Computers & Education 82 (2015) 263–279

Contents lists available at ScienceDirect

Computers & Education

journal homepage: www.elsevier.com/locate/compedu

Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis

Maya Israel[*], Jamie N. Pearson, Tanya Tapia, Quentin M. Wherfel, George Reese

University of Illinois at Urbana-Champaign, United States

A R T I C L E   I N F O

Article history:
Received 5 August 2014
Received in revised form
12 November 2014
Accepted 26 November 2014
Available online 3 December 2014

Keywords:
Computational thinking
Computing
Struggling learners
Diverse learners
Content integration

A B S T R A C T

The purpose of this study was to investigate how elementary school teachers with limited computer science experience in a high-need school integrated computational thinking into their instruction. The researchers conducted a cross-case analysis across different instructional contexts (e.g., general education classrooms, library, art) that included multiple observations and interviews over four months. Major themes included: (a) a wide range of implementation models emerged depending on teaching contexts, (b) ongoing professional development and embedded coaching resulted in increasing participation in computing education, (c) teachers and administrators viewed barriers to implementing computing from a problem solving framework, and (d) struggling learners, including students with disabilities and those living in poverty, benefitted from computing education that included scaffolding, modeling, and peer collaboration.

1. Introduction

Although a great deal is written about increasing the pipeline of people entering science, technology, engineering, and mathematics (STEM) careers, computer science (CS) has largely been ignored, which is alarming given the high demand for CS jobs (CSTA, 2003, 2010; Wilson & Moffat, 2010). Recently, though, there has been a growing interest in introducing computing into K-12 instruction. This new commitment to CS rests on research suggesting that children typically learn how to operate technologies rather than learn how to develop new technologies; in this way, they only experience the receiving end of technology (Burke & Kafai, 2014). Conversely, technologies should move towards being viewed as tools for thinking, learning, and creating (Burke & Kafai, 2014; Harel Caperton, 2010). The largest current example of introducing computing into K-12 education has been the Hour of Code initiative (http://code.org), organized through the non-profit organization Code.org that took place during the Computer Science Education Week (2014) in which approximately 15 million students had at least one hour of computing experience in school settings (http://csedweek.org/educate/hoc). This initiative represents the wider movement towards creating multiple experiences for young learners in the area of computer programming and computational thinking.

2. Definition of computing, computer programming, and computational thinking

Terms like computing, computer programming, and computational thinking are often used interchangeably, may cause definitional confusion, and are much debated (Grover & Pea, 2013; NRC, 2011). Additionally, these terms are sometimes used to describe other educational technology applications and general use of software such as word processing. To alleviate this confusion, the International Society of Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) operationalized the term computational thinking as a problem solving process that includes:

* Corresponding author. Department of Special Education, University of Illinois at Urbana-Champaign, 276B Education Building, 1310 South 6th Street, Champaign, IL 61820, United States. Tel.: +1 217 300 0338.
E-mail address: misrael@illinois.edu (M. Israel).

http://dx.doi.org/10.1016/j.compedu.2014.11.022
0360-1315/© 2014 Elsevier Ltd. All rights reserved.

Formulating problems in a way that enables us to use a computer and other tools to help solve them; logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking (a series of ordered steps); identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combinations of steps and resources; and generalizing and transferring this problem solving process to a wide variety of problems (ISTE and CSTA, 2011).

As the above definition reveals, although computational thinking includes concepts that are fundamental to computing and computer programming, it is a broader term that includes a problem-solving framework that incorporates problem representation, prediction, and abstraction (Kafai & Burke, 2014; NRC, 2008; Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013). Wing (2006), in fact, in her seminal article, defined computational thinking as a thought process for understanding problems and solutions for those problems through processes such as abstraction. Part of computational thinking involves the process of learning the skills needed to engage in computer programming. In this type of instruction, students learn to program in tile-based computing software (e.g., Scratch, Etoys, Alice) or programming languages (e.g., C++, Java). Harel Caperton (2010) compared learning how to program to learning basic literacy where engaging in technologies is analogous to reading while programming and computing is analogous to learning how to write. There is, however, still a great deal of discussion about the relationship between computational thinking and programming (NRC, 2010). In this paper, we utilize the term computational thinking to refer to students using computers to model their ideas and develop programs that enhance those programs similar to Wing's (2006) definition, and consider computer programming as one part of computational thinking.

3. Rationale for computing in K-12

The most prevalently cited rationale in the literature for including computing in K-12 instruction is the growing demand for workers with computer science skills (CSTA, 2003, 2010; Wilson & Moffat, 2010). Beyond the pipeline argument, however, research is beginning to suggest that learning computing has its own benefits including improving learners' higher-order thinking skills and the development of algorithmic problem-solving skills (CSTA, 2003; Fessakis & Mavroudi, 2013; Kafai & Burke, 2014; Kay & Knaack, 2005; Papert, 1991). To further the rationale for integrating computing into K-12 education, Sengupta et al. (2013) developed a framework for aligning concepts of computational thinking with scientific inquiry to showcase how and why computing should be integrated into science and math instruction. They argued that this integration is a natural way of applying algorithmic design and engineering within scientific inquiry. Others have stated that integrating computing into the content areas increases access to computational experiences and provides a way of introducing computing within authentic experiences rather than as isolated subject areas (Jona et al., 2014; Weintrop et al., 2014).

In fact, Liao and Bright (1991) conducted a meta-analysis of 65 studies of computer programming and children and found that 89% of the studies showed positive effect sizes for computer programming experiences. Fessakis and Mavroudi (2013) showed that children as young as kindergarten enjoyed and benefitted from computer programming and were able to learn simple tile-based programming that made use of graphically intuitive software designed for young learners. In their study, kindergarteners worked on one-to-one correspondence, counting, and angle-turn concepts through programming.

3.1. Confusion about computing in K-12

Even though there is growing evidence to support the integration of CS into K-12 education, many people, including teachers and their students, have inaccurate or naïve perceptions of CS and computing that influence their attitudes about CS learning and careers (Armoni, 2011). For teachers, these naïve perceptions and misconceptions directly influence whether and how they teach concepts of CS, and consequently, students often leave these experiences thinking that CS is boring, confusing, and too difficult to master (Wilson & Moffat, 2010). For example, if teachers believe that the only computing experiences available to students occur through learning programming languages such as Java or C++, they may never consider introducing computing at the earlier grades. If they believe that computing can only be learned in this manner, they may also not introduce learning experiences such as *CS Unplugged* or using tile-based programs such as Scratch or Etoys. Additionally, if teachers are only aware of career opportunities involving programming for large technology companies, that misconception will influence how they discuss computing careers.

4. Computing and CS in elementary school contexts previous research

Although sparse, the literature exploring emerging practices around computing initiatives in schools is on the rise (Clark, Rogers, Spradling, & Pais, 2013; Gardner & Feng, 2010; Lambert & Guiffre, 2009; Wolfe, 2011). With that said, very few of these studies focused on computing in elementary school settings. The majority focused on CS at the high school level with a direct focus on the career pipeline. In other STEM fields, the literature has shown that it is important to introduce key concepts and epistemic experiences early in students' education so that they will develop positive attitudes towards those disciplines (Aschbacher, Li, & Roth, 2010; Maltese & Tai, 2010). It is likely that it is equally important to provide students with early CS and computational thinking experiences as well as other STEM experiences. Despite this lack of research focus on elementary aged students, however, many of the programming tools available to students are aimed at young learners including those at the elementary school level (Good, 2011). As Good (2011) explained, these programs aim to “lower the computational floor” (p. 18) to allow for easy access to programming while at the same time maintaining challenging experiences as students' skills and knowledge of computing increases.

Of those studies conducted on computing education, findings indicate positive outcomes related to both attitudes about computing and CS as well as skills related to computing. For example, findings include students with increased positive attitudes about CS (Lambert & Guiffre, 2009; Lin, Yen, Yang, & Chen, 2005) as well as increased skills as computer programmers (Baytak & Land, 2011; Kwon, Kim, Shim, & Lee, 2012). Research on teaching practices indicated that teachers who were initially skeptical of implementing computing found computer programs such as Scratch and Etoys to be both valuable (Clark et al., 2013) and accessible (Lee, 2011).

Although there is a growing body of literature exploring computing education, no literature exists related to how teachers implement computing within the context of school-wide CS initiatives at the elementary level, especially with diverse learners. Research into the digital divide, for example, points towards the need to broaden the definition of access to technology beyond simply having access to tools, but having access to the skills necessary to use those tools (Kafai & Burke, 2014; Valadez & Durán, 2007). The aforementioned gap in this domain suggests that there is a need for additional research around computing that aims specifically to examine how elementary school teachers implement computing in different instructional contexts that include students from diverse backgrounds and those at-risk for academic failure due to poverty or disability. This study, therefore, addresses this gap in the literature by examining how elementary school teachers integrated computing into their instruction within the context of a school-wide computer science initiative for all learners, including those with different risk factors such as poverty and disability.

5. Theoretical foundations

5.1. Implementation science

When considering the introduction of computing education into school systems, Fixsen, Blase, Naoom, and Wallace (2005) provide a useful theoretical framework related to implementation science. Fixsen and colleagues have argued that for many years, we have unsuccessfully assumed that implementation would occur passively through diffusion and dissemination of information to teachers and school leaders. Rather, they suggested a strategic, deliberate implementation process that includes six stages of implementation: exploration, installation, initial implementation, full implementation, innovation, and sustainability. Although these stages move towards full implementation, the authors explained that these stages are not linear in nature and each stage influences the other stages. In addition to these stages, Fixsen and colleagues discussed core components that include staff selection, preservice and inservice training, ongoing coaching and consultation, staff performance assessment, data systems that support decision making, facilitative administration, and external system intervention to help navigate the needed organizational, financial and human resources (Fixsen et al., 2005). They further recognized that certain individuals drive implementation by actively working towards integrating the required practices into the education system, work through barriers, and generally problem solve as problems emerge; they referred to these people as purveyors. Odom (2009) and Harn, Parisi, and Stoolmiller (2013) expanded on this work by also describing practitioner adaptation of interventions. They recognized that teachers and other implementers often adapt interventions to their context based on their values, the administrative directives, and the values of the community. For teachers to sustain any instructional practice, it must meet their needs and be adapted to authentically address the specific needs of the instructional context. In essence, no teaching practice will look exactly the same in different contexts, but if the essential components are in place along with professional development and coaching, the fidelity of those practices will remain intact.

