NORTHWESTERN UNIVERSITY

Learning to Teach Computer Science:

Qualitative Insights into Secondary Teachers' Pedagogical Content Knowledge

A DISSERTATION

SUBMITTED TO THE GRADUATE SCHOOL
IN PARTIAL FULFILLMENT OF THE REQUIREMENTS

for the degree

DOCTOR OF PHILOSOPHY

Field of Learning Sciences

By

Aleata Kimberly Hubbard

EVANSTON, ILLINOIS

September 2017

ABSTRACT

Learning to Teach Computer Science:

Qualitative Insights into Secondary Teachers’ Pedagogical Content Knowledge

Aleata Kimberly Hubbard

In this dissertation, I explored the pedagogical content knowledge of in-service high school educators recently assigned to teach computer science for the first time. Teachers were participating in a professional development program where they co-taught introductory computing classes with tech industry professionals. The study was motivated by three questions: (1) what knowledge of computer science content, student thinking, and instructional strategies do teachers display? (2) what teaching tasks do teachers undertake when planning and implementing their lessons? and (3) how does teaching knowledge relate to the teaching tasks assumed? I conducted a year-long, collective case study with six teachers working in the San Francisco Bay region by gathering interview, questionnaire, observation, and assessment data at monthly classroom visits. Data suggest that (a) teachers know more about student difficulties than instructional strategies specific to computer science; (b) teaching tasks differ in the opportunities they provide for developing teaching knowledge; (c) the way teaching tasks support pedagogical content knowledge varies based on one’s level of content knowledge; and (d) teaching confidence and epistemological beliefs influenced instructional decisions but sometimes conflicted with external demands. The findings offer insight into the factors of practice-based training that influence the pedagogical content knowledge development of experienced teachers new to computer science. This work can inform the design of future initiatives to effectively prepare secondary computer science teachers.

ACKNOWLEDGEMENTS

During my first year of college, I took a class called *Freshman Immigration Course* where different professors in the computer science department visited each week and gave short presentations about their research. I still remember Roger Dannenberg demoing his music visualization software and Anastasia Ailamaki showcasing different real-world database applications. The projects I learned about, the enthusiasm of the speakers, and the potential for creative expression through technology all sparked my curiosity in this thing called ‘research’. Later that year I visited my undergraduate advisor Mark Stehlik and told him I was interested in research but I did not know where to start. He pointed me towards Maxine Eskenazi, a scholar who shared my interests in language and technology, who mentored me through my first research project, my first conference presentation at CALICO, and my first internship at a university spin-off company. These experiences were extremely validating and convinced me to pursue a career in research. For nearly fourteen years I have been pursuing my interest in understanding how people learn and how to support them in that process. This journey was not without its trials. Along the way, I got lost, took some detours, and started exploring other options. But, due to the tremendous support I received these past two years, I have rekindled my passion for research, restarted my journey, and can offer this dissertation as my first major contribution to the field. I owe the completion of this dissertation and the fulfillment of a dream to the guidance, collaboration, and encouragement of so many who I would like to thank here:

To Dr. Carol D. Lee, my committee chair, thank you for believing in me, making time to help me through this process, always pushing me to think in new ways, and serving as a role model for excellence in scholarship.

To the rest of my committee, Dr. Steve Schneider, Dr. Miriam Sherin, and Dr. Uri Wilensky, thank you for sharing your knowledge and insights and helping me shape this project into better a study than I would have produced on my own.

Thank you to Dr. Steve Schneider also for introducing me to the area of teacher learning through multiple projects at WestEd, making it possible for me to complete this project while still working, and sharing your many encouraging stories about your own dissertation process.

To Dean Penelope Peterson, thank you for allowing me to rejoin the Learning Sciences department so that I could complete this project.

To Ms. King, Mr. Edwards, Ms. Jones, Ms. Robinson, Mr. Miller, and Mr. Perez (all pseudonyms), thank you for participating in this study, allowing me into your classrooms, and opening up about your experiences as computer science teachers.

To Katie D’Silva, Danielle Oberbeck, Angela Knotts, and Joseph Green, thank you for helping me collect the data for this study and braving the early morning Bay Area traffic.

Thank you to Katie D’Silva also for helping code the study data and for sharing your insightful observations.

To Dr. Jodi Davenport, thank you for being the research group I did not have, for thinking through ideas with me, and sharing your wisdom on how to get things done well and on time.

To Dr. Yvonne Kao, thank you for letting me make this case study project my own and always being there to discuss ideas.

To Joel Cheuoua thank you for sharing your content knowledge and helping me understand new approaches to computer science I encountered during this project.

To Dr. Sarah Levine, thank you for providing feedback on earlier drafts of this dissertation.

To Adi Kalderon, thank you for your Hebrew translations.

To the STEM group at WestEd, thank you all for your constant encouragement that helped me make it through many long days.

To Joel Cheuoua, Finley Hubbard, Fenesha Hubbard, and Marilyn Hearns, thank you for your emotional support during this process and making sure I never gave up.

"I owe this to the hands and hearts of others. Through their love I found my soul and God and happiness. Don't you see what it means? We live by each other and for each other. Alone we can do so little; together we can do so much." ~Hellen Keller

This material is based upon work supported by the National Science Foundation under Grant No. 1348866. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the National Science Foundation.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1. INTRODUCTION

In the past decade, a powerful educational movement has emerged to introduce all of America's youth to computer science. Many advocates argue that studying computer science is imperative to preparing students for the growing number of careers across disciplines that will require an understanding of computing (e.g., Seehorn et al., 2011). Across the country, a wealth of programs and policies have been initiated to support students learning computer science. For example, New York, San Francisco, Chicago, and Arkansas are implementing computing curricula across their school districts (Arkansas Code Annotated, 2015; Chicago Public Schools, 2014; San Francisco Unified School District, 2015; The City of New York, 2016). Numerous not-for-profit organizations such as Girls Who Code, Black Girls Code, and Intel Computer Clubhouse have created out-of-school learning experiences targeting populations underrepresented in the computing field. Advocacy groups like Code.org and Alliance for California Computing Education for Students and Schools are promoting legislation to include computer science in the core K-12 curriculum. With the announcement of former President Obama's *Computer Science for All* initiative (The White House, 2016), funding for the research and development of computing education interventions is expected to increase. A major requisite to the sustainability of these plans is a sufficient pool of qualified teachers versed in both computing content and computing pedagogy who can instruct students.

However, there is a shortage of computer science teachers at the primary and secondary level. One reason for this scarce supply is a limited number of pre-service training programs available to prepare aspiring teachers. Given the paucity of pre-service opportunities, alternative pathways for in-service educators exist through computer science teaching endorsements and accreditation (Lang, Galanos, Goode, Seehorn, & Trees, 2013). For example, educators in California can teach computer science either with certification in mathematics, business, or industrial and technology education or with supplementary authorization in computer concepts and applications (Bernier, 2012). However, manifold pathways make it possible for teachers to enter computer science classrooms without adequate preparation. The computer science education community has recognized a need to support these teachers in developing the requisite knowledge and skills reflective of our current understanding of how to effectively teach and increase participation in computing (Ericson et al., 2008).

Studies of teaching have revealed that effective instruction requires a base of multiple skills and types of knowledge. For example, researchers have shown that positive student outcomes are related to teacher domain knowledge (Darling-Hammond, 2000; Wayne & Youngs, 2003) and inclusion of pedagogical strategies relevant to students' cultural backgrounds (Gay, 2002). These knowledge areas do not exist in isolation. Possessing a high level of one type of knowledge is not sufficient for effective teaching. One of the most influential conceptualizations of the interconnectedness of different types of teacher knowledge is Shulman's (1986) idea of pedagogical content knowledge (PCK). PCK describes the interplay between a teacher's understanding of subject matter, students, and instruction and represents a type of knowledge necessary for teaching that industry professionals (e.g., software engineers) or teachers of other domains generally do not possess or need.

In-service computer science teachers come from assorted backgrounds and thus vary in the PCK and experiences they bring into their classrooms (Ericson et al., 2008). For example, a tech industry professional transitioning into teaching may possess deep content knowledge of computing but have no training on assessing learning or pacing instruction. In contrast, an experienced mathematics educator teaching a computer science course for the first time might attend to selecting materials appropriate to students, but may lack relevant examples to help students confront their misconceptions. Furthermore, the field of computing is constantly changing, requiring teachers to stay one step ahead of their students to learn new material, programming languages, and paradigms (Gal-Ezer & Stephenson, 2010). A veteran teacher who taught computer science in the 1980s when BASIC and Pascal were popular programming languages may not be versed in current programming paradigms or equity practices to support diverse learners. Depending on their pathway into computer science classrooms, teachers will require different supports to supplement the weaker areas of their PCK.

Professional development (PD) programs are one avenue through which educators can develop their teaching knowledge. Various types of activities are used in delivering professional development including workshops, courses for credit, teacher study groups, conferences, teacher networks, committees, mentoring, internships, and resource centers (Garet, Porter, Desimone, Birman, & Yoon, 2001). A plethora of professional development opportunities have been created for computer science teachers. For example, Google's CS4HS initiative offers funding for the planning and development of K-12 computer science professional development programs across the globe and more than 400 programs have been funded ("Google CS4HS," n.d.). While teachers can learn about instructional strategies and subject matter at professional development programs, classroom experiences are needed to influence change in teaching practice (D. L. Ball & Cohen, 1999; Magnusson, Krajcik, & Borko, 1999; van Driel, Verloop, & de Vos, 1998).

In-service teachers, whether trained in other disciplines or possessing professional experience in the tech industry, are making computer science possible in American classrooms. These educators come from various backgrounds and differ in their readiness to adequately teach computing. Professional development opportunities help these teachers fill gaps in their teaching knowledge. Understanding how in-service teachers incorporate knowledge gained from professional development opportunities into their teaching practice can help us understand how best to support them.

1.1 Statement of the Problem

Research points to the value of linking professional development to authentic practice (D. L. Ball & Cohen, 1999; Garet et al., 2001). Educational partnerships with practitioners of a given discipline (e.g., a scientist working at a research institute, a software developer working in industry) are one form of professional development that allows teachers to connect their learning directly to their daily work. These collaborations tend to involve joint or distributed effort around team teaching, curriculum development, and learning opportunities for teachers (Grobe, Curnan, & Melchior, 1990). In science education, for example, partnerships like Columbia University's Summer Research Program for middle and high school teachers in New York City have benefitted teachers by improving their content knowledge, teaching skills, and self-efficacy (Powell-Moman & Brown-Schild, 2011; Silverstein, Dubner, Miller, Glied, & Loike, 2009). However, when situated within classrooms, the learning opportunities that these partnerships offer will depend on the particular contexts within which they occur and on the participating students, teachers, and external partners. For example, Nelson's (2005) study of the PIESS program demonstrated how pairs of K-12 teachers and graduate students either negotiated, exchanged, or rejected each other's expertise regarding science content knowledge, disciplinary practices, and education during their co-teaching. Each stance towards knowledge sharing had differing levels of impact on each partner's professional learning; some participants had transformative experiences while others resisted change. Although not discussed thoroughly in her study, Nelson also noted that "gender, personality, power, and the community, school, and classroom cultures were evident as factors affecting coparticipation and dialogue" (Nelson, 2005, p. 384).

Given the diversity of backgrounds of computer science teachers, they likely vary in the content knowledge they bring to the classroom and the beliefs they hold about teaching and learning in general and about computer science instruction specifically. Furthermore, the push to broaden participation in the field means the computer science student population will be more diverse than before (Cuny, 2015) and learners will bring differing experiences and interests into the classroom. Lastly, the instructional roles undertaken in classroom partnerships will likely depend on individual traits like teacher-volunteer compatibility and each participants' level of content knowledge (Scruggs, Mastropieri, & McDuffie, 2007). All the variety confronting teachers entering computer science will impact what happens in classrooms and, therefore, what instructional moments are available to help teachers improve their practice. The specific problem addressed in this dissertation is the development of experienced mathematics teachers' computer science PCK under highly idiosyncratic circumstances within on-the-job professional development.

1.2 Overview of the Study

This dissertation reports on findings from a collective case study conducted during the 2015-2016 school year with six teachers located in the San Francisco Bay Area. The goal of this study was to examine the PCK of experienced secondary mathematics teachers transitioning into computer science while collaborating with a content expert to deliver introductory computing courses. Each teacher was visited approximately once per month in their computer science courses and once in a class of their main discipline. During each visit, teachers were studied as they planned, enacted, and reflected on their lessons. Data collected included questionnaires completed before and after each visit, classroom observations, interviews, and tasks. This study is a component of *Developing Computer Science Pedagogical Content Knowledge Through On-the-Job Learning* (CSPCK), a five-year, NSF-funded project conducted by WestEd to study on-the-job training for computer science high-school teachers. CSPCK has two goals: (1) to identify the components of an on-the-job training model that effectively prepares in-service high school educators to teach CS and (2) to develop a theoretical framework and a suite of assessment items to measure PCK for CS. CSPCK focuses on teachers participating in TEALS (Technology Education and Literacy in Schools), a program that recruits and trains volunteers from the tech industry and places them into classrooms to co-teach with high school teachers. At the beginning of a TEALS partnership, volunteers lead lessons allowing teachers to focus on learning course content. As teachers gain experience and confidence with course materials, they take on more instructional responsibility. At the end of the partnership, high school teachers lead courses independently. The TEALS context was used to examine participants' computer science PCK and the relationship between their PCK and their teaching tasks.

1.3 Significance of the Study

The recent groundswell of interest in computer science education has created a pressing need for computing teachers at the secondary level. We don't just need teachers, we need qualified teachers. This study investigates how experienced educators from other disciplines develop the knowledge base needed for effective computer science instruction while working in authentic settings with the support of content experts. With a focus on in-service teachers new to computing, this dissertation differs from prior work on computer science PCK focused on either pre-service teachers or experienced educators. Exploring PCK development in a group of educators who have developed pedagogical knowledge but varying levels of subject matter knowledge is important because it reflects a reality for many computer science teachers. The domain is dynamic and computer science teachers will need to constantly update their content knowledge base as new tools and technologies emerge. This study takes steps towards helping us understand how PCK changes as teachers learn to teach new content.

The results of this study may also be useful to stakeholders invested in strengthening the computer science teaching force. Although the impetus to increase access to computer science in American schools exists, the field is still figuring how best to achieve this goal. Teacher educators can use the results of this dissertation to identify and adapt successful components of on-the-job teacher training into their professional development programs. Administrators who are implementing computer science programs in their states can benefit from understanding the growth trajectory of teachers new to computer science which will help in establishing realistic goals for transitioning teachers. Furthermore, this dissertation provides insight into the factors that motivate transitioning computer science teachers. Along with training educators in the requisite pedagogical and content knowledge, we also need to attend to how the initial experiences of teachers new to computer science influence retention. These efforts will be in vain if teachers leave the classroom because they are overwhelmed or lured towards other careers.

1.4 Limitations

This dissertation focuses on high school teachers working in the San Francisco Bay Area and unique features of this context may influence study results. The region contains Silicon Valley, home to many leaders of the technology sector (e.g., Google, Facebook, and Apple), and some of the wealthiest counties in America. Close proximity to this tech hub might provide teachers and students with more opportunities to interact with computing than their peers in other parts of the country, which, in turn, can influence the classroom experiences that support PCK development. While the goal of this work is not to generalize to all teaching contexts, I did seek variety in school and community contexts when recruiting participants. To capture a broad perspective on the different ways in which teachers develop PCK, I recruited participants from three distinct localities within the San Francisco Bay Area.

A second limitation concerns the relationship between mathematics, computer science, and subject matter knowledge. All the participating teachers were certified in mathematics. As mentioned above, certification in mathematics allows one to teach computer science in the state of California. Mathematical reasoning and techniques are integral to computer science (Joint Task Force on Computing Curricula & Society, 2013), and so mathematics teachers may have an easier time learning computer science content than teachers trained in other disciplines. This study does not provide insight into the PCK of teachers entering computer science from other subjects.

A third limitation of this study relates to the time frame of data collection. Teachers were visited approximately once per month during the school year, and, as such, observations were often conducted on one day of a lesson spanning multiple sessions. Thus, the evidence gathered in this study does not represent a complete picture of teachers' PCK across a topic. For example, a teacher might spend the first day of a lesson providing direct instruction on looping constructs and spend the second day reviewing those constructs. A visit on the first day of this lesson might reveal more information about the teacher's instructional practices while a visit on the second day might reveal more information about the teacher's knowledge of assessment. Monthly visits were chosen over lesson visits in order to capture evidence of teachers' PCK as they gained more experience in the classroom across the school year.

Lastly, the in-service professional development model observed in this study assumes that teachers will integrate their incipient content knowledge with their existing pedagogical knowledge to develop PCK for computer science. By focusing only on experiences in computer science classrooms, I would have no information on how other teaching experiences influenced participants’ PCK development in computer science. To partly address this methodological deficiency, teachers were observed once during a lesson in their main subject area. While a single visit cannot capture the entirety of a teacher's knowledge and practices, it does provide an initial point of contrast to the PCK observed in their computing classes.

1.5 Definition of Terms

Some of the terms central to this dissertation carry multiple meanings. Below I provide the definitions I will use for computer science, pedagogical content knowledge, transitioning computer science teacher, and volunteer content expert.

Computer Science (CS). Computer science is "the study of computers and algorithmic processes, including their hardware and software designs, their applications, and their impact on society" (Tucker et al., 2006, p. 2). It is a subfield within the field of computing along with computer engineering, information systems, information technology, and software engineering. The field is distinguished from other computing disciplines in its focus on the design and implementation of software, devising new ways to use computers, and the development of effective solutions for computing problems (The Joint Task Force for Computing Curricula 2005, 2005). Although CS is often conflated with other computing domains or computer-related activities with the K-12 landscape (CSTA, 2014), the aforementioned definition will be adopted in this dissertation.

Pedagogical Content Knowledge (PCK). Shulman's (1986) PCK framework describes the interplay between a teacher's understanding of subject matter, students, and instruction. It consists of "the ways of representing and formulating the subject that make it comprehensible to others…[it] also includes an understanding of what makes the learning of specific topics easy or difficult" (Shulman, 1986, p. 9). Many educational researchers have used the PCK construct to describe and investigate the knowledge teachers draw upon in their practice. After years of work in fields such as mathematics education and science education, scholars agree that PCK is essential to achieve the aims of teaching but they disagree on its specific components (Abell, 2008; Depaepe, Verschaffel, & Kelchtermans, 2013). In this dissertation, I focus on teachers' knowledge of computer science for instruction and teachers' knowledge of students. These PCK components relate to:

> teachers' abilities to: (1) organize instruction around an accurate, precise, and coherent set of interrelated conceptual learning goals; (2) anticipate, elicit, interpret, and address particular challenges the content poses for their students; and (3) sequence and represent that content during instruction in a way that advances their students' understanding. (Daehler, Heller, & Wong, 2015, p. 45)

Transitioning CS Teacher. The teachers who participated in this dissertation were experienced, in-service high school teachers certified in mathematics who are beginning to teach the CS courses offered through TEALS. To distinguish these participants from teachers who have received pre-service training or certification in CS, I will refer to them as transitioning CS teachers.

Volunteer Content Expert. The TEALS program recruits volunteers from the tech industry to support teachers in delivering CS courses. TEALS volunteers are not certified teachers, although some have experience teaching in informal learning programs or during their undergraduate studies. These volunteers contribute what Ball, Thames, and Phelps (2008) call common content knowledge or "the knowledge teachers need in order to be able to *do the work* that they are assigning their students" (D. L. Ball et al., 2008, p. 6). I will refer to these participants as volunteer content experts.

1.6 Outline

The remainder of this dissertation is divided into six chapters. In chapter two, I present a literature review focused on conceptualizations of CS PCK, methods of studying CS PCK, and factors influencing CS PCK development. Given the limited amount of literature on CS PCK, I supplement the review with a summary of PCK research in other domains. Given that PCK is domain specific, I also discuss the nature of CS, K-12 CS education, and challenges mathematics teachers might face transitioning into this discipline. I conclude chapter two with the presentation of the conceptual frameworks informing my dissertation and the specific research questions I addressed. In chapter three, I present the methodology undergirding my research, details of the TEALS professional development model, and the six teachers who participated in the case study. In the second half of the chapter, I detail my data sources, my procedures for data collection, reduction, and analysis, and my methods for establishing the trustworthiness of the data. At the end of the chapter I briefly describe my role as a researcher and the prior experiences that influenced my interpretation of the case study data. The study results are separated into three chapters. In chapter four, I focus on teachers' knowledge of CS content and student thinking. In chapter five, I explore the instructional responsibilities teachers undertook when planning and implementing their lessons and how those responsibilities related to their teaching knowledge. In chapter six, I investigate how confidence and epistemological beliefs influenced teachers' instructional responsibilities. While not an initial focus of this dissertation, teacher comments about their beliefs and confidence were prominent during interviews and encouraged me to devote more attention to the way these factors influenced PCK development. In the final chapter, I discuss the case study results, limitations, implications, and suggestions for future work.

CHAPTER 2. LITERATURE REVIEW

2.1 Introduction

For some years now, the CS education community has focused on training in-service teachers to satisfy the demand for more computing courses in K-12 schools. Although the number of professional learning opportunities for CS educators has increased, many of these programs do not last beyond a week and fail to include a focus on pedagogical content knowledge specific to computing (Menekse, 2015). Outside of this training, the classroom becomes a prominent milieu for CS teacher learning where unpredictable and idiosyncratic events provide opportunities to improve upon practice. How, then, can we better support the professional development of computing educators both in formal training programs and in their everyday teaching?

Specifying pedagogical content knowledge for CS and how it develops over time can help identify focal topics for professional development programs. Also, identifying CS teaching tasks can provide insight into how practice supports PCK development. Over many decades, efforts made towards understanding these topics in other domains have resulted in more effective teacher training (e.g., Desimone, Porter, Garet, Yoon, & Birman, 2002; Heller, Daehler, Wong, Shinohara, & Miratrix, 2012). However, the CS education community is only beginning to develop a base of similar work related to PCK and teaching. In the first part of this literature review, I synthesize the extant research on CS PCK and situate it within the extensive research base of PCK in other disciplines. Given that PCK is content specific, I also provide an overview of the discipline of CS as it relates to learning and teaching and contrast it against mathematics, a domain in which many transitioning CS teachers were trained. In the final section of this chapter, I introduce (a) a conceptual framework of CS PCK development, (b) a conceptual framework of CS PCK for transitioning teachers, and (c) the resultant research questions driving this dissertation.

2.2 Synthesis of CS PCK Literature

I focus on the following questions in this section of the literature review: (1) How is PCK conceptualized in CS educational research? (2) How is PCK investigated by CS education researchers? (3) What influences the development of PCK in CS teachers? I begin this synthesis review by describing my method for identifying relevant articles. Then, I summarize the literature on teacher knowledge and teacher learning in fields outside of CS. Lastly, I summarize literature on CS teacher knowledge, responding to the aforementioned questions.

2.2.1. Methods

Computer science education research (CSER) is a relatively young field influenced by the methods, research designs, and philosophical worldviews of various fields like psychology, computer science, and education (Fincher & Petre, 2004a). These communities vary in expectations for the dissemination of scholarly work and the value placed on different forms of dissemination (Joy, Sinclair, Sun, Sitthiworachart, & López-González, 2008). I conducted a review of the CSER literature by searching for both academic journals and conference proceedings in several databases that index scholarly work from the education, computer science, and engineering communities, namely, Education Research Complete, Psychology & Behavioral Sciences Collection, SocINDEX with Full Text, American Psychological Association PsycARTICLES, Education Resources Information Center (ERIC), ACM Digital Library, and IEEE Xplore Digital Library. Publication titles and abstracts were searched for the terms *computer science teachers* or *computer* in conjunction with *professional development, pedagogical content knowledge,* and *teacher preparation.* The initial search returned 314 results. Only papers written in English and focused on secondary teachers were included. Opinion pieces were excluded. Conference papers were excluded if they described a poster or provided only an abstract. Papers published prior to 2011 relating to the development of CS teaching knowledge were excluded because Armoni (2011) conducted a review of this literature spanning from 1975 to 2010. Papers that only provided descriptions of professional development programs were excluded because Meneske (2015) provides a review of CS professional development programs spanning from 2004 to 2014. Three conference papers could not be accessed and are excluded from this review. Inclusion and exclusion criteria winnowed the list to 25 papers. Each article was summarized and then categorized along the following dimensions: (a) literature review questions addressed, (b) country of research, (c) school level focus, and (d) a focus on pre-service or in-service teachers. Where appropriate, I also categorized articles based on the following: (a) category of teaching knowledge, (b) definition of teaching knowledge, (c) theoretical framework, (d) research methods, (e) study time span, and (f) number of participants. Lastly, I conducted a cross-case analysis of the papers to identify differences and similarities based on these dimensions.

I also searched for articles synthesizing research on teaching knowledge outside of CSER using the following databases: Education Research Complete, Psychology & Behavioral Sciences Collection, SocINDEX with Full Text, American Psychological Association PsycARTICLES, and ERIC. Publication titles and abstracts were searched for the term *review* in conjunction with *PCK, pedagogical content knowledge, knowledge of teaching, teaching* *knowledge,* and *teacher learning.* The initial search returned 297 results. Only peer-reviewed articles or book chapters written in English, focused on K-12 teaching, and published after 1999 were included. Articles also needed to focus on broad disciplines (e.g., mathematics) and not subdomains (e.g., geometry). Reviews were excluded if they focused on technological pedagogical content knowledge (TPACK) given my focus on teaching knowledge of specific subject areas. Reviews focused only on pre-service teachers were excluded given my focus on in-service teachers. Inclusion and exclusion criteria winnowed the resultant list to 17 papers. I conducted a narrative literature review of these articles to summarize how scholars in domains outside of CS conceptualize, investigate, and support the development of teacher knowledge.

