. The teachers in the study reported that their students had fun and the teachers planned to use robots again. Second, teachers used robots to teach computational thinking skills, as we had hoped they would and trained them to do. Third, we achieved a 100% implementation rate: every teacher who participated in the training used the robots with the students. However, each of these success measures comes with qualifications.

First, although participants reported positive experiences, we believe that using robots is more than just fun and future research should better evaluate how well these feelings of satisfaction affected learning and engagement. Students learned basic programming, which is a useful skill, but it is unknown whether the experience contributed to long-term interest in robotics or programming.

Second, teachers used robots to teach computational thinking skills; however, student gains in computational thinking skills were not assessed and may have been negligible. More research of student learning outcomes is needed.

Third, every teacher who participated in the training implemented the training, but changes in practice were short-term. Two factors that could explain the high rate of implementation are agency and immediacy. The five teachers who participated in the study elected to borrow the robots and receive training. Because the teachers borrowed the robots for 1-3 weeks, they were motivated to use them while they could. If the teachers had been required to participate, or had permanent, easy access to robots, the rate of implementation may have been lower. While having only short-term access to robots may have increased the rate of immediate implementation, it likely decreased the rate of long-term robot use. Teachers planned to use robots again, but nine months later, none had. By providing delivery and training, we overcame initial barriers to use, but for teachers to use robots on a regular basis, they need regular access to the technology, ongoing training and support, and incentives to change practice.

And although the rate of implementation in the current study was high, the number of teachers who elected to participate in the training was low—only two or three per school. The low rate of participation could be due to lack of time and motivation. Teachers were busy teaching required curriculum, and we asked them to spend time learning and teaching new, non-required skills. Unless computational thinking is required instruction, teachers may not elect to learn or teach it. To implement similar training on a larger scale, principals and teachers must prioritize the study of computational thinking, robotics, and programming. Furthermore, providing one-on-one training for teachers is expensive and requires qualified instructors. To make this type of training scalable, we recommend adding a sixth step to the intervention: training another teacher. Having teachers train other teachers in using robots to teach computational thinking could increase teacher confidence, promote collaboration among

teachers, promote a culture of innovation, and reduce the number of specialists needed for training.

The purpose of the PD was to provide teachers with the opportunity to gain the skills and confidence to use robots in the classroom, and that goal was achieved in the short term. Lasting change in teacher practice is unlikely to happen without ongoing training, supportive school culture, and incentive to teach robotics or CT, such as standards and curricular reform. This study supports assertions by Somekh (2008) and Mueller et al. (2008) that teachers gain technology self-efficacy by having positive experiences with technology. While more research is needed, our exploratory study supports previous findings that effective interventions for preparing teachers to teach computing include modeling, practice, instruction, lesson planning, and implementation. In addition, our training was school-based, elective, and mostly one-on-one. This individualized, agentic approach to PD takes resources and relies on supportive principals, policy, and school culture, but can be effective in providing teachers the skills and confidence to teach new concepts with new technologies.

References

Avalos, B. (2011). Teacher professional development in teaching and teacher education over ten years. *Teaching and Teacher Education*, *27*(1), 10-20.

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, *84*(2), 191.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145-157.

Cetin, I., & Andrews-Larson, C. (2016). Learning sorting algorithms through visualization construction. *Computer Science Education*, *26*(1), 27-43. doi:10.1080/08993408.2016.1160664

Darling-Hammond, L., Hyler, M. E., Gardner, M. (2017). *Effective teacher professional development* (research brief). Palo Alto, CA: Learning Policy Institute. Retrieved from https://learningpolicyinstitute.org/

Davies, R. S., & West, R. E. (2014). Technology integration in schools. In Handbook of research on educational communications and technology (pp. 841-853). Springer, New York, NY.

Desimone, L. M. (2009). Improving impact studies of teachers' professional development: Toward better conceptualizations and measures. *Educational Researcher, 38*(3), 181-199.

Ertmer, P. A., & Ottenbreit-Leftwich, A. T. (2010). Teacher technology change: How knowledge, confidence, beliefs, and culture intersect. *Journal of Research on Technology in Education*, *42*(3), 255-284.

Glaser, B., & Strauss, A. (1967). *The discovery of grounded theory*. New York, NY: Aldine.

Google (n.d.). *What is computational thinking?* Retrieved from https://computationalthinkingcourse.withgoogle.com/unit?lesson=8&unit=1

Guskey, T. R., & Yoon, K. S. (2009). What works in professional development?. *Phi Delta Kappan*, *90*(7), 495-500.

Holden, H., & Rada, R. (2011). Understanding the influence of perceived usability and technology self-efficacy on teachers' technology acceptance. *Journal of Research on Technology in Education*, *43*(4), 343-367.

ISTE (2014, September 11). *Computational thinking for all.* Retrieved from https://www.iste.org/explore/articledetail?articleid=152

Jaipal-Jamani, K., & Angeli, C. (2017). Effect of robotics on elementary preservice teachers' self-efficacy, science learning, and computational thinking. *Journal of Science Education and Technology, 26*(2), 175-192.

Kim, C., Kim, D., Yuan, J., Hill, R. B., Doshi, P., & Thai, C. N. (2015). Robotics to promote elementary education pre-service teachers' STEM engagement, learning, and teaching. *Computers & Education*, *91*, 14-31.

Leonard, J., Mitchell, M., Barnes-Johnson, J., Unertl, A., Outka-Hill, J., Robinson, R., & Hester-Croff, C. (2017). Preparing teachers to engage rural students in computational thinking through robotics, game design, and culturally responsive teaching. *Journal of Teacher Education*, 0022487117732317.

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.

Mason, S. L. & Rich, P. J. (In press). Preparing elementary school teachers to teach computing, coding, and computational thinking. *Contemporary Issues in Technology and Teacher Education*.

Mishra, P. & Koehler, M. J. (2006). Technological pedagogical content knowledge: A framework for integrating technology in teacher knowledge. *Teachers College Record*, *108*(6), 1017–1054.

Mueller, J., Wood, E., Willoughby, T., Ross, C., & Specht, J. (2008). Identifying discriminating variables between teachers who fully integrate computers and teachers with limited integration. *Computers and Education, 51,* 1523–1537.

Ng, W. S. (2017). Coding education for kids: What to learn? How to prepare teachers? ICICTE 2017 Proceedings. Retrieved from http://www.icicte.org/ICICTE_2017_Proceedings/6.2_Ng%202017.pdf

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

Papert, S. (1993). *The children's machine: Rethinking school in the age of the computer*. New York, NY: Basic Books.

Provo School District (2017). 2016-2017 Provo City School District School Data Profile, Lakeview Elementary. Retrieved from http://www.lakeview.provo.edu/Site_PDF/Lakeview.pdf

Provo School District (2017). 2016-2017 Provo City School District School Data Profile, Spring Creek Elementary. Retrieved from http://www.springcreek.provo.edu/Site_PDF/Spring%20Creek.pdf

Provo School District (2017). Annual Statistical Report. Retrieved from https://provo.edu/wp-content/uploads/2017/01/08162017-s3-Annual-Statistical-Report.pdf

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), 1-20.

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.

Somekh, B. (2008). Factors affecting teachers' pedagogical adoption of ICT. In J. Voogt & G. Knezek (Eds.), *International handbook of information technology in primary and secondary education* (pp. 449–460). New York, NY: Springer.

Vannatta, R. A., & Fordham, N. (2004). Teacher dispositions as predictors of classroom technology use. *Journal of Research on Technology in Education, 36*(3), 253–271.

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

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 (TOCE)*, *14*(1), 5.

ARTICLE 3

Development and Analysis of the Elementary Student Coding Attitudes Survey

Stacie L. Mason

Peter J. Rich

Brigham Young University

Abstract

There is an increasing emphasis on teaching young learners to code; yet, there are few tools designed to measure the effect of learning to code on young children. The purpose of this study was to develop and validate a tool to assess changes in young learners' attitudes toward coding: the Elementary Student Coding Attitudes Survey (ESCAS). We validated the scale using Confirmatory Factory Analysis and Structural Equation Modeling with responses from over 6,000 4th-6th grade students (aged 9-11 years). Survey validation revealed a scale consisting of five constructs that comprise young learners' attitudes toward coding: Social Value, Coding Confidence, Coding Interest, Perception of Coders, and Coding Utility. In our analysis, students' grade level, ethnicity, gender, coding frequency, coding experience, and math interest influenced Social Value, which in turn influenced Coding Interest, Perception of Coders, and Coding Utility. Students' math confidence, coding frequency, coding experience, ethnicity, and Coding Interest predicted their Coding Confidence. Among all observable variables, frequency had the greatest influence on Social Value, which substantially influenced all other factors. We discuss how this tool can help those who teach coding to young children to better measure and understand the factors that may influence young learners' attitudes toward coding over time.

Keywords: Elementary education, computational thinking, coding, attitude scale, instrument validation

1. Introduction

Computer science and coding are increasingly being taught in K-12 education globally. Throughout this study, we use the term "coding" to indicate content taught in elementary computer science, coding, computing, or software programming. Approximately 30% of all U.S. students and 15% of students world-wide have enrolled in Code.org (Code.org Statistics, 2019). What's more, enrollments have risen steadily from 10,000 teachers and 500,000 students in late 2013 to 1,000,000 teachers and 36,000,000 students at the end of 2018. Several countries including England, Finland, Australia, Greece, and France have made some form of coding/programming education compulsory, starting in primary school (Rich et al., 2018). In the U.S., the number of states that had enacted specific K12 computer science policies increased from 14 in 2013 to 44 in 2018 (Code.org, 2018). According to the U.S. Bureau of Labor Statistics (2019), information technology jobs will grow 12% over the next decade, which is "much faster than the average for all occupations" (para. 1). By some measures, 90% of the jobs performed by humans a century ago have now been automated (Balakrishnan, 2018). With this increasing push to teach coding, there is a need to understand how computer science/coding instruction is influencing students, both cognitively and affectively.

Cognitively, studies have long shown that participating in coding has improved students' math performance and problem-solving ability (Clements & Gullo, 1984; Rich, Leatham, & Wright, 2013; Schanzer, Fisler, & Krishnamurti, 2018; Scherer, Siddiq, & Sánchez-Viveros, 2018). Studies have also shown that computing experience improved students' attitudes toward STEM and computing (e.g., Gunbatar & Karalar, 2018; Kalelioğlu 2015; Master, Cheryan, Moscatelli, & Meltzoff, 2017; Rubio et al., 2015; Sáez-López, Román-González, & Vázquez-Cano, 2016). Scherer et al.'s (2018) recent meta-analysis of over 40 years of shows that there is

a rich knowledge-base of the cognitive effects of learning to code on young students. The research on the affective effect of learning to code is much less explored. Educators hoping to change students' perceptions about or attitudes toward coding need a way to measure these changes in order to gauge the effectiveness of different coding curricula. But few scales that assess primary students' attitudes toward coding have been validated. The purpose of this study was to develop and validate an instrument to assess primary students' attitudes toward coding. Our research was guided by two research questions:

RQ1. What factors affect elementary students' attitudes toward coding?

RQ2. How do we measure elementary students' attitudes toward coding?

2. Literature Review

Expectancy-value theory is a useful framework for understanding students' attitudes. In their expectancy-value model of achievement motivation, Eccles et al. (1983) suggested that a person's values and expectations of success are influenced by cultural factors, socializers' beliefs and behaviors, personal aptitudes, experience, achievement, perceptions, goals, and self-schemata (Wigfield & Eccles, 2000). Beyer (2014) likewise applied expectancy-value theory in her analysis of gender differences in computer science attitudes and predictors of CS grades and course taking.

Our interest is in primary-aged learners who may not yet have demonstrated much measurable accomplishments, but who are at an impressionable age where ideas about different life possibilities are beginning to emerge. Where expectancy-value theory focuses on how factors influence achievement-related choices, we are less interested in the choice outcomes than in the interplay among factors and the influence of experience on attitudes and beliefs. Inasmuch as *expectation of success* in expectancy-value theory is similar to Bandura's self-efficacy construct

(Wigfield & Eccles, 2010), we have chosen to use the terms *self-efficacy* and *confidence*, which may be more familiar to readers. Our concept of *interest* draws from definitions both inside and outside of expectancy-value theory. Thus, children's attitudes toward coding may best be assessed by measuring their interests in, self-efficacy for, perceived Utility of, and social biases toward coding.

To better understand which constructs might make up one's attitude toward coding, we reviewed existing attitude scales toward computer science. We searched csedresearch.org, google scholar, IEEE, and ACM databases for scales that measured students' attitudes toward coding, computer science, software programming, or computing. This resulted in a set of 16 scales (see Table 1). Most existing scales to assess computer science attitudes, self-efficacy, and interest are geared toward secondary or college students. Scales for elementary students primarily assess perceptions of STEM professionals.