To further complicate the issue of implementation of computing education initiatives, Hug and Reese (2006) using the Rogers (2003) dissemination of innovation framework, noted that while a maverick teacher will adopt a computing tool (such as Etoys or Scratch), it remains an open question whether such early adopters will lead to widespread use of a new tool. Existing classroom contexts that do not see computing as a priority and lack of access to technology (Norris, Sullivan, Poirot, & Soloway, 2003) make the challenge seem almost insurmountable. If the necessary infrastructure for computing is not in place (e.g., there is little flexibility in the curriculum to include computing during typical school hours), widespread implementation will typically not occur. In addition, as Agalianos, Noss, and Whitty (2001) explained in a historical analysis of implementation of Logo programming (an early programming language for children) in the US and UK, the creators of languages like Logo were skeptical of schools as organizational cultures that could include the effective use of tools like Logo. These scholars, however, did not investigate computing within the context of authentic school implementation practices.

Adopting a cautiously optimistic approach, the current study investigated school-wide computing education through the lens of Fixsen and colleagues, Odom, and other implementation scientists who have researched effective and long-lasting school implementation. This study made use of a cross-case research design to examine how different teachers implemented computing within the same school setting and the same implementation processes (e.g., access to coaching, administrative expectations). This design was chosen in order to understand how implementation could differ among teachers within the same school but with different instructional demands and students.

6. Methods

This study made use of a cross-case analysis (Merriam, 2009) to understand how computational thinking was integrated in K-5 instruction as part of a school-wide initiative with students from diverse backgrounds, including students living in poverty and those with disabilities. Case study methodology was chosen for this study because, according to Merriam (2009), case study methodology "offers a means of investigating complex social units consisting of multiple variables of potential importance in understanding the phenomenon" (p. 50). Stake (2006) explained that within cross-case analysis, single cases are seen as belonging to a collection of cases that share common characteristics and conditions that are categorically related. Merriam (2009) further stated that as the number of cases increase, the more compelling the findings. In this study, each participating teacher constituted a case within our cross-case analysis as different teachers integrated computing in different manners, had different instructional practices and attitudes about technology and computing, and faced different barriers to implementation. Four research questions guided this study:

1. How was computing integrated into instruction across different instructional contexts?
2. What types of supports did teachers request and use to implement computing education?
3. What barriers to implementation occurred during the adoption of computing into instruction?
4. How did teachers support diverse learners, including students with disabilities and those at risk for academic failure due to poverty?

6.1. Participants

The participants in this study were teachers and administrators in one Midwestern elementary school that made a commitment to systematically integrate school-wide computational thinking into instruction. The researchers purposefully selected this site due to (a) the commitment to include computing in instruction, and (b) its high-need classification. The student population included a high proportion of students living in poverty wherein 77.5% of the students received free or reduced lunch, and many students were classified as at risk for academic failure. Moreover, 45% of students received additional academic interventions as part of the Response to Intervention supports for academic at-risk factors and 15.4% received special education services.

The elementary school was “under-chosen” in a district where parents could rank their requests for elementary schools. The school's initial exploration of computing began with the combination of school leaders recognizing the need for change and seeking opportunities to work with the university around computational thinking. These school leaders recognized that integrating computational thinking into the curricula could increase the visibility of the school as well as the instructional experiences of their students at risk of academic failure. The emerging research study was shaped through collaboration between these school-based leaders, university faculty from both education and computer science departments at a local university, as well as local business leaders.

Data collection and analysis took place after approval by the institutional review board. The initiative began with a week-long summer workshop in which 20 of the teachers in the school volunteered to participate. The teachers were told that participation in the workshop could occur without participation in the research study. This workshop included a basic introduction to computing and computational thinking, modeling of computing tools, and time for teachers to practice using these computing tools. A follow up workshop took place during the winter break to offer additional professional development and allow the teachers time to work on integrating computing into their instruction. Throughout the year, energy and focus on the computational thinking escalated. For example, two teachers with experience in technology and computational thinking transferred to the school from other schools in the district. Teacher-led initiatives such as student presentations, a tech night, Twitter chats, and teacher-led professional development were all part of the evolution over this single year. At the same time, the researchers were gathering data on the process.

Within this school, seven teachers and two administrators were purposefully selected to participate in the study based on the following three criteria: (a) attendance at a summer workshop introducing computing and computer programming software, (b) willingness to integrate computing into their instruction, and (c) diverse learners in the classrooms. It was important to select teachers across grade levels and teaching duties in order to create meaningful comparisons across the cases. Table 1 provides information about the participating teachers and administrators.

6.2. Computing software

There are several programming tools specifically aimed at students in K-8. During the course of this study, the teachers primarily used the Etoys (http://www.squeakland.org/) and Scratch (http://scratch.mit.edu/) programming software platforms. Etoys and Scratch are both open-source media-rich programming environments. These programs allow teachers and students to develop programs such as games, simulations, and animations that can expand both understanding of computing and how complex systems work (Good, 2011; Kay & Knaack, 2005; Lee, 2011). The drag and drop method of programming within these environments allow users to have a graphically intuitive experience that can easily be integrated into K-12 CS teaching and learning (Bouras, Pouplooulos, & Tsogkas, 2008). These software aimed at engaging young learners provide an accessible starting point for learning with limited or no programming background (Good, 2011). Therefore, both Etoys and Scratch were used as computing platforms.

7. Data collection

Data collection occurred from September 2013 through February of 2014. Sources of data included teacher and administrator interviews as well as observations of teaching practices. A team of five researchers (3 university faculty members and two graduate students) collected both observational and interview data.

7.1. Observation data

Observations took place at times when the participating teachers implemented computing and made use of time-incremented field notes. No photos were taken of students' faces to protect their anonymity. During all observations, the researchers took detailed field notes of teachers' instructional practices, students' interactions, and students' computing activities. The researchers also took pictures of students' computer screens to capture the products that they developed. All field notes were typed and shared with the other researchers for data analysis purposes.

Table 1
Teacher participants.

Teacher (Pseudonym)	Years teaching	Content/Context	# of students
Mr. Ryan	15	Library/Media	300 (all students)
Mr. Thomas	8	Technology	300 (all students)
Ms. Lyle	18	Art teacher	300 (all students)
Ms. Smith	6	3rd grade teacher	26 students
Mrs. Marks	30+	Enrichment teacher	14 students
Ms. Rosen	27	4th/5th grade split	25 students
Mrs. Hawthorne	30+	2nd grade teacher	24 students

*Mr. Ryan, Mr. Thomas, and Ms. Lyle saw all the students in the school as part of their teaching roles.

7.2. Teacher and administrator interview data

To gain the perspectives of the teachers implementing computing and their administrators, semi-structured interview protocols were derived from the research questions and theoretical framework (see Appendix). The questions went through a vetting process in which the research team and computing experts across education and computer science provided feedback. The main focus of these interviews was to understand how teachers implemented computing, what barriers occurred during implementation, and how they supported students with diverse needs, including students with disabilities and those living in poverty. All interviews were audio-recorded and transcribed for analysis.

7.3. Cross-case data analysis

The researchers analyzed the observation field notes and interview transcripts through a basic interpretive methodology with constant comparative analysis (Glaser & Strauss, 1967) in which a structured coding scheme was developed. Data from each teacher observation and interview was coded separately in this first level of coding. These individual cases were then used for a cross-case analysis that focused on similarities and differences across the cases. The researchers conducted interrater reliability checks in both interview and observation data analysis. After each coder completed two interviews, interrater reliability ranged from 70 to 80 %. As the researchers debriefed and discussed their coding rationale, the team was able to more clearly operationalize the codes. After using the revised definitions, interrater reliability increased to 90% or higher.

7.4. Data analysis credibility

To address issues of trustworthiness and credibility of data analysis, the research team triangulated several sources of data to represent the perceptions of teachers implementing computing (i.e., teacher interviews), their instructional practices (i.e., classroom observations), and administrative views on computing (i.e., administrator interviews). For example, teachers discussed barriers to implementing computing, which was observed during instructional lessons. These two data sources corroborated each other. Additionally, the research team conducted member checks with study participants to corroborate themes.

8. Results

The following section describes the themes and subthemes that emerged in each of the four research questions.

8.1. Research question 1: how was computing integrated into instruction across different instructional contexts?

8.1.1. Teachers implemented computing differently based on classroom organization and the level of open-endedness of the tasks

The seven teachers in this study implemented computing differently in terms of how they organized instruction. Namely, they either used whole-class instruction or a center-based model. Additionally, they varied in the level of explicit instruction used within computing activities.

8.1.1.1. Whole-class versus center-based instruction. Five of the seven teachers generally used whole-class instruction in which they didactically taught a computing lesson, either as a distinct content area or integrated within core content instruction, and then provided time for students to work collaboratively on various computing tasks. For example, Ms. Smith (the 3rd grade teacher) typically co-taught computing lessons with the librarian and always began with whole class directions and demonstration followed by independent and collaborative peer work. They modeled specific computing tasks to the entire class and then the students worked on those tasks either independently or alongside peers. Unlike Ms. Smith and others that followed a similar structure (i.e., Ms. Rosen, Ms. Lyle, Mr. Thomas, and Ms. Marks), Ms. Hawthorne (the 2nd grade teacher) included computing instruction as a center activity in which students worked with peer mentors on computing tasks related to learning math concepts in a designated area within the classroom. The students in this case were expected to work with their peers without a great deal of teacher monitoring and support. Within this structure, the students had specific computing tasks related to math instruction and they worked on those fairly independently, with the support of their peer mentors.