Searching through online databases will only reveal a percentage of articles relevant to a literature review, so other techniques such as reference branching and expert review are recommended to help researchers identify additional articles of interest (Randolph, 2009). I identified an additional 25 articles to include in this review through reference branching and a review of journals focused on CS education research and teacher education research. In total, 37 CSER papers and 30 general papers are included in this review.

2.2.2. Findings

Research on Teacher Knowledge and Learning. In the 1980s, scholars began to turn their attention towards specialized teaching knowledge that went beyond general pedagogy and an understanding of content (Matthews, 2013). For example, the Cognitively Guided Instruction (CGI) group studied first grade teachers’ knowledge of teaching addition and subtraction by assessing their ability to distinguish problem types, judge the relative difficulty of problems, and anticipate student problem solving strategies (Carpenter, Fennema, Peterson, & Carey, 1988).

Shulman (1986) introduced PCK as a theoretical construct to analyze this specialized teaching knowledge which he described as knowledge of:

> the most regularly taught topics in one's subject area, the most useful forms of representation of those ideas, the most powerful analogies, illustrations, examples, explanations, and demonstrations…[it] also includes an understanding of what makes the learning of specific topics easy or difficulty. (p. 9)

This framework has been extremely influential in the study of teacher learning particularly in mathematics (Depaepe et al., 2013) and science (Schneider & Plasman, 2011).

Over the years, researchers have attempted to further specify PCK into subcomponents which have included: student conceptions and difficulties, instructional strategies and representations, types of tasks and their cognitive demands, assessment, educational ends, curriculum and media, context, content, and general pedagogy (Depaepe et al., 2013; S. Park & Oliver, 2008). Some of the more popular PCK conceptualizations derived from Shulman's work are Teacher Education and Development Study: Learning to Teach Mathematics (TEDS-M); Mathematical Knowledge for Teaching (MKT); Professional Competence of Teachers Cognitively Activating (COACTIV); and Knowledge for Algebra Teaching (KAT) (Blömeke & Delaney, 2012; Matthews, 2013). The many different conceptualizations of the PCK framework have not converged on a clear definition of the construct and many subdomains of PCK are difficult to distinguish (Hashweh, 2005; Matthews, 2013). It is important to note that PCK is subject specific and so it relates to content knowledge. Researchers have found the two types of knowledge to be strongly correlated (Blömeke & Delaney, 2012). Without strong content knowledge, teachers struggle to notice and understand student thinking and to participate in useful discussions with their colleagues (Goldsmith, Doerr, & Lewis, 2014). While researchers have progressed in specifying PCK components, the development of those components is less well understood (Abell, 2008).

Research on teacher learning in general can provide insight into how PCK develops. Learning to teach is a complex process mediated by both individual factors (e.g., beliefs, orientation to learning) and external factors (e.g., school culture, learning opportunities) (Avalos, 2011; Opfer & Pedder, 2011). Teacher learning is an incremental, career-long process that initially focuses on students and their ideas, then shifts towards a focus on teaching, and lastly concentrates on developing a repertoire of teaching practices (Schneider & Plasman, 2011). Some teachers direct their learning towards immediate use of new strategies in their practice while others focus more on understanding the theory behind teaching practice and why instructional strategies are effective (Vermunt & Endedijk, 2011). Teacher learning ideally leads to changes in the subject matter content of lessons, classroom discourse, and encouraging students to take responsibility for their own learning, which typically take at least a year to appear (Goldsmith et al., 2014; Postholm, 2012). Learning can also be reflected through changes in knowledge and beliefs, intentions for practice, emotions, task performance, awareness and understanding, personal development, teamwork, role performance, academic knowledge and skills, decision making and problem solving, and judgement (Vermunt & Endedijk, 2011).

A variety of activities can support teacher learning including professional conversations, collaboration with colleagues, and critical reflection (Dogan, Pringle, & Mesa, 2016; Goldsmith et al., 2014; Postholm, 2012; Randi & Zeichner, 2004). Professional development opportunities are more effective when they are coherent, provide a strong focus on content, extend over time, provide mentoring, involve teachers who work in comparable contexts, and model practices teachers can use in their classrooms (Dunst, Bruder, & Hamby, 2015; Postholm, 2012; Whitworth & Chiu, 2015). The differing backgrounds, amount of teaching experience, and prior knowledge participants bring to learning opportunities make it challenging for professional development providers to create effective learning opportunities (Wilson, Rozelle, & Mikeska, 2011). Some scholars argue that experiential learning opportunities should be favored over traditional in-service learning opportunities because they allow teachers to engage with materials of practice, can integrate into teachers' daily work, and allow teachers to generate knowledge by reflecting on their own practice (Opfer & Pedder, 2011; Randi & Zeichner, 2004).

In fact, some teacher education researchers are beginning to focus less on the domains of teacher knowledge and more on describing the core practices of teaching that enable educators to apply their knowledge in classrooms (Forzani, 2014; McDonald, Kazemi, & Kavanagh, 2013). Core practices, also called high-leverage practices, occur frequently in teaching, can be enacted in various types of classrooms, are acquirable, authentic, support learning about students and teaching, research-based, and might improve student achievement (Grossman, Hammerness, & McDonald, 2009). However, scholars are still undecided about which core practices best support student learning or what teaching knowledge is needed for effectively using core practices (D. L. Ball & Forzani, 2009).

The research base also indicates that beliefs and efficacy play an important role in acquiring teaching knowledge. Beliefs are ideas teachers hold about the nature of teaching and student learning and how the work of teaching should be enacted. These beliefs can depend on the particular school or students a teacher works with as well as the teacher's career stage and prior experiences (Luft & Roehrig, 2007; Postholm, 2012). Certain epistemological stances (e.g., constructivist views that see learners as active agents in constructing their own knowledge and see the aim of science as developing theories about the world) might encourage richer PCK development than other stances (e.g., views that support a transmission view of knowledge acquisition and see the aim of science as collecting facts about the world) (Hashweh, 1996, 2013). Misalignment between a teacher's beliefs and the theory undergirding professional development programs can lead teachers to reject their training (Goldsmith et al., 2014). Efficacy relates to teacher beliefs about their ability to accomplish professional duties. While a lack of efficacy can prevent teachers from learning, it can increase through peer collaboration (Goldsmith et al., 2014). Self-efficacy is also related to content knowledge and tends to be lower in teachers who have taken fewer content courses or who are teaching outside of their main subject (Mizzi, 2013; Ross, Cousins, Gadalla, & Hannay, 1999; Swackhamer, Koellner, Basile, & Kimbrough, 2009).

Another area of focus for this literature review was the methods used to study PCK. A variety of methods were mentioned in the review articles including tests, questionnaires, interviews, lesson observations, meeting observations, document analysis, conversation analysis, and concept mapping. Within mathematics education research, large-scale studies of PCK tend to use tests for study instruments while small-scale studies tend to use qualitative methods (Depaepe et al., 2013). Goldsmith, Doeer, and Lewis (2014) cautioned against the overreliance on self-reported data of instructional practices because there may be differences between the ways teachers and researchers perceive practices. Lastly, through a review of PCK studies conducted in multiple countries, Blömeke and Delaney (2012) shed light on the influence of language and culture on the development of PCK frameworks, assessments of teaching knowledge, and the results obtained when using these tools in different locales. First, they argued that since teaching is a cultural activity that varies from country to country, PCK frameworks and assessments may also be culture specific and bias towards the practices and views of teaching dominant in the countries within which they were developed. As an example, Blömeke and Delaney highlighted a study of Indonesian teachers' performance on the US-based MKT which showed lower performance on geometry items due in part to differences in how shapes are classified in each country. Second, there was empirical evidence that, in some countries, teachers whose mother tongue matched the language of their training program performed better on measures of content knowledge and PCK than teachers who had a different mother tongue.

Work conducted over the past four decades has advanced our understanding of the types of knowledge that are unique to teaching and researchers are still working towards a richer understanding of how teacher knowledge develops over career stages. However, "comprehension alone is not sufficient. The usefulness of such knowledge lies in its value for judgment and action" (Shulman, 1987, p. 14). So, recently there has been renewed interest in understanding how PCK relates to the practices of teaching. Aside from what teachers know and how they use that knowledge, studies within this domain have also emphasized the role that beliefs and efficacy play in teacher learning. Lastly, while a variety of methods exist to explore PCK, researchers need to attend to possible limitations of self-reported data and cultural biases embedded in their instruments.

Prior CSER Reviews. This literature review serves to summarize the knowledge base of CS teacher learning produced by the CSER community. Earlier reviews from this field can help to contextualize the PCK studies described in the following sections. Here I discuss two surveys that provide landscapes of the field, three methodological reviews, and two surveys related to CS teacher preparation.

Landscape surveys. In 2001, Holmboe, McIver, and George (2001) conducted a review of CSER from 1976 to 2000. They identified six types of projects prominent in CSER: ideas for new instructional methods, experience stories from practitioners, discussion of theory, computer aided systems, expert-novice studies, and empirical studies focused on how learners understand programming. One deficiency they identified in CSER articles was limited references to pedagogical theory or prior related work. They also noted a preponderance of papers that provided reflections from computer scientists on their teaching and suggested this was partly explained by a lack of researchers in the community dedicated specifically to CSER. The authors encouraged more empirical research that draws on existing educational theories and borrows methods from other disciplines. Lastly, the authors argued that CSER was still forming into its own discipline and that it was "important to keep some common ground in order to achieve a feeling of identity and belonging within the field. The common ground outlined in the present paper is…the facilitation of pedagogical content knowledge for practitioners" (Holmboe et al., 2001, p. 6).

A few years later, Fincher and Petre (2004a) edited *Computer Science Education*, a fourteen chapter book providing a landscape of the field and guidance on conducting CSER. They portrayed CSER as a nascent field still developing its own knowledge base and struggling to form an identity from amongst the diverse backgrounds scholars bring from other fields like education, psychology, and computer science. Like Holmboe et al. (2001), Fincher and Petre noted that most CSER papers were high in evidence but low in theory. They also categorized existing CSER efforts, but with a focus on motivations behind the work. Their classification scheme identified three CSER areas which overlap with four categories presented in Holmboe et al. (2001): student understanding; animations of algorithms, visualizations of machine processes, and simulation systems; and teaching methods. They highlighted seven additional areas motivating CSER including: assessment, educational technology, the transfer of professional practice into the classroom, the incorporation of new development and new technologies, transferring to remote teaching, student recruitment and retention, and the construction of the discipline. Although CSER stems from other disciplines, the authors proposed that it can be distinguished as its own field by focusing future research on questions uniquely relevant to computer science education such as "does student understanding of programming concepts differ based on the first programming language learned?"

These landscape surveys present CSER as a budding research field influenced by the backgrounds of a heterogeneous group of researchers coming from different disciplines where education research may not be their primary or sole focus. A lot of attention has been given towards students, tools for learning, and teaching methods, but not much research has focused on CS teachers as learners. In addition to strengthening the rigor of CSER, the field needs to attend more to building on theories of learning and pedagogy and to focusing on the aspects of education research that are unique to the discipline of CS.

Methodological reviews. Carbone & Kaasbøll (1998) conducted a review of 31 articles published in the March 1996 issue of the SIGCSE Bulletin and in the Communications of the ACM between 1992 and 1997. The particular SIGCSE Bulletin reviewed by the authors contained proceedings from the twenty-seventh SIGCSE symposium. The goal of their review was to identify methods used in evaluating novel strategies for teaching difficult CS topics. Six evaluation methods were identified: student questionnaires, student exam scores, controlled experiments, phenomenographic interviews, teacher descriptions of their own teaching, and mixed methods. Most articles (n = 23) focused on teacher descriptions, which the authors attributed to limited time for teachers to systematically evaluate their teaching while also delivering their courses. While the articles covered a range of computing topics difficult for students, algorithm analysis and object-oriented programming were most common. The authors recommended instructors improve their teaching evaluation by employing *low-road* solutions that require few resources (e.g., peer observation, class interaction diagrams, tutor feedback), multiple data sources, and iterative development.

Nearly a decade later, Berglund, Daniels, and Pears (2006) conducted a methodological review of qualitative CSER projects from 1992 to 2005. The authors focused on qualitative research to increase awareness of various methods that could be used by the CSER community, which had largely relied on quantitative approaches. The authors did not discuss the procedure used to identify articles, but they described 22 *pedagogically anchored qualitative research* studies that focused both on what students learn and how learning occurs. The studies appeared to focus exclusively on students as learners and not on teachers as learners. Studies were grouped into one of five research traditions: Vygotskian (n=4), phenomenography (n=5), constructivism (n=6), critical inquiry (n=4), and multi-faceted approaches (n=3). The authors noted how few qualitative CSER projects existed and encouraged more work in this area.

Randolph, Julnes, Sutinen, and Lehman (2008) used a content analysis approach to review the methodological properties of 352 articles published between 2000 and 2005 in eight major CSER publications. They found most articles came from conference proceedings, were written by first authors affiliated with American or Canadian institutions, focused on ways to organize computing courses, and included tertiary students in first-year courses as participants. In articles that included participants, experimental methods were most commonly used and the one-group, posttest-only design was the most commonly used research design. Questionnaires were used most frequently to measure outcomes of interest; attitudes and achievement were the most frequently measured outcomes. The authors acknowledged that given their backgrounds in quantitative educational research, their review focused more deeply on articles employing quantitative methods and less on qualitative studies. They cautioned researchers against relying solely on self-reported participant measures and they encouraged authors to include details about the reliability and validity of their study measures. The authors concluded by saying the CSER community has generated a wealth of research hypotheses that are based on anecdotal evidence and poorly designed studies which may lead the field to develop a weak knowledge base that is not empirically verified.

The resounding takeaway from these three reviews is that the CSER community needs to increase the methodological rigor of its studies. While experience reports from teachers and questionnaires collected from students can help us begin to understand CS learning and teaching, we should be cautious of relying only on self-report data to develop the CSER knowledge base. Lastly, by relying mostly on quantitative methods, the CSER community is missing out on the benefits of qualitative methods which include strategies for exploring new and understudied phenomena.

Teacher preparation reviews. The remaining articles discussed in this section focus specifically on teacher preparation, a critical component to improving CS education. Almost concurrent with the arrival of CS in high schools came concern about the shortage and unpreparedness of teachers. Unfortunately, problems with teacher availability and preparedness persist. Consider these nearly identical quotes published 29 years apart:

> The shortage of teachers in computer science stem from two sources. First, there is a definite lack of adequate teacher training programs…A second source of the teacher shortage problem is financial in nature. The relatively low pay (compared to that of industrial computer scientists) of secondary school teachers make it very difficult to retain those competent in computer science. (Poirot, 1979, p. 101)

> At present, all evidence points to a crisis in computer science education at the high school level…factors that contribute to this situation, [include]: a shortage of professional development opportunities that would allow teachers to develop and keep their technical and pedagogical skills current [and] the inability of school districts to attract or maintain highly qualified teachers in the face of salary and benefit competition from industry. (Ericson et al., 2008, p. 19)

The next two articles I summarize are reviews of work related to CS teacher preparation, one at the pre-service level and one at the in-service level.

Armoni (2011) reviewed pre-service CS teacher preparation dating back to the 1970s. This review included a discussion of the nature of CS and its connection to mathematics and science, pre-service teacher knowledge and development, the effects of teacher preparation programs, and publications on CS teacher education. Armoni noted that most of the work in this area was not empirical, but rather descriptive of training programs provided by CS teacher educators. Prior to the 1990s, CS was often confused with computer-related activities (e.g., computer applications or computer literacy) and many teachers were expected to assume duties as computer resource personnel and lab directors. Teacher training focused mainly on content knowledge, specifically programming. Due to high demand for CS courses in high schools, training was frequently targeted to in-service teachers from other disciplines who were assigned to CS courses. While some training programs created during this era included methods courses, there was a lack of material available to help guide the creation of these courses. Beginning in the 1990s, discussions of CS teacher education began to include Shulman’s concept of pedagogical content knowledge, but it was not sufficiently addressed in preparation programs. Teacher educators also began describing the body of knowledge CS teachers needed. A few empirical articles focused on the impact of teacher training, instructional strategies of effective teachers, and teachers’ use of an animation system as a pedagogical tool. Armoni concluded her review with several suggestions for pre-service CS training programs (see Table 2.1).

Table 2.1

Armoni's (2011) Recommendations for Pre-service CS Training

- Training programs should focus on all types of teacher knowledge, particularly content knowledge, pedagogical knowledge, PCK, and curricular knowledge
- Content knowledge should be comparable to the core of a CS undergraduate curriculum, emphasizing concepts more than specific technologies
- PCK should be included with examples; programs should revise PCK elements as new research comes out
- Training programs should encourage a student-centered, constructivist view of education that treats the teacher as a mediator and not a knowledge transmitter
- Given the changing nature of CS, teachers should be prepared to undertake in-service PD, whether organized or independently
- Training programs should include field experiences that relate to other elements of the training program
- Training programs should include a methods course that bridges different types of teaching knowledge with use of concrete examples and connections to practice
- The nature of CS should be included in training so teachers can represent the discipline accurately to colleagues and students

More recently, Menekse (2015) reviewed 21 articles on CS professional development for in-service teachers in the U.S. published between 2004 and 2014. By focusing on reports from journals and conference proceedings, the author acknowledged that the review excluded information about professional development programs offered by non-academic institutions. The 21 programs Menekse surveyed were traditional, summer workshops with most offering one-time encounters that lasted less than a week; no programs offered other forms of professional development like coaching. Most programs received funding from either NSF or Google and focused on training high school teachers. Many programs focused on incorporating computational thinking into courses, increasing participants' content knowledge, or broadening participation in CS courses. The concepts commonly covered in the professional development programs were algorithmic thinking, variables, conditions, loops, abstraction, and decomposition; only eight programs focused on pedagogical content knowledge. Professional development providers evaluated their programs primarily by surveying participants about their interests and opinions of the success of the workshops. Menekse also compared the 21 programs against the following six characteristics of effective professional development programs drawn from the broader teacher education literature: 50 or more contact hours, support for classroom implementation, focus on active learning methods, focus on PCK, collaboration with local district or school, and evidence of student learning resulting from the PD. More than half of the programs provided support for classroom implementation and a focus on active learning methods. However, fewer programs included the other four characteristics of effective programs and only one program incorporated all six characteristics.

These two reviews on the CS teacher preparation literature show that while CS teacher learning opportunities have evolved over the past few decades, there are several areas where these opportunities can be improved to better equip teachers with the necessary disciplinary and pedagogical knowledge needed in CS classrooms. Similar to the landscape surveys and the methodological reviews, research on teacher learning would benefit from more methodological rigor and a greater focus on PCK specific to CS.

Summary. The seven review articles described above provide a portrait of the CSER landscape and teacher training opportunities from the 1970s through today. These articles suggest that although CSER is a relatively young area, there is a significant body of work describing student learning and teaching in the field. The authors highlighted how much of the existing CSER work is anecdotal and that the field requires more empirical studies to strengthen its knowledge base. The articles also drew attention to the limited focus on pedagogical content knowledge and qualitative methods within CSER. In the following sections, I turn my attention towards CSER articles related to CS PCK, methods for investigating CS PCK, and factors influencing the development of this knowledge.

Conceptualizations of CS PCK. Seven articles within the dataset addressed the conceptualization of CS PCK. Four of these articles reported on the same line of work produced by the Competencies for Teaching Computer Science (KUI) research group, which I will summarize together below. Most of the articles were published between 2004 and 2015; one article was published in 1987. Researchers worked with educational materials and participants in Germany, Israel, U.K., and U.S. Unlike the majority of CSER work summarized in the reviews described in the previous section, all articles summarized here discussed the theories of learning influencing their work. While all articles mentioned Shulman's PCK framework and its derivatives, several authors drew upon other theories related to teacher knowledge, pedagogical reasoning, epistemological beliefs, and motivation. The conceptualizations presented in these articles can be distinguished by the research approaches used to create them, which were either inductive or abductive.

Baxter (1987) and Lapidot (2005) each used more inductive approaches in their dissertation projects to arrive at conceptualizations of CS PCK and PCK development, respectively. Baxter, a student of Shulman's, conducted a case study of two experienced secondary teachers in the U.S. to explore their content knowledge and PCK related to programming in general and to the topics of loops and sorting. Both participants identified as males and had over 19 years of high school teaching experience and at least three years of experience teaching programming. Their main teaching assignments were programming courses focused on the BASIC language. Each teacher participated in three interviews, two structured tasks (i.e., critique of a BASIC program, a word association task), and four weeks of classroom observations spread across eight weeks. Through a cross-case analysis of this data, Baxter arrived at four conclusions about PCK that extend beyond computer programming. First, content knowledge and PCK may be more integrated in expert teachers than in beginning teachers, with expert educators using teaching as an organizer of their content knowledge. This idea resembles Ball, Thames, and Phelps' (2008) MKT model of mathematics PCK which describes different categories of content knowledge related to teaching. Second, cognitive styles (i.e., preferred, routine ways of acquiring information) may influence how teachers attend to, store, and use their content knowledge. Baxter cited Witkin and Goodenough's description of field independence-dependence as an example: "field independence [w]as the tendency to correctly determine the upright in a deliberately misleading context; field dependency was the tendency to locate the upright incorrectly" (Baxter, 1987, p. 154). Baxter noted, however, that prior research on cognitive styles was problematic because the construct was underconceptualized and measures of cognitive style ignored the role of content knowledge. She recommended that more theoretically-based work be conducted to confirm her conclusion about the role that preferred ways of thinking influenced knowledge development. Third, teachers' goals for student learning seem related to their instructional practices and may stem from their views of the role of education (i.e., impart knowledge or develop problem solving skills) or their subject matter understanding. Lastly, through interactions with students, teachers are able to observe student conceptions and receive feedback on how well their instructional methods support student learning, all of which supports their PCK development. Figure 2.1 displays Baxter's conceptualization of teacher content knowledge and the ways in which cognitive styles influence how teachers attend to new information before incorporating it into their PCK.

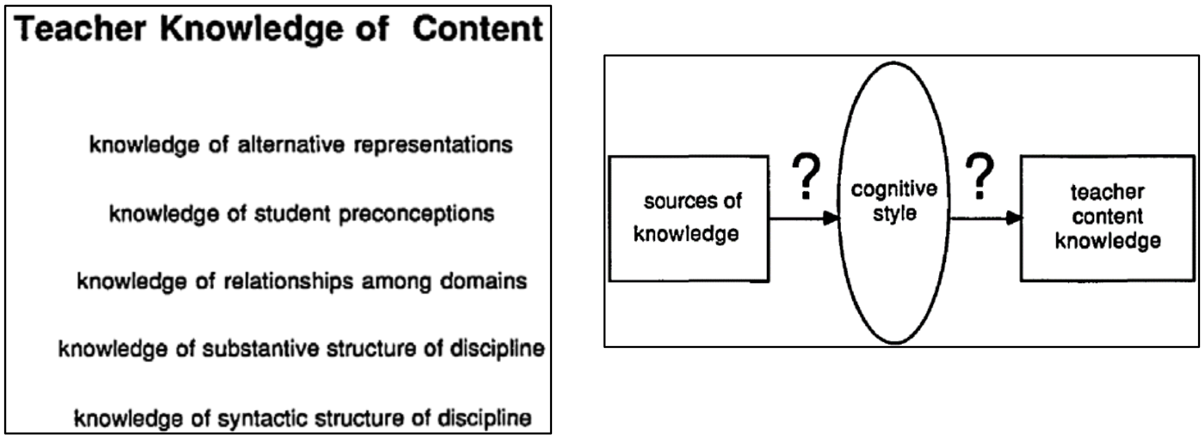

Teacher Knowledge of Content
knowledge of alternative representations
knowledge of student preconceptions
knowledge of relationships among domains
knowledge of substantive structure of discipline
knowledge of syntactic structure of discipline
sources of knowledge
?
cognitive style
?
teacher content knowledge

Figure 2.1. Baxter's *(1987, pp. 153, 157)* conceptualizations of PCK and CK (left) and the links between CK and cognitive styles (right).

Lapidot (2005) conducted a field study of 15 secondary teachers and pre-service teachers in Israel across four years using the methods of observations, interviews, and document analysis. Five pre-service teachers were analyzed more deeply during a practicum course. One in-service secondary teacher was followed across one year of teaching functional programming. Another in-service secondary teacher was followed across three years while teaching the first lesson on recursion. Lapidot used the results of this analysis to create a four-stage content reasoning model to explain how CS high school teachers learn, which was based off Shulman's (1987) model of pedagogical reasoning (see Figure 2.2). In the first stage, comprehension, teachers focus on learning the content they need to teach. In the second stage, transformation, teachers focus on ways of turning their content knowledge into instructional examples. During this stage, teachers may identify gaps in their content knowledge and return to the first stage. In the third stage, teaching, teachers focus on students and their understandings while they teach. In the final stage, reflection, teachers try to improve their practice by analyzing their own teaching and their students' understanding. Lapidot's model shares similarities with the science teacher learning progressions presented by Schneider and Plasman (2011). Lapidot also found that teacher learning was influenced by cognitive, social, and affective factors.