Table 1

Validated Scales to Assess Students' STEM Self-efficacy, Attitudes, and Perceptions

Authors, year	Name	Population	Subject	Constructs	Items	Scale	N
Scales for University Students							
Dorn & Tew (2015)	Computing Attitudes Survey (CAS)	College students	Computer science	Problem solving transfer, program solving strategies, problem solving fixed mindset, interest, real-world connections	26	Five-point	794
Hoegh & Moskal (2009)	Computing Survey	Colorado School of Mines undergraduate students	Computer Science	Confidence, interest, perceptions of gender, usefulness, perceptions of profession	38	Four-point	276
Ramalingam & Wiedenbeck (1998)	Computer Programming Self Efficacy	University students	C++ program-ming	Independence and persistence, complex programming, self-	32	Seven-point	421

Authors, year	Name	Population	Subject	Constructs	Items	Scale	N
	Scale (CPSES)			regulation, simple programming			
Washington, Grays, & Dasmohapatra (2016)	Computer Science Cultural Attitude and Identity Survey (CSAIS)	Undergraduate students of color	Computer science	Confidence, interest, gender, professional, identity	40	Four-point	65
Scales for High School Students							
Forssen, Moskal, & Harringer (2011)	Information Technology (IT) Attitude Survey	High school students in summer IT program	Informa-tion technology	General interest, gender stereotypes	20	Four-point	142
Hirsch, Gibbons, Kimmel, Rockland, & Bloom (2003)	High School Students' Attitude to Engineering and Engineering Self-Efficacy	High school students in funded summer program	Engineer-ing	Positive aspects of engineering, negative opinions of engineering, interest, job issues	33	Six-point	317
Mahoney (2010)	Student Attitudes Toward STEM	High school students (grades 9-12)	Science, technology, engineer-ing, mathema-tics	Awareness, perceived ability, value, commitment	96 (24 per cont ent area)	Four-point	378
Scales for Middle School Students							
Erkut & Marx (2005)	Attitudes toward Engineering, Math, and Science	Eighth graders	Math, science, engineering	Attitudes toward math, attitudes toward science, attitudes toward engineering	35	Five-point	436
Gibbons, Hirsch, Kimmel, Rockland, and Bloom (2004)	The Middle School Students' Attitude to Mathematics, Science and Engineering Survey	Middle school students (Grades 5-8)	Mathema-tics, science and engineering	Interest (stereotypic), Interest (non-stereotypic), positive opinions, negative opinions, problem solving, technical skills	35	Six-point	1701

Authors, year	Name	Population	Subject	Constructs	Items	Scale	N
Hirsch, Carpinelli, Kimmel, Rockland, & Bloom (2007)	Adapted Middle School Students' Attitude to Mathematics, Science and Engineering Survey	Middle school students (Grades 5-8)	Math, science, engineering	Interest (stereotypic), Interest (non-stereotypic), positive opinions, negative opinions, problem solving, technical skills, engineering	36	Six-point	890
Kukul, Gökçearslan, and Günbatar (2017)	Computer Programming Self-Efficacy Scale	Middle school students	Program-ming	Self-efficacy	31	Five-point	233
Owen et al. (2008)	Revised Simpson-Troost Attitude Questionnaire (STAQ-R)	Middle school students (grades 6-8)	Science	Motivating science class, self-directed effort, family models, science is fun for me, peer models	22	Five-point	1754
Scales for Elementary School Students							
Chambers (1983)	Draw A Scientist Test (DAST)	Elementary students (grades K-5)	Scientists	Stereotypic perceptions	1	Open-ended	4807
Hansen et al. (2017)	Draw-a-Computer-Scientist Test (DACST)	Children (grades 4-6)	Computer scientists	Perceptions	1	Open-ended	185, 87
Knight & Cunningham (2004)	Draw an Engineer Test (DAET)	Grades 3-12	Engineers	Perceptions	5	Open-ended	384
Kong, Chiu, and Lai (2018)	Programming empowerment survey	Primary school students (grades 4-6)	Programming	Interest, collaboration, meaningfulness, impact, creative self-efficacy, programming self-efficacy	23	Five-point	287

As shown in Table 1, we identified 16 scales that assess student attitudes and self-efficacy toward computer science or STEM subjects. For university students, four scales assess

attitudes toward programming. The three scales for high school students assess attitudes toward information technology (IT), engineering, and STEM. Of the five scales for middle school students, one scale assesses attitudes toward science and three scales assess attitudes toward math, science, and engineering. Three of the four elementary scales assess perceptions of scientists, engineers, or computer scientists. For elementary and middle schoolers, only one scale for each age group assesses attitudes or self-efficacy toward programming (Kong et al., 2018; Kukul et al., 2017).

We analyzed the content of these 16 scales to better understand which factors affect elementary student attitudes toward coding (RQ1). As shown in Table 2, seven of the 16 surveys assess confidence or self-efficacy (Erkut & Marx, 2005; Hoegh & Moskal, 2009; Kong et al., 2018; Kukul et al., 2017; Mahoney, 2010; Ramalingam & Wiedenbeck, 1998; Washington, et al., 2016). Eleven scales assess student interest in CS or STEM subjects (Dorn & Tew, 2015; Erkut & Marx, 2005; Forssen, et al., 2011; Gibbons et al., 2004; Hirsch et al., 2003; Hirsch et al., 2007; Hoegh & Moskal, 2009; Kong et al., 2018; Mahoney, 2010; Owen et al., 2008; Washington, et al., 2016). Seven scales assess students' value for, or perceived usefulness of, CS or STEM subjects (Erkut & Marx, 2005; Gibbons et al., 2004; Hirsch et al., 2003; Hirsch et al., 2007; Hoegh & Moskal, 2009; Kong et al., 2018; Mahoney, 2010). Eight scales assess students' perceptions of coders, scientists, or engineers (Chambers, 1983; Gibbons et al., 2004; Hansen et al., 2017; Hirsch et al., 2003; Hirsch et al., 2007; Hoegh & Moskal, 2009; Knight & Cunningham, 2004; Washington, et al., 2016). Three measures assess students' perceived gender stereotypes for CS or STEM subjects (Forssen et al., 2011; Hoegh & Moskal, 2009; Washington, et al., 2016). Only two of the 16 surveys assess social value (Gibbons et al., 2004; Owen et al., 2008).

Our goal in reviewing the scales was to find a scale that could be used on a large scale to measure changes in elementary student attitudes over time as students participate in coding activities (RQ1). Being able to measure and then track students' attitudes surrounding coding over time will help educators and administrators better understand how engaging with coding may shape students' affective biases. In our review, the scales that assess the greatest number of constructs that we were interested in (i.e., interest, utility, and social value) were created by Hoegh & Moskal (2009) and Washington, et al. (2016). However, the content, written for university students in programming classes is too advanced for elementary school children with limited coding experience. We were specifically focused on upper elementary (grades 4-6), the age by which many students are now being introduced to coding around the world. While each of the scales reviewed has its uses, none entirely matched the attitudes and perceptions we wished to assess among elementary students. Based on analysis of existing scales, and our experience working with elementary coding students, we developed the Elementary Student Coding Attitudes Survey (ESCAS) to assess elementary students' coding attitudes and beliefs, including perceived self-efficacy, interest, utility value, gender stereotypes, cultural stereotypes, and social value.

Table 2

ESCAS Constructs Assessed by Existing Scales

Authors, year	**Confidence**	**Interest**	**Useful-ness**	**Percep-tion of profession**	**Percep-tion of gender**	**Social value**
Scales for University Students						
Dorn & Tew (2015)		x				
Hoegh & Moskal (2009)	x	x	x	x	x	
Ramalingam & Wiedenbeck (1998)	x					

Authors, year	Confidence	Interest	Useful-ness	Percep-tion of profession	Percep-tion of gender	Social value
Washington, Grays, & Dasmohapatra (2016)	x	x		x	x	
Scales for High School Students						
Forssen, Moskal, & Harringer (2011)		x			x	
Hirsch, Gibbons, Kimmel, Rockland, & Bloom (2003)		x	x	x		
Mahoney (2010)	x	x	x			
Scales for Middle School Students						
Erkut & Marx (2005)	x	x	x			
Gibbons, Hirsch, Kimmel, Rockland, and Bloom (2004)		x	x	x		x
Hirsch, Carpinelli, Kimmel, Rockland, & Bloom (2007)		x	x	x		
Kukul, Gökçearslan, and Günbatar (2017)	x					
Owen et al. (2008)		x				x
Scales for Elementary School Students						
Chambers (1983)				x		
Hansen et al. (2017)				x		
Knight & Cunningham (2004)				x		
Kong, Chiu, and Lai (2018)	x	x	x			

3. Instrument Development

To develop the scale, we used steps described by DeVellis (2017): (1) determine what to measure; (2) generate items; (3) determine the scale format; (4) conduct expert review; (5) administer the instrument; (6) evaluate the items; and (7) optimize scale length.

3.1 Determining What to Measure

The ESCAS includes several factors that are of interest to educators, administrators, and researchers. Each of the six constructs we hoped to measure is complex. At the same time, we wanted a concise instrument to avoid participant fatigue, especially since the participants were

elementary-aged children. In designing the instrument, we tried to get at the salient features of each factor, but we have not tried to assess every aspect of every construct. We provide a brief review of each of these constructs in the following section.

3.1.1 Coding confidence or self-efficacy

The first construct our instrument is intended to assess is elementary students' perceived coding confidence or self-efficacy. Numerous studies have shown that perceived self-efficacy positively influences students' motivation and achievement (e.g., Manzano-Sanchez, Outley, Gonzalez, & Matarrita-Cascante, 2018; Multon, Brown, & Lent, 1991; Pajares & Schunk, 2001; Pintrich & de Groot, 1990). Perceived self-efficacy is a person's belief that they can complete a particular task or fulfill a particular role within a specific domain (Bandura, 2006). Although Bandura differentiated between confidence and self-efficacy, we use the terms interchangeably as has been done in related scales (e.g., Hoegh & Moskal, 2009), and as demonstrated by the language in Bandura's own sample scales.

Because efficacy varies by subject, self-efficacy scales "must be tailored to the particular domain of functioning that is the object of interest" (Bandura, 2006, pp. 307-308). Indeed, this was the problem we identified with existing scales—they were either more attuned to other STEM subjects or treated computing so broadly (e.g., "technology") as to render the term too ambiguous to pin down to students' self-efficacy for coding-related tasks. We needed to design a scale specific to coding self-efficacy. Since the ESCAS is intended for children who may have spent little or no time coding (i.e., so that it can be given prior to and following coding instruction), our scale does not include specific tasks that only experienced coders would understand. However, in addition to general statements (e.g., "I can learn to code") we included specific skills and aptitudes show to be useful in coding (e.g., "I am good at problem solving").

Several factors have been shown to interact with self-efficacy. Research demonstrates that computing experience increases with computing confidence (e.g., Gunbatar & Karalar, 2018; Shim, Kwon, & Lee, 2017). Several studies have indicated that males had higher computing self-efficacy than females (e.g., Beyer, 2014; Kong et al., 2018; Ramalingam & Wiedenbeck, 1998). Gunbatar and Karalar (2018) found that among middle school students, boys initially had higher programming self-efficacy than girls, but after completing a 12-week programming course, there was no significant difference between boys' and girls' self-efficacy and attitudes. Similarly, Master et al. (2017) found that experience programming robots eliminated differences between first grade boys' and girls' technology self-efficacy. In addition to the influence of gender, researchers have found that grade level helped predict programming self-efficacy (Kong et al., 2018), but ethnicity did not significantly influence STEM self-efficacy (Hirsch et al., 2003). Students may have high self-efficacy for a subject without really understanding related professions (Hirsch et al., 2003). In our analysis, we expected to see multiple predictors influence self-efficacy.

3.1.2 Interest and curiosity

The second factor our instrument measures is student interest in coding. Like self-efficacy, interest is correlated with student achievement and is content-specific (Schiefele, Krapp, & Winteler, 1992; Wininger, Adkins, Inman, & Roberts, 2014). In expectancy value theory, *interest value* refers to "how much the individual likes or is interested in the activity" (Wentzel & Wigfield, 1998, p. 158). According to Grossnickle (2016), "defining features of interest include knowledge of, positive feelings toward, and value for the object of interest" (p. 43). Of these three aspects, our interest factor focuses on positive feelings toward coding. We address value for coding in terms of its usefulness as a separate factor. We are more concerned

with assessing whether students are interested in coding than assessing why they are interested; however, two items include descriptors that indicate perceived properties of coding ("Solving coding problems seems fun"; "Coding is interesting").

In several studies, males showed more interest in computing or technology than females (Beyer, 2014; Dorn & Tew, 2015, Kong et al., 2018; Mahoney, 2010). However, studies have shown that the gender gap disappeared after children participated in robotics activities (Master et al., 2017; Sullivan & Bers, 2019). In another study, gender did not have a significant effect on IT interest, but ethnicity did have significant effect (Forssen et al., 2011). Computing experience has been shown to increase computing interest among university students (Dorn & Tew, 2015), but grade level did not significantly predict high school students' attitudes toward science, technology, or engineering (Mahoney, 2010).

3.1.3 Usefulness or utility value

The third factor in our scale is utility value. As defined by Wigfield & Cambria (2010), "utility value or usefulness refers to how a task fits into an individual's future plans" (p. 4). For example, taking a coding class to fulfill a requirement or to prepare for a profession demonstrates utility value. In expectancy-value theory (Wigfield & Eccles, 2000), utility value directly influences a person's achievement-related choices, and is influenced by a person's experiences, perceptions, goals, and self-schemata. Based on expectancy-value theory, then, we could expect that our utility value factor may correlate with many other dependent and independent variables in our model.

Though several scales we reviewed include items to assess students' value for computing or STEM (Erkut & Marx, 2005; Gibbons et al., 2004; Hirsch et al., 2003; Hirsch et al., 2007; Hoegh & Moskal, 2009; Kong et al., 2018; Mahoney, 2010), the studies say little about what

variables predict perceived usefulness. In Kong et al.'s (2018) model, gender predicted programming interest, which in turn predicted meaningfulness and impact (two aspects of perceived usefulness). Grade also helped predict meaningfulness.

3.1.4 Gender stereotype perceptions

Two types of negative stereotypes may influence children's attitudes toward computer science: (1) beliefs that girls are less able than boys, and (2) beliefs about computer scientist culture (Beyer, 2014; Cheryan et al., 2015). Stereotypes that boys are better at CS than girls may lower girls' self-efficacy or expectations of success.

Computer science is a male-dominated field. Whereas women have caught up to men in several STEM fields, in computer science the gender gap has actually increased (Beyer, 2014). According to data from the National Science Foundation (2017), during the years 2008-2014, women earned only 18% of computer science bachelor's degrees in the U.S., compared with 57% in all fields and 50% in science and engineering. Sullivan and Bers (2019) have asserted that only by starting computing instruction well before college can we increase the proportion of women in computing careers. By secondary school, male students have shown significantly more interest in computing than female students, as indicated by higher participation in robotics competitions and advanced computer science courses (Doerschuk, Liu, & Mann, 2007; Sullivan & Bers, 2019; Witherspoon, Schunn, Higashi, & Baehr, 2016). By teaching children coding, perhaps we can help them avoid forming the belief that coding is for boys only and start to decrease the gender gap in computer science.

