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).