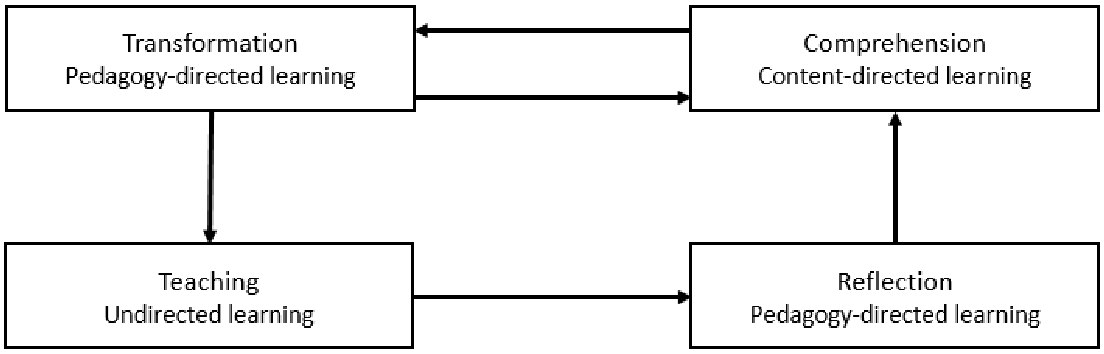

Transformation
Pedagogy-directed learning
Comprehension
Content-directed learning
Teaching
Undirected learning
Reflection
Pedagogy-directed learning

Figure 2.2. Lapidot's *(2005)* content reasoning model.

In contrast to the inductive approaches used by Baxter (1987) and Lapidot (2005), the other authors used abductive approaches to first identify PCK components from extant literature and pedagogical materials and then to confirm their models with data gathered from expert teachers. KUI was a multi-institutional research project funded by the German government between 2012 and 2015 with the goal of creating a competency model that could inform CS teacher training. KUI produced four articles included in this review (Bender et al., 2015; Hubwieser, Berges, et al., 2013; Hubwieser, Magenheim, Mühling, & Ruf, 2013; Margaritis et al., 2015). They defined competencies as "performance dispositions to solve complex situations" (Margaritis et al., 2015, p. 211). This group first created a system to categorize teacher education curricula by reviewing literature on PCK, teacher education standards, prior conceptualizations of PCK for CS, CSER publications, educational research in other disciplines, and textbooks on CS pedagogy. The resultant categorical system consisted of three dimensions: (a) *Fields of*

Pedagogical Operation which described three phases of the teaching processes that occur before, during, and after lessons; (b) *Aspects of Teaching and Learning* which described 15 categories related to other pedagogical components (i.e., subject and curriculum, teaching methods, learner issues, teacher issues, and educational system issues); and (c) *Non-cognitive Competencies* which described 17 categories related to social and communication skills, motivation and self-regulation, and beliefs and attitudes. Interviews were used to refine their model and provide descriptions of each dimension. Participants were presented with teaching scenarios related to each phase of the teaching process and then asked questions about how they would respond to the situation that were based on the aspects of teaching and learning dimension of the categorization scheme. The following is an example of a scenario related to the planning phase of the teaching process used by the KUI group to validate their framework:

> After a few hours on object-oriented modeling, you now want to start with the topic "Object Oriented Programming". You are planning that, after some time, your students should know what classes and objects are and that they are able to program their first small program. (Margaritis et al., 2015, p. 16)

After presenting the scenario to expert teachers, they were asked questions such as, “How would you proceed to plan the lesson so that your students can actively acquire the learning content?” or “What are the difficulties in relation to the planning of the lesson that can occur in this situation?” (Margaritis et al., 2015, p. 16). Thirteen experienced secondary CS teachers and ten CS teacher educators participated in the empirical validation of the model. Their work resulted in a framework of CS PCK that includes concrete descriptions of teaching competencies related to each framework category (see Figure 2.3).

Woollard (2005) used an approach similar to the KUI group to develop a framework for CS PCK based on data gathered from experienced teachers, existing curricular materials, and extant literature. Woollard argued that since metaphor is embedded in the design of computational tools (e.g., icons, window displays), it also plays an important role in the teaching of computer science in the form of pedagogic metaphor. Pedagogic metaphors, such as secret notes to represent the idea of encryption or a shopping checkout line to represent the idea of a queue, were defined as "a literally untrue description of a concept or body of knowledge…a description, in the form of words, actions, images or diagrams, of an element of teaching" (Woollard, 2005, p. 197). Six data sets were analyzed to identify and categorize pedagogic

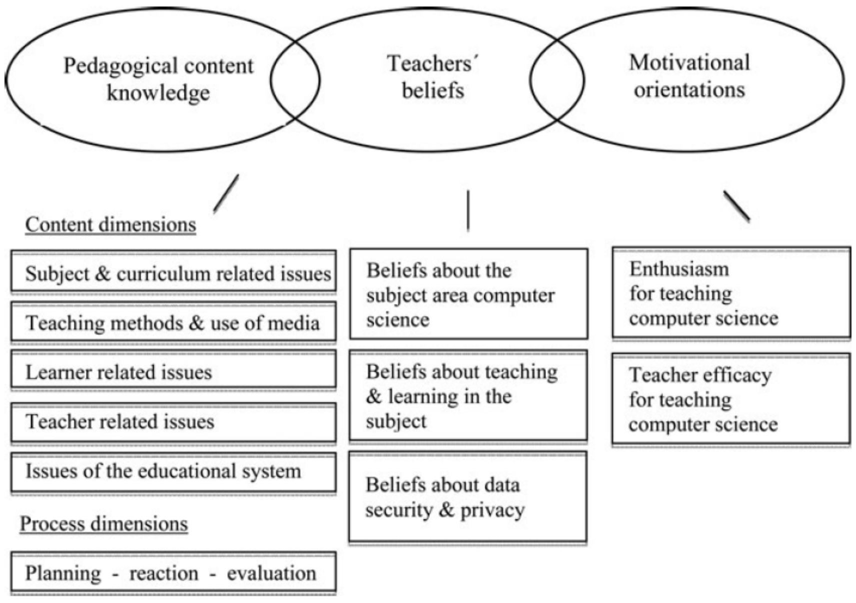

Pedagogical content knowledge
Teachers´ beliefs
Motivational orientations
Content dimensions
Subject & curriculum related issues
Teaching methods & use of media
Learner related issues
Teacher related issues
Issues of the educational system
Process dimensions
Planning - reaction - evaluation
Beliefs about the subject area computer science
Beliefs about teaching & learning in the subject
Beliefs about data security & privacy
Enthusiasm for teaching computer science
Teacher efficacy for teaching computer science

Figure 2.3. KUI's theoretical framework of CS PCK *(Bender et al., 2015).*

metaphors: 19 teacher interviews, 32 journal articles related to PCK, 18 teaching resources, three exams, 20 computing topics considered difficult to teach, and 20 computer metaphors Woollard identified in earlier work. Using a grounded theory approach, Woollard categorized the metaphoric approaches appearing in the data set along two axes (see Figure 2.4). Along the first axis, metaphors were either kinesthetic (i.e., concrete, based on devices and actions) or theoretical (i.e., conceptual, not linked to physical artifacts or activities). Along the second axis, metaphors were either traditional (i.e., common in the computing community) or novel (i.e., created by individuals in response to their teaching environments). An example of a kinesthetic, traditional metaphor of recursion is Russian dolls. An example of a theoretical, novel metaphor of recursion is a spiral turning. Woollard described the resultant PCK taxonomy as a tool to identify various non-literal explanations of computing concepts that can be used to address conceptual difficulties, isolated topics, learner misconceptions, and less interesting subject matter.

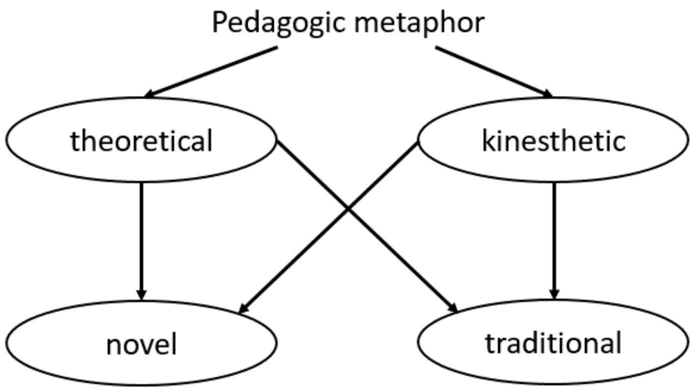


Figure 2.4. Woollard's *(2005)* framework of pedagogic metaphor.

Summary. The conceptualizations presented in this section provide different tools for describing the components, processes, and mediators of CS PCK. The frameworks confirm PCK research conducted in other fields by highlighting the relationship between content knowledge and PCK and the role of enactment in supporting PCK development. This body of work also adds to the CSER knowledge base by providing concrete examples of PCK within CS. While the authors hint at the developmental nature of PCK, three of the four frameworks were based on evidence gathered from experienced computing educators which might not provide a comprehensive view of PCK. As Baxter recommended:

> Studies of teachers who have been misassigned to subjects that they are not prepared to teach would enrich our understanding of the relationship between teacher content knowledge and explanations. These misassigned teachers might present explanations that are strikingly different from the explanations of knowledgeable teachers. (Baxter, 1987, p. 161)

A potential drawback of relying on the knowledge of experienced teachers to create PCK models is that it can lead to deficit views of the knowledge held by less experienced educators. Such models, for example, might be used to identify what areas of teacher knowledge are missing or underdeveloped in less experienced educators. Such models might also overlook components of teacher knowledge that appear in novice or proficient educators but not in experienced educators. As Schneider and Plasman concluded in their review of science PCK across teaching stages:

> New teachers tended to have ideas that were similar to preservice teachers yet showed some development. More experienced teachers…might have the same ideas as early career teachers or they might have much more development. Leader teachers…were often the most likely to have the most sophisticated ideas…Midcareer teachers, as well as illustrating novice ideas and more developed ideas, also illustrated ideas on a different overall path altogether. (Schneider & Plasman, 2011, p. 28)

Future work related to conceptualizing CS PCK could benefit from confirming existing models with less experienced educators or using similar methods to develop models of novice CS PCK.

Lastly, these conceptualizations of CS PCK were developed by scholars working within different countries where variation in teaching cultures might lead to culture-specific PCK models (Blömeke & Delaney, 2012). Using these models in diverse educational settings can also help to confirm their suitability for multiple contexts and identify possible areas of refinement (e.g., Zendler, McClung, & Klaudt, 2015).

Investigations of CS PCK. In this section of the literature review I focus on the methods used to study CS PCK. Following Depaepe et al. (2013), I summarize the research designs and participant counts of the identified studies. I also describe their countries of origin, school level of focus, teacher experience level, and time span to identify patterns in the types of studies that have been conducted. After summarizing the research designs, I provide detailed information on the instruments used within these studies.

Sixteen articles within the dataset were investigations of CS PCK and they are summarized in Table 2.2. The studies in this dataset were conducted mostly in Germany, Israel, or the U.S. Most articles reporting their school level focus concentrated exclusively on secondary education. Two articles also included a comparison to either primary education or tertiary education. There was a greater focus on in-service teachers than on pre-service teachers. Most studies collected data at one time point, but a handful of studies were longitudinal spanning 1 to 4 years. It is important to keep in mind that some articles reported on one aspect of a larger project which may have had a longer time span. Participant counts ranged from 2 to 1,127; one study did not include participants. Five types of data collection methods were used: interviews, classroom observations, meeting observations of teachers collaborating on an activity, a task (e.g., questionnaire, assessment), and document analysis. Similar to Depaepe et al.'s (2013) summary of studies on mathematics PCK, larger studies of CS PCK relied only on tasks while smaller studies tended to employ multiple data collection methods. Smaller studies were more prevalent than larger studies, which is likely explained by the fact that CS PCK is an understudied topic that needs to be explored more than validated at this point in time.

Table 2.2

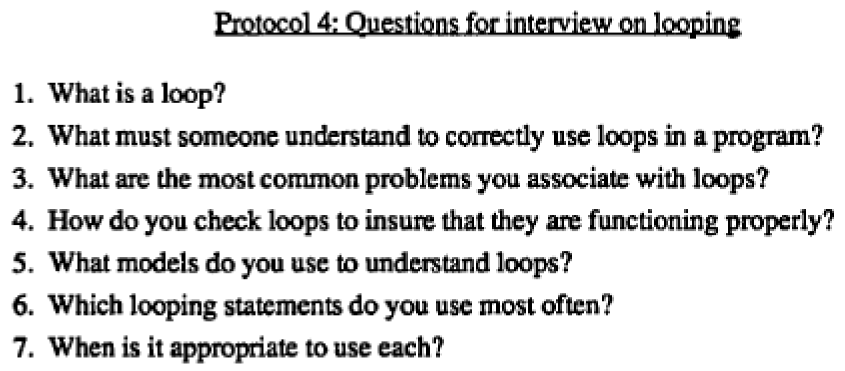
Research methods used to study CS PCK

Studies	Country	School Level	Teacher Level	Time Span	n	I	CO	MO	T	DA
Baxter (1987)	U.S.A.	Secondary	In-service	8 weeks	2	X	X		X	
Buchholz et al. (2013)	Germany	Not given	Pre-service	1 year	14				X	
Giannakos et al. (2014)	Greece	Secondary	In-service	Hours	1127				X	
Grgurina et al. (2014)	Netherlands	Secondary	In-service	Hours	10	X				
Griffin et al. (2016)	U.S.A.	Secondary	Pre-service In-service	1 year	2	X	X			X
Lapidot (2005)	Israel	Secondary	Pre-service In-service	4 years	15	X	X			X
Liberman et al. (2012)	Israel	Secondary	In-service	2 years	3	X	X		X	
Ohrndorf & Schubert (2013, 2014)	Germany	Not given	Pre-service	Hours	12				X	
Ragonis & Hazzan (2009)	Israel	Secondary	Pre-service[1]	1 year	11	X			X	
Saeli et al. (2012)	Netherlands	Secondary	N/A	N/A	N/A					X
Saeli et al. (2010)	Multiple	Secondary	In-service	Hours	30			X		
Schulte (2008)	Germany	Secondary	Pre-service	Hours	10			X		
Snelbecker & Bhote (1995)	U.S.A.	Primary Secondary	In-service	Hours	239				X	
Zendler & Hubwieser (2013), Zendler & Klaudt (2012)	Germany	Secondary Tertiary	In-service	Hours	240				X	

[1]One in-service teacher participated in this study, but the project was geared towards pre-service teachers.
Abbreviations in table: I = Interview; CO = Classroom observation; MO = Meeting observation; T = Task, test, or questionnaire; DA = Document analysis

While the majority of researchers developed their own instruments to investigate CS PCK, a few scholars incorporated existing instruments into their studies. Four projects (Buchholz et al., 2013; Grgurina et al., 2014; Saeli et al., 2012, 2010) used CoRe, an instrument created by Loughran, Mulhall, and Berry (2004) to study science PCK. With the CoRe, groups of teachers discuss eight prompts designed to elicit aspects of content knowledge and student understanding related to a big idea in a domain (e.g., matter is made up of particles). Some scholars administered the CoRe as intended with groups of teachers while others converted it into a questionnaire completed by individuals. Another project (Giannakos et al., 2014) used a subset of Schmidt et al.'s (2009) TPACK assessment. The TPACK assessment asks users to rate their level of agreement with 45 items related to their technology knowledge (e.g., I can learn technology easily), content knowledge (e.g., I have sufficient knowledge about mathematics), pedagogical knowledge (e.g., I know how to assess student performance in a classroom), PCK (e.g., I can select effective teaching approaches to guide student thinking and learning in mathematics), technological content knowledge (e.g., I know about technologies that I can use for understanding and doing mathematics), technological pedagogical knowledge (e.g., I can choose technologies that enhance the teaching approaches for a lesson), and their technological pedagogical content knowledge (e.g., I can teach lessons that appropriately combine mathematics, technologies, and teaching approaches). Snelbecker and Bhote (1995) administered two assessments to measure teachers' aptitudes and attitudes towards computing: CALIP and CAS. The Computer Aptitude, Literacy, and Interest Profile (CALIP) contains six subscales to measure total aptitude, interest, and literacy (Poplin, Drew, & Gable, 1984). The Computer Attitude Scale (CAS) contains 40 items across four subscales that measure computer anxiety, computer liking, computer confidence, and computer usefulness (Loyd & Gressard, 1984). One advantage of using existing instruments is that they will likely provide good measures of the variables of interest because they have been validated. Another benefit of using existing instruments is that there is potential to compare data gathered from CS teachers with data gathered from teachers of other disciplines and identify unique elements of CS teaching. One possible disadvantage to using widely available instruments to measure teacher knowledge is that they may not capture factors specific to computing. In the following paragraphs, I describe the other instruments created by researchers to study CS PCK.

Interviews. New interview protocols were developed for five of the studies. Ragonis and Hazzan (2009) mentioned collecting interview data but did not provide details on their interview prompts. Lapidot (2005) used interviews to understand participants' content knowledge, instructional plans, the impact of lessons on their knowledge, and any issues they faced. Each interview related to a specific lesson observed by Lapidot. Both Baxter (1987) and Griffin, Pirmann, and Gray (2016) dedicated some of their interviews to gathering background information about participating teachers, their students, and schools. Baxter (1987) also created two additional interview protocols to use in her study. The first interview asked participants to review a list of CS topics and rate the emphasis of each topic in their courses, rate the level of difficulty for students, and describe instructional strategies used for each topic. At the end of the interview participants were asked to rate five approaches to program development indicating the importance of each approach in their teaching. Baxter's second interview consisted of seven questions related to teaching the topic of loops (see Figure 2.5). Some of these items (e.g., 2, 3, 7) bear a resemblance to several items on the CoRe. Griffin et al.'s (2016) interview protocol focused mainly on teaching background and school context, but they also included prompts related to changes in teaching practices, ways of managing students' social interactions, teaching philosophies, and future course plans. Lastly, Liberman, Kolikant, and Beeri (2012) focused on eliciting descriptions of teaching episodes, instructional resources, and the pedagogical considerations made when selecting representations.



Protocol 4: Questions for interview on looping

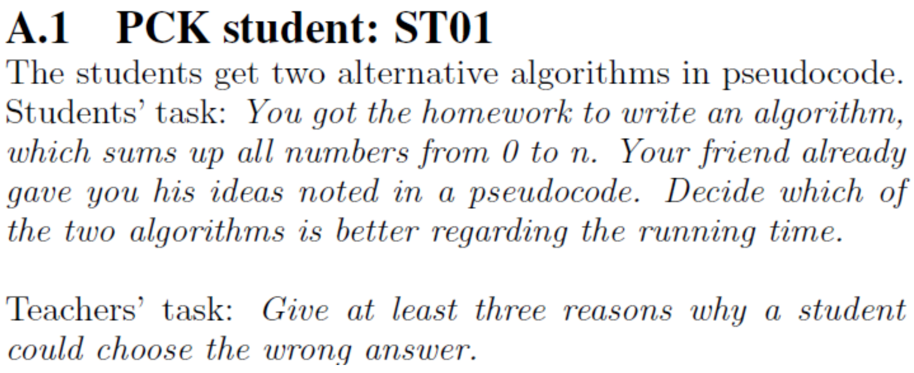
1. What is a loop?
2. What must someone understand to correctly use loops in a program?
3. What are the most common problems you associate with loops?
4. How do you check loops to insure that they are functioning properly?
5. What models do you use to understand loops?
6. Which looping statements do you use most often?
7. When is it appropriate to use each?

Figure 2.5. Baxter's *(1987)* interview protocol on teaching loops.

Classroom Observations. Classroom observations were mentioned in four of the articles: Baxter (1987), Griffin et al. (2016), Liberman et al. (2012), and Lapidot (2005). Little detail was provided on the structure of these observations, with most authors mentioning the use of fieldnotes. Griffin et al. (2016) described their observations as focusing on the initial state of the classroom, instructional activities, and social interactions.

Meeting observations. Schulte (2008) designed an observational study of teachers planning a lesson on bubble sort, an algorithm for arranging a series of numbers in order. Non-participant observers took notes as each group discussed how to plan lessons, analyzed a function to implement Bubble sort, brainstormed ideas on delivering lessons, and developed a lesson plan. Observation notes focused on ideas produced by participants, whether ideas were backed with evidence, and communication difficulties.

Tasks. New tasks were designed for six of the articles. Liberman et al. (2012) briefly mentioned the use of a subject matter knowledge assessment but they did not provide details on its contents. Baxter (1987) made use of a word association task where participants were given a programming term (e.g., variable, conditional) and asked to say the first thing that came to their minds. Similar to Schulte's (2008) meeting observation, Baxter also asked teachers to review and critique a computer program. Ohrndorf and Schubert (2013, 2014) created a set of tasks that presented teachers with instructional scenarios and asked them to explain possible student responses and misconceptions (see Figure 2.6). Ragonis and Hazzan (2009) asked participants to


A.1 PCK student: ST01

The students get two alternative algorithms in pseudocode. Students' task: *You got the homework to write an algorithm, which sums up all numbers from 0 to n. Your friend already gave you his ideas noted in a pseudocode. Decide which of the two algorithms is better regarding the running time.*

Teachers' task: *Give at least three reasons why a student could choose the wrong answer.*

Figure 2.6. Sample item from Ohrndorf and Schubert's PCK task *(2013).*

complete multiple tasks including feedback worksheets after each tutoring session, evaluation questionnaires completed at the end of the semester, and various homework assignments. The first three items of their feedback worksheets share similarities with the CoRe in their focus on student difficulties and ways of addressing those difficulties (see Figure 2.7). Lastly, Zendler and colleagues (Zendler & Hubwieser, 2013; Zendler & Klaudt, 2012) presented participants with a grid of 15 CS concepts (e.g., algorithm, data) by 16 CS process concepts (e.g., classifying, ordering) and asked them to rank the relevance of each process for each concept. Concepts included on the grid were identified through prior surveys of CS professors.

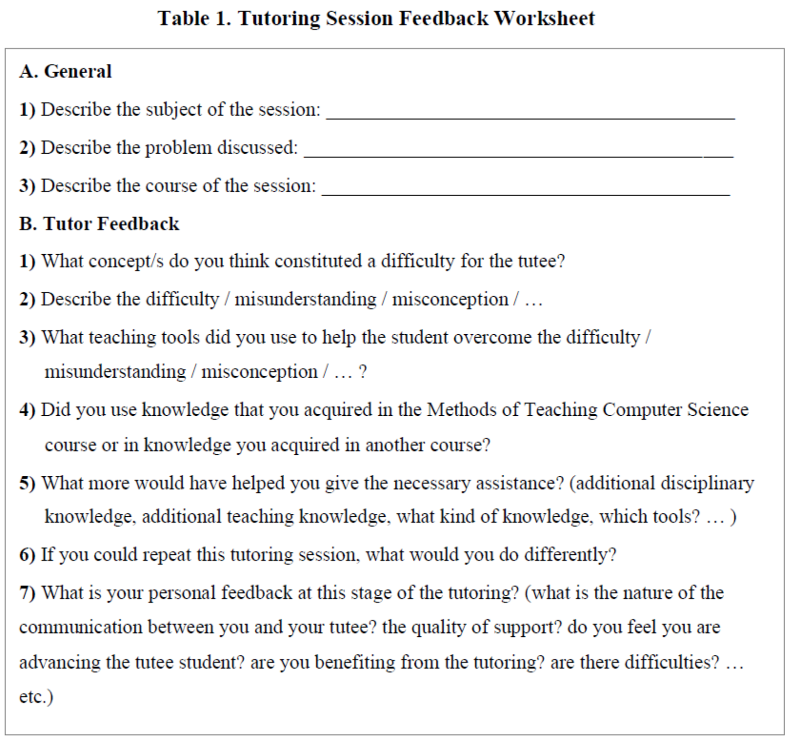


Table 1. Tutoring Session Feedback Worksheet

A. General

1) Describe the subject of the session: ______________________________

2) Describe the problem discussed: ______________________________

3) Describe the course of the session: ______________________________

B. Tutor Feedback

1) What concept/s do you think constituted a difficulty for the tutee?

2) Describe the difficulty / misunderstanding / misconception / …

3) What teaching tools did you use to help the student overcome the difficulty / misunderstanding / misconception / … ?

4) Did you use knowledge that you acquired in the Methods of Teaching Computer Science course or in knowledge you acquired in another course?

5) What more would have helped you give the necessary assistance? (additional disciplinary knowledge, additional teaching knowledge, what kind of knowledge, which tools? …)

6) If you could repeat this tutoring session, what would you do differently?

7) What is your personal feedback at this stage of the tutoring? (what is the nature of the communication between you and your tutee? the quality of support? do you feel you are advancing the tutee student? are you benefiting from the tutoring? are there difficulties? … etc.)

Figure 2.7. Ragonis and Hazzan *(2009)* Tutoring Session Feedback Worksheet.

Document analysis. Three articles included document analysis in their study of CS PCK. Griffin et al. (2016) reviewed materials from the CS Principles Framework to interpret teachers' implementation of the framework. Lapidot (2005) reviewed participants' lesson plans and curricular materials. Document analysis was the primary method used by Saeli et al. (2012) who analyzed the PCK information contained within the three Dutch textbooks available for CS instruction in the Netherlands. The authors used the CoRe to create an instrument for evaluating these materials.

Summary. CSER scholars employed a variety of methods to investigate PCK. Many of these methods were not unique to the field of computing, instead, they were contextualized to CS environments (e.g., asking questions about student misconceptions that could easily be ported to another discipline by replacing ‘computer science’ with ‘mathematics’). Multiple investigators developed their own tools to examine PCK, which often shared similarities with existing instruments. Future studies may benefit from adapting existing PCK instruments to their specific CS contexts as this would allow for greater comparison across studies and reduce the effort needed to develop robust measures. At the same time, Fincher, Tenenberg, and Robins (2011) argued that more methods specific to CS are needed to advanced CSER as its own discipline. The aforementioned tasks related to analyzing code, student work, or the relationship between CS concepts provide good examples of such approaches. Future work may attend to developing additional measures that focus on unique aspects of CS teaching.

Factors Influencing CS PCK Development. In the previous sections of this chapter, I examined how PCK is conceptualized and methods researchers used to study teacher knowledge. Another goal of this literature review was to identify factors influencing the development of teacher knowledge that should be considered when investigating PCK. This section is not a summary of components common to CS teacher learning opportunities. Rather, my focus is on the personal and contextual elements that play a role in what teachers absorb from their learning opportunities.