8.1.1.2. Level of explicit instruction. In addition to the structure of instruction (whole class vs. center-based), teachers used varying levels of explicit instruction, which resulted in differences in the level of open-endedness within the computing activities. Four of the teachers provided a great deal of flexibility (Mr. Ryan, Mr. Thomas, Ms. Smith, and Mrs. Marks) and openness within the tasks and the other three teachers had a more linear and structured approach (Ms. Lyle, Ms. Rosen, and Mrs. Hawthorne).

The level of open-endedness influenced how the teachers modeled and provided instruction. For example, Ms. Lyle, the art teacher, who taught in a more linear fashion, cycled between “carpet time” in which students received explicit instruction related to discrete computing tasks and “computer time” in which the students implemented those tasks. Carpet time allowed this teacher to frontload new information with the purpose of reducing student frustration. This instruction typically lasted approximately five minutes and included students as young as kindergarten. Computer time allowed students to move back to their computers to independently practice these new skills as the teacher circulated among the students and offered support when needed. As would be expected, the needs of students determined the level of teacher support during this work time. For example, some students required no support and worked independently. Other students needed minimal support and were encouraged to work collaboratively with their peers. A few students required more intensive support that was offered by the teacher or the instructional aides in the class. Computer time lasted approximately eight to ten minutes. During a class period, the teacher could cycle through two or three such rotations. For example, in one class session, Ms. Lyle wanted the students to learn how to make pen trails attached to shapes to display the path of moving objects. She started by modeling how to draw the shapes, add color to those shapes, and then provided step-by-step instruction and modeling in how to animate the stick figures and then how to add the pen trails to those animations. She explained to her class, “Guys, I want you to add movement too, but first I want you to make one shape and add color. Then come back [to the carpet].” Three minutes later, the majority of students created their shapes and were seated at the carpet. She continued, “Does anybody remember what we need to do to make it move?” She guided the students through the process with a cycle of explicit instruction and student work at the computers.

Mr. Ryan, the librarian, took a different approach. He typically started class with explicit instruction and modeling that focused on broader tasks such as a whole project. This instruction lasted approximately seven to ten minutes after which the students were then given time to work on the task. For example, he wanted to teach a group of third graders how to program a steering wheel to control the movement of a racecar. To do so, he modeled the steps at the beginning of the class on the interactive white board and kept the script and model on the board for students who wanted to look at the scripts and product. Throughout the rest of class, he circulated amongst the students and helped extend understanding. To one student, he stated, “Look at the script for your steering wheel. Can you steer your car?” Mr. Ryan then helped guide the student to problem solve methods to fix his script so that it would steer the car.

Ms. Marks, an enrichment teacher, had a great deal of freedom to use open-ended tasks without having to adhere to strict curriculum benchmarks. She wanted students to explore different computing tasks, but students had a great deal of freedom in how they would do so. For example, in one lesson, she had the students either draw a telephone or copy a picture of a phone from the Internet for an activity and then change this picture in the following ways: (a) combining them with other things, (b) adding something to their objects, (c) making their objects bigger or smaller, (d) putting their objects to different uses, (e) eliminating something, and (f) rearranging the objects on the screens. She walked around the classroom and offered feedback. When she observed a student struggling, she stated, “Do you remember what substitute means? It means replace something with another.” She then asked the entire class, “How many of you got to substitute something yet?” The majority of students raised their hands. During this activity, the students talked amongst themselves and said things such as, “I made it [the object of the phone] turn around. Now it's turning backwards.” Another student then stated, “How did you make it go backwards?” The student then answered, “I changed the numbers to make them negative.” These types of conversations were common in the open-ended activities as students could explore various computing and programming skills.

When asked about the decisions to offer explicit task instruction and open-ended activities, the teachers acknowledged the importance of both. They all worried that without some explicit instruction and modeling, the students would spend their time engaged in computing tasks that they already knew without moving towards more complex tasks and understandings. They explained that explicit instruction lent itself to more complex tasks because the students were consistently being guided through prerequisite steps to develop higher-order computational thinking.

8.1.2. *Integrating computing into the content areas was key to successful implementation*

Three classroom teachers (2nd, 3rd, 4th/5th grade split) taught computing by integrating it into their content areas. They all indicated that the pacing of the curriculum was too rapid to add computing as a distinct area of instruction. Ms. Smith, the 3rd grade teacher explained, “I think the time that we have here is so limited that I can't spare an hour or two a week with something that's not tied to the curriculum that the district has already given me.” Consequently, the way that they included computing in their instruction was by deliberately embedding it into their preexisting curricula. The librarian made note of the need for classroom teachers to integrate computing into their instruction as well. He explained:

> We don't see CS as a separate curriculum. That's a shift for me, because when we started this, I was really excited about CS on its own. I was, you know, and I still am…but we've realized that if it's going to get done, we really need to push it as embedded.

Thus, classroom teachers and specialists worked towards finding ways of integrating computing into instruction, either through whole class instruction or within centers (as stated above).

8.1.2.1. Examples of content and computing integration. Observations revealed numerous ways in which computing was integrated into the content in science, language arts, and mathematics. For example, Ms. Rosen, the 4th/5th grade split classroom teacher, taught a unit about the life cycle of a tree wherein the students used the Etoys computing software to journal about the process. Fig. 1 provides examples of student work within this activity. She explained that the students, “went into the computer lab and did an animation of a lifecycle of the tree, starting with an acorn and it grew into a sapling and then into a mature tree. The leaves then fell off and it's–they loved it! Each kid did something different…They said, ‘Can I add this? Can I put it on that? Can I make this background?’” To her, the students' enthusiasm drove her to further explore computing, especially in content with which students historically did not engage.

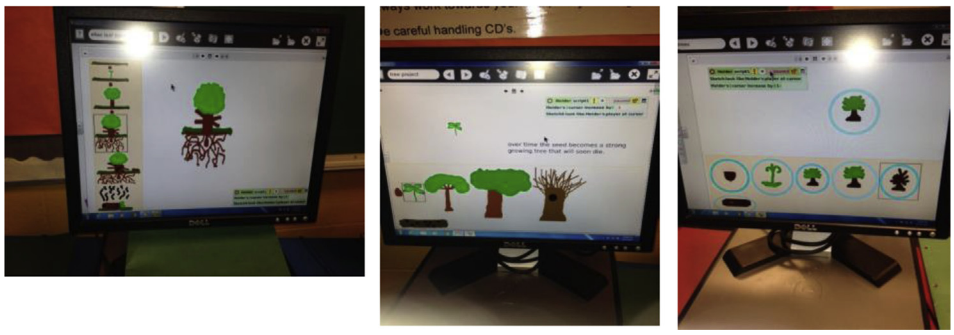

Fig. 1. Student work samples for lifecycle of a tree unit.

Ms. Rosen then stated:

Today we had to do the layers of the forest. Boring, boring. What's in the top layer? What's in the middle layer? What lives there? Their first question was, ‘Are we going to get to put it on Etoys?’ Etoys is only the tool to present what you know. It's not going to teach you the layers of the forest…So that's how I'm going to do the rest of my science. All their science notebooks will be an Etoys project.

She then indicated that now that she has experimented with computing in science, she would like to extend that to mathematics and reading instruction.

The second grade teacher, Ms. Hawthorne, integrated computing into her math instruction as a way to reinforce money skills. Rather than teaching money through computing in a whole-class manner, she decided to create a computing center in which students could collaboratively create e-books with money sentences using Etoys. She explained:

There is a series of pages that explain about each coin…my hope was that as we continued to learn about coins, we could just keep adding pages…like now we're doing change so somebody has so much money and they buy something for so much and it's less than a dollar, right? So I was thinking we could animate it.

Fig. 2 provides examples of students' coin e-book pages.

Because computing was taught in a center rather than whole class, Ms. Hawthorne relied on teaching classmates to work as mentors that supported the other students. She found this process considerably challenging initially as the children requested teacher support and could not work completely independently. She explained, “I keep trying different things, but the curriculum is so packed with stuff.” Hence, the center model was the only one she could implement consistently. Despite these challenges, Ms. Hawthorne stated that computing was worth the effort and time it required to teach peer mentors to support the other students. Later in the school year, she transitioned to a whole-class instructional model in which students worked together to create fraction e-books. In this context, where all the students worked on integrated computing within mathematics, there were more mentors to help within the class. Ms. Hawthorne stated that the students in her class supported each other and did not rely on her for a great deal of support.

8.2. Research question 2: what types of supports did teachers request and use to implement computing education?

8.2.1. Teachers were initially apprehensive about integrating K-5 computing, but were eager to integrate it into their instruction when given access to support and expertise

One of the main findings of this study was that teachers were generally willing to learn about computing and integrate computing into their core instruction as long as they received support from people they perceived as expert. Interestingly, though, at the beginning of the study, even the teachers who expressed excitement about computing education were apprehensive about integrating it into their instruction. This apprehension resulted in more passive participation wherein if computing experts such as the technology teacher, librarian, or university staff offered to co-teach and plan instruction, they accepted the support, but they did not initiate computing activities on their own or seek out experts to do so. As the study progressed, however, the teachers began to trust that they would be supported during computing instruction, gained experience, and became more confident, which resulted in their taking a much more active role. Once this occurred, they were more willing to implement computing and more actively sought support to do so. The third grade teacher, Ms. Smith, expressed both her anxiety and excitement for introducing computing into her instruction. She stated that during the initial week-long professional development workshop, she felt overwhelmed:

That week was really overwhelming I think more than anything because I am not a computer science type person. I am more like English language arts. (Laughter). So I think that was like a totally new language for me, and then when they were explaining all of the things, you know, the binary, that was really confusing for me…it was like a foreign language to me.