Girls' lower rate of participation in CS is attributable to a combination of factors, including lower interest and self-efficacy (Doerschuk, et al., 2007; Master, Cheryan, Moscatelli, and Meltzoff, 2017; Sullivan & Bers, 2019; Witherspoon, et al., 2016). But stereotypes about

gender and ability may also contribute. Master et al. (2017) found that first graders believed "boys were better than girls at robotics and programming," but not at science or math (p. 92). In their assessment of CS views and attitudes, Taub, Armoni, and Ben-Ari (2012) found that both boys and girls tended to disagree with the statement that "Boys are more likely than girls to study CS" (p. 8:14-15). Forssen et al. (2011), found that students who participated in a summer IT program did not show significant changes in their perceptions of gender stereotypes in IT. One goal of developing ESCAS is to assess changes in gender perceptions over time.

3.1.5 Perceptions of coders

Besides negative gender stereotypes related to computing, other negative stereotypes exist regarding coders and the coding profession. Stereotypes that computer science is socially isolating and that computer scientists are geeky males may lower children's interest in coding. Several scales we reviewed include items to assess students' perceptions of STEM and computing professionals (Chambers, 1983; Gibbons et al., 2004; Hansen et al., 2017; Hirsch et al., 2003; Hirsch et al., 2007; Hoegh & Moskal, 2009; Knight & Cunningham, 2004; Washington, et al., 2016).

Three instruments for elementary students use children's drawings to assess their perceptions of scientists, engineers, or computer scientists. Chambers (1983) developed The Draw A Scientist Test (DAST) "to determine at what age children first develop distinctive images of the scientist" (p. 257). Chambers found that students' depictions of stereotypic indicators increased with grade level and were mostly absent among students in kindergarten and first grade. Stereotypic indicators likewise increased with school income level. Only 28 students, all girls, drew women scientists. Similar assessments have been developed to assess children's attitudes toward engineers (Knight & Cunningham, 2004) and computer scientists

(Hansen et al., 2017). Knight and Cunningham (2004) found that students had preconceived ideas about engineers, including the idea that engineers are men. In pre-tests using the DACST, 71% of students drew a male computer scientist (Hansen et al., 2017). After 12 hours of programming instruction, 7% more students, all girls, "drew female computer scientists than before" (p. 279). In a study involving older students, Beyer (2014) found that among 1319 university freshman surveyed, women showed less-negative stereotypes of CS and CS majors than did men.

3.1.6 Social value

A factor related to perceptions of gender and cultural stereotypes is social value. Each person is influenced by the people around them. School children are especially influenced by their peers, family, and teachers. To gain peer and adult acceptance, children learn to adopt the values, behaviors and attitudes they observe (Wentzel & Wigfield, 1998). Therefore, we would expect that if a student thinks her parents, teachers, and peers value coding, that perception would influence the student's attitudes toward and participation in coding. If one of the socializers (parents, teachers, or peers) voiced a negative opinion of coding, that social value could negatively influence the student's coding attitudes and participation.

According to expectancy-value theory, socializers' beliefs and behaviors influence children's goals, self-schemata, interpretations of experience, memories, and perceptions of gender roles and activities, all of which influence children's task values and expectations of success, which values and expectancies directly influence achievement-related choices (Wigfield & Eccles, 2000). Based on expectancy-value theory, then, we may expect to see more indirect than direct influences among social value and other factors.

Owen et al. (2008) found that family models, peer models, and motivating science class predicted students' interest in science. Motivating science class had the largest effect, and family models had a stronger influence on interest for girls than for boys. Gibbons et al. (2004) found that middle school students in STEM outreach programs reported hearing about engineering jobs more often from television and movies than from friends, parents, teachers, or counselors.

3.1.7 School attitude

We expect that students' attitudes toward other subjects, such as math, science, and language arts, may predict students' coding attitudes. The two aspects of school attitudes included in this construct are confidence and interest in math, science, and language arts. Other scales we reviewed did not include school attitude as a predictor for coding or STEM attitudes.

3.1.8 Hypotheses

In previous studies of similar attitude scales, gender has been shown to influence self-efficacy and interest toward STEM subjects (Beyer, 2014; Dorn & Tew, 2015; Kong et al., 2018; Mahoney, 2010; Owen et al., 2008; Ramalingam & Wiedenbeck, 1998). Grade also influenced attitudes in the studies by Owen et al. (2008), and Kong et al. (2018) but not in the study by Mahoney (2010). Owen et al. (2008) found interest and efficacy to be correlated; we expect the same to be true in our model.

Based on previous studies, we predicted that

H1-1: Boys would have higher coding interest and confidence than girls;

H1-2: Confidence and interest would increase with grade level; and

H1-3: Interest, confidence, and utility would be correlated.

Based on expectancy value theory, which accounts for numerous additional influences on students' expectations of success and value for a task, we added several additional variables to

our model. In Eccles and Adler's 1983 model, a child's perceptions of gender roles, activity stereotypes, and parents' attitudes influence children's goals, which in turn influence the child's expectation of success (or self-efficacy, or confidence) and value for the task, which includes both interest and perceived utility. Previous experience also predicts interest and utility value. Based on this theory, we hypothesized that

H1-4: As coding experience increased, coding confidence, interest, and perceived utility would increase;

H1-5: Higher math confidence would predict higher coding confidence;

H1-6: Children of college-educated parents would demonstrate higher coding confidence, interest, and perceived utility than children of non-college-educated parents;

H1-7: Students who reported knowing a coder would have higher coding interest than students who did not;

H1-8: The more students thought their parents valued coding, the greater the students' coding interest and perceived utility.

3.2 Item Generation

To help us generate items, we first identified and examined several related survey instruments (Beyer, 2014; Dorn & Tew, 2015; Erkut & Marx, 2005; Forssen, Moskal, & Harringer, 2011; Gibbons, Hirsch, Kimmel, Rockland, & Bloom, 2004; Hansen et al., 2017; Hirsch, Carpinelli, Kimmel, Rockland, & Bloom, 2007; Hoegh & Moskal, 2009; Knight & Cunningham, 2004; Kukul, Gökçearslan, & Günbatar 2017; Mahoney, 2010; Owen et al., 2008; Ramalingam & Wiedenbeck, 1998; Suldo, & Shaffer, 2007; Taub, Armoni, Ben-Ari, 2012; Washington, Grays, & Dasmohapatra, 2016; Yadav, Zhou, Mayfield, Hambrusch, & Korb,

2011). After examining related instruments, we generated approximately 100 items, including both original items and items adapted from existing surveys. Through a review process described in the next section, we reduced the number of items to 40 that assess our six constructs of interest. As shown in Table 3, we adapted 10 items from Hoegh and Moskal's Computing Survey (2009); four items from Mahoney's Students' Attitudes Toward STEM scale (2010); two items from *The Middle School Students' Attitude to Mathematics, Science and Engineering Survey* (Gibbons et al., 2004), three items from the revised Simpson-Troost Attitude Questionnaire (STAQ-R; Owen et al., 2008), two items from Yadav et al.'s 2011 Computational Thinking Attitudes survey; three items from Dorn and Tew's 2015 Computing Attitudes Survey (CAS), and two items from Kukul, et al.'s (2017) Computer Programming Self-Efficacy Scale. In addition to the cited similarities, many of our items may resemble items from various scales that assess similar constructs.

3.3 Format

The original instrument included six multiple-choice items to collect demographic information, seven open-ended questions, and 39 Likert-type items, for 52 total items. Because the scale is designed for students in grades 4-6, we wrote item stems at approximately a third-grade reading level. All constructs were measured using a six-point Likert-type scale. We do not believe it is possible to possess "neutral" confidence; even-numbered scales avoid this problem by encouraging responses that demonstrate which way a respondent's confidence leans. Although Bandura (2006) recommended using a 100-point scale, Reeve, Kitchen, Sudweeks, Bell, and Bradshaw (2011) tested this claim and found that both and 11-point and 6-point scale provide adequate differentiation for self-efficacy scales. Due to these reasons, we chose to use a simple 6-point Likert-type scale (strongly disagree, disagree, somewhat disagree, somewhat

agree, agree, strongly agree) to measure all constructs. Nearly all items were positively worded to avoid causing confusion for a group that would consist of young readers.

3.4 Review and Field-test

After composing approximately 100 items to measure our constructs, each of the four research team members rated items for clarity and relevance, and based on combined ratings, we narrowed the list to 48 items. For the expert review, three university faculty in education and measurement reviewed the items and offered suggestions for revising or replacing problematic items. The two lead researchers again reviewed and revised items to try to convey clear meaning and accurately reflect the constructs we were trying to assess. Next, we tested the instrument with several upper elementary school-age children, asking them to verbalize their thoughts about scale items and format. Based on feedback from our field test, we further refined scale items. The resulting instrument includes 52 items: 6 demographic, 7 open-ended, and 39 Likert-type items (Table 3).

Table 3

Initial Factors and Items

Factor	Item		Source*
Coding	C1	I can learn to code.	C
Confidence	C2	I am good at coding.	D
	C3	I am good at problem solving.	D
	C4	I can write clear instructions for a robot or computer to follow.	F
	C5	If my code doesn't work, I can find my mistake and fix it.	F
	C6	I've been told I would be good at coding.	
Coding	I1	I like coding, or I think I would like coding.	D, E
Interest	I2	I would like to learn more about coding.	A, D
	I3	Solving coding problems seems fun.	A

	I4	Coding is interesting.	C
	I5	I would like to study coding in the future.	C
	I6	I think I would like a job that lets me code.	B
Utility	U1	It is valuable for me to learn coding.	D
	U2	I can use coding skills in other school subjects.	A
	U3	Knowing how to code will help me to create useful things.	
	U4	Knowing how to code will help me solve problems.	
	U5	I think I will need to understand coding for my future job.	C
	U6	Learning to code will make me better in math.	
	U7	Learning to code will make me better in science.	
	U8	Learning to code will make me better in technology.	
	U9	Learning to code will make me better in language arts.	
Gender	G1	Both girls and boys should learn coding.	C
Perception	G2	Both girls and boys can learn coding.	C
	G3	Both boys and girls can do well in coding classes.	C
	G4	Girls are better at coding than boys.	C
	G5	Boys are better at coding than girls.	C
	G6	Boys like to code more than girls do.	
	G7	Girls like to code more than boys do.	
Social	S1	Coding is cool.	E
Value	S2	Kids who are good at coding are smarter than average.	B
	S3	My friends think coding is cool.	E
	S4	My parents think coding is important.	E
	S5	My teachers think coding is important.	
School	SC1	I usually do well in math.	
Attitudes	SC2	I usually do well in science.	
	SC3	I usually do well in language arts.	
	SI1	I like math.	
	SI2	I like science.	
	SI3	I like language arts.	

Non-Likert Items		
Experience	X1	How much coding have you done?
	X2	In your view, what is coding? What is its purpose?
	X3	Do you know any coders are computer programmers? If yes, how do you know them?
	X4	Describe what a coder does at work.
Perceptions of coders	P1	Complete the following statement: Kids who code are _____.
	P2	Complete the following statement: Kids who code are also good at _____.
	P3	Complete the following statement: Kids who code also like to _____.
	P4	What kind of people tend to be good at coding?

*Items were adapted from the following scales:

A. Dorn & Tew, 2015
B. Gibbons, 2004
C. Hoegh & Moskal, 2009
D. Mahoney, 2010
E. Owen et al., 2008
F. Kukul, et al., 2017

3.5 Analytical Strategy

After initial scale development, we conducted two cycles of administration and analysis. Each cycle included administering the survey to students, confirmatory factor analysis (CFA) of each construct, and structural equation model (SEM) analysis. We ran CFAs and SEMs in Mplus. Because we used a 6-point Likert scale and had continuous coverage, we treated data as continuous. We used complex analysis to account for nesting within school groups. Missing data were estimated using the full information maximum likelihood (FIML) method.

For each factor, we ran a series of CFAs, starting with all items associated with the construct. For a model to have good fit, root mean square error of approximation (RMSEA) and standardized root mean square residual (SRMR) would be less than .08, and Comparative Fit

Index (CFI) and Tucker Lewis Index (TLI) would approach or exceed .95 (Hu & Bentler, 1999). In cases where the model had good fit, we would expect each item to have a factor loading above .32 with $p<.05$ (Worthington & Whittaker, 2006). For each factor we tested several models to find the best fit, removing items with the lowest factor loadings first. For each factor we also calculated internal reliability based on Cronbach's alpha in SPSS. We eliminated factors with values below $\alpha > .7$ for factors with four or more items and $\alpha > .65$ for three-item factors (Nunnally & Bernstein, 1994).

SEM assumes that the data meet all assumptions of multiple regression, including linearity, independence of observations, normality, equality of variances, and lack of multicollinearity. Using curve estimation in SPSS we spot checked items for linearity. Among the approximately 20 pairs of items checked for linearity, none showed a significant difference between the lines produced by linear and quadratic equations. To meet assumptions of independence of observations, our analysis accounted for grade level and school groups. It is also possible that the students were nested in classroom groups, for which we did not have data. According to the Central Limit Theorem, the assumption for normality was met, since the number of students taking the survey was 324 for the pilot study and 5725 for the second survey administration. Spot checks of equality of variance showed no cone-shaped residual plots. In tested models, there was significant correlation between Confidence and Utility, so multicollinearity could be an issue. Based on high correlations, we tested a single-factor model, which had good RMSEA and SRMR values (.063 and .053) but low CFI and TLI values (.839 and .826). Though factors were highly correlated, as expected, we maintain that Confidence, Interest, Utility, Perceptions of Coders, and Social Value are separate constructs.

3.6 Pilot Study

For our initial survey administration, the 52-item survey was administered to a convenience sample of 324 students in grades 4-6 from two local elementary schools. Table 4 shows the reported characteristics of our sample, based on grade, gender, race and ethnicity, parents' education, and coding experience, by school.