Nine articles within the dataset described factors influencing CS PCK development. Two of these articles discussed the same line of work produced by the Exploring Computer Science (ECS) group at UCLA. The nine articles, written between 2011 and 2016, represent work conducted in the U.S. (5), Greece (2), Argentina (1), and multiple countries (1). Eight articles focused on secondary teachers and one article focused on both primary and secondary teachers. Information on influential PCK factors were gathered by asking teachers through interviews and surveys what best supported them during their learning experiences. Some of this work was also supplemented with observation data and log file data. While each article had a primary focal area (see Table 2.3), I noticed similar ideas across the papers related to the dynamic nature of the CS field, teachers' working environments, need for community, beliefs and identity, and classroom implementation. Insights related to these ideas are summarized below.

Table 2.3

Articles describing factors influencing CS PCK

Article	Primary Focus	Country	School Level
Dagdilelis & Xinogalos (2012)	Professional Development	Greece	Secondary
Ericson, Rogers, Parker, Morrison, & Guzdial (2016)	Professional Development	Multiple	Secondary
Goode, Margolis, & Chapman (2014) Ryoo, Goode, & Margolis (2015)	Professional Learning Community	USA	Secondary
Kordaki (2013)	Beliefs	Greece	Secondary
Leake & Lewis (2016)	Professional Learning Community	USA	Secondary
Guzdial (2011)	Review	USA	Secondary
Martinez, Gomez, Moresi, & Benotti (2016)	Professional Development	Argentina	Primary, Secondary
Ni & Guzdial (2012)	Identity	USA	Secondary

Dynamic discipline. The field of CS is constantly advancing. In response to these changes, CS educators often need to incorporate new material, programming languages, and paradigms into their courses. For example, in just twenty years, the College Board switched the language of instruction for the AP CS A course from Pascal, to C++, and then to Java (Roberts, 2004). In the most recent survey administered by the Computer Science Teacher Association (CSTA) to learn about the state of CS education in America, respondents reported that rapidly changing technology was one of their greatest challenges to teaching CS (CSTA, 2015). For some teachers, this creates a need for continual learning in order to keep one's knowledge and skills updated (Ni & Guzdial, 2012) and can lead to feelings of inefficacy (Kordaki, 2013). Teachers sometimes find it difficult to locate resources to support their practice and try to identify materials from websites of other educators working in similar teaching contexts (Leake & Lewis, 2016). For other teachers, the dynamic nature of CS makes sharing practices with colleagues during PD opportunities a productive activity (Ryoo et al., 2015).

Working environments. The realities of teachers' working environments also influence learning opportunities. Three articles highlighted how teaching workloads limited the amount of time educators could devote to participating in communities or learning CS content, particularly through systems not designed specifically for educators (Ericson et al., 2016; Guzdial, 2011; Leake & Lewis, 2016). For example, in a study of interactive electronic books designed to support high school CS teachers new to programming, Ericson et al. (2016) found teachers tended to work through the books in short spurts of time and they recommended designing modules that could be completed in 20 minutes. Leake and Lewis' (2016) interviews with secondary teachers showed that teachers found participation in online teaching communities as distracting from their primary goal of helping students. Factors at the school system level also influenced opportunities to learn. In some situations, teaching courses influential to college admissions restricted educators' choice of instructional methods (Dagdilelis & Xinogalos, 2012; Kordaki, 2013), limiting their use of new approaches or technologies. The status of CS as an elective course led some teachers in Greece to use direct teaching methods and drill and practice techniques to maintain student attention (Kordaki, 2013). In the U.S., this may also be an issue because CS is often relegated to the periphery of secondary instruction while courses that satisfy college entrance requirements and align with standardized testing are prioritized. Only twenty-six states and the District of Columbia allow CS courses to fulfill high school math or science graduation requirements (“Where computer science counts,” n.d.) and only 30% of teachers and administrators report prioritizing CS in their districts (Google & Gallup, 2015).

Community. Professional learning communities also play a role in supporting CS teacher knowledge development. While CS teachers have opportunities to connect and exchange information with peers at public gatherings like conferences (e.g., Garcia, Franke, Hoeppner, & Paley, 2014), they are often the only CS teachers at their schools. This isolation makes it difficult for them to interact with other CS educators on a regular basis and discuss issues of practice (Ni & Guzdial, 2012). ECS professional development providers designed their program specifically to address this issue of isolation. Their multi-year professional development program brought together CS educators from different schools to learn about inquiry-based and equity-based practices used in the ECS curriculum. Surveys collected from participants over three years showed that the community of colleagues was the most impactful PD component for ECS teachers’ professional growth and a source of new pedagogical knowledge (Goode et al., 2014; Ryoo et al., 2015).

Beliefs and identity. Ni and Guzdial (2012) argued that a teacher’s sense of identity influenced how one teaches, how one develops as a teacher, and one’s commitment to a secondary CS teaching career. They interviewed nine teachers with CS course assignments to gain insight into CS teaching identities. Teachers identifying as CS teachers understood the scope of the field, saw CS as valuable for students, wanted to keep updated with changes in CS, and wanted to connect with colleagues. Participants seeing themselves as both a CS teacher and a teacher of another discipline would incorporate content from their other discipline into CS courses and desired to connect with likeminded CS teachers. Teachers identifying only as business teachers struggled to distinguish computer applications from CS, did not want to pursue further learning or peer collaboration, and did not see a need for their schools to offer more CS courses. Kordaki (2013) discovered a similar range of ideas amongst twenty-five CS teachers in Greece. She also noticed some patterns between teachers' beliefs, motivations, and teaching practices. Teachers employing direct teaching strategies with minimal student involvement (e.g., reading textbooks aloud) were extrinsically motivated to pursue their careers because of the advantages of a teaching career (e.g., job security, work-life balance). Teachers who did not teach and simply occupied students time (e.g., allowing them to surf the internet) had low expectations for students such as becoming familiar with Microsoft applications. Lastly, teachers who used project work and encouraged problem solving and critical thinking in their courses wanted to become more effective teachers and were intrinsically motivated to enter the profession because they loved teaching.

Classroom implementation. A few articles discussed observations of CS teaching practices that can inform our understanding of how teachers make use of their PCK. In discussing their prior work that led to the design of electronic books for CS teachers, Ericson et al. (2016) described how expert teachers often spent time debugging student code and that they rarely wrote their own code. In an evaluation of a professional development program for CS educators in Argentina, Martinez et al. (2016) found that nearly half of participants used the same strategies presented and practiced in the workshop, but that less than a third of participants used strategies that were only presented. Martinez et al. also found that in attempting to recreate inquiry based tasks in their classrooms, teachers lacking in content knowledge introduced tasks that did not focus specifically on CS.

Summary. PCK does not exist or operate in isolation. The work summarized in this section demonstrates that various personal and contextual factors influence how teachers learn, what teachers can learn, and the ways teachers implement their knowledge in practice. We cannot assume that teachers will simply increase their PCK if provided with materials describing common student misconceptions, examples of instructional strategies, or tutorials to learn CS content. The dynamic nature of CS requires teachers to be lifelong learners and learning opportunities need to account for the limited time teachers have for professional development. Camaraderie goes a long way to help teachers develop a sense of identity as CS educators, persist in their careers, and learn about alternative teaching strategies. Beliefs about teaching and learning can be reflected in classroom teaching practices, with some beliefs more productive than others in supporting teacher learning. Also, CS teachers work differently than professional software engineers, so we cannot expect them to learn in ways shown useful for professional computer scientists. Lastly, external factors at the student and administrative level play a hand in the decisions teachers make. Thus, we must consider the sociocultural contexts within which teachers work when investigating how they develop PCK. This attention to factors beyond teacher cognition is not unlike ecological models of human development that consider cultural socialization and identity development in the learning process (e.g., Lee, Spencer, & Harpalani, 2003).

2.3 Computer Science Learning and Teaching

Given that PCK is domain specific, what are the unique features of CS related to acquiring teacher knowledge of the discipline? To address this question, I provide an overview of CS learning and teaching by discussing (a) the nature of the discipline, (b) CS in K-12

American classrooms, and (c) possible challenges for teachers transitioning into CS from other disciplines.

2.3.1. Nature of CS

What is computer science? What processes and tools can we use to learn computer science? These questions relate to epistemology, or the ideas people hold about the nature and acquisition of knowledge. Chinn, Buckland, and Samarapungavan (2011) described epistemic cognitions as relating to (a) inquiry goals and the worth associated with achieving goals (i.e., epistemic aims and values); (b) how knowledge is structured (e.g., knowledge as deterministic versus stochastic); (c) the origins of knowledge, the reasons for one's beliefs, and the attitudes taken towards ideas; (d) dispositions that support or hinder epistemic aims; and (e) processes for achieving epistemic aims. They argue that these cognitions are important to consider because they relate to how people approach learning and teaching. As an example, "a student who aims for explanations in a history class may not be satisfied just to learn the tragic sequence of events of the Great Depression; the student may seek deeper economic explanations of why it began and why it lasted so long" (Chinn et al., 2011, p. 147). Furthermore, epistemic cognitions vary across disciplines. For example, while scientists aim to make the most accurate claims possible by building theoretical models of the natural world based on data, those interpreting literary texts aim to explore the human experience by attending to the language content and form of texts which are open to multiple interpretations (Lee, Goldman, Levine, & Magliano, 2016). When investigating learning and teaching, we cannot expect an individual to necessarily bring the same epistemic cognitions into different disciplines.

Inside the classroom, the dominant epistemic cognitions of authentic disciplinary practice are sometimes not reflected in curricula, leaving students with a limited view of the domains they study. Chinn and Malhotra (2002) demonstrated that activities in science textbooks designed for middle school and upper elementary lacked many of the cognitive processes used in authentic science practice and assumed a conflicting epistemology where “scientific reasoning is viewed as simple, certain, algorithmic, and focused at a surface level of observation” (2002, p. 190). To address this, some education researchers are using epistemic cognition as a guide in designing instructional frameworks for disciplinary reading that incorporate the inquiry and argumentation processes of authentic disciplinary practice (Goldman et al., 2016). Lastly, teachers’ epistemic cognitions are reflected in their instructional decisions (e.g., minimizing required curricular material they believe do not support disciplinary thinking) and in the ways in which they engage with content in class (e.g., how they respond to students using alternative methods), which may convey ideas to students about the preferred epistemic cognitions to assume for a given discipline (Buehl & Fives, 2016). Teachers and students entering CS classrooms for the first time may confront disciplinary epistemologies that differ from their known school experiences. These differences may present challenges as they adjust to new ways of learning. In the following paragraphs, I address the epistemologies undergirding the CS discipline and K-12 CS education.

In the mid-1980s, nearly forty years after the start of modern CS, the Association for Computing Machinery (ACM) and the Institute of Electrical and Electronics Engineers (IEEE) appointed the Task Force on the Core of Computer Science to create a framework and teaching paradigm for CS. The task force, viewing no need to distinguish the core of CS from the core of computer engineering broadened their scope to *computing* and offered the following definition:

“The discipline of computing is the systematic study of algorithmic processes that describe and transform information: their theory, analysis, design, efficiency, implementation, and application. The fundamental question underlying all computing is, ‘What can be (efficiently) automated’?” (Denning et al., 1989, p. 12). More recently, Denning (2003, see Table 2.4) described the discipline as consisting of five mechanics that govern the operation of computations, five conventions for designing computation, five standard practices for building and deploying computing systems, and thirty core technologies (e.g., artificial intelligence, data structures, graphics, natural language processing, operating systems).

Three paradigms dominate the discipline that differ in their aims, processes, and epistemologies: a rationalist approach, a technocratic approach, and a scientific approach (Eden, 2007; Tedre & Sutinen, 2008). The rationalist approach, prominent in subfields such as theoretical CS, views CS as a branch of mathematics, aims for coherent theoretical structures and systems, and seeks a priori knowledge of program correctness through deductive reasoning. The technocratic approach, prominent in subfields such as software design, views CS as a branch of engineering, aims for useful, efficient, and reliable systems, and seeks a posterior knowledge of program reliability through empirical methods. The scientific approach, prominent in subfields such as artificial intelligence, views CS as a natural science, aims to investigate and explain phenomena, and seeks to explain, model, and predict the behavior of programs using both deductive and empirical methods.

As mentioned above, disciplines are sometimes practiced differently in school settings. Recent efforts such as the CSTA K-12 Computer Science Standards (Seehorn et al., 2016) and the K-12 Computer Science Framework (“K–12 Computer Science Framework,” 2016) offer

Table 2.4

Denning's (2003) Great Principles of Computing

Area	Components	Description
Mechanics	Computation	What can be computed; limits of computing
	Communication	Sending messages from one point to another
	Coordination	Multiple entities cooperating toward a single result
	Automation	Performing cognitive tasks by computer
	Recollection	Storing and retrieving information
Design	Simplicity	Various forms of abstraction and structure that overcome the apparent complexity of the applications
	Performance	Predicting throughput, response time, bottlenecks, capacity planning
	Reliability	Redundancy, recovery, checkpoint, integrity, system trust
	Evolvability	Adapting to changes in function and scale
	Security	Access control, secrecy, privacy, authentication, integrity, safety
Practice	Programming	Using programming languages to build software systems that meet specifications created in cooperation with the users of those systems
	Engineering systems	Designing and constructing systems of software and hardware components running on servers connected by networks
	Modeling and validation	Building models of systems to make predictions about their behavior under various conditions; and designing experiments to validate algorithms and systems
	Innovating	Exercising leadership to design and bring about lasting changes to the ways groups and communities operate
	Applying:	Working with practitioners in application domains to produce computing systems that support their work

insight into educators’ ideas of how CS curricula should be implemented in schools. The frameworks describe five core concepts that capture major CS content areas, five crosscutting concepts used across the core concepts, and seven core practices needed to make use of the concepts (see Table 2.5).

Although structured differently, the K-12 frameworks touch on most elements presented in Denning’s model. The K-12 frameworks do not appear to foreground the topics of automation (i.e., deciding when and how to automate cognitive tasks) or evolvability (i.e., adapting to changes in function and scale). In contrast to Denning’s model, the K-12 frameworks place more emphasis on understanding the impacts of computing, collaborating with peers, and communicating about computing. Also, with their focus on (a) creating and refining computational artifacts and (b) developing and using abstractions, the K-12 frameworks appear to support technocratic and scientific paradigms of CS and less so the rationalist paradigm that views CS as a branch of mathematics. These differences might be explained by the K-12 frameworks’ focus on CS knowledge and skills beneficial to all students, not just those who plan to pursue tertiary studies or careers in CS. While frameworks provide guidance on what to include in curricula and when, they do not detail how core concepts and practices should be implemented. In the next section I address different ways CS has been implemented in K-12 classrooms.

Table 2.5

K-12 Computer Science Framework ("K–12 Computer Science Framework," 2016; Seehorn et al.,

Area	Components	Description
Core Concepts	Computing systems	The physical components (hardware) and instructions (software) that make up a computing system communicate and process information in digital form
	Networks and the internet	Networks connect computing devices to share information and resources
	Data and analysis	Data is collected and stored so that it can be analyzed to better understand the world and make more accurate predictions
	Algorithms and programming	An algorithm is a sequence of steps designed to accomplish a specific task. Algorithms are translated into programs, or code, to provide instructions for computing devices.
	Impacts of computing	Computing influences culture and culture shapes how people engage with and access computing
Crosscutting Concepts	Abstraction	The process, or its result, of reducing a process or set of information to a set of important characteristics for computational use
	Systems relationship	A systems perspective provides the opportunity to decompose complex problems into parts that are easier to understand, develop, fix, and maintain. General systems principles include feedback, control, efficiency, modularity, synthesis, emergence, and hierarchy
	Human-computer interaction	The study of how people interact with computers and to what extent computing systems are or are not developed for successful interaction with human beings
	Privacy and security	Secluding information and expressing it selectively; safeguards surrounding information systems including protection from theft or damage to hardware, software, and the information in the systems
	Communication and coordination	The reliable exchange of information between autonomous agents (communication) that cooperate toward common outcomes that no agent could produce alone (coordination)
Practices	Fostering an inclusive computing culture	
	Collaborating around computing	
	Recognizing and defining computational problems	
	Developing and using abstractions	
	Creating computational artifacts	
	Testing and refining computational artifacts	
	Communicating about computing	

2.3.2. Computing Education

Origins of Computing Education. The arrival of digital computers had a profound effect on education. In discussing the educational implications of modern computers, Forsythe (1963) predicted computing would be critical for technical and non-technical students alike and require both new ways of teaching existing subject areas and a new academic discipline focused on computer technology that he called computer science. Around that time, Goodlad, O'Toole, and Tyler estimated approximately 300 American high schools were using computers for instructional purposes (as cited in Korotkin, Darby, Jr., & Romashko, 1970, p. 34). During the 1970s, pioneers such as Seymour Papert and Alan Kay spearheaded research on the use of computers for education with visionary ideas about the ways computing could transform learning (Solomon, 1986). These efforts have had lasting effects on computing education.

Papert cautioned against using computers to simply replicate existing educational practices and advocated for a more exploratory use of computers to introduce learners to programming and computation, which he demonstrated with several activities involving the Logo programming language (Papert & Solomon, 1971). Logo is an educational programming language designed to support young learners and those without extensive knowledge of mathematics (Feurzeig, Papert, Bloom, Grant, & Solomon, 1969). Due to widespread adoption of microcomputers in the 1980s and the release of Papert's book *Mindstorms: Children, Computers, and Powerful Ideas*, Logo became more accessible, grew in popularity, and was even required in many school curricula (Kahn, 2015). Influenced by the work of Papert, Kay envisioned a new technological medium that could augment learning by allowing children to express ideas and explore areas of interest (Kay, 1972). He brought the idea of personal, portable computing into the realm of education with Dynabook and the Smalltalk programming language (Kay & Goldberg, 1977). Kay's efforts led to the creation of the desktop user interface as well as the visual programming environment Etoys (Guzdial, 2004).

Another important development in the expansion of computing education was the emergence of Papert's theory of constructionism. This perspective views learning as the construction of knowledge that occurs through the creation, debugging, and sharing of artifacts (Papert & Harel, 1991). Also important in this theory are *learning cultures*, or social contexts that offer learners opportunities to interact with members of a community and develop connections to the ideas they are learning (Kafai, 2006). Some of Papert's students extended this theory by highlighting the value of cultural identity expression and shared constructionist activity for knowledge development (Hooper, 1996; Shaw, 1995). Papert's Epistemology and Learning Group further advanced computing education by taking up constructionist ideas in the design of computer applications to support learning. For example, Harel and Papert (1990) described a year-long project where constructionist ideas and the programming language LogoWriter were used to allow fourth grade students to create their own instructional software to explain fractions. As another example, microworlds, or open-ended exploratory computing environments, designed to explore mathematical and scientific ideas resulted from the constructionist tradition (Brandes & Wilensky, 1991; Edwards, 1998; Papert, 1972).

The efforts of Papert, Kay, and their contemporaries as well as constructionist ideas continue to profoundly influence the computing education movement today. Their impact is reflected in the numerous tools and initiatives derived from their work such as the educational programming environment Scratch, the robotic construction kit Lego Mindstorms, and the One

Laptop per Child initiative (Kahn, 2015). Although current computing education initiatives vary widely across the globe, there are two main approaches used in their curricula: teaching CS as its own subject and integrating computing into other disciplines (Ragonis, 2009). Although this dissertation focuses on CS education, I provide a brief summary of both approaches in order to situate my work within the larger computing education landscape.

Computer Science. Computer science entered American high schools over sixty years ago. Buchman (1956) provided an early example of a course designed for high school seniors in advanced mathematics that included (a) instruction on number systems, the history and mechanics of computers, and programming; (b) lab exercises to design and implement a program to compute a formula; and (c) a visit to a local science and technology company that used automatic computers. As another example, researchers at Stanford University organized three courses designed to introduce fundamental programming concepts through a computer-assisted instruction platform to ninety workplace-bound high school students between 1968-1970 (Lorton & Cole, 1981). The courses focused on three commonly used programming languages (i.e., Simper, (S)Logo – a variant of the Logo language, and BASIC). Since that time, CS education has evolved in its reach, focus, and format. In the following paragraphs, I provide a glimpse of the CS education landscape at the secondary level in the U.S. over the past five decades.

During the 1969-1970 academic year, Korotkin, Darby, and Romashko (1970) surveyed 12, 396 public secondary schools in the United States to learn about their use of computers and found: approximately 13% of schools reported using computers for instructional purposes (e.g., problem solving and electronic data processing, guidance and counseling, gaming and simulation, computer-assisted instruction, and management of instruction); computers were most frequently incorporated into math, science, data processing, and business education courses; and the most commonly used programming languages were Fortran and BASIC. About ten years later, the Johns Hopkins University Center for Social Organization of Schools (1983) reported preliminary results of 990 survey responses collected from public, private, and parochial primary and secondary schools in the United States during the 1982-1983 academic year. Their results showed that, at the high school level, programming was the preferred computer instructional activity and BASIC was the most popular language, with a minority of schools teaching Fortran, Logo, or Pascal. It was during this decade that the College Board released its first Advanced Placement exam for CS which used Pascal to test high school students' understanding of CS (Roberts, 2004). ACM also released the first edition of its secondary computer science curriculum guide which included two courses focused on algorithms and problem-solving to prepare students for CS studies in college, one course to introduce a broader audience of students to programming, and a computer literacy course (Turner, 1985).

In the 1990s, the Strategic Directions in Computing Research Education Work Group, a coalition of eighteen CS scholars, reported on issues in CS education across the grade span which included the lack of a coherent, widely-used high school curriculum (Tucker, 1996). Harvey (1991) described how the lack of an official high school CS curriculum led many secondary CS courses to be strongly influenced by the existing AP CS course, which followed an engineering paradigm of CS prevalent in colleges and appropriate for students preparing to enter careers as professional programmers. Instead, he argued high school CS curricula should be more exploratory, project-based, and driven by learners' interest to encourage budding curiosity in programming. Stephenson (2000), drawing on extant literature and survey results from teachers in five states, reported a decline in the number of college-bound students taking programming courses compared to prior years and an increase in the use of object-oriented programming languages (e.g., Java, C++) in schools. During this decade, ACM released its second high school CS curricular guide which focused on seven topic areas (i.e., algorithms; programming languages; operating systems and user support; computer architecture; social, ethical, and professional context; computer applications; and advanced applications like artificial intelligence or computational science) (Merritt et al., 1993). Also, the College Board switched from using Pascal to using C++ on the AP CS exam (Roberts, 2004).

In the summer of 1999, an ACM task force met to discuss ways of improving K-12 education in science, mathematics, and computing (De Blasi, 1999) and later produced a four-tiered curriculum model for CS (Tucker et al., 2003). From this committee, CSTA was formed for the purposes of "supporting teachers and pursuing excellence in computing education for students age 5-18 (K-12)" (Stephenson, 2005, p. 29). In 2004, just six years after the previous programming language change, the College Board switched from using C++ to Java on the AP CS exam, forcing teachers to learn another program language and leading many schools to remove AP CS from the curriculum (Roberts, 2004). Many described the state of CS during this decade as a crisis. According to the CSTA Curriculum Improvement Task Force, the crisis manifested in a decrease of CS courses, enrollments, and levels of involvement from female and minority students (Stephenson, Gal-Ezer, Haberman, & Verno, 2005).

Within the last seven years we have seen the advent of several initiatives to drastically increase and improve CS opportunities in the U.S. at the national level (Astrachan, Cuny, Stephenson, & Wilson, 2011) such as CS10K, an NSF-initiative to "develop effective new high school computing curricula and get it into 10,000 high schools taught by 10,000 well-prepared teachers by 2016" (Cuny, 2012, p. 35). CSTA surveys high school computer science teachers biennially to learn about the state of CS education in America. The most recent survey (CSTA, 2015) was administered in spring 2015 to 10,182 educators identified as CS, computer programming, or AP CS teachers; 1,354 educators responded. Respondents reported their schools offered introductory CS courses (89%), Computer Science Principles which is now offered as an AP course (66%), AP CS A (78%), Code.org curricula (66%), and other courses (88%). In introductory courses, the most commonly covered topics were game programming, problem solving, and logic applications in information technology and information systems. The most commonly used programming languages were Scratch, Python, and C++ or C#. The most common other courses that were not AP courses or introductory courses included computer graphics, web design, and networking. CS courses are offered under a variety of departments including business (67%), technology (51%), science (42.8%), and math (15%); only 5% of teachers reported their CS courses were offered within a computing department.

The CS education community is moving towards wide adoption of a few curricula and curriculum frameworks including ECS, AP CS Principles, and AP CS A (Cuny, 2012, 2015). ECS is a pre-AP CS course built around inquiry-based instruction and culturally relevant and meaningful curriculum (Goode & Margolis, 2011). It includes six units that each conclude with a unit project: human-computer interaction, problem solving, web design, introduction to programming, robotics, and computing applications. The curriculum incorporates CS pedagogical tools such as Scratch, Lego Mindstorms, NetLogo and CS Unplugged. AP CS Principles is a new College Board course focused on seven big CS ideas and six computational practices which overlap greatly with the concepts and practices of the K-12 CS Framework described above (Kick & Trees, 2015). During the course, students complete two performance tasks: one where they investigate and report on a computing innovation and on where they create and communicate about a program related to an area they find interesting. Programming is just one component of the course and teachers are given the flexibility of selecting the programming language they want to use with their classes. AP CS A focuses on problem solving and introduces students to algorithms, data structures, and programming in the Java language (The College Board, 2014). During the course, students complete at least 20 hours of structured labs where they design, implement, and refine programming solutions. Some labs include a focus on string processing, array manipulation, and object-oriented program design. As of 2015, ECS and AP CS Principles were being used in more than 1,000 schools and AP CS A was offered in 4, 310 schools (Cuny, 2015; The College Board, 2015).