She later explained that although she expected computing to be too difficult to implement, she found it easier than she anticipated. Her enthusiasm came from the students' engagement and excitement. Similarly, Ms. Rosen explained:

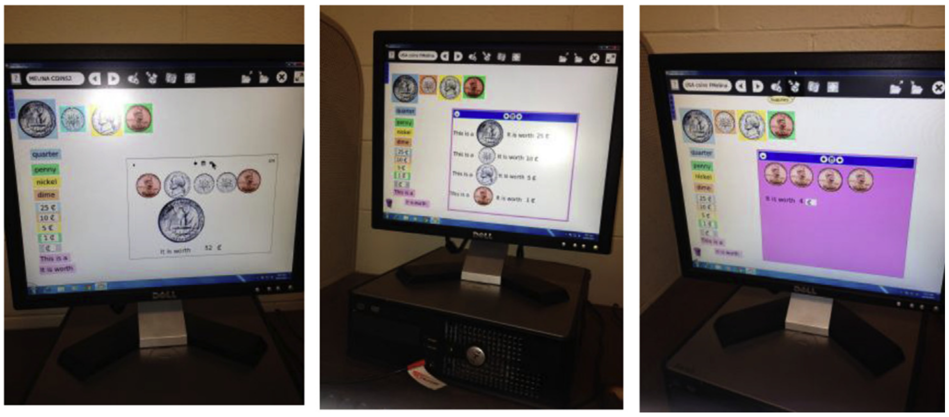

Fig. 2. Student work samples of coin e-books.

> It's fairly new to me. I've been teaching a long time and computers weren't really part of what we did, so I've had to learn along with the kids. They are much more daring and adventurous than I am, and I'm finding that they are not afraid to try it, do it, make mistakes…I want to make sure I'm doing it right before I try it, so I've had to learn to just do it and see what happens.

When asked what made her start teaching computing, she stated, “What made me start it? [Librarian's name]. (Laughter)…He said ‘You're not doing this by yourself. Tell me what you need and I'm going to be in there.’”

The technology teacher for the school, Mr. Thomas, explained:

> One of the biggest surprises to me has been how readily people will take it on, given the right supports. I initially was skeptical that people would take it on just because of time and because people are afraid to do things that they aren't immediately good at. I found that that's not necessarily the case.

The teachers all explained that integrating computing into their instruction resulted in increased preparation time as they needed to learn about the technology as well as how it could be integrated into core instruction. The art teacher described the lengths she took to prepare for teaching computing, “I sat at home and I made a list of what the icons mean and what each of them were and just kind of started from there…I wrote that kind of stuff out for myself.” In this way, she increased her own capacity and comfort with the new software. Despite the added work, however, all the teachers expressed pride in efforts taking place at their school around computing. There was a collective belief that they were engaging in knowledge creation that could inform other teachers and the educational community.

8.2.2. *Professional development, embedded coaching, and computing expertise were key to successful implementation*

There was a strong commitment to providing professional development and coaching to address teachers' computational skills and need for curricular alignment among both the administrators and technology specialists. This professional development included the summer workshop for twenty of the teachers in the school, as well as additional workshops during the winter break. Ms. Lyle, the art teacher, described her workshop experience:

> I see how much we packed in one week and how I went from completely not getting how to get rid of something if I dragged it…and it took me all week to figure that out, where I can just click on it and get rid of it.

She strongly expressed that it was important for a large number of teachers to undergo the summer professional development together, “I think it did help that we had a big chunk of us get trained all at once.”

The administrators and technology specialists recognized that these workshops would not be enough to sustain the computing effort at the school, so they also instituted embedded coaching that was done both by the librarian and the technology specialist. This embedded coaching involved co-planning lessons that would integrate computing into the teachers' core curriculum as well as co-teaching with teachers who wanted additional support during instruction. The primary coaches in the study included the technology teacher, the librarian, and university staff with experience in embedded computing. The librarian explained that teachers were willing to accept his coaching because of the trust that he has built in the community, “I think by the fact that I've been here for 10 years and that in my position as the librarian I see everybody. I know everybody…. But yet I'm never judgmental with people, you know. I don't evaluate them, so they feel safe coming to me for ideas, for advice.” This level of trust was crucial as teachers had different comfort levels with computing and could seek help regardless of how complex or rudimentary their computing needs.

Ms. Smith, the 3rd grade teacher, expressed her reliance on the librarian:

> He doesn't just push me, you know. He's always there, always asking questions…he'll give encouragement too. He'll say, ‘Oh, your kids are doing really well. They're probably doing what the 4th and 5th graders are doing.’ It's nice that he gives encouragement.

Ms. Rosen also expressed how meaningful it was that administrators and coaches publically acknowledged the computing work she was doing with her students, “Do you know how proud I was at the staff meeting when he said, ‘You ought to see what her kids are doing with Etoys in science.’ I'm like, yeah, we're going to do more. Yeah!” That recognition for effort and trying something new spurred these teachers to continue to explore computing with their students.

Ms. Lyle similarly expressed, “It has been wonderful to have people in here that know what it [computing] is and if I could turn to them, I could, you know. At first, those first three days, it was so nice to have helpers.” She stated that this support extended to feedback:

> It was good to hear feedback because she just kept pumping me up…it was really nice to have somebody that kind of saw the different levels and what was working and what wasn't working and just someone to discuss it with was helpful.

Thus, it was a combination of opportunities for professional development in which teachers had time to learn distinct computing skills and consider how to apply those within their instructional settings and access to embedded coaches that could help the teachers develop lesson plans as well as co-teach with them until they became more confident and familiar with integrating computing into their instruction. As teachers became more comfortable with computing education, they relied less on the coaches for co-teaching although they continued to work with them as co-planners.

8.2.3. *Administrative support was crucial and educating administrators about the realities of computing education was important*

In all teacher interviews, the teachers indicated that administrative support for computing was key for successful implementation. Ms. Lyle, the art teacher, explained, “I think it's huge that our assistant principal is trained as a technology person. You know, I think that is one of the driving things." Although the teachers acknowledged the support of the administration, the librarian expressed that he needed to communicate the different instructional practices used during computing to the principal. He explained:

> I went to our administrator and said I need you to tell the teachers it's ok that they're doing computing, their kids are up and moving around and that they're talking and that they are - things look crazy at times and that's ok. We want that. And we've gotten great support for that.

In this way, he advocated for the classroom teachers who had concerns about the unstructured nature of computing exploration in their classrooms by explaining to the principal that students were encouraged to collaborate and move to see each other's work. While the librarian cited the administration support, he also acknowledged that there are top-down initiatives that could significantly influence administrative support for computing including the PARCC assessment and the new teacher evaluation. Therefore, despite the clear support of the administrators, the teachers still had reservations about how computing education would be viewed in light of competing important initiatives such as the new state assessments and the teacher evaluation system. The collective sentiment was gratitude for the administrative support and hope that it would continue despite these other initiatives.

The administrators, when asked about their roles in the school-wide computing initiative, indicated that they recognized that teachers may consider integrating computing into their instruction as risky, given new teacher evaluation expectations, the demands to cover all the content standards, and the different classroom management practices associated with computing as compared to traditional instruction. They stated that they needed to alleviate these concerns by clearly communicating administrative support. The principal explained:

> So with the new teacher evaluation, it made our teachers very nervous. So the thought of technology and new teacher evaluation in the same year…I realized I had to convey to them that it was going to be okay to be messy and that it was going to be okay to be scared and nothing was going to be perfect this year, and that it was not a 'gotcha' year with the evaluations.

The assistant principal further described the administrative support:

> I have been shockingly surprised at how many people have embraced the use of it [computing education] with very little, I would say, directives as far from the admin. It's just been supportive. It's been encouraged. We've tried to give resources. We have had our partners, you know, providing resources and just saying we are available.

Overall, both teachers and administration mentioned the vital importance of administration support for computing to be implemented and sustained.

8.3. Research question three: what barriers to implementation occurred during the adoption of computing into instruction?

All the teachers as well as the administrators described barriers to teaching computing. These barriers became apparent during observations. Interestingly, when the teachers and administrators discussed these barriers, they did so within the context of problem solving and moving towards solutions.

As could be expected, some barriers were easier to solve whereas others were more institutional in nature. Table 2 includes the six major barriers described by the teachers and administrators and observed during implementation observations as well as strategies that they used in overcoming those barriers. Of these, lack of technology, lack of computing expertise, and students' status as at risk for academic failure due to poverty and disability were the three main barriers identified by the teachers and are described below.

8.3.1. Lack of technology

The most common barrier cited was lack of technology. All the teachers and administrators described technology-related barriers, but they all found ways around these barriers. The principal described this mindset as related to the lack of technology infrastructure in the school building:

> It's like, let's build a house, but you can't have the hammer. I'll give you the nails…So figure it out…and that's kind of where we are. We're figuring it out. We're going to build the house anyway. [We've] just got to figure out how to hammer those nails without the hammer.

Table 2
Teacher-identified barriers to computing education and strategies.