Table 4

Reported Characteristics of Survey Respondents in Pilot Study

Characteristic	Category	School 1	School 2	Total
Grade	4	36	78	114
	5	39	75	114
	6	42	45	87
	Total	117	198	315
Gender	Female	57	90	147
	Male	59	98	157
	Other	1	10	11
Race/Ethnicity*	White	44	148	192
	Native Hawaiian or Pacific Islander	14	7	21
	Hispanic	47	23	70
	Black or African American	3	6	9
	Asian	4	7	11
	Native American or Alaska Native	4	6	10
	Other	15	18	33
Mom College	Yes	58	158	216
	No	10	9	19
	I don't know	47	30	77
Dad College	Yes	57	167	224
	No	12	7	19
	I don't know	45	24	69
Coding Hours	None	24	12	36
	1-10 hours	70	105	175
	>10 hours	20	79	99

*Students could select more than one option.
Due to missing data, totals may be less than 324.

As shown in Table 4, our sample included approximately equal numbers of girls and boys. Students at School 2 were predominately white, non-Hispanic, while School 1 included approximately equal numbers of white non-Hispanic and Hispanic students, and few non-white students.

3.7 Evaluation and Scale Optimization

With the data from this initial sample, we conducted CFA and SEM analyses, using procedures described in the Analytical Strategy section. Table 5 shows the best-fitting model for each of six factors.

Table 5

CFA Factors, Pilot Study

Factor	Items	Cronbach's Alpha	RMSEA	CFI	TLI	SRMR
Confidence	C1, C2, C3, C4, C5, C6	.785	.066	.975	.958	.028
Interest	I1, I2, I3, I4, I5	.896	.064	.993	.986	.014
Utility	U1, U2, U4, U5	.727	.045	.995	.985	.017
Gender Perception	G1, G2, G3	.687	.073	.974	.961	.043
Social Value	S1, S3, S4	.630	.070	.979	.970	.024
School Attitude	SC1-3, SI1-3	.674	.281	.513	.188	.138

Model fit and internal reliability statistics for the best-fitting model of each construct.

Since Gender Perception and Social Value were just-identified models, we tested each in the presence of Interest. We tested School Attitude both as a single factor and as two separate factors (School Confidence and School Interest), neither of which produced good model fit. As shown, only three factors—Confidence, Interest, and Utility—obtained both the desired fit statistics and a Cronbach's alpha above .7. Since Gender Perception and Social Value were three-item factors, the low alphas were deemed acceptable; however, because of the complexity of the model and relatively small sample, our SEM analysis included models comprised of only

the three strongest factors and 15 items. Our best-fitting SEM from the pilot study is shown in Figure 1.

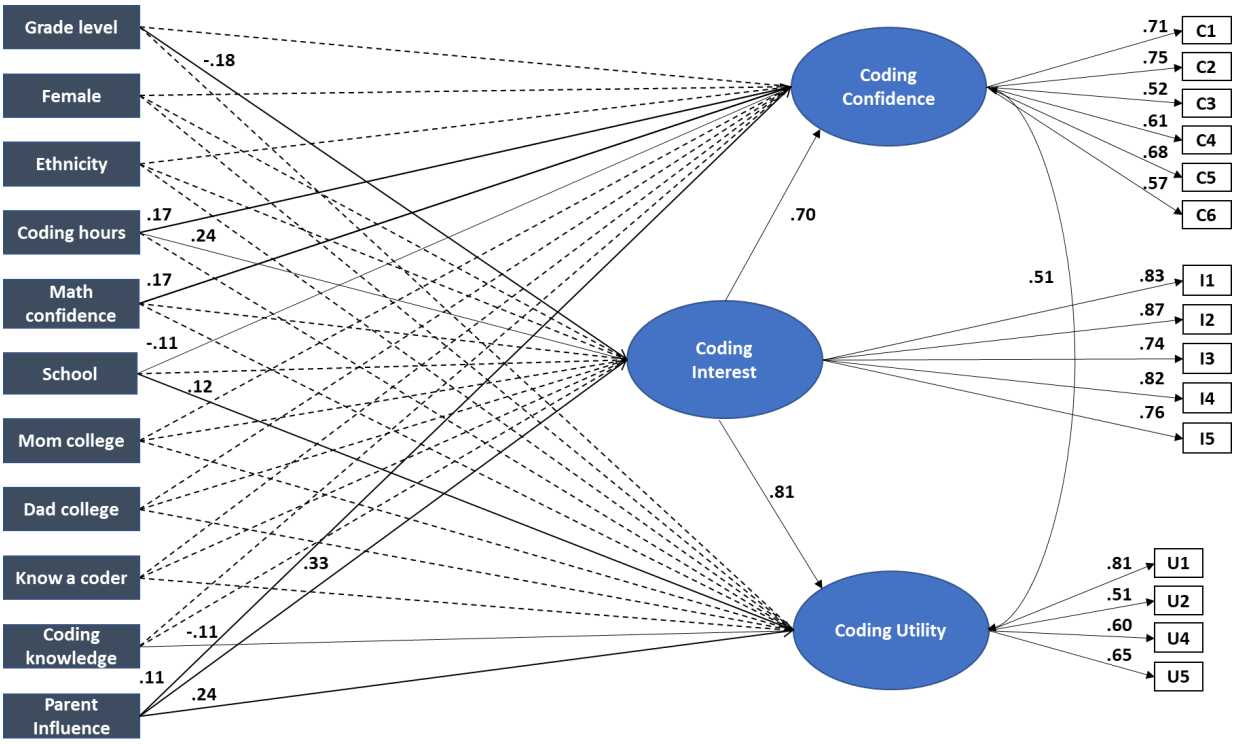

Figure 1. SEM for pilot study. Dotted lines indicate insignificant relationships. All shown values were significant, with unstandardized p<.05. Beta values are standardized: STDYX for grade level, coding hours, math confidence, and parent influence; STDY for dichotomous predictors: female, ethnicity, school, mom college, dad college, know a coder, and coding knowledge. For every one standard deviation increase in coding hours, we saw a .17 standard unit increase in Coding Confidence, holding all other variables in the model constant.

As shown in Figure 1, gender, ethnicity, parents' education, and knowing a coder were not significant predictors in this model. Grade level negatively influenced Interest; experience (coding hours) positively influenced Interest and Confidence; math confidence positively influenced Coding Confidence; and coding knowledge positively influenced perceived Coding Utility. The most influential variable was parent influence, which positively predicted Coding Interest, Confidence, and Utility. Coding Interest also significantly influenced Confidence and Utility.

Before administering the survey to a larger student population, we revised several items. The initial survey included seven open-ended items that assessed students' perceptions of coders, knowledge of and experience with coding. The purpose of open-ended items was to avoid shaping children's perceptions. However, the ESCAS is intended to be used with large student populations. The use of open-ended items would require a qualitative analysis of results after each administration. To simplify data analysis, for the second survey administration we replaced open-ended items with Likert-type scale items based on students' responses to the open-ended items in the initial administration. Using "constant comparative analysis" (Glaser & Strauss, 2017), we coded and categorized open-ended survey responses, allowing themes to emerge from the data as we looked for common patterns across cases. Using themes from the data, we generated Likert-scale items to replace the open-ended items (Table 6).

Table 6

Survey II New Items

Factor	Item	
Interest	I7	I know what coding is.
Social	S7	I am friends with kids who code.
Perceptions	St3	Kids who code like to play video games.
of coders	St4	Kids who code spend less time outside than other kids.
	St5	Kids who code enjoy doing sports.
	St6	Coders are nerdy.
	St7	Coders are good at math.
	St8	Coders are good at science.
	St9	Coders are good at language arts.

All new items except S7 replaced and were generated from responses to open-ended items in the initial survey. S7 was added to strengthen the Social Value construct.

Besides replacing nine items in the second survey administration, we also revised our hypotheses. Drawing from theory, previous studies, and our pilot study, we made the following hypotheses for our second phase:

H2-1: Grade level may negatively influence students' coding attitudes (Confidence, Interest, Utility, and Perceptions of Coders);

H2-2: Male students may have slightly more positive coding attitudes than students who identify as female or other;

H2-3: As coding frequency increases, coding attitudes may become more positive;

H2-4: As coding experience increases, coding attitudes may become more positive;

H2-5: As math confidence increases, Coding Confidence will increase;

H2-6: As math interest increases, Coding Interest will increase;

H2-7: The more students think their parents and peers valued coding, the more positive students' coding attitudes; and

H2-8: Ethnicity and parents' education would not significantly influence students' coding attitudes.

3.8 Second Survey Administration

The revised 52-item survey was administered to a sample of 5725 students in grades 4-6 from 28 elementary schools in a single school district. These students were all participating in weekly coding classes for roughly 30-45 minutes. This was part of a district-wide initiative to teach all elementary-aged children to code. Coding teachers all reported teaching the same group of students once a week for 30-45 coding sessions throughout the entire school year.

Upper-elementary students' coding experiences focused on interest-based projects in Scratch (see https://bootuppd.org/curriculum-3rd-grade-plus/).

Parental permission was sought to allow students to complete an attitude toward coding survey as well as the Computational Thinking test (Román-González, Pérez-González, & Jiménez-Fernández, 2016). Students completed the ESCAS anonymously. Using SPSS, we split the sample randomly into two approximately equal halves. Table 7 shows the reported characteristics of our sample, based on grade, gender, race and ethnicity, parents' education, and coding experience, by group. We used group 1 data for CFAs and group 2 data for SEM analysis.

Table 7

Reported Characteristics of Survey Respondents for Second Survey Administration

Characteristic	Category	Group 1	Group 2
Grade	4	930	860
	5	924	964
	6	1,044	1,003
	Total	2,898	2,827
Gender	Female	1,367	1,350
	Male	1,422	1,364
	Other	109	113
Race/Ethnicity*	White	2,062	1,989
	Native Hawaiian or Pacific Islander	62	59
	Hispanic	156	167
	Black or African American	48	59
	Asian	62	45
	Native American or Alaska Native	31	35
	Other/multiple/missing	477	473
Mom College	Yes	2,048	1,997
	No	326	302
	I don't know	524	528
Dad College	Yes	2,035	2,024
	No	312	275
	I don't know	551	528

Coding Experience	<1 year	741	739
	1-2 years	1,075	1,043
	2-3 years	613	591
	>3 years	469	454
Coding Frequency	Daily	145	130
	3 x per week	275	264
	Weekly	1,741	1,719
	Monthly	294	293
	< monthly	443	421

*Students could select more than one option. Students who chose more than one option were counted in the "multiple" category.

3.8.1 Confirmatory Factor Analysis

Using group 1 data from the second survey administration and the same methods from the pilot study, we performed confirmatory factor analysis (CFA) for each latent construct in our model: Coding Confidence, Coding Interest, perceived Utility, Gender Stereotypes, Perceptions of Coders, Social Value, and School Attitude. Table 8 shows the best-fitting model for each construct, for the second survey administration.

Table 8

CFA Factors, Second Survey Administration

Factor	Items	Cronbach's Alpha	RMSEA	CFI	TLI	SRMR
Confidence	C1, C2, C3, C4, C5, C6	.812	.068	.976	.961	.022
Interest	I1, I2, I3, I4, I5	.933	.069	.994	.988	.010
Utility	U2, U3, U4, U5	.817	.019	.999	.998	.005
Gender	G1, G2, G3	.715	.089	.975	.959	.041
Social	S3, S4, S7	.653	.060	.989	.982	.026
School	SC1-3, SI1-3	.722	.276	.571	.285	.107
Coders	St2, St5, St7, St8, St9	.752	.055	.986	.972	.019

Model fit and internal reliability statistics for the best-fitting model of each construct. Latent factor "Coders" was not included in the pilot study.

As shown in Table 8, CFA results from the second survey administration were similar to results from the pilot study CFAs, though statistics for internal reliability and fit were generally better for the larger group than for the pilot study. Again, Gender Perception and Social Value were tested in the presence of Interest. Items assessing School Attitude did not form a cogent construct. All other factors obtained desired statistics for internal reliability ($\alpha > .7$ for factors with four or more items; $\alpha > .65$ for three-item factors) and model fit in at least three of four measures (RMSEA and SRMR <. 08, CFA and TLI > .9).

Although the gender construct performed reasonably well, we chose not to include it in the SEM because it did not provide the information we had hoped the factor would provide, namely, showing perceptions of gender stereotypes. The three items forming the construct state that both boys and girls can or should code. Only 3% of participants disagreed with the statement, "Both girls and boys can learn coding" (see Figure 2). A higher portion, 14% disagreed with the statement, "Both girls and boys should learn coding"; however, responses may reveal students' value for coding rather than a belief in gender stereotypes. None of the items stating gender stereotypes loaded with the gender factor, and the paired items (e.g., "Boys are better at coding than girls." "Girls are better at coding than boys.") did not function together in a predictable way. However, separate analysis showed interesting trends in boys' and girls' responses. As shown in Figure 2, 36% of girls said that girls are better at coding, and 36% of boys said that boys are better at coding; in other words, across genders and in equal proportions, students disagreed with the idea that members of their own gender were better at coding. Both groups emphatically rejected the idea that members of the opposite gender were better at coding; however, girls were more likely than boys to reject the idea (83% vs 77%).

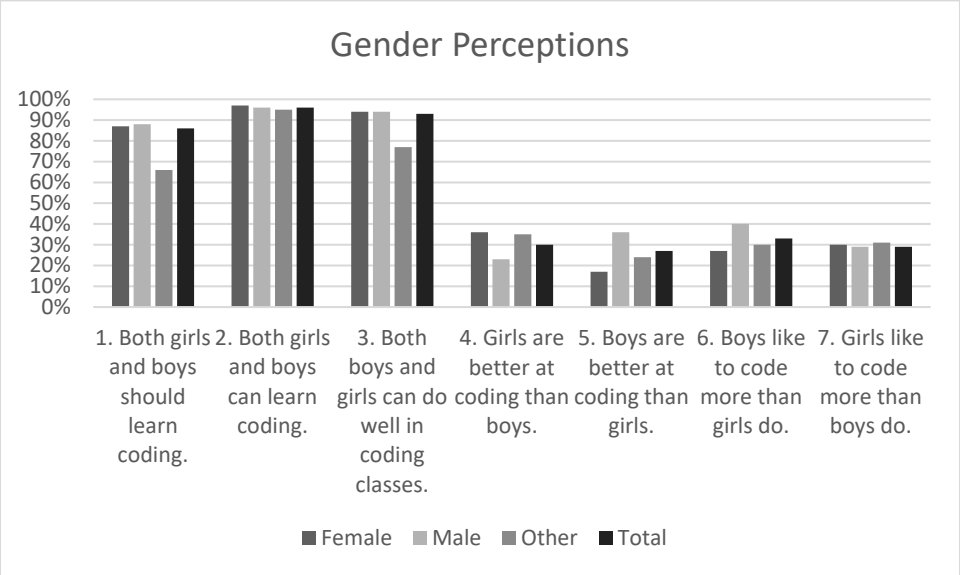

Figure 2. Gender perceptions. The figure shows the portion of participants who agreed with each statement, by gender identity.