Computing in Other Disciplines. As mentioned above, another significant approach to computing education in high schools is the integration of computing into other disciplines. Some argue that training a sufficient number of teachers to meet the demand for CS as its own course is an enormous task and advocate for the inclusion of computing in other disciplines (e.g., Wilensky, Brady, & Horn, 2014). Furthermore, computational approaches are becoming integral to professional practice in STEM fields, providing another reason to integrate computing into high school courses (Orton et al., 2016). Beyond considerations of feasibility and professional preparation, the integration of computing into other courses is beneficial for learning because, in the constructionist tradition, it provides students with computational tools with which they can better build, assess, and understand models of phenomena in their primary disciplines (e.g., Blikstein & Wilensky, 2009; Wilensky, 1999a; Wilkerson-Jerde & Wilensky, 2015).

Major advocates of this approach include diSessa and Abelson (1986) who envisioned that programming would become a widespread literacy and created Boxer as an example of a programming environment that could serve as an expressive medium for laypeople and not just computer experts. They also explored this vision in their book *Turtle Geometry: The Computer as a Medium for Exploring Mathematics* (Abelson & DiSessa, 1986) where they discussed ways that Logo's turtle geometry could support the exploration of advanced mathematical topics. During the 1990s, researchers continued to build on Papert's constructionist ideas and developed environments to support youth in exploring other topics with computing (Strohecker, 1991; Wilensky, 1995, 1997b). This decade saw a rise in agent-based modeling platforms like NetLogo (Wilensky, 1999a), robotic construction kits like Programmable Bricks (Resnick, Martin, Sargent, & Silverman, 1996), and visual programming languages like AgentSheets (Repenning & Sumner, 1995) and LogoBlocks (Begel, 1996). These computing environments and their derivatives continue to support new ways of learning varied topics such as statistics, music, engineering, and electricity (e.g., Abrahamson & Wilensky, 2007; Bamberger & diSessa, 2003; Blikstein & Wilensky, 2010; Sengupta & Wilensky, 2009).

Today, many courses under this computing education approach focus on computational thinking. Popularized by computer scientist Jeanette Wing, computational thinking encompasses ways of "solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science" (2006, p. 33). Wing predicted that computational thinking would be instrumental to both innovation in every discipline and childhood education (Wing, 2008). In 2010, a committee of scholars convened to discuss the implications of computational thinking on K-12 education, which highlighted the varying perspectives that currently exist about the nature of computational thinking and its related pedagogy (National Research Council, 2011).

To aid in the expansion of computational thinking, researchers have created frameworks to guide the development of K-12 curricula. For example, Weintrop et al. (2016) developed a taxonomy of computational thinking practices in high school science and mathematics courses to guide the development of curricula and assessments. With a focus on (a) gathering, analyzing, and sharing data, (b) modeling and simulating phenomena, (c) problem solving with computational tools, and (d) systems-level thinking, the taxonomy aligns with a scientific view of CS that aims to investigate and explain phenomena through the development and refinement of computational models. Similarly, Angeli et al. (2016) developed a framework that delineates the computational thinking competencies primary school students should master related to five skills (i.e., abstraction, generalization, decomposition, algorithmic thinking, and debugging).

Within the past few years, the National Science Foundation has stimulated research in this area through its STEM+C program that seeks to “build the evidence base for effective pedagogy and pedagogical environments that will make the integration of computing within STEM disciplines more age-appropriate and contemporaneously relevant to pre-K-12 STEM education” (“STEM + Computing Partnerships (STEM+C),” 2016, p. 3). One example of computing integrated into secondary courses is the Computational Thinking in Science and Math (CT-STEM) project at Northwestern University that is developing NGSS-aligned lessons, assessments, standards, and teacher training for biology, chemistry, physics, and mathematics courses (“CT-STEM,” 2016). The CT-STEM project emerges from the constructionist tradition, the integration of computing in other disciplines, and computational thinking. One CT-STEM lesson, Wolf Vs. Sheep (Wilensky, 1997a), uses an agent-based model created in NetLogo (Wilensky, 1999a) to guide students through an exploration of population growth that can be incorporated into biology or environmental science classes. Another example is the Bootstrap curriculum that uses Common Core aligned units to help students learn algebra through video game creation (Schanzer, Fisler, Krishnamurthi, & Felleisen, 2015). Each curriculum unit focuses on a game feature (e.g., locating elements on a screen), a programming concept (e.g., expressions), and a math concept (e.g., coordinates) that students must integrate using the WeScheme programming environment to add new features to their games. Beyond the intersection of computational thinking and STEM, others have also focused on integrating computational thinking into areas such as e-textiles and game design (e.g., Fields, Searle, & Kafai, 2016; Holbert & Wilensky, 2011).

Summary. In the above sections, I summarized two approaches to computing education, both stemming largely from the pioneering work of Seymour Papert. I have not yet mentioned the role of teachers in these visions of computing education. Given that computing literacy is still not on par with other literacies like reading, many teachers vary in their knowledge and experience of how to use computers to support learning and teaching. In both approaches to computing education, many teachers are confronted with new content and disciplinary practices they need to learn to deliver their computing courses. As Solomon (1986) noted:

> Computers offer a new opportunity to help teachers to enhance their teaching and understanding of children and to keep schooling from becoming an alienating experience.

Whether or not this will happen is unclear. To make it happen, action needs to be taken now to reeducate educators to develop models of what might be possible. (p. 147)

To address this need, various professional development opportunities have been created for teachers working in CS courses (e.g., Blum & Cortina, 2007; Goode & Margolis, 2011; Gray et al., 2015; Guzdial, Ericson, Mcklin, & Engelman, 2014; “Professional Development,” n.d.) and for teachers integrating computing into other courses (e.g., Bort & Brylow, 2013; Jenkins, Jerkins, & Stenger, 2012; Yadav, Mayfield, Zhou, Hambrusch, & Korb, 2014). However, transitioning teachers do not enter these experiences as blank slates. They bring with them their ideas and habits developed from teaching in other disciplines. The question then is, how do existing teacher knowledge, experiences, and epistemic cognitions influence learning to teach CS? I address this question in the next section.

2.3.3. Transitioning to CS Teaching

In this section, I contrast CS with mathematics to identify aspects beyond content knowledge that may be new or different for mathematics teachers transitioning into CS classrooms. I focus on mathematics as a comparison discipline because all teachers who participated in the case study described in this dissertation were certified as mathematics teachers. Also, many states in the U.S. allow teachers certified in mathematics to teach CS and for CS to count as a high school graduation requirement by substituting mathematics credits (Ericson et al., 2008; Stanton et al., 2017). In the following paragraphs, I discuss how the subculture practices, epistemological beliefs, and sources of teacher efficacy in mathematics may present challenges in the transition to teaching CS.

Teaching subcultures differ in their beliefs and practices. In a comparison of teaching subcultures in mathematics, English, social studies, science, and foreign languages, Grossman and Stodolsky (1995) found mathematics teachers reported less freedom to decide on course content, more departmental coordination, a greater view of their discipline as static and unchanging, and a greater belief in grouping students by prior achievement for beneficial instruction. They also found that, like foreign language teachers, mathematics teachers viewed their courses as highly sequential (i.e., prior course learning is required for future course learning), well defined (i.e., greater consensus amongst teachers about course content), and more in need of covering all the course curriculum. Depaepe, De Corte, and Verschaffel (2016), in summarizing the literature on epistemological beliefs in mathematics education, described two epistemological paradigms dominating the discipline: an absolutist perspective that views mathematical knowledge as fixed and objective and a fallibilist perspective that views mathematical knowledge as dynamic, relativist, and the outcome of social processes. Teachers espousing more absolutist ideas tend to use transmission teaching methods and view learners as dependent on teachers for learning. Teachers espousing more fallibilist ideas tend to use inquiry-based teaching methods and view students as autonomous learners who construct their own mental representations of mathematics knowledge. Smith (1996) offered insights into the source of mathematics teachers' efficacy. School mathematics in the U.S. is dominated by *telling mathematics* where (a) instructional delivery focuses on teachers stating facts and demonstrating procedures, (b) teachers follow course sequencing as presented in textbooks, and (c) effectiveness is measured by the computational proficiency students achieve. In such an environment, teachers feel efficacious in implementing telling mathematics when they can extensively study a manageable amount of math content and have clear guidance on how to deliver that content. In other words, both the content and practices they need to use are predictable. Despite efforts to reform mathematics into a more fallibilist view where teachers guide students in exploring math concepts, teachers may continue to use traditional teaching methods because this is the type of mathematics instruction they saw in their own schooling or due to external pressures from students, teachers, and parents (Handal, 2003).

Transitioning CS teachers who bring the cultural practices and ideas of school mathematics into their new CS courses may be challenged, or liberated, by a number of differences. First, computing is a dynamic discipline. Although there are core principles to the discipline (see Table 2.4), the field continues to evolve. In the late 1980s, there were only nine core technology areas, but today there are thirty (Denning, 2003). The most recent curricular guide for undergraduate CS programs (Joint Task Force on Computing Curricula & Society, 2013), written five years after its previous incarnation, introduced three new topical areas of study to be covered in CS curricula: information assurance and security, platform-based development, and software development fundamentals. At the K-12 level, the most commonly used programming languages have changed nearly every decade from Fortran, BASIC, Logo, and Pascal in the 1970s and 1980s (Johns Hopkins University Center for Social Organization of Schools, 1983; Korotkin, Darby, Jr., & Romashko, 1970), to object-oriented programming languages like Java and C++ starting in the 1990s (Stephenson, 2000), and more recently to blocks-based programming tools like Scratch and Alice (Maloney, Resnick, Rusk, Silverman, & Eastmond, 2010; Weintrop & Wilensky, 2015). Mathematics teachers who may be accustomed to covering a predetermined knowledge base in their classes will need to become active learners in CS classrooms, updating their skills as the discipline evolves, new curricula emerge, and new programming environments are developed. This need for continual learning can lead to feelings of inefficacy (Kordaki, 2013), especially from mathematics teachers who may customarily derive their efficacy from mastery of a predetermined knowledge base.

Second, CS invites methodological pluralism. That is, many open-ended CS problems can be approached in different yet acceptable ways that arrive at accurate solutions. Looking beyond correctness, these approaches can vary along multiple attributes such as efficiency in the use of computational resources and readability (Bentley, 1982). Learners bring different styles of problem solving to CS tasks such as the concrete, bottom-up style and the logical, abstract, top-down style (Turkle & Papert, 1992). When given the same programming task, students follow different pathways, some more productive than others, to reach their goal (Blikstein et al., 2014). Enumerating all the possible approaches and pathways learners might follow is an onerous undertaking. Instead, CS teachers need to anticipate common approaches and prepare pedagogical tools to support the unexpected and alternative solutions that students might provide (Hazzan et al., 2015). For mathematics teachers accustomed to problems with determinant answers that can be solved using a fixed set of procedures, the plurality of CS problem-solving might feel unsettling and require educators to develop comfort with some level of uncertainty in their teaching.

Third, CS teachers are often the only CS teacher in their schools and may not work within CS departments (Century et al., 2013; CSTA, 2015; Ni & Guzdial, 2012). This limits the number of colleagues available to discuss issues related to CS teaching. Mathematics teachers who may be used to coordinating frequently with other mathematics teachers in their building may need to search for professional learning communities beyond their campuses. Within CS, these communities exist through professional networks like the CSTA, virtual communities (e.g., Cooper, Grover, & Simon, 2014; Morrison, Ni, & Guzdial, 2012), and groups formed at in-person professional development workshops (e.g., Ryoo et al., 2015).

Fourth, compared to other core disciplines, CS is a relatively new subject that has only been in American schools since the 1950s (e.g., Buchman, 1956). CS is not universally accessible across American schools and students who gain exposure to CS do so through a variety of opportunities including formal classes, self-teaching, online activities, and extracurricular groups (Google Inc. & Gallup Inc., 2016b, 2016a), each giving students a different knowledge base and skill set that teachers need to attend to in their classes. Many transitioning teachers probably do not have extensive prior experiences with CS in a school context or notions of how it is typically taught. Also, many transitioning teachers have limited CS content knowledge (Ericson et al., 2008). In contrast, all mathematics teachers have experienced some version of school mathematics as learners, have notions of how it can be taught, and have been exposed to relevant content knowledge. CS may appear hard or seem unknown for these teachers because, as Ben-Ari (1998) suggests, many novices have no conceptual models of how computers operate or computing upon which to develop new knowledge. They may, for example, not know how to organize a computer lab for effective instruction or have useful metaphors from everyday life to draw upon in explaining concepts. With a limited understanding of the discipline, they may underestimate the value of a course like AP CS Principles because it does not fit stereotypical models of CS courses that have focused heavily on programming.

In sum, mathematics teachers transitioning into CS are facing two new arenas, the discipline of CS and the discipline of school CS. Although mathematics teachers may benefit from knowledge of a discipline closely related to CS, the subculture of school mathematics may not translate directly to school CS. The prior knowledge, social supports, and curricular tools from their mathematics teaching may not be useful or available in their CS classrooms. Thus, in addition to learning new content, mathematics teachers transitioning into CS may also need to learn new ways of teaching that align with the nature of computing.

2.4 Conceptual Frameworks

This dissertation study was designed to describe the evolution of teacher knowledge and practices used by experienced mathematics teachers transitioning into computer science classrooms with the support of tech industry professionals over the course of one school year. Unlike most other studies that focus on pre-service teachers or expert teachers, I focus on educators in the middle of the teaching experience spectrum to better understand how PCK develops. I created two conceptual frameworks to guide this study: a framework of CS PCK development and a framework of CS PCK. I used an abductive approach to develop these frameworks drawing first on pilot data gathered during the 2014-2015 school year and then revising these frameworks based on the literature reviewed above.

Although models of CS PCK and CS PCK development were presented earlier in this chapter, I decided not to use them because of (a) distinctions I make between content knowledge and PCK, (b) distinctions I make between knowledge and knowledge development, and (c) my focus on transitioning teachers instead of expert teachers. Baxter's (1987) model includes aspects of content knowledge, what she calls knowledge of relationships among domains, knowledge of substantive structure of discipline (i.e., concepts and theories of a subject such as homeostasis in biology), and knowledge of syntactic structure of discipline (i.e., ways new knowledge is acquired in a field such as inquiry processes in biology). While she argued that content knowledge and PCK are integrated in experienced teachers, I focus on transitioning teachers where I believe these knowledge domains are more distinct. The KUI model (Bender et al., 2015) contains four components I did not include: curricular knowledge, issues of the educational system, teacher-related issues, and process dimensions. While I do not deny the importance of knowledge about curricula and issues of the educational system, I consider these knowledge bases separate from PCK. I also include a professional development component in my PCK development model since I focus on describing how CS PCK develops within a particular teacher learning context.

My frameworks do, however, share similarities with the existing frameworks of CS PCK and CS PCK development. Like the framework produced by the KUI group (see Figure 2.3), my CS PCK development framework includes components related to content dimensions, process dimensions, teachers' beliefs, and teacher efficacy. From Baxter's model (1987), I include the idea that the relationship between content knowledge and PCK differ based on a teacher's experience level. Like Lapidot (see Figure 2.2), I also include the cyclical nature of the teacher learning process.

2.4.1. CS PCK Development Framework

The results of this literature review were used to refine a conceptual framework of CS PCK development that I created based on pilot data gathered during the 2014-2015 school year (see Figure 2.8). The framework presents a model of the cyclic mechanisms by which teachers trained in particular subject matters (e.g., mathematics) but new to teaching computer science develop CS PCK through classroom experiences as part of an in situ professional development program that includes support from volunteer content experts (i.e., tech industry professionals outside the education field). The progression presented in this framework resembles generative change, where teachers:

> learn to talk to their students about their thinking, puzzle about what the responses tell them about students' understanding, decide how to use this knowledge in planning instruction and interacting with students, and figure out how to learn more about the students' thinking. (Franke, Carpenter, Levi, & Fennema, 2001, p. 656)

Transitioning CS teachers are confronted constantly with unfamiliar content and new student understandings in their classrooms. By attending to these experiences, making sense of them, and incorporating them into their lessons, their learning becomes simultaneous with their teaching and allows for continued growth (A. Ball, 2009; Franke, Carpenter, Fennema, Ansell, & Behrend, 1998). The framework consists of seven components and the processes between them, each of which is described below.

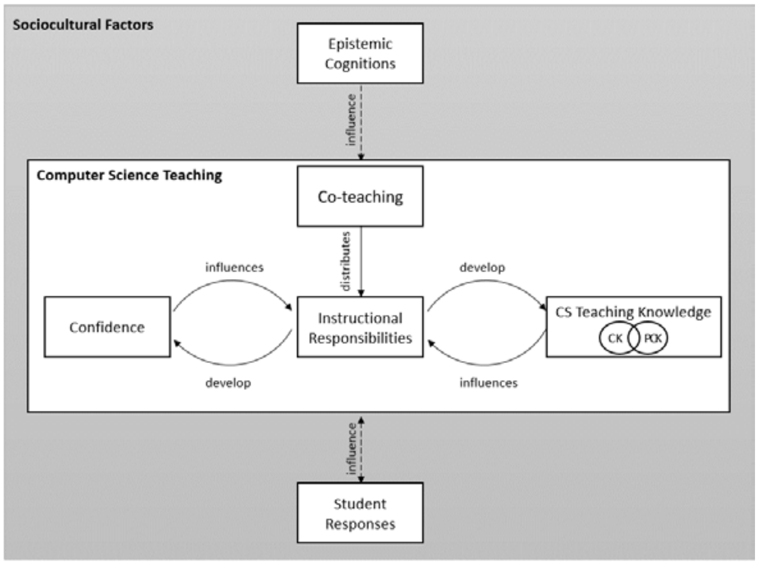


Figure 2.8. Theoretical framework of CS PCK development.

Co-teaching. This component captures the specific professional development model studied in this dissertation, which is described in chapter 3. In a co-teaching approach, transitioning CS teachers collaborate with volunteer content experts to deliver computing courses. Other models of CS PCK development might replace this component with different learning activities (e.g., professional learning communities, summer workshops, pedagogical repositories), and, depending on their intended outcomes, link directly to other components in the model such as CS teaching knowledge. While the specific co-teaching model studied here intends for transitioning CS teachers to increase their teaching knowledge and confidence, I argue that instructional responsibilities requiring subject matter knowledge (e.g., creating unit assessments) as opposed to general pedagogical tasks (e.g., classroom management) serve as the primary mechanism guiding this development.

Instructional responsibilities. This component describes the practices instructors partake in during their teaching. Examples of teaching practices include managing the classroom, presenting a lesson, and evaluating student work. Instructional responsibilities relate to core practices focused on in mathematics education research (Grossman et al., 2009) and resembles the fields of pedagogical operation identified in the KUI's model of CS PCK (Hubwieser, Magenheim, et al., 2013). I label this component as instructional responsibilities to highlight that practices are divided between transitioning CS teachers and their volunteers, and that each person might be responsible for a subset of all teaching practices used in the classroom. This distinction is captured by the arrow labeled distributes which extends from the co-teaching component to the instructional responsibilities component.

CS teaching knowledge. While scholars have identified many areas of teaching knowledge, I focus on content knowledge and PCK. Content knowledge describes one's understanding of subject matter (e.g., how to use loops in the Java programming language). PCK describes one's understanding of instruction and student understanding (e.g., representations of loops and common difficulties students have in creating loops). Shulman (1986) originally described PCK as a subset of content knowledge, while Ball and colleagues (2008) distinguished different types of content knowledge specific to teaching. Regardless of how content knowledge and PCK are delineated, it is clear from prior research that the two knowledge areas are tightly interwoven. Baxter (1987) and Liederman et al.'s (2012)'s work with experienced CS teachers suggest the relationship between content knowledge and PCK differs based on amount of teaching experience, with greater experience leading to more integrated content knowledge and PCK. I convey this developmental view of content knowledge and PCK with two concentric circles that might be separate in novice teachers and completing overlapping in the most experienced teachers. The specific categories of teacher knowledge will be discussed below in the CS PCK framework.

Instructional Responsibilities to CS Teaching Knowledge. The arrows in the framework suggest that an individual's teaching knowledge will influence which instructional responsibilities they assume and how they enact them. Also, as teachers gain experience enacting their instructional responsibilities, they will develop greater teaching knowledge. Simply enacting responsibilities is insufficient for knowledge growth; teachers must also make sense of the feedback they receive from students when performing their teaching tasks (Franke et al., 1998). But not all teachers respond generatively when faced with unexpected or new experiences and they may need additional supports (e.g., collaboration with teachers or researchers, examining artifacts of practice outside of class) to stimulate their reflection and change their practices (Baumfield, 2006). Lastly, it can be difficult initially for teachers trained in more traditionalist, teacher-centered educational approaches to assume a stance of a learner who gains knowledge from enacting instructional responsibilities (A. Ball, 2009). This is captured by the *epistemic cognitions* component of the framework, which is described below.

Confidence. Related to Bandura's theories of self-efficacy (1977), this component captures a teacher's belief in his or her ability to accomplish professional duties. As summarized above, a lack of confidence can prevent teachers from learning and tends to be lower in teachers with weaker content knowledge (Goldsmith et al., 2014; Mizzi, 2013; Ross et al., 1999; Swackhamer et al., 2009). A lack of confidence can also lead teachers to avoid teaching, rely on prepared instructional materials, and minimize teacher-student discourse (Harlen & Holroyd, 1997; Schneider & Plasman, 2011). Within CSER, Ni and colleagues (Morrison et al., 2012; Ni, 2009; Ni & Guzdial, 2012) have found that lack of confidence prevents some teachers from implementing contextualized computing curricula. The arrows in the framework suggest that an individual's confidence will influence which instructional responsibilities they assume and how they enact them. Also, as teachers gain experience enacting their instructional responsibilities, they will develop greater confidence in their teaching abilities.

Epistemic cognitions. Epistemic cognitions encompass ideas about the nature and acquisition of knowledge. These ideas can depend on a teacher's particular working context and career stage, with certain ideas encouraging PCK development more than others (Hashweh, 1996, 2013; Luft & Roehrig, 2007; Postholm, 2012). In my theoretical framework, I focus on epistemic cognitions about how learners construct CS knowledge (i.e., their sources and justification of knowledge; Chinn et al., 2011) and teachers' aims in helping students acquire CS knowledge (Buehl & Fives, 2016). This component is distinct from the CS teaching block because I assume that transitioning CS teachers will begin their CS assignments with existing ideas about teaching and student learning based on teaching experiences in other subjects. For example, a teacher trained in mathematics might begin her CS teaching assignment with the belief that learners acquire knowledge through the transmission of information and that her goal is to help students acquire procedural fluency. These existing beliefs will likely influence how teachers decide to implement their co-teaching partnership, conduct their instructional responsibilities, and learn from their experiences.

Student responses. Berliner once commented that "good teaching is judged through reliance on standards applied to the tasks of teaching and related to norms for professional behavior, including moral considerations. Successful teaching is about whether intended learnings were achieved” (Berliner, 2001, p. 468). The student responses component relates to successful teaching emerging from the CS teaching box. One would expect that as educators improve their CS teaching, their students will benefit from improved outcomes. At the same time, students serve as a source of PCK development for teachers (Baxter, 1987). Their homework solutions and in-class questions provide teachers with examples of student understanding. Their reactions to course projects and lessons give teachers feedback on the effectiveness of their instruction, which might lead them to reinforce productive practices and refine less effective ones.

Sociocultural factors. Teacher cognition is not detached from the sociocultural contexts within which teachers work (Avalos, 2011; Opfer & Pedder, 2011). Boards of education decide who can be a CS teacher, which computing courses are taught, and whether these courses are offered as electives or required classes. School cultures may be more or less supportive of encouraging professional growth amongst staff. Participation in a professional learning community and access to professional training can support teacher development, but there may be disparities in access to these opportunities. The field of CS influences what technologies are commonly used and taught in secondary classrooms. Students bring their own interests and prior backgrounds into computing courses that educators must respond to in their teaching. All of these factors can influence opportunities, motivation, and mechanisms for teacher learning.

2.4.2. CS PCK Framework

The second framework I developed describes the PCK central to computer science teaching for transitioning teachers (see Figure 2.9). This framework, which closely resembles Shulman’s (1986) original definition of PCK, shares similarities with the models presented by

Baxter (1987) and the KUI group (Bender et al., 2015) as it relates to knowledge of student understanding, representations, and teaching strategies. The model contains three components: knowledge of student understanding, knowledge of student interest and motivation, and knowledge of content and teaching. In my descriptions of these components, I also briefly summarize related literature on the specific knowledge computing teachers need to draw upon in their work.

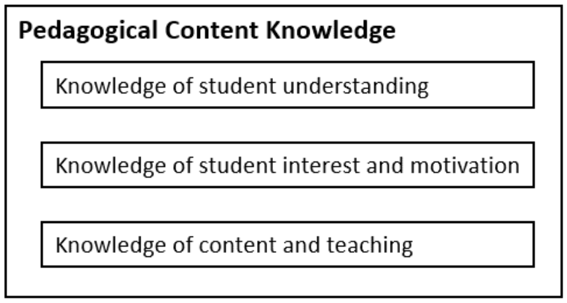

Pedagogical Content Knowledge
Knowledge of student understanding
Knowledge of student interest and motivation
Knowledge of content and teaching

Figure 2.9. Theoretical framework of CS PCK.

Knowledge of student understanding. Borrowing from Shulman, this component relates to a teacher's "understanding of what makes the learning of specific topics easy or difficult: the conceptions and preconceptions that students of different ages and backgrounds bring with them to the learning of those most frequently taught topics and lessons" (Shulman, 1986, p. 9). This component also includes an understanding of students' common problem-solving strategies. While CS is not just programming, research on novice programmers provides insights into common misconceptions, difficulties, and problem-solving approaches of students new to CS.