Barriers to computing implementation	Strategies implemented to overcome barriers
Access to technology (e.g., computers and computer mice)	Students rotated to different classrooms using computers in library and computer lab; Teachers created Donors Choose accounts to ask for computer and other technology.
Access to expertise support in classroom	Tech teacher/librarian support; University faculty and graduate student support and coaching; utilizing online resources (e.g., EtoysIllinois.org website).
Computing access issues due to poverty and disability/ Struggling learners	Peer support & collaboration; one-on-one support; balance of explicit and open instruction; technology supports
Limited instructional Time	Teachers integrated computing into core curriculum.
Lack of students' computing experience	Students encouraged to explore without "correct" answers; Teachers differentiated for various levels of exposure; Individualization; Student choice
Classroom space (classroom computer center)	Some teachers created computing stations with 3–4 computers. Others used computers in library or lab or reserved one of two laptop carts.

The teachers shared this problem-solving mindset as well. For example, when the school was having Internet server issues, the teachers communicated that to the administrators, who quickly worked with district personnel to replace the server. When the art teacher found that the kindergarteners were having difficulties with the standard-sized mice and that some kindergarteners with disabilities had difficulties with the small mice, she sought information from the librarian as well as university faculty in order to find appropriate mice for her students.

Additionally, the administration recognized that they could not fully remediate this issue on their own so they worked with local entrepreneurs with expertise in community organization, who recommended that the teachers create Donors Choose accounts to request classroom technologies. Donors Choose (http://donorschoose.org) is an online charity website where teachers post project requests and the needed funds to complete those projects. When the projects raise the required amount of funds, the organization ships the materials to the schools. In this way, teachers could obtain additional laptop computers for computing activities.

The librarian stated that lack of technology was an initial barrier that lessened throughout the course of the year. He explained that with the addition of extra laptops, the issue shifted towards challenging teachers to use all the available technology throughout the school day. For example, when the school received two laptop carts, challenges emerged around procedures for accessing the laptop carts. Procedures were set for checking these carts out by teachers, but the fact that the carts were not in the teachers' classrooms resulted in a barrier as the teachers did not want to disrupt instruction to retrieve the carts. With time, however, the teachers trained students to safely transfer the carts to the classrooms, thus bypassing this issue.

8.3.2. Lack of computing expertise

All of the classroom teachers were new to computing. The only personnel with extensive technology experience were the technology teacher, the librarian, and the assistant principal. As stated above, many considered computing important but had concerns about their ability to teach with technology with which they were not fully comfortable. The assistant principal stated that one of her questions was, “Can we get staff to learn [computing] and try something new when time is a struggle?” She recognized that it would be unrealistic to expect the teachers to fully implement computing education without a certain level of expertise. Consequently, she and the other administrators encouraged teachers to start small and work towards increased computing experiences.

Mr. Ryan, the librarian, explained that at the beginning of the year, teachers made comments such as, “You know we're not experts.” He acknowledged this lack of expertise and addressed that need through embedded coaching, co-teaching, and professional development. He also expressed teachers' discomfort with the shifting control from teachers as “sage on stage” to a more student-centric model. Ms. Rosen, the 4th/5th grade split classroom teacher, expressed her concerns about her lack of expertise and the need for support, “My biggest fear was I would look stupid. If you don't make me look foolish in front of the kids and it's safe for me to try, I'll do it.” Mr. Ryan indicated that this response slowly shifted throughout the course of study, especially after the Hour of Code week, in which computing was implemented in all classes and the teachers had more experience with how to integrate computing into their classrooms. Mr. Thomas, the technology teacher, stated that these teachers, “moved from the acknowledgement of not being experts to by the end of the Hour [of Code Week], them wanting to learn more.”

The teachers also realized that they did not need to be technology and computing experts because they could rely on the students as experts. In fact, all the teachers acknowledged that the students were learning about computing more rapidly than they anticipated. Once the teachers found that the students could serve as peer helpers and experts, they shifted their instruction to offer more student ownership, thus relieving them of some of the anxiety for not having all the answers.

8.3.3. The role of poverty and disability on students' computing experience

Another commonly mentioned teacher concern related to students' diverse levels of access to technology as well as the role of disability on how fully students could participate in computing activities. Like the other barriers that emerged in this study, the teachers and administrators acknowledged the real barriers associated with academic risk factors, poverty, and disability on students' full participation in computing, but did so in a proactive manner that encouraged participation rather than accept these limitations.

Issues that emerged for students without access to technology at home and those with disabilities that influenced fine motor skills included lack of mouse skills including how to double click, how to drag an object, and how to right click. These students also had difficulty with keyboarding functions such as using the “control, alt, delete” sequence. Students with disabilities who struggled with reading had difficulty reading the commands and reading within Scratch and Etoys as well as with complex problem solving involved in some of the computing activities.

Ms. Smith explained that only a few of her students have access to technology and use computers daily:

> Those kids [with access to technology at home], compared to the kids that don't have access at home, it's like a stark contrast in their level of how they use Etoys and what they know how to do … They don't know how to use the mouse, you know … They don't know how to click and drag as the same time. They're not familiar with that because they haven't touched a mouse … and then right click–the difference between right and left.

Despite these students' difficulty with some computing tasks, teachers observed that most of the students were engaged and experienced successes with computing. The factors that they attributed to student success included peer support and collaboration, encouragement of risk-taking, and differentiation. Mrs. Hawthorne, the 2nd grade teacher, explained, “[Computing] breaks that straight line [of thinking] and allows freedom and in a way that's really not unsafe for the students.”

Overall, issues of students' meaningful access to computing activities remained a constant barrier to implementation that both the administrators and teachers acknowledged and with computing activities could learn to compute, collaboratively problem solve, and persevere through challenging instructional tasks. All participants were optimistic that with increased emphasis on computing education, students would continue to benefit from these experiences.

8.3.4. Limited instructional time

The teachers and administrators all expressed lack of instructional time as a barrier to implementation throughout the study. Although they remediated this issue by integrating computing into the content areas, this is the one barrier that remained a consistent challenge for the teachers. They were all committed to preserving time for computing while at the same time teaching the mandated curriculum, preparing their students for state assessments, and meeting the demands of other important school-wide initiatives such as their rigorous reading program to support all the struggling readers in their school.

Despite this challenge, the teachers and administrators found ways to find instructional time for computing including: (a) integration into curriculum as described above, (b) participating in the Hour of Code (See Code.org) and creating a second Hour of Code experience a few months after the one organized by Code.org and (c) including computing within specials such as library and art. In addition to these immediate changes, the teachers and school administrators communicated with the district administration about the need for more flexibility in the curriculum so that computing could more easily be integrated. For example, during the course of the study, the school staff had specific amounts of time for reading and mathematics in 90-min blocks. There was no flexibility in subject integration such as teaching science and reading concurrently. By the end of the study, these conversations began to result in action plans towards this goal such as developing a school-wide plan for ensuring content would be covered through this integrated model.

Overall, the barriers identified by the teachers and those identified during observations lessened throughout the data collection period as a result of teachers' and administrators' problem-solving as issues and barriers emerged. Additionally, as teachers' expertise and experience increased, they were able to discover solutions to problems that initially appeared challenging, such as finding time to integrate computing into their instruction.

8.4. Research question four: how did teachers support diverse learners, including students with disabilities and those at risk for academic failure due to poverty?

8.4.1. Poverty and disability status influenced students' computing experiences

As mentioned above, both observation and interview data revealed that students' meaningful access within computing instruction and with technology in general were influenced by both (a) lack of access to technology due to poverty and (b) difficulty with problem solving and computing due to disability or other academic risk factors. Initially, the researchers' hypothesized that students' disability status would be a greater barrier to meaningful access than poverty. Although this was the case for students with more significant disabilities, this hypothesis was negated as students who lacked access to technology due to poverty struggled more than students with mild to moderate disabilities (e.g., learning or emotional and behavioral disorders). Students living in poverty were understandably much less likely to have computers in their homes and access to mobile devices. Consequently, they did not have opportunities to learn fundamental skills such as using a mouse or trackpad, dragging, double clicking, etc. This lack of experience in basic computer navigation was an initial barrier for many students in poverty.

Both students in poverty and those with disabilities and other risk factors required additional support from the teachers, who differentiated assignments and offered necessary scaffolding. Ms. Smith described:

> I think we try to stick with them a little bit more and I think I've tried to change the project in different ways to kind of meet their needs. I think that's one thing that I do like about Etoys, that I feel like I can use it with my diverse group of kids, you know, like some kids who might just not make the maze really work the way that it's supposed to work. They might just be drawing down the maze and learning how to use the flap and writing something in the flap. Whereas other kids worked out their maze and now are trying to animate other things within their maze … I feel like I can meet all my kids' needs with the same project.

8.4.2. Regardless of learning challenges, most struggling learners benefited from computing

Despite the challenges that students faced with computing instruction due to poverty or disabilities, both observation and interview data highlighted that these struggling learners generally thrived in the computing environments. Ms. Rosen, the teacher in the 4th/5th grade split class, explained:

> I find that even the students who are most challenged find computers friendly because it doesn't care if you're right or wrong … even my lowest kids, the ones who are so academically challenged, they're the first ones to want to get on a computer and try it because it's safer.

The art teacher, Ms. Lyle, similarly stated:

> Kids that were functioning lower in the classroom, some of them shined on this [computing]. Great shining moments for them because they could do the drawing, they could handle the mouse, and they could read enough to do it. It was really rewarding for them.

She later expressed that some of these typically struggling learners who found success with computing began to take leadership roles in helping other students. This was a common phenomenon stated among the teachers. Because computing was often taught within the context of flexible, differentiated instruction with multiple options for expressing understanding, these struggling learners found success and could share that with their peers. Therefore, students who generally struggled academically did not necessarily struggle with computing and could take on leadership roles that were not available to them during traditional instruction.