Across genders, approximately 70% of students disagreed with the statement, "Girls like to code more than boys do." A higher portion of girls, 73%, disagreed with the statement, "Boys like to code more than girls do," while only 60% of boys disagreed with the latter statement. The responses to items G1-G7 suggest that most students in the study did not embrace gender stereotypes about coding. Although items G4-G7 were not part of a strong factor, researchers using ESCAS and hoping to assess changes in perceptions of gender stereotypes over time may wish to include items such as G4-G7 for separate analysis. We kept our other just-identified construct, Social Value, which had better model fit than the Gender construct and, more importantly in this case, provided useful information. Table 9 shows the final constructs and items included in our SEM.

Table 9

Final Factors and Items

Factor	Item	
Coding	C1	I can learn to code.
Confidence	C2	I am good at coding.
	C3	I am good at problem solving.
	C4	I can write clear instructions for a robot or computer to follow.
	C5	If my code doesn't work, I can find my mistake and fix it.
	C6	I've been told I would be good at coding.
Coding	I1	I like coding, or I think I would like coding.
Interest	I2	I would like to learn more about coding.
	I3	Solving coding problems seems fun.
	I4	Coding is interesting.
	I5	I would like to study coding in the future.
Utility	U2	I can use coding skills in other school subjects.
	U3	Knowing how to code will help me to create useful things.
	U4	Knowing how to code will help me solve problems.
	U5	I think I will need to understand coding for my future job.
Social	S3	My friends think coding is cool.
Value	S4	My parents think coding is important.
	S7	I am friends with kids who code.
Perceptions	S2	Kids who are good at coding are smarter than average.
of coders	St5	Kids who code enjoy doing sports.
	St7	Coders are good at math.
	St8	Coders are good at science.
	St9	Coders are good at language arts.

Final five factors and 23 items.

3.8.2 Measurement invariance across groups

To ensure that the scale was performing the same across genders and grade levels, we tested for configural, metric, and scalar invariance. First, we ran a CFA for each gender group (Table 6). All three gender groups showed good fit statistics (RMSEA and SRMR < .08; CFI and TLI > .9). Next, we tested configural, metric, and scalar models in Mplus (Table 10). According to Chen's (2007) guidelines for goodness of fit, change in CFI from one model to the next should be no more than 0.010, and preferably no more than 0.005. Change in CFI from the configural to the metric model was -0.001; change in CFI from the metric to the scalar model was -.004. Therefore, the scale had measurement invariance across gender, indicating that the same constructs were being assessed, regardless of gender.

Table 10

Measurement Invariance Across Gender

	RMSEA	CFI	TLI	SRMR
Female	.046	.953	.947	.039
Male	.052	.943	.935	.040
Other	.054	.952	.946	.058
Configural	.049	.948	.941	.040
Metric	.049	.947	.942	.046
Scalar	.049	.943	.941	.048

We used the same procedures and criteria to test measurement invariance across grade levels. Table 11 shows fit statistics for each grade level group, configural, metric, and scalar models. All three grade level groups showed good fit statistics (RMSEA and SRMR < .08; CFI and TLI > .9). Change in CFI from the configural to the metric model was -0.003; change in CFI from the metric to the scalar model was -.001. Using Chen's (2007) criteria, the scale had

measurement invariance across grade level, suggesting that the scale is measuring the same construct across grade levels and can be used for students in grades 4-6.

Table 11

Measurement Invariance Across Grade Level

	RMSEA	CFI	TLI	SRMR
Grade 4	.047	.944	.936	.040
Grade 5	.052	.941	.933	.042
Grade 6	.051	.955	.948	.042
Configural	.051	.950	.942	.042
Metric	.051	.947	.942	.047
Scalar	.051	.946	.944	.046

3.9 Structural Equation Model

Once we had identified reliable factors and items, we created a structural equation model, shown in Figure 3. We used our model to test the effects of grade level, ethnicity, gender, coding frequency, coding experience, math confidence, and math interest on Coding Confidence, Coding Interest, Social Value, Perception of Coders, and Coding Utility.

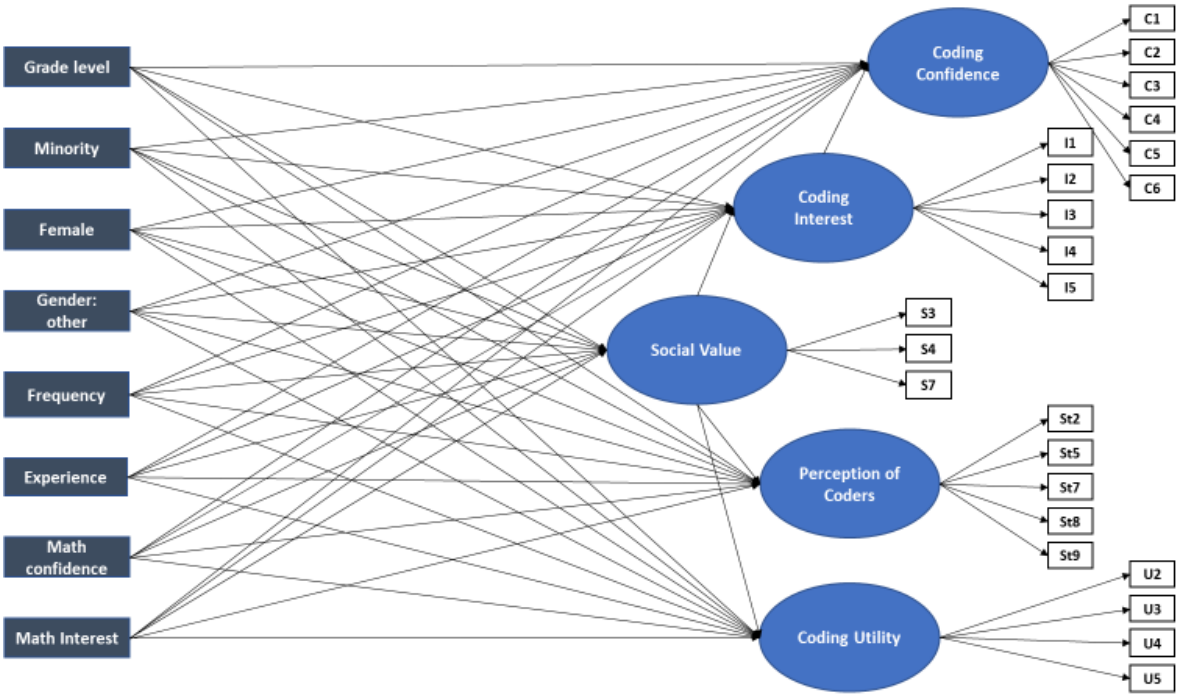

Figure 3. Hypothesized model.

4. Results

Our hypothesized model showed good model fit, as shown in Table 12. Numerous alternative models were also tested, but among tested models, this model obtained the best fit and explained the greatest amount of variance, indicated by R-squared values. Figure 3 shows the final model with coefficient values added and nonsignificant pathways removed.

Table 12

SEM Results

RMSEA	CFI	TLI	SRMR	R^2-Con	R^2-Int	R^2-Util	R^2-Soc	R^2-Cod
.037	.950	.940	.032	.752	.829	.901	.307	.471

Con=Coding Confidence
Int= Coding Interest
Util= Perceived Utility
Soc=Social Value
Cod=Perceptions of Coders

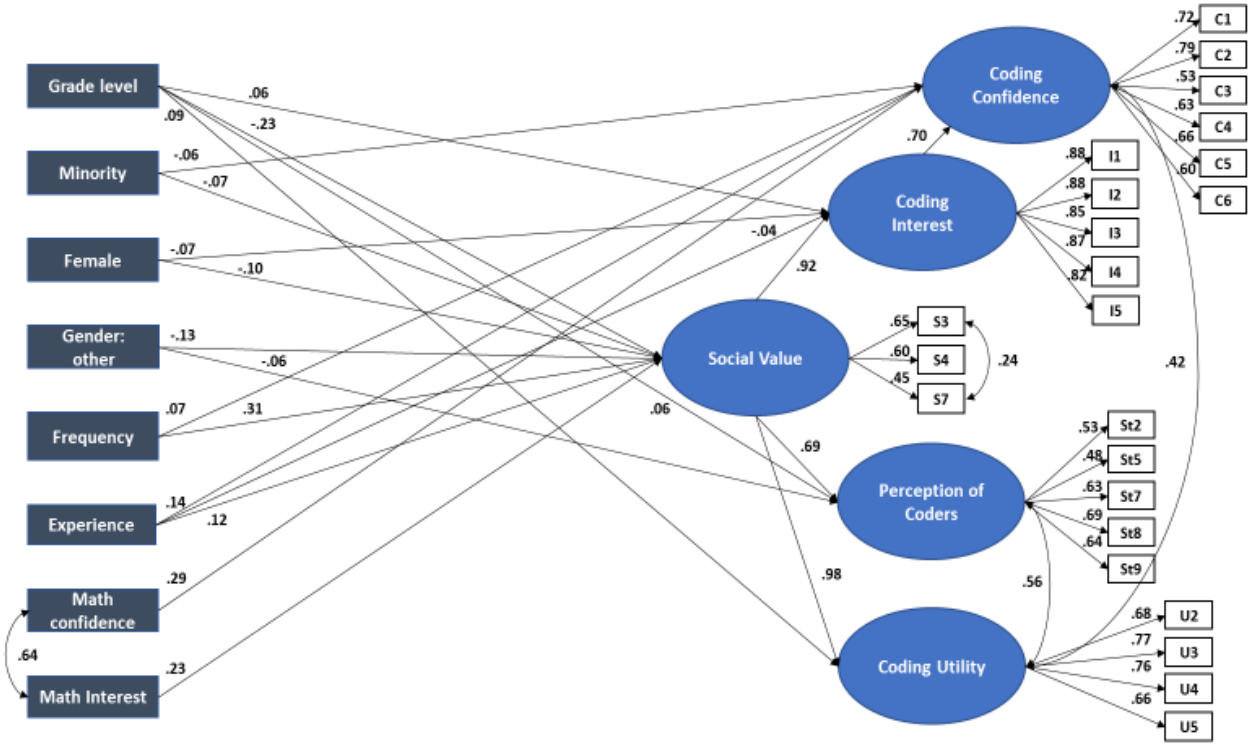

Figure 4. SEM results. Nonsignificant paths have been removed from the model; all shown values were significant, with unstandardized p<.05. Beta values are standardized: STDYX for grade level, frequency, experience, math confidence, and math interest; STDY for dichotomous predictors: minority, female, gender: other. RMSEA = 0.037; CFI = 0.950; TLI = 0.940; SRMR = 0.032.

As shown in Figure 4, seven variables predicted Social Value, which influenced Coding Interest, Perception of Coders, and Coding Utility. Coding Interest predicted Coding Confidence (β = 0.70). Of the seven independent variables that predicted Social Value, frequency had the greatest influence (β = 0.31), followed by math interest (β = 0.23) and grade level (β = -0.23). The more frequently children coded and the greater their interest in math, the more likely they were to say that their parents and friends valued coding. Experience also positively predicted Social Value (β = 0.12). Grade level had a negative effect: as grade level increased, students were less likely to say that their parents and friends valued coding. Gender and ethnicity had a statistically significant but only slightly negative effect.

Social Value significantly predicted Coding Interest (β = 0.92), Perception of Coders (β = 0.69), and Utility (β = 0.98). The more students felt their parents and friends valued coding, the greater a student's interest in coding. Grade level, gender, and Experience also had statistically significant but slight effects on Interest. The more students felt their parents and friends valued coding, the more positive a student's perception of coders. Grade level and gender also had statistically significant but slight effects on Perception of Coders. The more a student felt their parents and friends valued coding, the more the student valued coding. Grade level also had a slight positive effect on Coding Utility.

Coding Interest significantly predicted Coding Confidence (β = 0.70). The greater a student's interest in coding, the greater that student's coding self-efficacy. Math confidence also had a strong, positive influence on Coding Confidence (β = 0.29). Coding experience and frequency had smaller, positive effects on Coding Confidence. Ethnicity had a slightly negative influence on Coding Confidence.

5. Discussion

Based on our pilot study results, we hypothesized that grade level may negatively influence students' coding attitudes (Confidence, Interest, Utility, and Perceptions of Coders). Our analysis showed that indeed, as grade level increased, Social Value decreased, and Social Value positively influenced all other factors, directly or indirectly. However, the negative effect of grade on attitude was slightly mitigated by the positive direct effect of grade on Coding Interest, perception of coders, and coding Utility.

In our pilot study, gender did not have a significant effect on students' coding attitudes; however, in other studies (Beyer, 2014; Dorn & Tew, 2015; Kong et al., 2018; Mahoney, 2010; Owen et al., 2008; Ramalingam & Wiedenbeck, 1998) males had more confidence or interest in

CS or STEM subjects than did female students. Therefore, we anticipated a small effect for gender (H2-1). Our analysis showed that compared to male students, students who identified as female or other had slightly lower scores for Social Value (β = -.10; β = -.13). Compared to males, females indicated slightly lower Coding Interest (β = -.07), and students who identified as other had slightly less positive Perceptions of Coders (β = -.06). These differences were small but statistically significant. It may be that gender biases toward coding are naturally less developed by elementary students. The results from the current study demonstrate this slightly, indicating that older students experience more gender bias toward coding. Future use of the ESCAS may serve as a measure to see if, over time, students who engage in coding develop these gender biases at a lesser rate than those who do not (or, hopefully, reverse the trend!).