Studies of novice programmers date back to the 1970s. For example, Miller (1974) asked non-programmers to arrange a set of commands into a program that would accomplish a name sorting task and found participants struggled with statements involving logical disjunction, negatively expressed statements, and debugging. Sorva (2012) synthesized this literature into a list of 150 misconceptions and identified the following topics that were often cited in research on student difficulties: variables, assignment, references and pointers, classes, objects, constructors, and recursion. Other difficulties center around assembling programs from subcomponents, conditionals, looping constructs, and tracing code (Guzdial, 2004; Kaczmarczyk, Petrick, East, & Herman, 2010). Another area of research has explored student difficulties with planning and implementing solutions (Ginat, 2008; Hanks & Brandt, 2009; Rist, 2004). When planning problem solving approaches, novices tend to pick one solution and stick with it, make hasty decisions, use a bottom-up approach, and ignore disconfirming evidence. When implementing their solutions, novices tend to jump quickly to coding, write out all their code before executing it, do not adequately test their solutions, and patch errors instead of finding complete solutions.

Beyond these specific misconceptions and less effective approaches, researchers have also investigated the underlying causes of student difficulties. Clancy (2004) identified six sources of novice misconceptions which he argued resulted from students transferring existing knowledge into their nascent understanding of programming. These sources include inconsistencies between English and programming terminology (e.g., *while* means continuous testing in English but one test per iteration in programming), inconsistencies between mathematical notation and programming syntax (e.g., = represents equality in mathematics but assignment in programming languages), prior experience with other programming languages, overgeneralizing examples, modifying correct rules, and misapplying pedagogic metaphors. Similarly, du Boulay (1986) described five areas of difficulty that novices confront when learning to program: recognizing what problems programming can solve, modelling how computers execute commands, learning the syntax and semantics of programming languages, acquiring standard templates for common tasks, and planning, implementing, and testing programs. He also identified interactions (e.g., how components of a program work together) as a source of confusion for learners. Rist (2004) suggested that novice programming behavior can be explained by memory overload and limited design patterns upon which to judge their solutions. Lastly, Pea (1986) presented three conceptual programming bugs based on studies of elementary and secondary students using the Logo and BASIC programming languages: parallelism bug (assuming different lines of code can be executed simultaneously), intentionality bug (attributing foresightedness to the program), and egocentrism bug (assuming there is more meaning behind their code). As Spohrer and Soloway observed, "instruction can be improved when educators gain a better understanding of what students do and do not know … instructors should strive to familiarize themselves with specific high-frequency bugs, and to learn as much as possible about the origins of all bugs" (1986, p. 632).

Knowledge of student interest and motivation. This component relates to a teacher's understanding of what students find engaging in computer science, what dissuades them, and how those factors can be accounted for in teaching practices. As Ball et al. noted, "when choosing an example, teachers need to predict what students will find interesting and motivating…each of these tasks requires an interaction between specific mathematical understanding and familiarity with students and their mathematical thinking" (2008, p. 9). This knowledge component is particularly important for teachers today as the field attempts to create inclusive learning environments where more students feel welcomed in CS courses. Research related to this component has focused on engaging topics, contextual and individual influences, and student aversion to CS.

Social factors play a role in encouraging students studying computer science. For some students, family support, friends in CS, and relationships with mentors increase persistence in introductory programming courses (Barker, McDowell, & Kalahar, 2009; Katz, Allbritton, Aronis, Wilson, & Soffa, 2006). Students are also motivated by work they find relevant to their own experiences and that allows them to express who they are (Margolis, Goode, Chapman, & Ryoo, 2014). At the same time, many students are dissuaded from persisting in computing courses for various reasons. Negative perceptions of CS as isolating, constant interaction with computers, and solely programming can repulse students (Biggers, Brauer, & Yilmaz, 2008; Carter, 2006). Other factors such as intense workloads, falling behind in class, and a lack of prior programming experience can lead students to withdraw from CS (Kinnunen & Malmi, 2006; Petersen, Craig, Campbell, & Tafliovich, 2016).

Researchers and educators have tried various methods to attract and retain students in computing. Embedding topics such as music, media computation, video games, e-textiles, and robotics into CS have enthused many students (e.g., Basawapatna, Koh, & Repenning, 2010; Buechley, Eisenberg, & Elumeze, 2007; Freeman et al., 2014; Guzdial, 2003; Petre & Price, 2004). Culturally relevant computing approaches also engage youth by bridging their computing experiences with their cultural identities (e.g., Eglash, Bennett, O'donnell, Jennings, & Cintorino, 2006; Hooper, 1996; Pinkard, Erete, Martin, & Royston, 2017; Scott & White, 2013; Searle & Kafai, 2015). Researchers caution against relying exclusively on contextualized approaches that appeal to the interests and backgrounds of students. Frieze (2007) argued that learning environments balanced in terms of gender, student personalities, and support provided to students – instead of environments that target specific groups – encourages a broader range of students in successfully participating in CS courses. Guzdial (2010) suggested that while contextualized approaches can increase retention, students who eventually take decontextualized CS courses will be able to apply their knowledge to a wider variety of applications.

Knowledge of content and teaching. Ball et al. (2008) included this component in their model of Mathematical Knowledge for Teaching. Knowledge of content and teaching concerns teacher understanding of appropriate representations of CS ideas, methods of presenting those ideas to students, and sequencing content for effective instrument. In their guide to creating a methods course for prospective secondary CS teachers, Hazzan, Lapidot, and Ragonis (2015) described various pedagogical tools teachers can use in their practice to introduce and reinforce CS concepts (see Table 2.6).

Computer labs are another integral component of CS instruction where teachers need to make decisions regarding organization and which programming and visualization tools to incorporate (Hazzan et al., 2015). Pair programming is a method of collaboration for computer labs where one partner controls the computer while the other partner constantly reviews and comments on the work. Studies of pair programming provide some evidence that students using pair programming write better programs, ask instructors less questions, have greater satisfaction and confidence, and are more likely to complete their courses than students who work independently (McDowell, Werner, Bullock, & Fernald, 2003; L. A. Williams & Kessler, 2001). Integrated development environments (IDE) are tools that facilitate program development providing users with supports such as debuggers, autocompletion, and compilers. Many IDEs have been developed for novices to address issues that make programming difficult to learn. Kelleher and Pausch (2005) categorized nearly 80 IDEs designed to alleviate common novice

Table 2.6

Pedagogical Tools for Computer Science (Hazzan et al., 2015)

Tool	Definition	Example
Pedagogical games	Social games that aim at teaching computer science ideas, which can be played either with or without computers	A version of bingo where board cells contain conditional statements whose execution depends on the announced value
CS-Unplugged	Concepts are presented through engaging activities and puzzles (e.g., using cards, crayons, active playing)	Introduce sorting algorithms by having students use a scale to order a set of objects of unknown weights
Rich tasks	Programming exercises that can be solved in a variety of ways and promote discussion about major CS ideas	Write a method that checks whether a given date is valid. The method should check whether three given integers (a day, a month, and a year) can represent a valid date in the twenty-first century. Assume that each month has 30 days.
Concept maps	A graphical tool for representing concepts and the connections between them	Construct a concept map which represents all topics covered in the course so far
Classification of objects and phenomena from life	Classify a set of images from everyday life related to a computing concept	Present students with a set of pictures that they classify according to their own criteria. During a share out period, teachers introduce terminology related to the underlying computing concept.
Metaphors	Analogies between CS topics and ideas familiar to students	Students create posters to explain the concept of variable through metaphors, they discuss the advantages and disadvantages of each poster

problems such as mastering the mechanics of writing programs or knowing where to begin and what is possible to create through programming. Similarly, Guzdial (2004) provided a genealogical grouping of several environments stemming from Logo, Smalltalk-72, and more traditional languages, where relevance, applicability, and immediate feedback are incorporated into their designs to sustain engagement with programming. These tools minimize common programming obstacles and make it possible for youth to create their own computational artifacts.

Problem solving is an integral component of CS with which novice learners often struggle (Robins, Rountree, & Rountree, 2003). Through a content analysis of textbooks, worksheets, and exams, Ragonis and Shilo (2013) identified nine categories of problem solving questions used in CS: address or define criteria, argue and justify, analyze, compare, complete, convert, discover, develop, and integrate. These categories of problem solving are reflected in a set of question types identified by Hazzan et al. (2015) for use in CS teaching, which can be presented as pure algorithmic tasks that focus on computing structures, narrative algorithmic tasks that are embedded within a context, or selected response items with a limited number of answer choices (see Table 2.7).

While these instructional tools have shown promise as effective teaching methods, there are times when it may not be appropriate to use them or when they need to be modified for specific contexts. With pair programming, for example, some students will resist collaborative work and others will leave all the tasks to their partners (L. Williams, 2007). Regarding programming environments designed specifically for students, it is still unclear how useful they are in helping students transition to high-level languages like Python or Java (Armoni, Meerbaum-Salant, & Ben-Ari, 2015; Powers, Ecott, & Hirshfield, 2007). Also, some students find these environments inauthentic, less powerful, and difficult to manage compared to high-level languages (Weintrop & Wilensky, 2015), which may be demotivating. Lastly, many people report CS Unplugged activities to be engaging, but some studies suggest these activities need to be modified to better meet learning goals and link more directly with core CS topics (e.g., Rodriguez, Rader, & Camp, 2016; Taub, Ben-Ari, & Armoni, 2009). Teachers need to be aware of the strengths and limitations of these instructional devices and how they might need to be adapted for their particular students.

Table 2.7

Computer Science Question Types (Hazzan et al., 2015)

Type	Definition
Develop a solution	An open problem to which learners develop their own solutions (e.g., algorithm, pseudo-code, a program script)
Develop a solution using a given module	A problem to which learners develop a solution while making use of a given module
Trace a given solution	Learners are presented with code and trace the execution of that code
Analyze code execution	Learners review a program and analyze aspects of its execution
Find the purpose of a given solution	Learners are presented with a solution to an unknown problem and asked to identify what problem it solves
Examine the correctness of a given solution	Learners review a solution and determine if it solves a given problem correctly
Complete a given solution	A given problem and an incomplete solution are given, students complete the missing instructions so that the final code solves the problem correctly
Manipulate instructions	A given problem and its solution are given, students address different manipulations on the solution
Estimate efficiency	Learners estimate the efficiency (i.e., the order of magnitude of runtime) of a given solution
Design a question	Learners create their own questions
Examine programming style	Learners examine the programming style of different solutions to the same task
Transform a solution	Learners transform a given solution into a different programming approach, language, or paradigm

2.4.3. Research Questions

Drawing on the literature reviewed in this chapter and the two conceptual frameworks presented above, I focused on the following research questions in this dissertation:

1. What knowledge of computer science content, student thinking, and instructional strategies do teachers develop?
2. What teaching tasks (i.e., instructional responsibilities) do teachers undertake when planning and implementing their CS lessons?
3. How does teaching knowledge support the implementation of teaching tasks?

CHAPTER 3. METHODS

In this chapter I describe the procedures I used to examine PCK in computer science teachers participating in a co-teaching, on-the-job professional development program. I begin with a discussion of the methodology underpinning my work. Then I describe the professional development program explored in this study, the study participants, and the contexts within which the participants worked. The second half of the chapter is devoted to the data sources, procedures, and analyses used in the study. At the end of the chapter, I provide an overview of my background to highlight the perspective I brought to this study based on my personal, educational, and professional experiences.

3.1 Methodology

Creswell (2008) suggests researchers explicate the beliefs underpinning their work so that audiences understand their selection of methods and analytic approaches. After years of schooling in both computing and the humanities, training in an interdisciplinary graduate program, and professional experience with a multifaceted research organization, I have adopted a pragmatic worldview that foregrounds practical applications over philosophical distinctions and embraces a pluralistic approach for understanding phenomena. Morgan (2007) distinguishes pragmatism from other worldviews in its use of (a) abductive reasoning that cycles between induction and deduction, (b) intersubjectivity that acknowledges researchers assume different frames of reference during their work, and (c) transferability to determine how findings can be applied to some settings beyond the study context. Below I explain how a pragmatic worldview informed the research design and analysis of this study.

This study, and the larger body of research within which it is situated, explores knowledge for teaching computer science at the secondary level. Teacher knowledge is not a new topic to the education research community and so the CSPCK team at WestEd drew on existing theoretical frameworks to inform decisions during our study planning phase. We selected Shulman's PCK framework (1986) as a guide given its influence on the study of teacher knowledge over the past thirty years. The wealth of existing research on PCK in other domains provided us with a starting point for forming questions, selecting methods, and thinking about analysis. However, the limited amount of research on PCK within computer science, particularly at the secondary level, gave rise to the need for exploratory approaches to understand the nuances of teacher knowledge within this understudied domain. Our team decided on a mixed method approach to address the question of how do experienced educators learn to teach computer science.

The quantitative arm of our work (i.e., the main study) consisted of survey and assessment data collected from teachers, volunteers, and students at the beginning and end of the school year. We used four instruments: background questionnaires focused on the professional experiences of teachers and volunteers and their feelings of readiness to teach CS; a survey to measure students' attitudes towards CS; content assessments for teachers and students; and a CS PCK assessment for teachers and volunteers. The qualitative arm of our work, which is the focus of this dissertation, consisted of a collective case study with six TEALS teams located in the San Francisco Bay region of the United States.

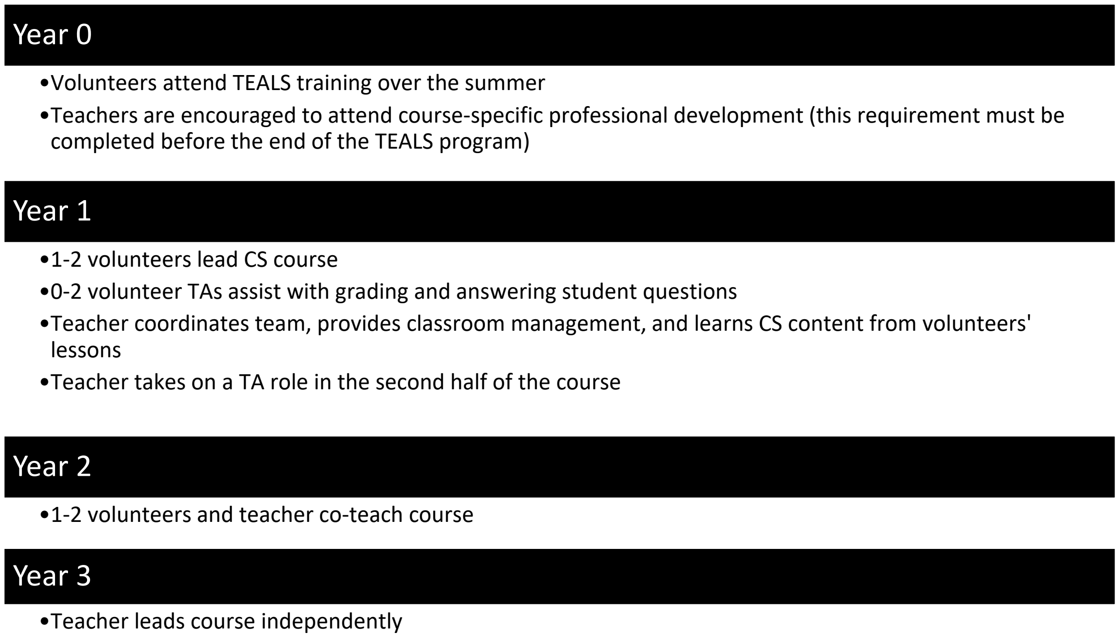
A collective case study involves selecting multiple cases with both redundant and varying characteristics and examining them jointly to understand an issue (Stake, 2000). Multiple case studies support theory building better than a single case because cross-case comparison makes it easier to distinguish idiosyncratic findings from significant patterns (Eisenhardt & Graebner, 2007). Our rationale for selecting a case study approach was to gather information on how teachers developed CS PCK during the school year, explain the mechanisms behind the results of the main study, and identify emergent themes related to PCK development. Stake also notes that case study work involves the triangulation of data to “reduce the likelihood of misinterpretation…also to clarify meaning by identifying different ways the case is being seen” (2000, pp. 453–4). For this reason, I incorporated a combination of self-reported data, observations, and performance measures into the case study design. I also supplemented the case study data with assessment and questionnaire data gathered as part of the main study. While the type of methods I selected for the case study remained consistent during the study year, the instruments underwent multiple rounds of revisions. This iterative development process allowed me to incorporate early stage findings into the instruments and to address methodological gaps that became apparent once the study began (Eisenhardt, 1989). Instrument changes resulting from this iterative process and their impact on study analysis will be discussed below. Lastly, I began analysis using a deductive approach where I attempted to understand the collected data through the lens of Shulman’s PCK framework (1986) and its derivatives. However, I found the deductive approach somewhat limiting because there was little prior research specific to CS PCK to which I could compare my results. So, I also employed inductive techniques to generate patterns in the data specific to CS content.

Lastly, it is worth noting that a case study, like other approaches, is not without its limits. While the case study strategy provides a way to build theory about an understudied phenomenon, it can be inundated by large amounts of data and the peculiarities of individual cases (Eisenhardt, 1989). However, as Stake commented, “the purpose of a case report is not to represent the world, but to represent the case…the utility of case research to practitioners and policy makers is in its extension of experience” (2000, p. 460). Furthermore, case studies are useful for identifying important constructs and less effective at identifying their relative importance (Eisenhardt & Graebner, 2007). So, by using a case study approach, this work focused on identifying and understanding the constructs important to CS PCK development. Future studies employing other methods will be needed to generate results and theory that are generalizable to the larger community of transitioning CS teachers.

3.2 Professional Development Program

Case study participants were enrolled in a professional development program offered by TEALS, an organization focused on increasing the number of computing courses offered in U.S. high schools. TEALS began in 2009 when a Microsoft employee volunteered to teach computer science at a high school in Washington and since then has experienced rapid growth. At the start of this dissertation, 7,000 students at 131 schools across 19 states were enrolled in TEALS courses. In the TEALS PD model, high school teachers and volunteers from the tech industry collaboratively teach computing courses. The year prior to starting in the program, potential volunteers and interested schools undergo a rigorous recruitment process to ensure their commitment and preparedness for the program. Volunteers are required to have a CS degree or equivalent industry experience and participate in interviews with representatives from TEALS and partner high schools. Similarly, schools interested in participating in the program complete applications and attend interviews including a district administrator, a school administrator, and the classroom teacher. Selected schools sign formal agreements detailing the responsibilities of all parties involved in the program and designating a teacher for the TEALS course. The summer prior to their courses, teachers and their volunteers communicate to discuss their plan for the school year. Teachers are required to participate in a summer professional development program appropriate for their course; these programs are not offered by TEALS. Volunteers attend a 40-hour, hybrid summer training offered by TEALS. TEALS employs regional managers who mentor teaching teams, occasionally visit classrooms, and organize local meetings for participants. TEALS also hosts a private online forum where teachers and volunteers can interact with other participants. Lastly, school administrators commit to conducting multiple classroom observations during the school year and providing teaching teams with feedback. Most courses are offered during the first period of the day, before volunteers begin their regular work day.

During the 2015-2016 school year, TEALS offered three models in their PD program: teaching assistant, consulting support, and co-teaching. The teaching assistant model was designed for experienced CS teachers. In this model, one to two volunteers provide support with grading and assisting students every class session. The consulting support model was also designed for experienced CS teachers. In this model, one to two volunteers are available to support teachers as needed and may visit the classroom monthly. The co-teaching model (see Figure 3.1) was designed for teachers new to computer science. In this model, three to four volunteers participate in delivering instruction, grading, and assisting students every class session; volunteers usually alternate days with each volunteer attending class half the week. This model begins with volunteers assuming the bulk of instructional responsibilities while teachers focus on learning course content. As teachers gain experience and confidence with course materials, responsibilities shift to where teachers assume most of the instructional responsibilities. At the end of the co-teaching model, high school teachers lead courses independently. In this case study, I focus on teachers who used the co-teaching model.



Year 0

- Volunteers attend TEALS training over the summer
- Teachers are encouraged to attend course-specific professional development (this requirement must be completed before the end of the TEALS program)

Year 1

- 1-2 volunteers lead CS course
- 0-2 volunteer TAs assist with grading and answering student questions
- Teacher coordinates team, provides classroom management, and learns CS content from volunteers' lessons
- Teacher takes on a TA role in the second half of the course

Year 2

- 1-2 volunteers and teacher co-teach course

Year 3

- Teacher leads course independently

Figure 3.1. TEALS co-teaching model.

Two computing courses are offered through the TEALS program. The semester-long *Introduction to Computer Science Principles* (Intro) course is based on the University of California at Berkeley's largely successfully CS10 course for non-CS majors called *The Beauty and Joy of Computing* (BJC). The course introduces some of the big ideas of computing, discusses the history and future of the field, and teaches students programming with the block language Snap!. The year-long *AP Computer Science A* (AP) course is based on the University of Washington's CSE 142 course for CS majors. AP CS A introduces students to the Java programming language. Neither course requires prior programming experience. As a young organization, the TEALS program evolves each year. Starting in the 2015-16 school year, TEALS provided instructional teams with complete curriculum packages for the Intro and AP courses containing detailed weekly lesson plans. They also supplemented the Intro course with a second semester introduction to the Python programming language. This additional semester was included to support teachers who needed to extend their Intro course through the entire school year. Some Intro teams, having already developed a plan for their second semester, continued using their own curricula (e.g., an introduction to HTML or an introduction to Java).

TEALS differs from other PD programs because their model is integrated into teachers' classes and involves content experts outside of the education world. Some might judge this model as less than effective because (a) learning in the moment provides teachers with little time for the reflective activities that support teacher knowledge development and (b) the collaboration of pedagogical experts (the teachers) with content experts (the volunteers) does not guarantee teachers will automatically obtain PCK specific to CS. Is it worth studying PCK development in teachers if they are participating in an ineffective PD program? Next, I compare the TEALS model against models of effective PD to highlight the strengths and weaknesses of this program.

Desimone (2009) identified the following five features of teacher learning activities that are associated with change in teacher knowledge, practice, and to some extent student achievement: content focus, active learning, coherence, duration, and collective participation. First, effective teacher learning activities focus on both subject matter content and how students learn that content. Teachers in the TEALS program are immersed in the subject matter content of their classes while delivering lessons and assisting students. Their exposure to how students learn that content is variable. This can depend on the summer PD they attend and the ways in which they support students in their courses. Second, effective teacher learning also involves active learning activities as opposed to passive activities like lectures. TEALS provides multiple opportunities for teachers to learn, for example, by observing the lessons delivered by volunteers, discussing student progress in their teaching teams, or evaluating student work. However, how teachers benefit from these opportunities can depend on factors such as teacher-volunteer dynamics and teachers' feelings of readiness. Another feature of effective teacher learning activities is coherence with teachers' knowledge and beliefs. While such coherence will depend on individual characteristics, TEALS encourages schools to identify teachers for the program who believe in the co-teaching PD model and are committed to leading the course independently after two years. Another feature of effective teacher learning activities is sufficient duration, which the literature suggests includes activities spread over a semester and 20 hours or more of contact time. The TEALS program is embedded in teachers' courses which are either a semester or a year in length and last well over 20 hours. Lastly, collective participation where teachers can work with colleagues in their school is a characteristic of effective teacher learning activities. Many CS teachers are the only CS teachers in their schools, so meeting this criterion is difficult for CS PD programs across the nation. TEALS does not explicitly arrange for collective participation amongst teachers within the same school. However, TEALS fosters collective participation amongst teachers and their volunteers as they work together to deliver a course, amongst teachers in the same locale who attend optional meetings organized by regional managers, and amongst all teachers and volunteers who elect to participate in the online TEALS teaching forum. While these methods of collective participation are not identical to in-school partnerships, they do provide teachers multiple venues to discuss their teaching with other educators who are participating in the same PD program.

In creating a model that addresses the urgent need for CS courses, TEALS may have compromised efficacy for immediacy. A comparison against Desimone’s (2009) five characteristics of effective PD suggests that the program is of sufficient duration and supports collective participation that can lead to teacher learning and change in practice. However, the way in which TEALS incorporates a focus on subject matter content, active learning tasks, and coherence with teacher knowledge and beliefs might lead to differing outcomes based on how individual teachers and volunteers implement their co-teaching partnerships.

3.3 Participants and Context

Recruitment for this case study began in fall 2014 and ran parallel to recruitment for the main study. I opted for an opportunistic, theoretical sampling approach for this study. Selecting among the first participants to consent allowed me to maximize the number of case study visits conducted during the 2014-2015 school year and provided more time to build rapport with participants. I also considered course assignments and school location when selecting potential participants so that a range of experiences and school contexts would be represented in the study. Seven teachers joined the case study during the 2014-2015 school year. Two of these teachers withdrew from the case study at the end of the school year, so I recruited an additional teacher to participate in the 2015-2016 school year. Five returning participants and one new participant took part in the case study presented here. Teachers received a stipend of $1,000 for each year of participation in the case study.

Also, while this dissertation does not focus explicitly on the impact of context on teaching, it is important to note that the contextual factors within which educators work can shape their PCK. As Berliner notes, "context has to be thought of as a third variable and probably of equal status with talent and practice in the debate over important influences in the development of accomplished, exemplary, or expert teachers" (2001, p. 466). In the following paragraphs, I provide a brief description of each teacher’s background, their stage in the TEALS program, and the context (i.e., locale and schools) within which they worked during the study timeframe.

3.3.1. Participating Teachers

Mr. Edwards. Mr. Edwards, a fourth-year TEALS participant, taught the AP course. His teaching team comprised two volunteers who also worked with him the previous two years. His volunteers participated in the course intermittently throughout the study year. Mr. Edwards had more than 25 years of teaching experience and was certified to teach mathematics and art. During the study, he also taught digital arts and animation courses that involved some programming. In the 1990s, he taught AP CS A when it focused on the Pascal programming language. He did not provide information about previous careers, age, or ethnicity.