All of the teachers discussed the role of peers in supporting struggling learners. Observational data supported this sentiment. Peer support often resulted in increased perseverance and increased positive attitudes about computing tasks. Ms. Rosen explained, “If a peer is sitting next to them and said, ‘Oh, you did that wrong, do that one again,’ there isn't any judgment. It's like, ‘Oh yeah, I did.’” She stated that if she offered correction, some students would be less likely to accept help compared to feedback from peers. The combination of scaffolding, differentiated assignments, and peer support resulted in increased student engagement for those at risk of academic failure due to poverty or mild to moderate disability.

Students with more significant disabilities, such as those with intellectual disabilities, often did not participate in computing activities. When talking to the teachers about this issue, they stated that they did not know how to support this group of learners in fully engaging in the classroom activities. The teachers also stated that the students with more significant needs displayed major fine motor, executive functioning, and cognitive needs that would have required assistive technology as well as more structured activities that the teachers did not feel prepared to address. They stated that this is an area they are committed to addressing as the year continues and have reached out to professionals with experience in this area.

8.4.3. *Balance between explicit instruction, individualization, and scaffolded inquiry helped to support all learners*

All seven teachers in the study relied heavily on both modeling and scaffolding the open inquiry experiences for their learners. They explained that this was important to support those students that needed modeling and step-by-step instruction. Ms. Lyle explained, “You know, it's just slowly getting those things added on. It's just like adding on a little bit each time, like a new vocabulary word.”

When instruction began, the teachers always provided some level of explicit instruction. In one lesson, for example, Mr. Ryan provided students with directions for how to import images from Google into Etoys. He guided students through choosing an image, saving that image, selecting that image, and then dragging it into Etoys. Throughout this process, he explained the steps, modeled the steps, and checked student understanding.

They all expressed the inherent ease with which computing can be individualized. Ms.
Ms. Lyle explained:

> I had a couple students that they could draw on it if you set it up … They had to have somebody right next to them and walk them through it … I really focused on them getting comfortable with the mouse, drawing, it's ok. It's the experience more than anything … whether they're completely functional with it all and making something move.

8.4.4. *Student collaboration in computing elicited problem solving and minimized the role of the teacher as expert*

Both observation and interview data indicated that encouraging student-to-student collaboration was an effective method for engaging students with and without disabilities in computational thinking. Ms. Smith explained, “I think in the beginning, it was frustrating for a lot of them, like what is this X, Y thing and how do I get it to move, but I think when we implemented peer teaching, that worked really well.” Ms. Smith discussed how she (and several other teachers) strategically implemented peer teaching in order to help students become more self-reliant. She explained:

> The first thing we did was they had to ask two peers around them before they could ask an adult and then their peers had to figure out, you know, had to help them out and if they didn't know, then they would ask an adult. We had to train a couple of peers before we actually went into the project … then those kids were kind of the experts and they would actually go around and help other kids.

Although they were able to successfully collaborate with their peers on computing, the students in this study were initially skeptical of peer-to-peer collaboration. For example, the librarian explained:

> When you tell the kids you don't have to do this on your own, this is something you can work together on and you're not stuck at your spot and your table by yourself; get up and move, talk, and look at what that other person is doing. And then they sit there and it's like no.

Once students understood that collaboration was encouraged, three primary modes of student-to-student collaboration emerged over time. These three themes included teacher prompted collaboration, organized collaboration, and student-initiated collaboration.

8.4.4.1. Teacher prompted collaboration. In many instances, student-to-student collaboration was achieved through teacher prompting. For example, teachers were observed prompting their students to collaborate with each other by using priming phrases such as, “remember that your friends are helpers too!” or “please ask for help from a friend before coming to me.” In this way, students were given a reminder to initiate collaborative relationships with their peers around computing before seeking teacher support.

8.4.4.2. Peer mentoring as collaboration. In addition to prompted peer collaboration, data indicate that organized peer mentoring was also an effective mode of student-to-student collaboration during computing. The teacher organized this partnership. Each pair of students was assigned as either the “mentor” or “mentee” and each student was responsible for guiding the other through various computing tasks without adult support. The technology teacher shared his experiences observing their interactions:

> So fifth graders, for instance, they go in and help or pair up with kindergarteners and I just love how they interact with little kids. At first they try to do everything on their own. But then once you give them instruction on how to help them, then they really become the teacher.

In many cases, students felt more empowered by their ability to help support each other, rather than being directed by teacher instruction. Through this more organized collaboration, students had the opportunity to offer support and receive support from their peers. The dynamics of these relationships seemed to be particularly rewarding for struggling learners who, in other instances were not able to provide support for their peers.

8.4.4.3. Student-initiated collaboration. Findings also revealed that students initiated collaboration to support problem solving as well as to share successes. That is, students naturally and independently problem solved with their peers when they needed computing support. On many occasions, when teachers identified student-initiated collaboration, they reinforced students' efforts through verbal praise and shared their successes with other peers. This was, in a way, a shift away from traditional schooling models in which teachers directed information, activities, and student conversations. Ms. Rosen explained, "In a computer setting, when someone's sitting next to you and has got something really cool going on, you can say, 'How did you do that? I want to make mine do that!'" Thus, all three types of student-to-student collaborative models supported struggling learners both because they could rely on their peers for help and because they could shift roles and become the helper or mentor.

9. Discussion

This study sought to answer four questions related to implementing school-wide computing in a diverse elementary school environment with many students at risk for academic failure. In doing so, it became apparent that teachers were extremely positive about computing and their rationale for embedding computing into instruction was consistent with the literature that points towards the power of computing in increasing students' problem solving and higher-order thinking skills (CSTA, 2003; Fessakis & Mavroudi, 2013; Kay & Knaack, 2005; Papert, 1991).

9.1. RQ one: how was computing integrated across different instructional contexts?

Teachers took different instructional approaches to computing instruction contingent on their instructional context. All four classroom teachers embedded computing into their core curricula in order to teach computing while at the same time meet the curricular benchmarks set by the school district. Computing was integrated primarily into mathematics and science instruction. In interviews, all four of these teachers pointed to the need to maintain the pacing guides set by the school district in teaching content, and that computing was successfully taught because its integration did not take additional instructional time.

This finding interestingly aligns with recommendations from the field, but for pragmatic school-related context reasons rather than the reasons cited in the literature. The teachers taught computing within the content areas in order to create the time to teach computing. The literature on embedding computing into the content areas posits, on the other hand, that this integration helps students apply computing within the various domains, thus avoiding teaching computing in an isolated manner devoid of its real-world application (Jona et al., 2014; NRC, 2011; Weintrop et al., 2014). Thus, this integration occurred for different reasons.

On the other hand, the three specialized teachers (i.e., librarian, art teacher, and technology teacher) taught computing within their disciplines, but had more flexibility. Consequently, these specialized teachers taught more discrete computing skills. Like Fessakis and Mavroudi (2013), the teachers found that even very young learners benefited from computing and could engage in fairly sophisticated processes within the tile-based programming software. These teachers stated that although the content teachers taught computing within their content areas, there was still a need for discrete computing instruction, which the content teachers did not have time to teach. These specialized teachers used CS Unplugged activities, Code.org curricula, and Scratch tutorials to teach specific computational thinking and programming skills with the hope that these would then be applied in the content areas.

9.1.1. Implications for research and practice

This study highlighted the importance of both embedded computing tied to instructional content and explicit computing education so that students can both learn the discrete skills and thinking processes associated with computing and apply them within authentic instructional contexts. Given the confusion and level of debate regarding the definition of computational thinking and how it is operationalized in K-12 instruction (Grover & Pea, 2013; NRC, 2011), both researchers and practitioners should carefully describe their computing initiatives.

Future research should evaluate (a) how discrete skills learned through computing and computational thinking education is generalized by students applying these skills within the content areas, (b) how content teachers and technology teachers collaborate to align core curriculum with computing instruction, and (c) how the readily available computing resources can best be integrated into both instructional contexts.

As teachers consider ways of introducing computing and computational thinking into their instruction, they should consider both integrated and discrete methods of instruction and how these fit within their core curriculum and computing education goals. There is a growing body of resources for teachers related to both discrete computing instruction and computing within the content areas including:

- Code.Org (http://code.org)
- EtoysIllinois (http://etoysillinois.org)
- Scratch-Ed resources (http://scratched.media.mit.edu/resources)
- Computer Science Unplugged (2010) (http://csunplugged.org)

Although these resources are helpful to teachers, it is important to note that the teachers in this study needed to do a great deal of their own curriculum development, especially those who tried to integrate computing into their own instructional contexts.

9.2. RQ two: what types of supports did teachers request and use to implement computing education?

The classroom teachers in this study (2nd, 3rd, 4th/5th, enrichment, and art) did not have previous computing education knowledge or experience. Therefore, without professional development, they would not have attempted to integrate computing into their instructional practice. The week-long professional development, although helpful in giving the teachers an introduction to computing education, was not sufficient. This is consistent with research on professional development and implementation science that posits that professional development should be embedded into the instructional context (Fixsen et al., 2005). The coaching that was offered after the initial and winter-break workshop helped the teachers apply what they learned in the workshop to their settings. Because they did not have the expertise to implement computing on their own, the teachers relied on the librarian, technology teacher, and university staff who could help them learn how to implement computing within their unique settings.

In addition to professional development, all the teachers expressed the importance of administrative support as well as administrative understanding of how school-wide computing would impact instruction. It was not enough for the administrators to simply state that they wanted computing to take place and that they supported teachers' efforts. The teachers expressed that it was important for administrators to have a deep understanding of how computing would impact pedagogy, classroom management, and other initiatives such as teacher evaluation and state assessments and that the administration would communicate their priorities regarding this interplay to the teachers. They stated that without this type of communication, the teachers would be less likely to take instructional risks. This sentiment was also consistent with the implementation science literature that discusses the roles of administrators are crucial to any long-term instructional practice to take root (Fixsen et al., 2005).