We hypothesized that as coding frequency increased, coding attitudes would become more positive (H2-3). Among all observable variables, frequency had the greatest influence on Social Value (β = .31), which in turn substantially influenced all other factors. Besides its indirect effect on Coding Confidence, coding frequency had an additional, direct, slightly positive effect on Coding Confidence (β = .07). As expected, coding experience also had a net positive effect on coding attitudes, with direct effects to Social Value (β = .12), Coding Confidence (β = .14), and Coding Interest (β = -.04).

We hypothesized that math confidence (H2-5) and interest (H2-6) would predict Coding Confidence and Interest. In our model, math confidence was the variable with the greatest predictive power for Coding Confidence (β = .29). Math interest had high predictive power for Social Value (β = .23), which substantially influenced Coding Interest (β = .92). This correlation between math and coding reinforces findings from others' research (Rich, Leatham, & Wright, 2013; Scherer, et al., 2018), and serves to strengthen the notion that those who feel comfortable

with and interested in mathematics are more likely to also feel that way toward coding, even at the elementary level.

We also hypothesized that the more students thought their parents and peers valued coding (H2-7), the more positive the students' coding attitudes. Our analysis showed this hypothesis to be true: Social Value positively influenced Coding Interest, Utility, and perception of coders, and Coding Interest positively influenced Coding Confidence. Social Value itself played a revealing role in this study, strongly influencing all other factors. The three questions that accounted for Social Value (i.e., "My parents think coding is important", "I have friends who code," and "My friends think coding is cool") demonstrate that elementary students' attitudes toward coding are strongly filtered through their social relations. It is curious that the influence of the students' teacher ("My teacher thinks coding is important) did not adequately load onto this factor; students' perceptions of their teachers' value of coding did not consistently predict students' attitudes toward coding. More research is needed to adequately explore the influence that parents and peers exert on elementary students' attitudes toward coding. Social value could be divided into two constructs, family models and peer models, as done in the STAQ-R (Owen et al., 2008). More items would be needed to measure this Social Value effect.

In our pilot study, ethnicity and parents' education were not significant predictors. In the second administration, parents' education remained insignificant and therefore was removed from the model. Ethnicity had a slight but statistically significant influence on Social Value and Coding Confidence. Most studies we reviewed did not mention ethnicity as a variable. Of two that did, one found a significant effect for ethnicity on IT interest (Forssen et al., 2011), the other found no significant effect for ethnicity on STEM self-efficacy (Hirsch et al., 2003). Our large

sample was less diverse than our pilot study sample. It may be that as minority population decreases, influence of ethnicity increases, though, again, the influence was small.

The ESCAS did not assess negative perceptions of stereotypes about gender or coders. We were able to form constructs related to gender perceptions and Perceptions of Coders, but items conveying negative stereotypes did not perform consistently or load strongly to the factors. It may be that students were too young to have formed negative stereotypes (Chambers, 1983), or that young students were afraid to share negative stereotypes. For example, in one case, we received an email from a teacher who said her students were concerned about the message inherent in the statement, "Coders are nerdy." Curiously, we created this item as a response to other elementary students' open-ended comments to this effect. It is possible that similar scale items about negative stereotypes would perform more consistently with older students.

5.1 Limitations

While we took several steps to ensure that the ESCAS is a reliable and valid scale for measuring elementary students' attitudes toward coding, there are a few limitations to its development. First, the area we live in is less racially diverse than many other locales; thus, the ESCAS may not be representative of all other groups or may not have been as sensitive to issues of racial diversity or socio-economic status. In addition, all students in the second survey administration had participated in weekly, in-school coding class for 30-45 minutes per week, for several months or more. While the ESCAS was written in a way that it could be answered by students with little to no coding experience, it may perform differently among students who have not coded or who are not receiving regular coding instruction.

Although the model explains 30-90 percent of variance for each of the five constructs, observable variables (grade, gender, ethnicity, frequency, experience) account for a relatively

small portion of variance on their own. For all factors other than Social Value, unobservable variables explained most of the variance. For example, a student's Social Value score predicts a student's Coding Interest, Utility, and perception of coders. One latent variable predicts another. The observable variable with the greatest influence was reported frequency—how often students said they code, which itself could be predicted by a student's coding attitude.

6. Conclusion

We set out to better understand what factors affect elementary student attitudes toward coding (RQ1) and how to measure those (RQ2). With increasing pressure to teach coding at the elementary level, educators, administrators, and researchers need a way to measure the effect these programs have on young children. To answer our questions, we reviewed existing computer science attitude scales, but found these lacking in one dimension or another. In order to have a multi-faceted understanding of attitude (interest, Utility, self-efficacy, and social bias), we developed and validated the Elementary Student Coding Attitudes Survey (ESCAS), a 23-item instrument to assess elementary students' coding attitudes and self-efficacy. Using data from nearly 6000 4^{th}-6^{th} grade students, we conducted a confirmatory factor analysis (CFA) and built a structural equation model. As a response to RQ1, the CFA identified five strong factors: Coding Confidence, Coding Interest, Social Value, Perceptions of Coders, and Coding Utility. This combination of factors has not been used in similar scales for elementary students (Chambers 1983; Hansen et al., 2017; Knight & Cunningham, 2004; Kong et al., 2018). We had hoped to include a sixth factor to assess students' perceptions of gender stereotypes and coding; however, our items did not form a strong construct to assess gender perceptions. Understanding and assessing this important, nuanced construct will require further study.

Using a structural equation model (SEM) helped us further examine the influence of several predictors (RQ2) to the factors identified as making up attitude (RQ1). Previous analyses have tended to focus on two or three predictors, principally demographic in nature. Using SEM revealed that students' grade level, ethnicity, gender, coding frequency, coding experience, and math interest all influenced Social Value, which in turn influenced Coding Interest, Perception of Coders, and Coding Utility. Students' math confidence, coding frequency, coding experience, ethnicity, and Coding Interest predicted their Coding Confidence.

Two findings from this analysis were particularly revealing: first, among observable variables, coding frequency had the greatest influence on outcome variables. Thus, it is important to engage students in coding often. An "hour of code" or once a month coding lessons may be insufficient to positively encourage children toward coding. The finding that frequent coding positively influenced attitudes is consistent with previous findings that coding experience improves attitudes (Gunbatar & Karalar, 2018; Kalelioğlu 2015; Master, et al., 2017; Rubio et al., 2015; Sáez-López, et al., 2016; Shim et al., 2017; Sullivan & Bers, 2019). Second, Social Value—parent and peer influence—had a substantial, significant mediating effect on young students' confidence with, interest in, and perception of coding. Many programs that encourage elementary coding are directed toward the individual student or to educators. Given the results of this study, it might be more effective to consider the child's social sphere as an essential factor in influencing their attitudes toward coding. Future research might consider examining attitudes through an ecological model, such as that proposed by Bronfenbrenner (1994).

More and more elementary educators are teaching coding to young children (Rich et al., 2018). In order to measure its effect on young children, we need instruments to measure both cognitive and affective effects of learning to code on young children. The ESCAS is a tool that is

relatively quick to administer (5-10 minutes), that specifically targets young learners' coding self-efficacy, value, interests, and perceptions. We hope to be able now use the scale to assess changes in attitude toward coding over time.

Declaration of Interest

This work is original and has not been published or submitted for publication in any other venue. The opinions expressed herein are solely those of the authors.

Funding Source

This research did not receive any specific grant from funding agencies in the public, commercial, or not-for-profit sectors.

7. References

Balakrishnan, A. (17 May 2018). Fascinating debates: Should everyone learn to code? Medium.com. Retrieved from: https://medium.com/@akbgunner4ever/fascinating-debates-should-everyone-learn-to-code-4354fba30bbf

Bandura, A. (2006). Guide for constructing self-efficacy scales. In T. Urdan & F. Pajares (Eds.), *Self-efficacy beliefs of adolescents* (pp. 307-337). Charlotte, NC: Information Age Publishing.

Beyer, S. (2014). Why are women underrepresented in Computer Science? Gender differences in stereotypes, self-efficacy, values, and interests and predictors of future CS course-taking and grades. *Computer Science Education, 24*(2-3), 153-192.

Bronfenbrenner, U. (1994). Ecological models of human development. In M. Gauvain and M. Cole (Eds.), *Readings on the development of children* (4th ed., pp. 3-9). New York, NY: Worth Publishers.

Bureau of Labor Statistics (2019). Computers and information technology. Retrieved from: https://www.bls.gov/ooh/computer-and-information-technology/home.htm

Chambers, D. W. (1983). Stereotypic images of the scientist: The Draw-a-Scientist Test. *Science Education, 67*(2), 255-265.

Chen, F. F. (2007). Sensitivity of goodness of fit indexes to lack of measurement invariance. *Structural Equation Modeling, 14*(3), 464-504.

Cheryan, S., Master, A., & Meltzoff, A. N. (2015). Cultural stereotypes as gatekeepers: Increasing girls' interest in computer science and engineering by diversifying stereotypes. *Frontiers in Psychology, 6*, 49. https://doi.org/10.3389/fpsyg.2015.00049

DeVellis, R. (2017). *Scale development* (4th ed.). Los Angeles, CA: Sage.

Doerschuk, P., Liu, J., & Mann, J. (2007). Pilot summer camps in computing for middle school girls. *ACM SIGCSE Bulletin, 39*(3), 4-8. https://doi.org/10.1145/1269900.1268789

Dorn, B. & Tew, A. E. (2015). Empirical validation and application of the Computing Attitudes Survey. *Computer Science Education, 25*(1):1-36.

Eccles J. S., Adler, T. F., Futterman, R., Goff, S. B., Kaczala, C. M., Meece, J. L., & Midgley, C. (1983). Expectancies, values, and academic behaviors. In J. T. Spence (Ed.), *Achievement and achievement motivation* (pp. 75–146). San Francisco, CA: W. H. Freeman.

Erkut, S. & Marx, F. (2005). 4 schools for WIE (Evaluation Report). Wellesley, MA: Wellesley College, Center for Research on Women. Retrieved January 2, 2012 from http://www.coe.neu.edu/Groups/stemteams /evaluation.pdf

Forssen, A. V., Moskal, B. M., & Harriger, A. R. (2011). Measuring the impact of a high school intervention on students' attitudes in information technology: Validation and use of an attitude survey. *The ASEE Computers in Education (CoED) Journal, 3*(2), 2.

Gibbons, S.J., Hirsch, L.S., Kimmel, H., Rockland, R., & Bloom, J. (2004). Middle school students' attitudes to and knowledge about engineering. Paper presented at ICEE Conference 2004, Gainesville, FL.

Glaser, B. G., & Strauss, A. L. (2017). *Discovery of grounded theory: Strategies for qualitative research*. New York: Routledge. https://doi.org/10.4324/9780203793206

Grossnickle, E. M. (2016). Disentangling curiosity: Dimensionality, definitions, and distinctions from interest in educational contexts. *Educational Psychology Review, 28*(1), 23-60.

Gunbatar, M. S. (2018). Examination of undergraduate and associate degree students' computer programming attitude and self-efficacy according to thinking style, gender and experience. *Contemporary Educational Technology, 9*(4), 354-373.

Gunbatar, M. S., & Karalar, H. (2018). Gender differences in middle school students' attitudes and self-efficacy perceptions towards mBlock programming. *European Journal of Educational Research, 7*(4), 925-933. doi: 10.12973/eu-jer.7.4.923

Hansen, A. K., Dwyer, H. A., Iveland, A., Talesfore, M., Wright, L., Harlow, D. B., & Franklin, D. (2017, March). Assessing children's understanding of the work of computer scientists: the draw-a-computer-scientist test. In *Proceedings of the 2017 ACM SIGCSE technical symposium on computer science education* (pp. 279-284). ACM.

Hirsch, L. S., Carpinelli, J. D., Kimmel, H., Rockland, R., & Bloom, J. (2007, October). The differential effects of pre-engineering curricula on middle school students' attitudes to and knowledge of engineering careers. In *Frontiers in Education Conference-Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007. FIE'07. 37th Annual* (pp. S2B-17-21). IEEE.

Hirsch, L. S., Gibbons, S. J., Kimmel, H., Rockland, R., & Bloom, J. (2003, November). High school students' attitudes to and knowledge about engineering. In *Frontiers in Education Conference, 2003. FIE'03. 33rd IEEE* (pp. F2A-7-12). IEEE.

Hoegh, A., & Moskal, B. M. (2009, October). Examining science and engineering students' attitudes toward computer science. In *Frontiers in Education Conference, 2009. FIE'09. 39th IEEE* (pp. W1G-1-6). IEEE.

Hu, L. T., & Bentler, P. M. (1999). Cutoff criteria for fit indexes in covariance structure analysis: Conventional criteria versus new alternatives. *Structural Equation Modeling: A Multidisciplinary Journal, 6*(1), 1-55.

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior, 52*, 200-210. https://doi.org/10.1016/j.chb.2015.05.047

Knight, M., & Cunningham, C. (2004, June). Draw an engineer test (DAET): Development of a tool to investigate students' ideas about engineers and engineering. In *ASEE Annual Conference and Exposition* (Vol. 2004). ASEE.

Kong, S. C., Chiu, M. M., & Lai, M. (2018). A study of primary school students' interest, collaboration attitude, and programming empowerment in computational thinking education. *Computers & Education, 127*, 178-189. https://doi.org/10.1016/j.compedu.2018.08.026

Kukul, V., Gökçearslan, Ş., and Günbatar, M. S. (2017). Computer programming self-efficacy scale (CPSES) for secondary school students: Development, validation and reliability. *Educational Technology Theory and Practice, 7*(1), 158-179. doi:10.17943/ETKU.72918

Mahoney, M. P. (2010). Students' Attitudes toward STEM: Development of an instrument for high school STEM-based programs. *Journal of Technology Studies*, *36*(1), 24-34.

Manzano-Sanchez, H., Outley, C., Gonzalez, J. E., & Matarrita-Cascante, D. (2018). The influence of self-efficacy beliefs in the academic performance of Latina/o students in the United States: A systematic literature review. *Hispanic Journal of Behavioral Sciences, 40*(2), 176-209.

Master, A., Cheryan, S., Moscatelli, A., & Meltzoff, A. N. (2017). Programming experience promotes higher STEM motivation among first-grade girls. *Journal of Experimental Child Psychology, 160*, 92-106.