Ms. Jones. Ms. Jones, a third-year TEALS participant, taught multiple sections of a two-part course. During the first semester, she taught the Intro curriculum, and during the second semester she taught an introduction to Java programming. Her teaching team comprised four volunteers who worked with her the two prior years. One of her volunteers was previously a teaching assistant at the University of California at Berkeley for the BJC course upon which the TEALS Intro curriculum is based. Ms. Jones had 13 years of teaching experience and was certified to teach mathematics. During the study, she also taught trigonometry. Her professional experience priors to teaching involved a career in the film and entertainment industry. She identified as female and Asian and she was in her early 40s.

Ms. King. Ms. King, a second-year TEALS participant, taught three sections of the AP course. Her teaching team comprised two male volunteers who also worked with her during the 2014-2015 school year. She had 11 years of teaching experience and was certified to teach mathematics. During the study, she also taught support mathematics. Towards the end of the study year, Ms. King was appointed the first computer science department chair at her high school. Before TEALS, she taught an introductory Java course based on an Oracle Academy curriculum. Her professional experiences prior to teaching involved a career in user interface design. She identified as female and Caucasian and she was in her late 50s.

Mr. Miller. Mr. Miller, a second-year TEALS participant, taught two sections of the Intro course. He taught the first section collaboratively with volunteers and the second section independently. His teaching team comprised two volunteers, one of whom he worked with the previous year. He had 38 years of teaching experience and was certified to teach mathematics, life sciences, and French. During the study, he also taught algebra. Mr. Miller mentioned no careers outside of teaching. He identified as male and Caucasian and he was in his early 60s.

Mr. Perez. Mr. Perez, a second-year TEALS participant, taught one section of the Intro course. During the 2014-2015 school year, he taught the AP curriculum. He joined the case study during the 2015-2016 school year. His teaching team comprised three volunteers who were only active during the first semester. He had two years of teaching experience and was certified to teach mathematics. During the study, he also taught algebra. Prior to becoming a high school teacher, Mr. Perez had experience tutoring college students in computer science. He identified as male and Caucasian and he was in his mid-20s.

Ms. Robinson. Ms. Robinson, a second-year TEALS participant, taught one section of the AP course. Her teaching team comprised four volunteers who also worked with her during the 2014-2015 school year. She had 11 years of teaching experience and was certified to teach mathematics. During the study, she also taught geometry and an introductory computing course based on the curriculum used by other teachers in her school district. Ms. Robinson’s professional experiences prior to teaching involved multiple roles in the tech industry including web designer, quality insurance engineer, and software engineer. She identified as female and Latina and she was in her mid-40s.

3.3.2. Professional Development Stages

Although all study participants employed the co-teaching model, each participant implemented their co-teaching partnerships differently. These differences correspond to three stages in the TEALS co-teaching model. Ms. Robinson, was in the *volunteer-led* stage. She relied heavily on volunteers to lead her AP course during the first term and she began transitioning into a lead role on her teaching team during the second term. Two teachers, Ms. Jones and Mr. Miller, were in the *collaborative* stage. They worked with volunteers to plan, implement, and revise their courses throughout the year. The remaining three teachers, Ms. King, Mr. Edwards, and Mr. Perez, were in the *teacher-led* stage for the entire school year. They directed their courses receiving support from volunteers to assist students, grade assignments, and explain topics unfamiliar to the teachers. The volunteers working with Mr. Edwards and Mr.

Perez gradually withdrew from their responsibilities during the school year. I use these three PD stages (see Table 3.1) to frame some of the study results.

Table 3.1

Professional Development Stage

	Volunteer-led	Collaborative		Teacher-led		
Teacher	Ms. Robinson	Ms. Jones	Mr. Miller	Ms. King	Mr. Edwards	Mr. Perez
TEALS Course	AP	Intro	Intro	AP	AP	Intro
Year in TEALS	2	3	2	2	4	2
Volunteers	4	4	2	2	2	2

3.3.3. Locale

This dissertation was conducted in the San Francisco Bay area in the western United States, specifically in the counties of San Francisco, San Mateo, and Santa Clara. Demographic and economic profiles of the three counties represented in the study are provided in Figure 3.2 and Figure 3.3. This region is wealthier than most other regions of the United States with a population that is largely Caucasian, Asian, and Hispanic. This region is also home to Silicon Valley, the preeminent information technology hub in the world where many high-tech corporations and start-up companies are housed.

Proximity to the world's preeminent technology hub may influence transitioning CS teachers' development differently than educators in other regions. For example, living in a community where news about companies like Facebook and Google are broadcasted on the evening news may make teachers more aware of the relationship between the content of their courses and professional computing. As another example, due to an expansive tech community in this region, volunteers who joined the TEALS program in the San Francisco Bay area may bring different content knowledge and experiences to the participants' classrooms than volunteers in other regions of the country.

County	San Francisco	San Mateo	Santa Clara
Median Household Income	$75, 000	$88, 000	$91, 000
25 years or older w/Bachelor's degree or higher	52%	44%	47%
Language other than English spoken at home	45%	46%	51%

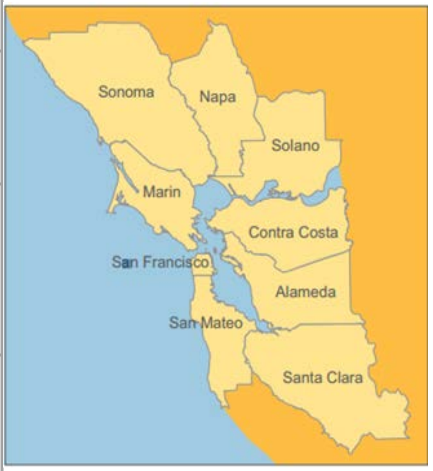


Figure 3.2. San Francisco Bay Area counties represented in case studies. Sources: U.S. Census Bureau *(2015)*, Bay Area Council Economic Institute *(2012)*.

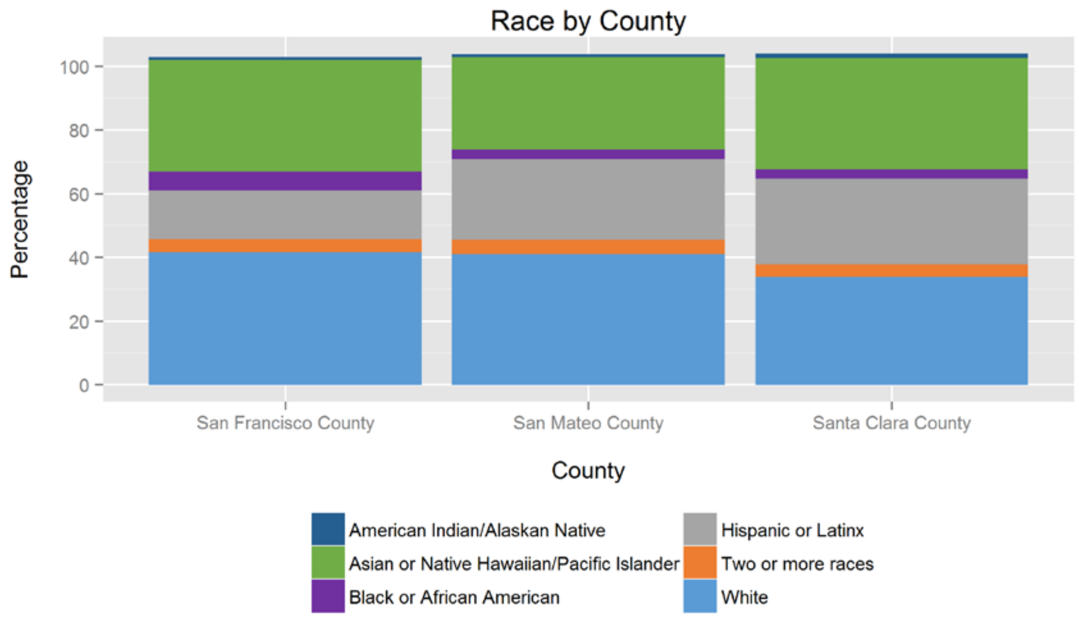


Figure 3.3. Race within San Francisco Bay counties represented in the case study. Totals exceed 100% because Hispanics may be of any race and are also included in applicable race categories. Asian includes Asian, Native Hawaiian, and Pacific Islander. Source: U.S. Census Bureau *(2015)*

3.3.4. Schools

Each teacher worked at a different school and there were two schools within each of the counties listed above. The demographic profiles of each school differed from the demographic profiles of the counties within which the schools resided. At both schools within San Francisco county, students identifying as Asian represented more than 55% of the student population. In both San Mateo county and Santa Clara county, the largest ethnic group at one school identified as Caucasian and the largest ethnic group at the second school identified as Hispanic. The percentage of students receiving free or reduced lunch and per-pupil expenditures also varied. The percent of students receiving free or reduced lunch ranged from about 15% to 65% (“Common Core of Data,” 2016) and per-pupil expenditures ranged from about $8,000 to $13,000. As a point of comparison, the average per-pupil expenditure in the U.S. in 2013 was $11,841 (K. Park, Hurt, Fisher, & Rost, 2016). Table 3.2 summarizes school information by teacher.

Table 3.2

School Profiles by Teacher

Teacher	Mr. Miller	Mr. Perez	Ms. King	Mr. Edwards	Ms. Jones	Ms. Robinson
County	San Francisco	San Francisco	San Mateo	San Mateo	Santa Clara	Santa Clara
Free/reduced lunch eligible	65%	60%	17%	49%	16%	54%
Per-pupil expenditure	$9, 000	$9, 000	$10, 000	$10, 000	$13, 000	$8, 000

3.4 Data Collection Procedures

This case study focuses on the PCK and instructional responsibilities of teachers transitioning into CS classrooms. Selecting methods to explore PCK requires attending to the nature of teacher knowledge. Hashweh (2005) described teacher knowledge as units that are private and personal, content-specific, event-based and story-based, and developed through repeated experiences of planning and teaching particular topics. Kagan (1990) noted that teacher knowledge is sometimes stored in metaphors, can reside at an unconscious level, and may not be describable by teachers. Ball (1988) commented on the challenges of inferring teaching knowledge from teacher action because the ability to perform tasks does not guarantee complete or explicit understanding of those tasks nor does the inability to explain solutions always imply a lack of understanding. Given the often tacit and multifarious nature of teacher knowledge, researchers attempting to elicit PCK need to employ multiple methods and ask teachers to articulate their knowledge (Baxter & Lederman, 1999).

Questionnaires, interviews, and observations were the primary data sources used in this study. These data sources were designed to elicit experiences related to the planning, enactment, and reflection of teachers' lessons. Observations were included given limitations with using self-reported data of teaching practices identified in the literature (e.g., differences in how teachers and researchers understand practices). Towards the end of the study, questionnaires were introduced to address two gaps identified in the primary data sources. First, the data collected from teachers related to the specific content covered during observed lessons. Towards the end of the study, there was no one topic that all teachers covered during the visits. A questionnaire focused on a common topic of which I expected all participants to have some degree of knowledge and experience was added to allow for more comparison across teachers. Second, beliefs about teaching and learning continued to surface in discussions with teachers when they explained their rationale for certain actions. Since this topic was not explicitly included in the study instruments, I included a short questionnaire about epistemological beliefs. Lastly, a background questionnaire and content assessments used in the main study were also included.

This study spanned an entire school year, beginning in September 2015 and ending in June 2016. Data gathering centered around *visits,* or a collection of activities related to one classroom lesson. Visits were centered around particular lessons so that participants could draw on their recent experiences when responding to questionnaires and interviews. Visits were spaced approximately a month apart so that I could capture data about participants teaching different topics. Three visits were conducted during the first semester and another three visits were conducted during the second semester. Towards the end of the second semester, one visit was conducted in each teacher's main subject area to provide a comparison against their CS teaching. Table 3.3 provides a description of the activities involved in a case study visit.

Table 3.3

Case study visit activities

Visit Activity	Description
Scheduling	Observer and teacher arrange a date for the observation.
Pre-lesson questionnaire	Teacher completes pre-lesson questionnaire online.
Interview selection	Observer and researcher review pre-lesson questionnaire responses to identify which interview to use.
Classroom observation	Observer attends one class session and completes observation protocol. Observations lasted from 50 minutes to 90 minutes.
Interviews	Observer interviews teacher in person directly after the observed lesson. If the teacher is unavailable directly after the lesson, the observer interviews the teacher either in person or on the phone later in the day or on the following day. Interviews are audio recorded.
Post-lesson questionnaire	Teacher completes post-lesson questionnaire online.
Visit write-up	Observer compiles all the data gathered from the visit and completes a post-visit form.
Review of visit write-up	Researcher reviews the observer's write-up and works with observer to clarify any issues.

Study data was gathered from eleven different data sources, which will be described later in more detail. Some data sources were collected across the entire year, others were used either in the first semester or the second semester, and the rest were administered at either the very beginning or very end of the school year (see Figure 3.5). The pre-lesson questionnaire, post-lesson questionnaire, main study questionnaire, and main study assessment were administered through online survey tools. Most interviews were conducted in person at each participant's school directly after an observation. When teachers were not able to meet with observers directly after their lessons, they were interviewed later either at the participant's school or on the phone. All interviews were audio recorded. Teachers received the PCK questionnaire and teaching beliefs questionnaire over email and returned their responses over email. Lastly, observation protocols were completed either on paper or on a computer, depending on the observer's preference.

	2015				2016					
Data Source	Sept.	Oct.	Nov.	Dec.	Jan.	Feb.	Mar.	Apr.	May	Jun.
Observation Protocol I	▨	▨	▨	▨						
Observation Protocol II					▨	▨	▨	▨	▨	
Pre-lesson Questionnaire	▨	▨	▨	▨	▨	▨	▨	▨	▨	
Post-lesson Questionnaire	▨	▨	▨	▨	▨	▨	▨	▨	▨	
PCK Questionnaire									▨	▨
Beliefs Questionnaire									▨	▨
Lesson Reflection Interview	▨	▨	▨	▨	▨	▨	▨	▨	▨	
Think-aloud Interview	▨	▨	▨	▨						
CoRe Reflection Interview					▨	▨	▨	▨	▨	
Main study questionnaire	▨									
Main study assessment								▨	▨	

Figure 3.5. Data source collection timeline.

Three observers assisted me in collecting data for the case study and each observer was responsible for collecting data from one or two participants. I was responsible for collecting data from Ms. King and Mr. Edwards. To help me understand the data gathered from the other four participants, I joined the observers for one visit with Ms. Robinson, Ms. Jones, Mr. Miller, and Mr. Perez. All three observers were also involved in data collection during the 2014-2015 school year and were familiar with the case study methods, data sources, and participants when this dissertation study began.

Although I created a preferred procedure for the case study tasks, there were some deviations to the procedure. First, three teachers did not complete all activities. Mr. Edwards did not complete two post-lesson questionnaires, the PCK questionnaire, the beliefs questionnaire, or the main study tasks. I gathered some of this missing information during interviews with Mr. Edwards. Ms. Robinson did not complete one post-lesson questionnaire or the observation and interviews for her math visit. Mr. Perez did not complete the pre-lesson questionnaires, post-lesson questionnaires, or most of the interviews for his last CS visit and his math visit. He also did not complete the PCK questionnaire or the beliefs questionnaire. While he did complete both main study tasks, he was mistakenly given the content assessment for the AP course instead of the Intro course. Second, I tested the CoRe reflection interview with Ms. King and Ms. Robinson during the first semester which replaced their think-aloud interviews. So, these teachers only completed two think-aloud interviews while all other teachers completed three. Figure 3.6 summarizes the data collected from each participant.

Data Source	Mr. Edwards	Ms. Jones	Ms. King	Mr. Miller	Mr. Perez	Ms. Robinson
Observations	7	7	7	7	7	6
Pre-lesson Questionnaires	7	7	7	7	5	7
Post-lesson Questionnaires	5	7	7	7	5	6
PCK Questionnaire	No	Yes	Yes	Yes	No	Yes
Beliefs Questionnaire	No	Yes	Yes	Yes	No	Yes
Lesson Reflection Interviews	7	7	7	7	6	6
Think-aloud Interviews	3	3	2	3	3	2
CoRe Reflection Interviews	4	4	5	4	2	4
Main study questionnaire	No	Yes	Yes	Yes	Yes	Yes
Main study assessment	No	Yes	Yes	Yes	Yes	Yes

Figure 3.6. Data accounting sheet. Cells highlighted in red indicate missing data.

3.5 Data Sources

3.5.1. Pre-lesson Questionnaire

A seventeen-item pre-lesson questionnaire was created to probe teachers about their feelings of comfort, feelings of preparedness, and teaching knowledge for an upcoming lesson. Most questionnaire items were drawn from Loughran, Mulhall, and Berry's (2004) interview tool named CoRe (Content Representation) which prompts small groups of teachers to discuss PCK related to big ideas in a domain (e.g., ecosystems in science). The CoRe was developed over the course of two years through interviews and observations with more than 50 high school science teachers. A subset of CoRe items is listed here:

- What you intend students to learn about this idea
- Why it is important for students to know this
- Difficulties/limitations connected with teaching this idea
- Teaching procedures (and particular reasons for using these to engage with this idea)

Since I converted the interview prompts into a questionnaire format, I modified the CoRe in three ways for the present study. First, prompts were rephrased as questions given that teachers would read the items alone without someone present to explain the prompts. Second, the focus of the questionnaire shifted from a "big idea" selected by participants to the learning objective for a given class period. Given the various goals of classroom lessons (e.g., learn new content, complete an assignment, research a topic), I replaced the CoRe prompt about assessing student learning with 'how will you know if this lesson is a success'. And third, teachers completed the pre-questionnaire individually and not in groups, which might have influenced the level and type of detail provided in their responses. I also included four additional items on the questionnaire. Two close-ended items asked teachers about their level of comfort and preparedness for their current units. To gauge teacher's perceptions of CS content, another close-ended item asked about the difficulty of the given learning objective compared to other topics in the course. Lastly, given the role of programming environments in the TEALS courses, an item was included asking about the technologies used in the given lesson. This questionnaire also gathered details about lessons that helped in planning observer visits. The complete pre-lesson questionnaire is included in Appendix A.

3.5.2. Post-lesson Questionnaire

A fourteen-item post-lesson questionnaire was created to gather teachers' reflections on their observed lesson and details about their co-teaching. Items focused on methods teachers used to prepare for class and instructional resources used during the lesson. Most items were drawn from the Horizon Inside the Classroom Interview Protocol(Weiss, Pasley, Smith, Banilower, & Heck, 2003). This protocol was developed as part of the Looking Inside the Classroom study that captured portraits of science and mathematics instruction across the U.S from 364 teachers across 31 elementary, middle, and high schools. Given that teachers can gain PCK through informal exchanges with fellow teachers (Desimone, 2009), I also included an item asking teachers for advice they would offer to someone teaching the topic the first time. The second half of the questionnaire asked about the responsibilities undertaken by the teacher and volunteers and the impact of co-teaching on classroom instruction. The post-lesson questionnaire is included in Appendix B.

3.5.3. Lesson Reflection Interview

The lesson reflection interview consisted of two main prompts, one focused on the lesson and one focused on the co-teaching model (see Appendix C). Each main prompt also contained probing questions. Several probing questions related to the lesson prompt were adapted from protocols used by Niess, Lee, Sadri, and Suharwoto (2006) to interview mathematics teachers about the development of their TPCK after attending a summer professional development course. Probing questions focused on how the lesson supported students, unexpected occurrences, and revisions for future implementations of the lesson. Three probing questions related to the co-teaching model were also included. These items focused on the quality of the co-teaching experience, the roles assumed by teachers and volunteers, and the effectiveness of the model in preparing teachers to lead the lesson independently.

3.5.4. Think-aloud Interview

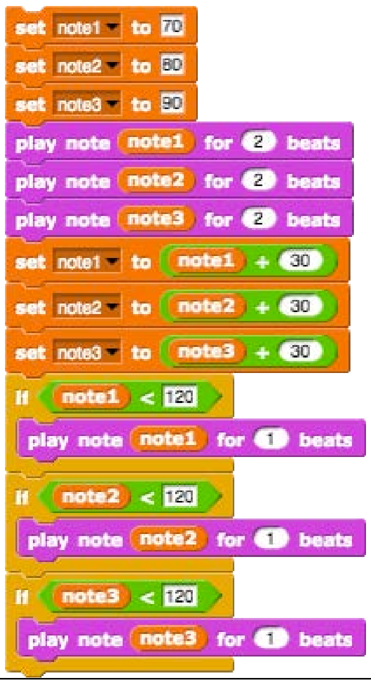
Ten think-aloud interviews were created to provide teachers with scenarios that might evoke memories of recent pedagogical experiences and drive discussion about their PCK. The development of the interviews consisted of: (1) identifying types of prompts to include, (2) aligning prompts to curricula, and (3) designing prompts to reflect situations encountered in classrooms. Details on each step are described below.

Three types of prompts were selected for the interviews: assessment, student work, and instructional materials. *Assessment prompts* asked participants to review three items and decide which should be included on an exam. This prompt type was borrowed from Davis (2004), who used a similar technique to explore science teachers' content knowledge. *Student work prompts* asked participants to review student solutions to a programming problem. This prompt type was modified from an activity created by Hazzan, Lapidot, and Ragnois' (2011) to support CS educators in examining programming tasks that can be solved in a variety of ways. *Instructional materials prompts* asked participants to watch and critique a video that presented an explanation of a CS topic. After each prompt, teachers were asked to discuss how their own students might respond to the items. The interview prompts are presented in Appendices D, E, and F.

After selecting prompt types, I focused on aligning interviews with topics covered in the AP and Intro courses. I reviewed TEALS curricular guides and identified one topic for each unit to address in the interviews. To coordinate interviews with recent classroom experiences, teachers were asked to identify the topic of their lessons a few days before a case study visit and observers selected the prompt whose topic most closely related the focus of the lesson.

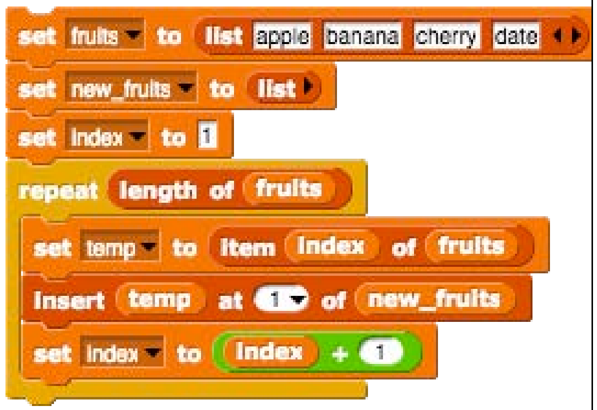
I also focused on designing tasks that might simulate teachers' everyday experiences. While designing assessment prompts, I drew on a classification of twelve question types used in CS teaching (Hazzan et al., 2011) to vary the types of items presented in each interview. For example, the assessment prompt shown in Figure 3.7 includes items that ask students to transform a solution (item 1), trace a solution (item 2), and complete a given solution (item 3). For student work prompts, I incorporated items reflecting common misconceptions. For example, the prompt shown in Figure 3.8 reflects common difficulties students have with loops including confusion with nested loops (solution 1), assuming the entire for loop statement is executed before the for loop body (solution 2), and confusing comparison operators (Götschi, Sanders, & Galpin, 2003). Lastly, I reviewed recommended course textbooks, AP CS A practice exam guides, a repository of assessments created by prior TEALS participants, and assessment instruments developed by a project advisor. Some interview items were borrowed and modified from these materials while I created others.

Item 1
Rewrite the following script using one list to represent the notes:



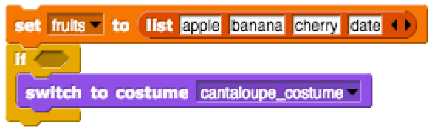
```
set note1 to 70
set note2 to 80
set note3 to 90
play note (note1) for 2 beats
play note (note2) for 2 beats
play note (note3) for 2 beats
set note1 to (note1 + 30)
set note2 to (note2 + 30)
set note3 to (note3 + 30)
if <note1 < 120>
    play note (note1) for 1 beats
if <note2 < 120>
    play note (note2) for 1 beats
if <note3 < 120>
    play note (note3) for 1 beats
```

Item 2
Create a tracing table to show the value of fruits, new_fruits, and index after each block is called in the following script:



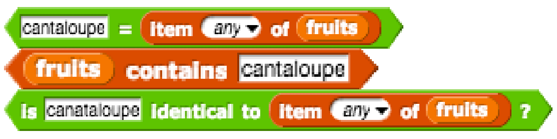
```
set fruits to (list apple banana cherry date)
set new_fruits to (list)
set index to 1
repeat (length of fruits)
    set temp to (item (index) of fruits)
    insert (temp) at 1 of (new_fruits)
    set index to (index + 1)
```

Item 3
Your friend is trying to write a script that checks if the word ‘cantaloupe’ exists in a list of fruits, and, if so, changes the sprite’s costume.



```
set fruits to (list apple banana cherry date)
if < >
    switch to costume cantaloupe_costume
```

Which of the following blocks could your friend use in the if block to make the script run correctly? Select all the blocks that could work.



a) `cantaloupe = (item any of fruits)`
b) `(fruits) contains cantaloupe`
c) `is canataloupe identical to (item any of fruits) ?`

Figure 3.7. Think-aloud assessment prompt.