9.2.1. Implications for research and practice

There is still a great deal to learn about the roles of professional development and administrative support in beginning and maintaining school-wide computational thinking initiatives. Future research should evaluate (a) how to sequence professional development for teachers with no or limited computing experience so that they could integrate those experiences into their instruction effectively, (b) the types of coaching experiences that teachers receive as novices and as they progress towards more experienced with computing, and (c) the interplay between computing education and other educational initiatives and how administrators balance the demands of these as well as communicate with a staff that is actively engaged in positive risk taking.

Teachers and administrators considering ways of implementing school-wide computing should consider the need for both targeted and embedded professional development as well as time for teachers to work with coaches on curriculum development. Without the adequate time and financial resources to do so, it is unlikely that computing will take root across elementary classrooms. Additionally, administrators should clearly articulate their expectations regarding computing and assure teachers that as they learn how to integrate computing into their instruction, they will not be penalized for their instructional risk taking.

9.3. RQ three: what barriers to implementation occurred during the adoption of computing into instruction?

Several barriers to implementation became apparent during this study, although all but one of these barriers lessened over time. One of the barriers to implementation that was anticipated by the researchers and still emerged was the teachers' lack of experience with technology and tile-based programming software. The researchers developed both targeted and embedded professional development to address this need and had to continue to ensure that teachers had the time and access to expertise to address this barrier. Findings related to lack of teacher expertise was consistent with the literature that points towards people's naive or inaccurate perceptions of computer science and programming (Armoni, 2011) as well as concerns regarding perceptions of the difficulty of computing. Wilson and Moffat (2010) explained that teachers' lack of understanding of computational thinking can lead teachers to assume that computational thinking and computer science is too difficult to learn. Findings from this study confirmed this issue, although when teachers were given appropriate professional development and embedded coaching, these perceptions of computational thinking diminished. The teachers expressed initial concern about teaching computing, but all expressed diminished concerns as their experiences increased. The teachers also relied on their students to serve as computing experts to help the other students in the class when the teachers did not know solutions to individual problems. This finding was consistent with the NRC (2011) report that stated teachers often rely on students to help troubleshoot and act as experts. This report also strongly noted the importance of professional development to build teachers' skills in computing that aligns to teachers' curricular as well as computational needs.

9.3.1. Implications for research and practice

Computing and computational science in schools is now getting significant press attention and resources are being focused on growing computational thinking in the schools. It is important for researchers to systematically document the processes taking place in schools as they venture into this new curricular space. This study has attempted to do this for the case of a single school. Future research must evaluate ways of alleviating the various computing barriers related to technology access, teacher knowledge, student demographics, and other barriers identified in this study occur in different contexts. For example, researchers should investigate how different computing software creates opportunities to introduce computing into tightly packed K-12 instructional contexts and whether some of the software, associated professional development, and learning progression models available reduce these barriers more than others.

As teachers and administrators consider ways of successfully including computing into their own instructional contexts, it is helpful to consider the strategies used by teachers in this study, particularly the entrepreneurial and collaborative methods they utilized to gain access to technology and professional development as well as communicate their needs to their administrators.

9.4. RQ four: how did teachers in different instructional contexts support struggling learners, including students with disabilities and those at risk for academic failure due to failure?

Both observation and interview data indicated that teachers spent a great deal of time and effort making computing accessible to students with diverse learning needs. Teachers, for example, created learning experiences that were flexible, included modeling, and scaffolded computing instruction and independent work through both guided practice and collaborative problem solving. Early in the year, they noticed that many struggling learners displayed signs of learned helplessness and overreliance on teachers, so they instituted a collaborative problem solving process. This phenomenon is consistent with literature that indicates that struggling learners often have below average perceptions of their academic skills, often attribute failure to external factors, perceive tasks as too difficult, and lack confidence to complete tasks (Montague & Applegate, 2001; Nunez et al., 2005). By focusing on collaborative problem solving, the teachers encouraged peer support that facilitated increased socialization and engagement for struggling learners.

In speaking with the teachers and administrators, it became apparent that they viewed computing from a social justice perspective. They recognized that because the majority of their students do not have access to technology at home and have numerous academic risk factors, a focus on computing and digital citizenship could extrapolate to other areas of learning and possibly lead to increased career options. This realization is consistent with the literature related to the digital divide that states that students in poverty do not have the computing skills of their more affluent peers (Valadez & Durán, 2007). Observational data showed that teachers used instructional best practices that were effective in other instructional contexts such as modeling, explicit instruction, and peer collaboration. These instructional strategies, according to the teachers, supported their struggling learners in the context of computing.

9.4.1. Implications for research and practice

Teachers of struggling learners (including students with disabilities and those at risk for academic failure due to poverty) stated that it appeared that these students thrived within this computing environment but needed additional scaffolds to help them fully engage. Future research should more closely evaluate students' engagement and learning across demographics including disability status and socioeconomic status in order to more fully understand the computing needs of students at risk for academic failure and then develop interventions and instructional supports to address those needs. This research should include students from a wide range of demographics including students with a range of disabilities. Future research should evaluate the role of scaffolding, modeling, and peer collaboration more closely and identify ways in which these instructional best practices can be tailored to the context of computing.

10. Limitations and implications for future research

This study included several limitations that should be noted. First, this study took place in one school environment so findings from this study may not necessarily generalize to other contexts. Because this study used an implementation science perspective that accounted for practitioner adaptation (Fixsen et al., 2005; Harn et al., 2013; Odom, 2009), it holds true that each instructional site will approach computing education in a slightly different manner, depending on the goals of the schools, the computing experiences of the teachers, and other features that distinguish one school from another. Although cross case analysis took place among teachers within this school, findings would likely be somewhat different in other elementary schools. Future research, therefore, should replicate this study across different elementary school sites to learn what features of school-wide K-5 computing education may be consistent across settings, and what features change with instructional contexts.

Second, this study primarily focused on teacher implementation of computing rather than on specific student outcomes. Future research should examine student engagement and learning in more detail. Although such studies exist, as noted above, very few studies examine students from diverse backgrounds. For example, there were no studies in the literature examining students with disabilities. Future research should use multiple data sources including data related to engagement, collaborative problem solving, and learning progressions of computing skills across grades and with diverse learners. For example, research should begin to align the CSTA standards with curricula available through sites such as Code.org as well as through open-ended computing activities in software such as Scratch and Etoys and evaluate how students at different ages and grades and with different abilities and backgrounds work through various computing tasks.

Lastly, it is important to note that this study took place during four months of instruction and teachers' implementation of computing changed over time. For example, after the end of the study, teachers transitioned to less reliance on coaches for co-teaching. Future research should examine computing over time to understand how to create a sustainable culture of school-wide computing education.

11. Conclusions

This study provides exploratory findings related to how elementary school teachers integrated computational thinking into their instruction within the context of a school-wide computing initiative. It showcased both the challenges and benefits that teachers encountered during a remarkable year in which they moved from no experience to a place of pride in being a school with an identity based on use of new technology. This study opens the door for additional research both into teachers' experiences as well as their students' learning and engagement. More research is necessary in order to understand implementation, supports that teachers require, and the types of instructional strategies that could support diverse learners in engaging in computational thinking. This research is essential given the national movement towards including more computing in K-12 settings.

Appendix. Semi-structured Interview protocol for Administrators, technology teachers/specialists, and classroom teachers

Administrator Interview Questions

1. What is your impression on how computing and computer programming started at your school?
2. How do you envision school-wide CS at your school?
3. What has surprised you most about computing at your school? Probe for answers related to students and teachers.
4. What are your expectations around computing for struggling learners and kids with disabilities? Has this shifted since the beginning of this project?
5. Did you encounter any barriers to implementing computing this year? Probe: did you anticipate those barriers at the beginning of the year? Probe: Have these barriers shifted as the year has progressed?
6. What kind of feedback have you received from the teachers who have implemented computing?
7. Tell us about some of the teachers who are hesitant to teach computing? Probe: What kinds of supports have you provided to teachers implementing computing?
8. If you could go back to the beginning of the year, is there anything you would change in how computing was implemented this year?

Technology Teachers Interview Questions

1. What has surprised you most about computing at your school? Probe for answers related to students and classroom teachers.
2. What do you hope computing will look like next year?
3. How do you envision school-wide CS moving forward at your school?
4. What are the challenges/barriers that you face with that vision?
5. Tell us about how students are responding to computing education? Probe for examples
6. Describe your collaboration with classroom teachers in computing implementation? Probe for information about collaboration with high implementers and hesitant implementers.
7. What do you find to be the most effective way to provide professional development to classroom teachers?
8. How have your interactions with your colleagues changed since starting the computing education project?
9. What do you think the role of computing/CS is for kids who struggle because of disability, poverty, or other issues making them at risk for academic failure?
10. What are your expectations around computing for struggling learners and kids with disabilities? Probe: Has this shifted since the beginning of this project?
11. What kinds of direct supports are you providing? Probe: What strategies appear to be most effective to support struggling learners? Novice computer users? Kids that struggle with computing?
12. What do you consider measures of success for students using computing software?
13. What kind of feedback have you received from the teachers who have implemented computing?

Classroom Teacher Questions

1. Describe your experience with computing education this year? Probe: How have you implemented computing? How much instructional time have you dedicated to computing?
2. What has surprised you most about computing education at your school?
3. Have you faced any challenges in implementing computing? Probe for additional information and examples.
4. What are your expectations for your students around computing? Probe for information about struggling learners and kids living in poverty and without access to technology.
5. What are your goals for moving forward with teaching computing next year?
6. Are you providing any different support/instruction to struggling learners? If so, how are you providing CS instruction to struggling learners? Probe: (If yes) What kinds of direct supports are you providing? What strategies appear to be most effective to support struggling learners?
7. Describe your collaboration with [names of computing exerts] at your school. Probe: What aspects of these collaborations were helpful and why?
8. What has the administration communicated to you regarding their expectations around computing? Probe: Have the administrators provided any support/feedback related to teaching computing?