Multon, K. D., Brown, S. D., & Lent, R. W. (1991). Relation of self-efficacy beliefs to academic outcomes: A meta-analytic investigation. *Journal of Counseling Psychology, 38*(1), 30.

National Science Foundation. (2017). TABLE 5-2. Bachelor's Degrees Awarded, by Field and Sex: 2004–2014. Retrieved 11/9/2018 from https://www.nsf.gov/statistics/2017/nsf17310/static/data/tab5-2.pdf

Nunnally, J., & Bernstein, I. (1994). *Psychometric theory* (3rd ed.). New York, NY: McGraw-Hill.

Owen, S. V., Toepperwein, M. A., Marshall, C. E., Lichtenstein, M. J., Blalock, C. L., Liu, Y., ... & Grimes, K. (2008). Finding pearls: Psychometric reevaluation of the Simpson–Troost Attitude Questionnaire (STAQ). *Science Education*, *92*(6), 1076-1095.

Pajares, F., & Schunk, D. H. (2001). Self-beliefs and school success: Self-efficacy, self-concept, and school achievement. In R. Riding & S. G. Rayner (Eds.), *Self-perception* (pp. 239-266). Westport, CT: Greenwood Publishing Group.

Pintrich, P. R., & De Groot, E. V. (1990). Motivational and self-regulated learning components of classroom academic performance. *Journal of Educational Psychology, 82*(1), 33-40.

Ramalingam, V., & Wiedenbeck, S. (1998). Development and validation of scores on a computer programming self-efficacy scale and group analyses of novice programmer self-efficacy. *Journal of Educational Computing Research, 19*(4), 367-381. doi: 10.2190/C670-Y3C8-LTJ1-CT3P

Reeve, S., Kitchen, E., Sudweeks, R. R., Bell, J. D., & Bradshaw, W. S. (2011). Development of an instrument for measuring self-efficacy in cell biology. *Journal of Applied Measurement, 12*(3), 242-260.

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, 63*(3), 311-329. doi:10.1007/s11528-018-0295-4

Rich, P. J., Leatham, K. R., & Wright, G. A. (2013). Convergent cognition. *Instructional Science, 41*(2), 431-453. Retrieved from https://link.springer.com/article/10.1007/s11251-012-9240-7

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2016). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking test. *Computers in Human Behavior, 72,* 678-691. http://dx.doi.org/10.1016/j.chb.2016.08.047

Rubio, M. A., Romero-Zaliz, R., Mañoso, C., & Angel, P. (2015). Closing the gender gap in an introductory programming course. *Computers & Education, 82*, 409-420.

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two-year case study using "Scratch" in five schools. *Computers & Education, 97*, 129-141. http://dx.doi.org/10.1016/j.compedu.2016.03.003

Scherer, R., Siddiq, F., & Sánchez Viveros, B. (2019). The cognitive benefits of learning computer programming: A meta-analysis of transfer effects. *Journal of Educational Psychology, 111*(5), 764-792. doi:10.1037/edu0000314

Schiefele, U., Krapp, A., & Winteler, A. (1992). Interest as a predictor of academic achievement: A meta-analysis of research. In K. A. Renninger, S. Hidi, & A. Krapp (Eds.), *The role of interest in learning and development* (pp. 183-212). Hillsdale, NJ: Lawrence Erlbaum.

Shim, J., Kwon, D., & Lee, W. (2016). The effects of a robot game environment on computer programming education for elementary school students. *IEEE Transactions on Education, 60*(2), 164-172.

Suldo, S. M., & Shaffer, E. J. (2007). Evaluation of the self-efficacy questionnaire for children in two samples of American adolescents. *Journal of Psychoeducational Assessment, 25*(4), 341-355.

Sullivan, A., & Bers, M. U. (2019). VEX robotics competitions: Gender differences in student attitudes and experiences. *Journal of Information Technology Education*, *18*, 97-112. https://doi.org/10.28945/4193.

Taub, R., Armoni, M., & Ben-Ari, M. (2012). CS unplugged and middle-school students' views, attitudes, and intentions regarding CS. *ACM Transactions on Computing Education (TOCE), 12*(2), 8.

Washington, A. N., Grays, S., & Dasmohapatra, S. (2016). The Computer Science Attitude and Identity Survey (CSAIS): A novel tool for measuring the impact of ethnic identity in underrepresented computer science students. In Proceedings of the *ASEE's 123rd Annual Conference & Exposition 2016*. Retrieved from https://www.asee.org/search/proceedings

Wentzel, K. R., & Wigfield, A. (1998). Academic and social motivational influences on students' academic performance. *Educational Psychology Review, 10*(2), 155-175.

Wigfield, A., & Cambria, J. (2010). Students' achievement values, goal orientations, and interest: Definitions, development, and relations to achievement outcomes. *Developmental Review, 30*(1), 1-35.

Wigfield, A., & Eccles, J. S. (2000). Expectancy–value theory of achievement motivation. *Contemporary Educational Psychology, 25*(1), 68-81.

Wininger, S. R., Adkins, O., Inman, T. F., & Roberts, J. (2014). Development of a student interest in mathematics scale for gifted and talented programming identification. *Journal of Advanced Academics, 25*(4), 403-421.

Witherspoon, E. B., Schunn, C. D., Higashi, R. M., & Baehr, E. C. (2016). Gender, interest, and prior experience shape opportunities to learn programming in robotics competitions. *International Journal of STEM Education, 3*(1), 18.

Worthington, R.L. & Whittaker, T.A. (2006). Scale development research: A content analysis and recommendations for best practices. *The Counseling Psychologist, 34*, 806-838.

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., & Korb, J. T. (2011, March). Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM technical symposium on Computer science education* (pp. 465-470). ACM.

DISSERTATION CONCLUSION

This dissertation includes three articles in which we explored teachers' and students' self-efficacy and attitudes toward learning computing, coding, and computational thinking. As the demand for coding instruction has increased, elementary schools have started to provide coding instruction. However, most elementary school teachers lack the skills and confidence to teach computing. To better understand how to effectively prepare teachers to teach computing, in our first article we examined recent research on programs to prepare elementary school teachers to teach computing, coding, or computational thinking. We found that training that includes active participation has improved teachers' computing self-efficacy, attitudes, and knowledge. However, few interventions in our review included practice teaching computing, and none of the studies examined student outcomes after interventions. The second and third articles help to address these two gaps in the literature.

The second paper describes a professional development program for preparing K–6 teachers to teach computational thinking concepts using robots. Key components of the intervention were modeling, practice, instruction, planning, and implementation. The purpose of the intervention was to provide teachers with the opportunity to gain the skills and confidence to use robots in the classroom, and all five teachers showed increased skills and confidence, and successfully taught their students to use robots. In surveys and informal interviews, the teachers indicated that they valued the in-school training as well as informal learning activities engaged in during implementation. The findings support previous research suggesting that positive experiences improve self-efficacy.

In the third article, we further explored factors that influence self-efficacy for coding, this time focusing on students' attitudes and beliefs. In the article, we described how we developed

the Elementary Student Coding Attitudes Survey (ESCAS). Confirmatory Factory Analysis suggested five constructs that comprise young learners' attitudes toward coding: Social Value, Coding Confidence, Coding Interest, Perception of Coders, and Coding Utility. In our analysis, coding frequency and math interest had the greatest influence on Social Value, or how children perceive their peers' and parents' value of coding, which substantially influenced all other constructs.

A theme that ran through all three articles is that experience improves self-efficacy and attitudes. This finding is not new but supports Bandura's theory of self-efficacy as well as previous research findings. To prepare teachers to teach coding and computational thinking, the professional development program we developed included practice both coding and teaching coding. Coding programs for children should likewise include frequent practice. More research is needed to understand long-term outcomes for teachers and students from teacher professional development interventions. Our scale, the ESCAS, is a tool that may be used to assess changes in student attitudes over time, and therefore help researchers evaluate and improve teacher professional development interventions to prepare elementary teachers to teach coding.

APPENDIX A

IRB Approval, "Professional Development for Teaching Computational Thinking with Robots: An Exploratory Study"

INSTITUTIONAL REVIEW BOARD
FOR HUMAN SUBJECTS

Memorandum

To: Stacie Mason

Department: IP&T

College: EDUC

From: Sandee Aina, MPA, IRB Administrator

Bob Ridge, PhD, IRB Chair

Date: February 23, 2018

IRB#: E18065

Title: *"Developing Elementary Teachers' Self-Efficacy to Teach Computing"*

Brigham Young University's IRB has approved the research study referenced in the subject heading as exempt level, category 1. The approval period is from **February 23, 2018 to February 22, 2019**. Please reference your assigned IRB identification number in any correspondence with the IRB. Continued approval is conditional upon your compliance with the following requirements:

1. CONTINGENCIES
   - Submit the recruiting script
   - Submit Provo School District approval
   - Add mentor's name and contact information to the "Questions" section of the consent form.
2. A copy of the informed consent statement is attached. No other consent statement should be used. Each research subject must be provided with a copy or a way to access the consent statement.
3. Any modifications to the approved protocol must be submitted, reviewed, and approved by the IRB before modifications are incorporated in the study.
4. All recruiting tools must be submitted and approved by the IRB prior to use.
5. In addition, serious adverse events must be reported to the IRB immediately, with a written report by the PI within 24 hours of the PI's becoming aware of the event. Serious adverse events are (1) death of a research participant; or (2) serious injury to a research participant.
6. All other non-serious unanticipated problems should be reported to the IRB within 2 weeks of the first awareness of the problem by the PI. Prompt reporting is important, as unanticipated problems often require some modification of study procedures, protocols, and/or informed consent processes. Such modifications require the review and approval of the IRB.
7. A few months before the expiration date, you will receive a continuing review form. There will be two reminders. Please complete the form in a timely manner to ensure that there is no lapse in the study approval.

IRB Secretary

A 285 ASB

Brigham Young University

(801)422-3606

APPENDIX B

Informed Consent, "Professional Development for Teaching Computational Thinking with Robots: An Exploratory Study"

Consent to be a Research Subject

Introduction

This research study is being conducted by Stacie Mason under the supervision of Rick West at Brigham Young University to help understand K-6 teachers' a) confidence about using robots, b) beliefs about the benefits of using robots in the classroom, and c) professional development needs and preferences. You were invited to participate because you indicated an interest in participating in professional development training.

Procedures

If you agree to participate in this research study, the following will occur:

- You will be asked to complete a 10-question pre-training survey
- I will provide professional development training in which
  - I demonstrate the use of robots
  - You practice using robots
  - I provide instruction about computational thinking and technology integration
  - I try to answer your questions
  - We collaboratively plan lessons that use robots to meet specific learning objectives
- During this training I will take notes about your questions and comments and our activities. The meeting will not be recorded.
- You will have 1-3 weeks to use the robots in your classroom.
- When I collect the robots, I will ask you to complete a 12-question survey.
- In an informal interview, I will ask you a few questions about your experience using robots to teach.
- I will ask you if you would like to share any of the lesson plans you used to teach with robots.
- All surveys, informal interviews, and instruction will take place in your classroom or a room of your choice.
- Surveys and informal interviews will take approximately 5 minutes each.
- Training will take 30-60 minutes.
- You may participate in training without choosing to complete the surveys or informal interviews.

Risks/Discomforts

The only anticipated risk to you is the loss of time. I will try to minimize this risk by trying to follow your lead during training, and by allowing you to end the training at any time.

Benefits

I anticipate that by participating in the training, you will be more confident in your ability to use robots to meet specific learning objectives. Participating in surveys or interviews may help you to reflect on your practice. I also hope that your participation will help increase our understanding of teachers' professional development needs and preferences, and beliefs and attitudes toward using robots to teach.

Confidentiality

For this study we will not collect personal or private information. To protect your confidentiality, surveys will be anonymous. Data from informal interviews will be identified by your grade level, not your name. In any presentations or publications resulting from this research, I will use pseudonyms to refer to data. Data will be stored in my office files.

	Institutional Review Board	
BYU	3-5-2018	2-22-2019
	Approved	Expires

Participation

Participation in this research study is voluntary. You have the right to withdraw at any time or refuse to participate entirely.

Questions about the Research

If you have questions regarding this study, you may contact Stacie Mason at stcmason@gmail.com or Rick West at rickwest@byu.edu for further information.

Questions about Your Rights as Research Participants

If you have questions regarding your rights as a research participant contact IRB Administrator at (801) 422-1461; A-285 ASB, Brigham Young University, Provo, UT 84602; irb@byu.edu.

Statement of Consent

I have read, understood, and received a copy of the above consent and desire of my own free will to participate in this study.

Name (Printed): _____ Signature _____ Date: _____

APPENDIX C

District Approval, "Professional Development for Teaching Computational Thinking with Robots: An Exploratory Study"

[redacted] City SCHOOL DISTRICT

ASSESSMENT, DATA & RESEARCH
DEPARTMENT OF TEACHING & LEARNING

ANNE-MARIE HARRISON-PAULSEN
EXECUTIVE DIRECTOR

RON TWITCHELL, PH.D.
DIRECTOR

March 1, 2018

To Whom It May Concern:

This letter is to give approval for Stacie Mason to conduct a research study titled *"Developing Elementary School Teachers' Self-Efficacy to Teach Computing"* in [redacted] City School District (PCSD), specifically at [redacted]. I have reviewed your research request and give permission for the research to move forward. This means that PCSD personnel may be contacted, but PCSD personnel are not required to participate. No undo pressure should be applied in order to gain participation.

Per this approval, you also agree to provide PCSD with the results of your findings within 30 days of its conclusion.

If you have any questions or need additional information, feel free to contact me at (801) 374-4924 or at ront@[redacted].edu.

Sincerely,

Ron Twitchell, Ph.D.
Director of Assessment, Data & Research
[redacted] *City School District*

280 WEST 940 NORTH [redacted] [redacted]

APPENDIX D

Teacher Recruitment Scripts, "Professional Development for Teaching Computational Thinking with Robots: An Exploratory Study"

Teacher Recruitment Scripts

Teacher Initial Contact Email:

The BYU McKay School TEC Lab has robots available for teachers to check out for classroom use. We have found that using robots not only promotes student engagement, but also helps students learn useful skills. To promote use of these resources, we are currently providing free delivery and training. If you are interested in either of these services, please complete the attached google form:

https://goo.gl/forms/PIaLtB6fklNKCykH3

If you would like to check out robots but are not interested in delivery or training, please contact the McKay School TEC lab: teclab@byu.edu 801-422-2229.

Teacher Follow-up Email:

You have requested ___ (number) _____ (type of robot).

I will deliver them to your classroom _____ (date) at _____ (time). At that time, I will work with you for 30-60 minutes (or as long as you like) to help you prepare to use the robots in the classroom.

In conjunction with the training, I am conducting a small study to help me understand teachers' teachers' professional development needs and preferences, and attitudes and beliefs toward using robots in the classroom. Participation in the study would take about 10 minutes beyond the training time (approximately 5 minutes when the robots are delivered and 5 minutes when robots are returned). Attached is an Informed Consent form that provides more information about the study. You may use the robots and receive training without participating in the study.

Please email me if you would like to make changes to your order or if you have questions about the study.

Sincerely,
Stacie Mason

APPENDIX E

IRB Approvals, "Development and Analysis of the Elementary Student Coding Attitudes Survey"

INSTITUTIONAL REVIEW BOARD
FOR HUMAN SUBJECTS

Memorandum

To: Professor Peter Rich
Department: IP&T
College: EDUC
From: Sandee Aina, MPA, IRB Administrator
Bob Ridge, PhD, IRB Chair
Date: August 31, 2018
IRB#: X18351
Title: *"Measuring the Effects of Computing in Elementary Education"*

Brigham Young University's IRB has approved the research study referenced in the subject heading as expedited level, category 7. The approval period is from **August 31, 2018 to August 30, 2019**. Please reference your assigned IRB identification number in any correspondence with the IRB. Continued approval is conditional upon your compliance with the following requirements:

1. CONTINGENCIES:
   - Additional site approvals as you get them
   - Instruments in Spanish
2. A copy of the informed consent statement is attached. No other consent statement should be used. Each research subject must be provided with a copy or a way to access the consent statement.
3. Any modifications to the approved protocol must be submitted, reviewed, and approved by the IRB before modifications are incorporated in the study.
4. All recruiting tools must be submitted and approved by the IRB prior to use.
5. In addition, serious adverse events must be reported to the IRB immediately, with a written report by the PI within 24 hours of the PI's becoming aware of the event. Serious adverse events are (1) death of a research participant; or (2) serious injury to a research participant.
6. All other non-serious unanticipated problems should be reported to the IRB within 2 weeks of the first awareness of the problem by the PI. Prompt reporting is important, as unanticipated problems often require some modification of study procedures, protocols, and/or informed consent processes. Such modifications require the review and approval of the IRB.
7. A few months before the expiration date, you will receive a continuing review form. There will be two reminders. Please complete the form in a timely manner to ensure that there is no lapse in the study approval.

IRB Secretary
A 285 ASB
Brigham Young University
(801)422-3606

INSTITUTIONAL REVIEW BOARD
FOR HUMAN SUBJECTS

Memorandum

To: Professor Peter Rich
Department: IP&T
College: EDUC
From: Sandee Aina, MPA, IRB Administrator
Bob Ridge, PhD, IRB Chair
Date: September 12, 2018
IRB#: X18351
Title: *"Measuring the Effects of Computing in Elementary Education*

Brigham Young University's IRB has reviewed the amendment submitted on September 6, 2018. The IRB determined that the amendment does not increase risks to the research subject and the aims of the study remain as originally approved. The amendment has been approved. The revised parental permission forms have been approve and stamped for your files.

The approval of this protocol expires on **August 30, 2019**. All conditions for continued approval period remain in effect. Any modifications to the approved protocol must be submitted, reviewed and approved by the IRB before modifications are incorporated in the study.

IRB Secretary
A 285 ASB
Brigham Young University
(801)422-3606

INSTITUTIONAL REVIEW BOARD
FOR HUMAN SUBJECTS

Memorandum

To: Professor Peter Rich
Department: IP&T
College: EDUC
From: Sandee Aina, MPA, IRB Administrator
Bob Ridge, PhD, IRB Chair
Date: January 31, 2019
IRB#: X18351
Title: *"Measuring the Effects of Computing in Elementary Education"*

Brigham Young University's IRB has reviewed the amendment submitted for the above-referenced protocol. The IRB determined that the amendment does not increase risks to the research subject and the aims of the study remain as originally approved. The amendment has been approved with the following contingency: Provo School District approval, with specific language approving passive parental permission. The revised consent statement and recruiting script have been approved and stamped for your files.

The approval of this protocol expires on **August 30, 2019**. All conditions for continued approval period remain in effect. Any modifications to the approved protocol must be submitted, reviewed and approved by the IRB before modifications are incorporated in the study.

IRB Secretary
A 285 ASB
Brigham Young University
(801)422-3606

APPENDIX F

District Approvals, "Development and Analysis of the

Elementary Student Coding Attitudes Survey"

[redacted] City
SCHOOL DISTRICT

ASSESSMENT, DATA & RESEARCH
DEPARTMENT OF TEACHING & LEARNING

ANNE-MARIE HARRISON-PAULSEN
EXECUTIVE DIRECTOR

RON TWITCHELL, PH.D.
DIRECTOR

Jan 29, 2017

To Whom It May Concern:

This letter gives approval for Peter Rich to conduct a research study titled *"Measuring the Effects of Computing in Elementary Education"* in [redacted] City School District (PCSD). Your research request has been reviewed and you are given permission for the research to move forward. This means that PCSD personnel and/or students may be contacted, but neither PCSD personnel nor students are required to participate. No undo pressure should be applied in order to gain participation. Any report of such pressure will result in your approval being rescinded.

Per this approval, you also agree to provide PCSD with the results of your findings within 30 days of the study's conclusion.

If you have any questions or need additional information, feel free to contact me at (801) 374-4924 or at ront@[redacted].edu.

Regards,

Ron Twitchell, Ph.D.
Director of Assessment, Data & Research
[redacted] *City School District*

[redacted] School District
Established 1911
Reid P. Newey, Superintendent

Learning First!

September 5, 2018

Dr. Peter Rich
peter_rich@byu.edu

Dear Dr. Rich,

The application you submitted has been reviewed by the Superintendent's Executive Staff. The research project entitled "Measuring the Effects of Computing in Elementary Education" is approved for the 2018-2021.

As a researcher you are responsible for all aspects of the study. District resources may not be used to conduct the study. All costs associated with the study are paid by the researcher. Please request data, as needed, from the [redacted] District Assessment Department by submitting a data request (https://www[redacted].k12.ut.us/departments/assessment) under "Tools" or calling (801) 402-5305.

Approval at the district level allows each site to then determine whether to participate in your proposed research study or project. District approval, therefore, is not a guarantee that you will be able to conduct the study at the locations, with the employees, or with the students you wish to include in the study or project.

Please remember that any anticipated changes to the study and approved procedures must be submitted to this office prior to implementation. It is our understanding that you will protect the anonymity of individuals involved in the research.

We hope your research proves insightful and fulfilling.

Sincerely,

Ms. Amy Siegel
Assessment Supervisor

APPENDIX G

Parental Permission, "Development and Analysis of the Elementary Student Coding Attitudes Survey"

Parental Permission for a Minor

Introduction

My name is Peter Rich. I am a professor at Brigham Young University, conducting a research study about elementary school students' attitudes toward coding. I am inviting your child to take part in the research because the school your child attends offers coding instruction.

Procedures

If you agree to let your child participate in this research study, the following will occur:

- Your child will be asked to take an online survey.
- The survey has 52 items and is designed to take approximately 10-15 minutes.
- The survey will be taken in the regular classroom or computer classroom during computer time.

You may preview the survey by using this URL: https://byu.az1.qualtrics.com/jfe/form/SV_cMTAICSuS2FbZOJ

Risks

The only risk we anticipate from participating in this study is that students may worry that their performance will affect their grades. We will assure them that they will not be graded on their survey responses.

Confidentiality

We are very mindful of the need to protect student privacy. To that end, we will not collect names or other identifying personal data from students. The researcher will not use identifiers in the written report. Only the researchers will have access to the data, which will be stored in a password-protected computer file. At the end of the study, data will be securely stored for three years, as is required by the university internal review board.

Benefits

There are no direct benefits for your child's participation in this project. We hope that the data gathered in the survey will help researchers and stakeholders understand children's attitudes toward coding.

Compensation

There will be no compensation or reward for participation in this project.

Questions about the Research

Please direct any further questions about the study to Peter Rich at peter_rich@byu.edu or Stacie Mason at stcmason@gmail.com.

Questions about your child's rights as a study participant or to submit comment or complaints about the study should be directed to the IRB Administrator, Brigham Young University, A-285 ASB, Provo, UT 84602. Call (801) 422-1461 or send emails to irb@byu.edu.

Participation

Participation in this research study is voluntary. You are free to decline to have your child participate in this research study. You may withdraw you child's participation at any point without affecting your child's grade or standing in school.

IF YOU DO NOT WISH FOR YOUR CHILD TO PARTICIPATE IN THE STUDY, please sign and return this form. If you permit your child to participate, you do not need to return this form.

APPENDIX H

Teacher Script, "Development and Analysis of the Elementary Student Coding Attitudes Survey"

Teacher Instructions and Script

Teachers, before reading the script, please either load the survey onto students' computers, or be prepared to share the link or QR code (found at the bottom of the page) with students.

In the survey there are a few questions that may be confusing for students who have not coded; for example, "I am good at coding." If a student would answer "I don't know" (which is not an option), please advise them to disagree with the statement.

Please read the following aloud to your students before they take the survey:

> "Our class has been invited to take a survey about what we think about coding. Coding is another name for computer programming. To code is to write instructions for a computer or robot to follow. You may have done coding using robots, Code.org, Tynker, or Scratch.
>
> The survey includes 52 questions about what you think about coding and school. The survey will take about 15 minutes. Your answers will help other people understand what students think about coding.
>
> You will not get a grade for taking the survey. There are no right or wrong answers in the survey. If there are questions you do not understand, you can ask for help.
>
> You don't have to be in this study. It's up to you. If you do not want to take this survey, please indicate by raising your hand. Before you say yes to take the survey, you can ask me to tell you more about anything that you don't understand.
>
> If you agree to take the survey, please press the link to the survey" (or, if the survey is already loaded on students' screens, "please press the forward arrow on your screen"): Coding_attitudes_survey

APPENDIX I

Survey, "Development and Analysis of the Elementary Student Coding Attitudes Survey"

What is this survey about?

The purpose of this survey is to learn about what students think about coding. Coding is another name for computer programming. To code is to write instructions for a computer or robot to follow. You may have done coding using robots, Code.org, Tynker, or Scratch. You are being asked to take this survey because your school offers coding instruction.

The survey includes 55 questions about what you think about coding and school. The survey will take about 15 minutes.

You will not get a grade for taking the survey. There are no right or wrong answers in the survey. We just want to know what you think. If there are questions you do not understand, you can ask for help.

This survey was designed by Peter Rich, a BYU professor, and a few of his students. We don't know if being in this study will help you, but we hope to learn something that will help other people some day. When we are done with the study, we will write a report about what we learned. We won't use your name in the report.

You don't have to be in this study. It's up to you. If you say yes now, but change your mind later, that's okay too. All you have to do is tell us. Before you say yes to be in this study, you can ask your teacher to tell you more about anything that you don't understand.

If you choose to participate, please click the "next" arrow.
Thank you!

For each of the following items, choose the answer that shows how much you agree.

I can learn to code.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I am good at coding.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I am good at problem solving.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I can write clear instructions for a robot or computer to follow.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

If my code does not work, I can find my mistake and fix it.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I've been told I would be good at coding.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I like coding, or I think I would like coding.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I would like to learn more about coding.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Solving coding problems seems fun.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Coding is interesting.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I would like to study coding in the future.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I think I would like a job that lets me code.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I know what coding is.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

It is valuable for me to learn coding.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I can use coding skills in other school subjects.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Knowing how to code will help me to create useful things.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Knowing how to code will help me solve problems.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I think I will need to understand coding for my future job.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Learning to code will make me better in math.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Learning to code will make me better in science.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Learning to code will make me better in technology.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Learning to code will make me better in language arts.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Both girls and boys should learn coding.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Both girls and boys can learn coding.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Both boys and girls can do well in coding classes.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Girls are better at coding than boys.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Boys are better at coding than girls.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Boys like to code more than girls do.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Girls like to code more than boys do.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Coding is cool.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Kids who code are smarter than average.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Kids who code like to play video games.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Kids who code spend less time outside than other kids.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Kids who code enjoy doing sports.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Coders are nerdy.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Coders are good at math.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Coders are good at science.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Coders are good at language arts.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I usually do well in math.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I like math.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I usually do well in science.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I like science.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I usually do well in language arts.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I like language arts.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

My friends think coding is cool.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

My parents think coding is important.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

My teachers think coding is important.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I know someone who can code.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

I am friends with kids who code.

- ○ Strongly agree
- ○ Agree
- ○ Somewhat agree
- ○ Somewhat disagree
- ○ Disagree
- ○ Strongly disagree

Demographics

What grade are you in?

- ○ 3
- ○ 4
- ○ 5
- ○ 6

What is your gender?

- ○ Girl
- ○ Boy
- ○ Other

How would you describe yourself?

- ☐ White
- ☐ Native Hawaiian or Pacific Islander
- ☐ Hispanic
- ☐ Black or African American
- ☐ Asian
- ☐ Native American or Alaska Native
- ☐ Other

Did your mom go to college?

- ○ Yes
- ○ No
- ○ I don't know

Did your dad go to college?

- ○ Yes
- ○ No
- ○ I don't know

How long have you been coding? (For example, Tynker, Scratch, Code.org)

- ○ Less than 1 year (or, I started this school year)
- ○ 1-2 years
- ○ 2-3 years
- ○ More than 3 years

How often do you code?

- ○ Almost every day
- ○ About 3 times a week
- ○ About once a week
- ○ About once a month
- ○ Less than once a month