References

Agalianos, A., Noss, R., & Whitty, G. (2001). Logo in mainstream schools: the struggle over the soul of an educational innovation. *British Journal of Sociology of Education, 22*(4). http://dx.doi.org/10.1080/01425690120094449.

Anybody can learn. (2014). Retrieved from https://code.org.

Armoni, M. (2011). The nature of CS in K–12 curricula: the roots of confusion. *ACM Inroads, 2*(4), 19–20. http://dx.doi.org/10.1145/2038876.2038883.

Aschbacher, P. R., Li, E., & Roth, E. J. (2010). Is science me? High school students' identities, participation and aspirations in science, engineering, and medicine. *Journal of Research in Science Teaching, 47*(5), 564–582.

Baytak, A., & Land, S. M. (2011). An investigation of the artifacts and process of constructing computers games about environmental science in a fifth grade classroom. *Educational Technology Research and Development, 59*, 765–782. http://dx.doi.org/10.1007/s11423-010-9184-z.

Bouras, C., Poulopoulos, V., & Tsogkas, V. (2008). PeRSSonal's core functionality evaluation: enhancing text labeling through personalized summaries. *Data & Knowledge Engineering, 64*(1), 330–345. http://dx.doi.org/10.1016/j.datak.2007.07.007.

Burke, Q., & Kafai, Y. B. (2014). *Decade of game making for learning: From tools to communities. Handbook of digital games* (pp. 689–709). http://dx.doi.org/10.1002/9781118796443.ch26.

Clark, J., Rogers, M. P., Spradling, C., & Pais, J. (2013). What, no canoes? Lessons learned while hosting a scratch summer camp. *Journal of Computing Sciences in Colleges, 28*, 204–210.

Computer Science Education Week. (2014). Retrieved from http://csedweek.org.

Computer Science Teachers Association (2003). Retrieved from http://csta.acm.org.

Computer Science Teachers Association (2010). Retrieved from http://csta.acm.org.

Computer Science Unplugged. (2010). Computer Science Unplugged Creative Commons Attributions-Noncommercial-No Derivatives Works 3.0 License. Retrieved from http://csunplugged.org/.

DonorsChoose.org. (2000–2014). *Donors choose: teacher ask. You choose*. DonorsChoose.org. Retrieved from http://www.donorschoose.org/.

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

Fixsen, D. L., Naoom, S. F., Blase, K. A., Friedman, R. M., & Wallace, F. (2005). *Implementation research: A synthesis of the literature*. Tampa, FL: University of South Florida, Louis de la Parte Florida Mental Health Institute, The National Implementation Research Network (FMHI Publication #231) http://ctndisseminationlibrary.org/PDF/nirnmonograph.pdf.

Gardner, M. K., & Feng, W. C. (2010, June). Broadening accessibility to computer science for K-12 education. In *Proceedings of the fifteenth annual conference on innovation and technology in computer science education* (pp. 229–233). ACM. http://dx.doi.org/10.1145/1822090.1822155.

Glaser, B., & Strauss, A. (1967). *The discovery of grounded theory*. Hawthorne, NY: Aldine Publishing Company.

Good, J. (2011). Learners at the wheel: novice programming environments come of age. *International Journal of People-Oriented Programming, 1*(1), 1–24. http://dx.doi.org/10.4018/ijpop.2011010101.

Grover, S., & Pea, R. (2013). Computational thinking in K–12 a review of the state of the field. *Educational Researcher, 42*(1), 38–43. http://dx.doi.org/10.3102/0013189x12463051.

Harel Caperton, I. (2010). Toward a theory of game-media literacy: playing and building as reading and writing. *International Journal of Gaming and Computer-Mediated Simulations (IJGCMS), 2*(1), 1–16. http://dx.doi.org/10.4018/jgcms.2010010101.

Harn, B., Parisi, D., & Stoolmiller, M. (2013). Balancing fidelity with flexibility and fit: what do we really know about fidelity of implementation in schools? *Exceptional Children, 79*(2), 181–193.

Hug, B., & Reese, G. (2006). How technology integration in mathematics and science teaching can occur: the role of the maverick teacher. *Teaching Education, 17*(2), 167–179. http://dx.doi.org/10.1080/10476210600680390.

International Society for Technology in Education and The Computer Science Teachers Association. (2011). *Operational definition of computational thinking for K-12 thinking operational-definition-flyer.pdf*. Retrieved from http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf?sfvrsn=2.

Jona, K., Wilensky, U., Trouille, L., Horn, M., Orton, K., Weintrop, D., et al. (2014). *Embedding computational thinking in science, technology, engineering, and math (CT-STEM)*. Presented at the Future Directions in Computer Science Education Summit Meeting, Orlando, FL.

Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. MIT Press.

Kay, R. H., & Knaack, L. (2005). A case for ubiquitous, integrated computing in teacher education. *Technology, Pedagogy and Education, 14*(3), 391–412. http://dx.doi.org/10.1080/14759390500200213.

Kwon, D. Y., Kim, H. S., Shim, J. K., & Lee, W. G. (2012). Algorithmic bricks: a tangible robot programming tool for elementary school students. *Education, IEEE Transactions on, 55*(4), 474–479. http://dx.doi.org/10.1109/TE.2012.2190071.

Lambert, L., & Guiffre, H. (2009). Computer science outreach in an elementary school. *Journal of Computing Sciences in Colleges, 24*(3), 118–124.

Lee, Y. J. (2011). Empowering teachers to create educational software: a constructivist approach utilizing Etoys, pair programming and cognitive apprenticeship. *Computers & Education, 56*(2), 527–538. http://dx.doi.org/10.1016/j.compedu.2010.09.018.

Liao, Y. K. C., & Bright, G. W. (1991). Effects of computer programming on cognitive outcomes: a meta-analysis. *Journal of Educational Computing Research, 7*(3), 251–266. http://dx.doi.org/10.2190/E53G-HH8K-AJRR-K69M.

Lin, J. M. C., Yen, L. Y., Yang, M. C., & Chen, C. F. (2005, June). Teaching computer programming in elementary schools: a pilot study. In *National Educational Computing Conference*.

Maltese, A. V., & Tai, R. H. (2010). Eyeballs in the fridge: sources of early interest in science. *International Journal of Science Education, 32*(5), 669–685.

Merriam, S. B. (2009). *Qualitative research and case study applications in education. Revised and expanded from "Case Study Research in Education"*. 350 Sansome St, San Francisco, CA 94104: Jossey-Bass Publishers.

Montague, M., & Applegate, B. (2001). Middle school students' perceptions, persistence, and performance in mathematical problem solving. *Learning Disability Quarterly, 23*, 215–228. http://dx.doi.org/10.2307/1511165.

National Research Council. (2008). *Taking science to school: Learning and teaching science in grades K–8*. Washington, DC: National Academy Press.

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

National Research Council. (2011). *A framework for K-12 science education: Practices, crosscutting concepts, and core ideas*. Committee on a Conceptual Framework for New K-12 Science Education Standards. Board on Science Education, Division of Behavioral and Social Sciences and Education. Washington, DC: The National Academies Press.

Norris, C., Sullivan, T., Poirot, J., & Soloway, E. (2003). No access, no use, no impact: snapshot surveys of educational technology in K# x2013; 12. *Journal of Research on Technology in Education, 36*(1), 15–27. http://dx.doi.org/10.1080/15391523.2003.10782400.

Nunez, J. C., González-Pienda, J. A., González-Pumariega, S., Roces, C., Alvarez, L., Gonzalez, P., et al. (2005). Subgroups of attributional profiles in students with learning difficulties and their relation to self-concept and academic goals. *Learning Disabilities Research & Practice, 20*(2), 86–97. http://dx.doi.org/10.1111/j.1540-5826.2005.00124.x.

Odom, S. L. (2009). The tie that binds: evidence-based practice, implementation science, and outcomes for children. *Topics in Early Childhood Special Education, 29*, 53–61. http://dx.doi.org/10.1177/0271121408329171.

Papert, S. (1991). Situating constructionism. In S. Papert, & I. Harel (Eds.), *Constructionism* (pp. 1–11). Norwood, NJ: Ablex.

Rogers, E. M. (2003). *Diffusion of innovation* (5th ed.). New York: The Free Press.

Scratch-Ed Featured Resources. (year). Retrieved from http://scratched.media.mit.edu/resources.

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

Stake, R. E. (2006). *Multiple case study analysis*. New York: Guilford Press.

Valadez, J. R., & Durán, R. P. (2007). Redefining the digital divide: beyond access to computers and the Internet. *The High School Journal, 90*(3), 31–44.

Weintrop, D., Beheshti, E., Horn, M. S., Orton, K., Jona, K., Trouille, L., et al. (2014). *Defining computational thinking for science, technology, engineering, and math*. Poster presented at the Annual Meeting of the American Educational Research Association (AERA 2014), Philadelphia, USA.

Welcome to EtoysIllinois.org!. (n.d.). EtoysIllinois. Retrieved from http://etoysillinois.org/.

Wilson, A., & Moffat, D. C. (2010). Evaluating scratch to introduce younger schoolchildren to programming. In *Proceedings of the 22nd Annual Psychology of Programming Interest Group (Universidad Carlos III de Madrid, Leganés, Spain*.

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

Wolfe, T. J. (2011, September–October). Students teaching students. *Learning & Leading with Technology, 39*(2), 32+. Retrieved from http://go.galegroup.com/ps/i.do?id=GALE%7CA268962259&v=2.1&u=uiuc_uc&it=r&p=AONE&sw=w&asid=e3b2bc28b3992394cac53a1fecb0e365.