The following code simulates a counter that goes from 0 0 to 1 9. Rewrite this code using for loops.

```
int num1 = 0;
int num2 = 0;

System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);

num1 = 0;
num2 = num2 + 1;

System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
```

Solution 1

```
for (int num2 = 0; num2 <= 1; num2++){
        System.out.println(num2);
}
for (int num1 = 0; num1 <= 9; num1++){
        System.out.println(" " + num1);
}
```

Solution 2

```
for (int num2 = 0; num2 <= 1; num2++){
  num2 = num2 - 1;
  for (int num1 = 0; num1 <= 9; num1++){
    num1 = num1 - 1;
    System.out.println(num2 + " " + num1);
  }
}
```

Solution 3

```
for (int num2 = 0; num2 < 1; num2++){
  for (int num1 = 0; num1 < 9; num1++){
    System.out.println(num2 + " " + num1);
  }
}
```

Figure 3.8. Think-aloud student work prompt.

3.5.5. CoRe Reflection Interview

At the end of the first semester, I reviewed the pre-lesson questionnaire data and noticed variation in the level of detail provided in teachers' responses. Some participants provided clear and detailed responses while others occasionally provided terse or ambiguous comments. Since most teachers had been vocal in their interviews, I decided to explore their pre-lesson questionnaire responses through interviews during the second semester. Starting in the second semester, teachers still completed the pre-lesson questionnaire and I reviewed their responses to see which items might require further explanation. During the interview, teachers were reminded of their responses to the selected items and asked to provide more details about their comments.

The CoRe reflection interviews replaced the think-aloud interviews during the second semester. One potential drawback to this change is that I did not capture think-aloud interview data when some participants were starting to assume more instructional responsibility in their classrooms and when, presumably, course content became more complex. These changes in context may have influenced PCK development differently than first semester experiences and resulted in different outcomes during the think-aloud interviews.

3.5.6. Observation Protocol I

During first semester classroom visits, observers made jottings to describe the type of activities happening (i.e., direct instruction, student work time, assessment, and exam preparation), the responsibilities undertaken by the teacher and volunteers (i.e., leading class and assisting students), and the teaching practices used (i.e., interpret student productions, demonstrate solutions, describe approaches to solve problems, provide justification for solutions, justify the importance of a topic, make use of metaphor or stories, and assess whole class learning). At the end of the observations, observers completed a form indicating (a) the presence of each activity, instructional responsibility, and teaching practice, (b) the amount of class time each item occurred, and (c) whether the item was performed by the teacher or a volunteer. The observation recording sheet (see Appendix G) was adapted from Goss, Powers, and Hauk's Case Study Teaching Observations: Recording Sheet (2013) and Park and Oliver's PCK Evidence Reporting Table (2008).

3.5.7. Observation Protocol II

The observation protocol was modified at the end of the first semester to better reflect the activities of computer science classrooms and to address issues related to capturing activities in the moment (see Appendix H). First, activity type and instructor responsibilities were reorganized into two categories: instructional activities and classroom activities. Instructional activities focused on teacher-led activities and included direct instruction, initiate-response-evaluate sequences, instructional conversations, and non-content related activities. Classroom activities focused on individual student work time, student group work time, assessment, lab time at computers, and AP exam preparation. Second, due to difficulties in determining teaching discourse moves during live observations, the teaching practices section of the protocol was changed from a focus on *what* instructors were saying to *who* was speaking, to *whom*, and with which *discourse moves*. These changes were inspired by research on PCK in science education (e.g., Carlsen, 1987; Schneider & Plasman, 2011) that has shown a relationship between the amount of conversational control teachers give to students in class and the teacher's confidence and content knowledge. Three new categories were used on this version of the protocol: voices in the room (i.e., student, teacher, volunteer), interactions (i.e., between students, between student and teacher, between student and volunteer, and between teacher and volunteer), and classroom discourse (i.e., pose questions, respond to questions, provide explanations). Lastly, to increase the accuracy of the observation reports, observers completed the protocol form for each six-minute classroom segment, instead of once at the end of the observation. Changes in the observation protocol restricted the comparisons I could make across semesters to two types of instructional activities (i.e., teacher-led activities and student-instructor interactions).

3.5.8. PCK Questionnaire

Evidence gathered from the aforementioned data sources centered around specific lessons, whose topic and format varied based on teacher availability, course, and time of school year. The variability in observed lessons provided an authentic view into teachers' daily practice but complicated a comparison across cases. To facilitate more cross-case analysis, I asked participants to complete the same task designed to elicit knowledge about student understanding and teaching strategies for specific topics (see Appendix I). The task contained three prompts:

1. List the difficulties students have with linear data structures (i.e., arrays and lists in AP, lists in Intro). For each difficulty: provide a description of the difficulty, indicate the frequency with which you see the difficulty when teaching, and describe how you address the difficulty. List as many difficulties as you can.
2. List all of the types of lessons and activities you would plan around linear data structures. For each lesson/activity: provide a description of the lesson/activity, describe your rationale for using the lesson/activity, and describe all the ways you help students who have difficulty understanding the lesson/activity.

3. What are the most challenging topics covered in your course? For each topic, do you address these topics differently than other topics? If so, how?

Linear data structures were selected as a focus for this task given their difficulty for novice programmers (Robins et al., 2003), their prolonged appearance in both the Intro and AP curricula, and their inclusion in multiple case study visits. The last item asks teachers to identify the most challenging topics in their courses. Topics might be conceptually difficult for students to understand because of their abstractness or because of their interconnectedness with multiple other topics. As teachers increase their CS teaching knowledge, they should be able to better distinguish the more difficult course topics from the less difficult topics.

3.5.9. Teaching Beliefs Questionnaire

During both the 2014-2015 and 2015-2016 school years, teachers participating in the study occasionally made comments reflecting their ideas about the value of learning CS, the effectiveness of various activities in attracting more diverse students to their courses, and the impact of course curricula on student motivation. For example, two AP teachers made the following comments at the end of the 2014-15 school year:

> So, teaching to the AP is entirely different than the idea that I want the kids to explore and have fun and just come away with a love for computer science...So, what I need to do unfortunately is give them this kind of [practice AP] test with every chapter so that they're getting the rigor as we go, as opposed to all at the end. (Ms. King, 4/29/2015)

> I think what we really need to do at the high school level is get everybody exposed to at least getting enough experience with coding that they really have a better sense of what it's

> about, and doing fun activities that are going to get those kids, and the more artsy kids, or the kids who would never choose to take a computer programing class. That's why I think the animation is a great place to bring them both together. (Mr. Edwards, 4/10/2015)

Beliefs about teaching and learning are important to consider when examining teaching knowledge because beliefs can influence instructional decisions (Fang, 1996). While not a primary focus of this dissertation, teacher beliefs were examined to identify potential relationships with teaching knowledge.

A subset of items from the Teacher Beliefs Interview (TBI; Luft & Roehrig, 2007) were administered to participating teachers. The TBI consists of seven items, four focused on beliefs about learning and three focused on the acquisition of knowledge. The TBI was developed through a three-phase, iterative process of creating items, testing them with science teachers, and revising items so that they better elicited responses that were “highly personalized, often constructed in episodic ways, and contained affective and evaluative components” (Luft & Roehrig, 2007, p. 42). Over one hundred pre-service, induction, and experienced science teachers were interviewed during the creation of the TBI. Reliability and validity of the final TBI were established through interviewing teachers in another domain (i.e., math) and calculating the internal consistency of the items ($\alpha = .70$).

The TBI items used in this dissertation were:

1. How do you maximize student learning in your classroom?
2. How do you students learn computer science best?
3. How do you describe your role as a teacher?
4. In the school setting, how do you decide what to teach and what not to teach?

The first two items relate to beliefs about student learning and the last two items relate to beliefs about the acquisition of knowledge. The second item was rephrased to use the term *computer science* instead of *science.*

Three TBI items were excluded from this questionnaire. The teaching beliefs questionnaire along with the PCK questionnaire were not included in the teachers' original list of study tasks. Furthermore, these tasks were introduced at the end of the study year, which is a hectic time for teachers dealing with their courses and final exams. In order to encourage greater completion of these two tasks, only half of the TBI items were included.

3.5.10. Main Study Tasks

Teachers participating in the case study were also participating in the main study. As part of the main study, teachers completed a background questionnaire asking about their educational and professional background and readiness to teach CS. Teachers and their students also completed an assessment of the content covered in their courses. Results from these instruments were used to supplement data gathered from the case study data sources.

3.6 Data Reduction

Data gathered for this study were not all received in a form that facilitates analysis. Interview data and open-ended questionnaire responses went through a process of data reduction, which Miles and Huberman define as "the process of selecting, focusing, simplifying, abstracting, and transforming the data" (1994, p. 10). Data reduction occurs throughout the entire study process and decisions made about data reduction influence what conclusions can be drawn. Here I explain the data reduction processes applied to the case study data and how those processes bound the study conclusions.

3.6.1. Interview Transcripts and Unitization

Participants partook in three types of semi-structured interviews for this study: lesson reflection interviews, think-aloud interviews, and CoRe reflection interviews. The interview data were transformed to allow for exploration of the knowledge and practices of individual teachers and for comparison across teachers. First, audio recordings were transcribed by a member of the WestEd study team. Transcriptions can range from a naturalized[1] style that captures details of how interlocutors talk (e.g., recording overlapping speech, response tokens such as mm hm, or non-verbal vocalizations such as gesturing) to a denaturalized style that focuses on the substance of a speaker's discourse (Oliver, Serovich, & Mason, 2005). I opted for a denaturalized style as I found it more important to identify, for example, participants' ideas around strategies to teach students algorithms than to explore the meaning behind their choice of discourse moves.

Second, I reviewed each transcript to resolve phrases the transcriber could not decipher and to begin unitizing the interviews. After exploring different methods of unitization, I settled on the approach of dividing transcripts into chunks of texts that responded to prompts listed on the interview protocols. I decided on this approach because it made it easier to compare responses across participants, to summarize which interview prompts each participant received, and to identify the frequency of questions asked that were not on the protocol. In dividing the transcripts using this approach, a unit might contain multiple questions. For example, consider the following excerpt I had with Ms. King:

[1] Other scholars use the terms naturalized and denaturalized to mean the opposite of Oliver, Serovich, and Mason's (2005) definition. For example, Bucholtz (2000) refers to transcription style that that resembles written text as *naturalized* and the transcription style that reflects oral speech as *denaturalized*.

[1] Interviewer: And what did you expect students to walk away with at the end of the lesson?

[2] Ms. King: I expected that they would then be able to do this worksheet and understand how do you - which covered all those methods, um, without too much trouble. That they could figure out what I really didn't know. As opposed to if I hadn't done that little mini lesson I would have been - hands would have been up all over the class.

[3] Interviewer: And so when you were walking around to give it a check off, it seemed like they were getting it?

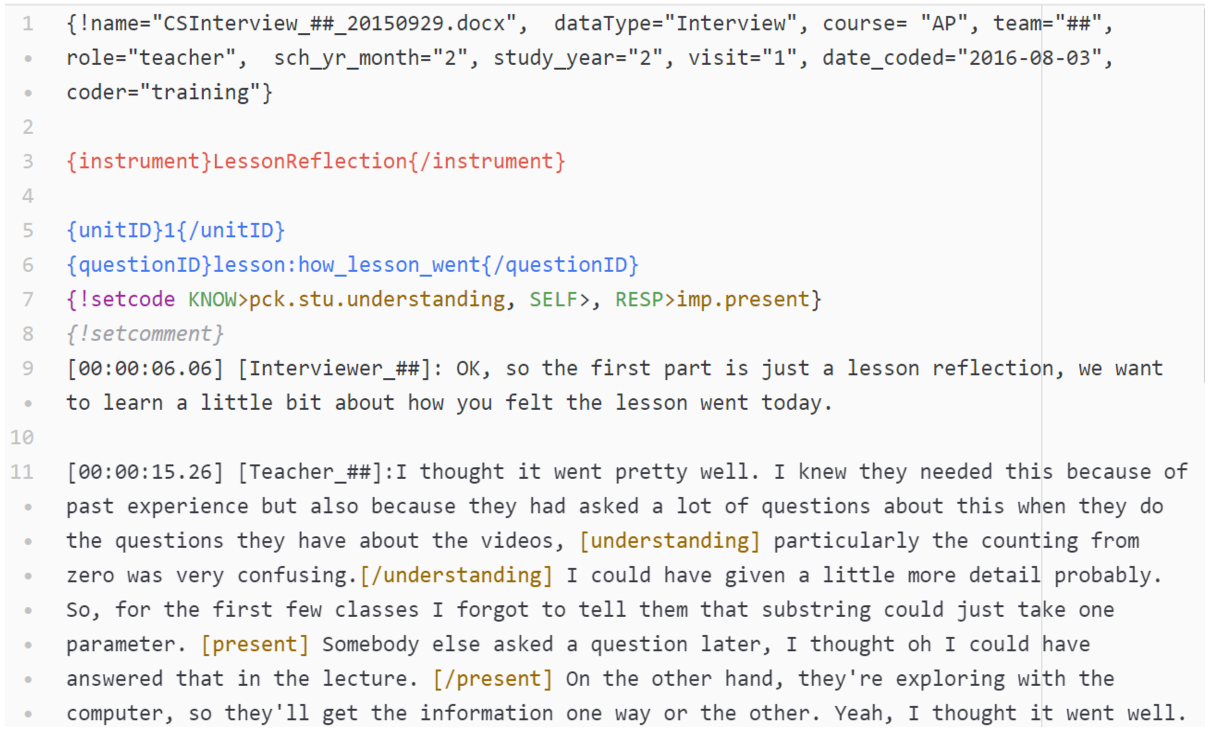
[4] Ms. King: Yes so when I was checking off, so some people were working on their code. Other people were definitely working on their worksheet. Now the people who were behind in their code are also the people who have a harder time with the worksheet, so there's always that some kids are taking three times as long as others, or four days more to finish things, and it's, it's a little bit troubling to try to not let the ones who are ahead get bored, and not to have too many assignments lined up, because then you start to panic, the kids start to get confused.

This entire excerpt was grouped into one unit. In line [1] I pose a question from the interview protocol. In line [3] I ask a question to clarify her response from line [2]. While line [3] was a separate question, I treated it as a continuation to the exchange in lines [1-2]. It is also important to note that in the conversational style of a semi-structured interview, there were times when interviewers asked relevant questions that were not listed on the interview protocol. In these instances, the questions and the participant's response were grouped into a unit labeled as 'other'. The 'other' category was not divided into subcategories, but units of this type tended to focus on clarification questions, specific events observed during the classroom visit, and final thoughts shared after all protocol questions were asked. There were also instances, particularly in think-aloud interviews, where participants would respond to multiple prompts in the same response. In these instances, the question and the participant's response was grouped into a unit labeled as 'multiple'. Lastly, most transcripts included two interviews, both the lesson reflection interview and either a think-aloud interview or a CoRe reflection interview. While these interviews were contained in the same transcript file, they were treated as two separate interviews.

As a final step in data reduction of the participant interviews, I tagged the transcription files with metadata to describe both the interviews and the individual units within them. This tagging process was also applied to the pre-lesson questionnaires and the post-lesson questionnaires. The metadata scheme I adopted is based on the TAMS scheme used in the qualitative software tool called TAMS Analyzer (Weinstein, 2006). I included a header tag at the start of each file that listed: the file name, the data source type (i.e., interview, pre-lesson questionnaire, or post-lesson questionnaire), the teacher's course (i.e., AP, Intro, or non-CS), a numeric identifier associated with the teacher, participant's role (i.e., teacher or volunteer), month of the school year with August as month 1, year of the WestEd study (i.e., year 2 for this dissertation work), visit number, coding date, and either the name of the coder or phase of interrater coding (i.e., training, agreement, reliability). Next, the start of each interview was labeled with a tag to identify the specific data source (e.g., lesson reflection interview, think-aloud interview). Within an interview, each unit was prefaced with tags to indicate their sequence in the interview and a label for the prompt or question to which participants responded.

Units that were coded for analysis included two additional tags to contain the codes applied to it as well as comments from the coder. Figure 3.9 shows an example of the metatags applied to a lesson reflection interview and a think-aloud interview conducted during the same session.

Tagging the interview and lesson questionnaires using this metadata scheme made it easy to process the files using pattern matching techniques and to tabulate units, codes, and interrater reliability. With numerous qualitative data analysis software available (e.g., TAMS Analyzer, NVivo, HyperResearch), readers may wonder why I opted to avoid these tools and simply apply a metadata scheme to text files. Two reasons motivated my decision. I found the effort required by the coding team to learn the intricacies of each tool required a lot of effort and reduced the amount of time available for actual coding. Second, each tool provides its own methods of retrieving and summarizing information from the data files and were sometimes restricted to specific operating systems. By using text files, the coders could easily manipulate files on any machine and edit them collaboratively if needed. While there was some time invested on my part to create the metadata scheme and to create programming scripts to prepare and process files, this effort seemed to outweigh the disadvantages of using an existing software package.

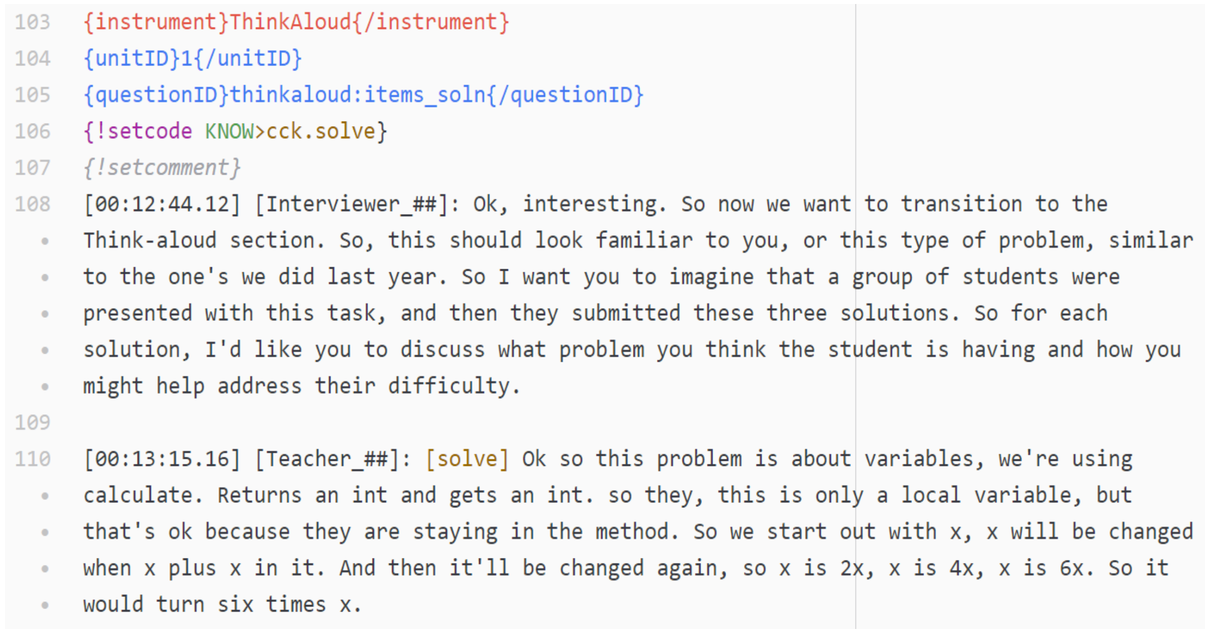


```
{!name="CSInterview_##_20150929.docx",  dataType="Interview", course= "AP", team="##",
role="teacher",  sch_yr_month="2", study_year="2", visit="1", date_coded="2016-08-03",
coder="training"}

{instrument}LessonReflection{/instrument}

{unitID}1{/unitID}
{questionID}lesson:how_lesson_went{/questionID}
{!setcode KNOW>pck.stu.understanding, SELF>, RESP>imp.present}
{!setcomment}
[00:00:06.06] [Interviewer_##]: OK, so the first part is just a lesson reflection, we want
to learn a little bit about how you felt the lesson went today.

[00:00:15.26] [Teacher_##]:I thought it went pretty well. I knew they needed this because of
past experience but also because they had asked a lot of questions about this when they do
the questions they have about the videos, [understanding] particularly the counting from
zero was very confusing.[/understanding] I could have given a little more detail probably.
So, for the first few classes I forgot to tell them that substring could just take one
parameter. [present] Somebody else asked a question later, I thought oh I could have
answered that in the lecture. [/present] On the other hand, they're exploring with the
computer, so they'll get the information one way or the other. Yeah, I thought it went well.
```



```
{instrument}ThinkAloud{/instrument}
{unitID}1{/unitID}
{questionID}thinkaloud:items_soln{/questionID}
{!setcode KNOW>cck.solve}
{!setcomment}
[00:12:44.12] [Interviewer_##]: Ok, interesting. So now we want to transition to the
Think-aloud section. So, this should look familiar to you, or this type of problem, similar
to the one's we did last year. So I want you to imagine that a group of students were
presented with this task, and then they submitted these three solutions. So for each
solution, I'd like you to discuss what problem you think the student is having and how you
might help address their difficulty.

[00:13:15.16] [Teacher_##]: [solve] Ok so this problem is about variables, we're using
calculate. Returns an int and gets an int. so they, this is only a local variable, but
that's ok because they are staying in the method. So we start out with x, x will be changed
when x plus x in it. And then it'll be changed again, so x is 2x, x is 4x, x is 6x. So it
would turn six times x.
```

Figure 3.9. Metatags applied to transcription and questionnaire files. ## indicates a number used to identify the teacher.

3.6.2.Questionnaire Item Selection

Another form of data reduction is selecting a subset of collected data to include in analysis. After reading and rereading responses provided on the pre-lesson questionnaires and the post-lesson questionnaires, I decided to exclude certain items because they did not contribute much analytic power to my understanding of the participants' PCK or instructional practices specific to CS (see Tables 3.4 and 3.5). First, I excluded items that tended to elicit descriptions of lesson topics, progress through the curriculum, or tools used in the classroom without mention of student learning or impact on teaching. These excluded items were items 9 and 15 on the pre-lesson questionnaire and items 7 and 10 on the post-lesson questionnaire. The following list provides examples of typical responses to these items:

- We've finished the basic constructs of Strings, if/else, loops. We haven't done arrays yet
- We are using Java because that is what AP CS teaches.
- Traditional paper notes accompanied by a visual lesson using Snap!

Table 3.4

Pre-lesson Questionnaire Open-ended Items Included in Analysis

#	Items	Included	Excluded
6	What is learning objective of this specific lesson?	X	
7	Why is it important for students to know this?	X	
9	Where does this lesson fit in the sequence of the unit you are working on? What have the students experienced prior to the lesson?What will they learn after the lesson?		X
10	What else do you know about this idea that you do not intend students to know yet?	X	
11	What are the difficulties or limitations connected with teaching this topic?	X	
12	What do you know about students' thinking that will influence your teaching of this topic?	X	
13	What other factors will influence your teaching of this topic?	X	
14	What teaching procedures will (would) you use to engage with this topic and why?		X
15	What technology will (would) you use to engage with this topic and why? (e.g., programming languages, programming environments, visualizations)		X
16	How will you know if this lesson is a success?		X
17	Is there anything else you would like to share about the lesson I will observe?		X

Table 3.5

Post-lesson Questionnaire Open-ended Items Included in Analysis

Items	Included	Excluded
What was the main topic of this lesson?		X
How did you prepare to teach this topic?	X	
What additional preparation do you need to teach this topic again?	X	
What advice would you offer someone teaching this topic for the first time?	X	
What resources were used to plan this lesson?		X
How did these resources support instruction and learning?		X
How did these resources hinder instruction and learning?		X
Will you plan another lesson to revisit the topic(s) covered today? / Please explain why or why not.		X
How did the co-teaching model used during today's lesson support and/or hinder instruction and learning?		X
How did the co-teaching model used during today's lesson support your development as a computer science high school teacher?		X
Is there anything else you would like to share about the lesson I observed?		X

Second, I excluded items that tended to elicit descriptions of general pedagogical knowledge and practices that might apply across domains, such as ideas related to lesson planning or factors affecting learning not specific to CS. These were items 4 and 16 on the pre-lesson questionnaire and items 8, 9, and 10 on the post-lesson questionnaire. The following list provides examples of typical responses to these items:

- I have already introduced the overview. At this point we will look at more examples to deepen students understanding
- If the students are engaged and not just filling in anything.
- I will like to see students successfully complete the project.
- Gave the students more practice than they would have had doing a couple of coding exercises.
- Some students work ahead on the worksheet, getting it wrong as they ignore the lesson. Giving the examples as notes would prevent that, but have other issues.
- Taking paper notes may seem boring sometimes.

I also excluded optional questionnaire items (i.e., item 17 on the pre-lesson questionnaire and item 14 on the post-lesson questionnaire). More than half (60%) of the responses to these two items were of the form ‘No, I have nothing else to add.’ The remaining responses were mostly affective comments (e.g., ‘I was proud of my students’) or additional details about lesson sequencing or content. Lastly, I also excluded two post-lesson questionnaire items that asked teachers to describe the impact of the co-teaching model on student learning, instruction, and their own development (i.e., items 12 and 13). Several (30%) of the responses to these two items were of the form ‘N/A’ or ‘no volunteer present’. Most other responses described the co-teaching model used; information about the co-teaching implementation was captured more consistently with closed-ended items on the post-lesson questionnaire. Also, participants were asked about their co-teaching implementation during interviews, which tended to elicit more detailed responses than those provided on post-lesson questionnaires.

In excluding these items from analysis, I may have lost some insightful information about participants’ CS PCK development. However, given the infrequency of comments about their CS PCK or their CS related instructional practices on these items, I decided to focus on other items that seemed to elicit richer responses. While I did not systematically analyze these items, I did review them to better understand the classroom contexts within which participants taught.

3.6.3. Observation Protocols

Another layer of data reduction occurred during classroom observations. Our study did not include consent to video record classrooms so I had to rely on observers to take field notes and interpret teacher, volunteer, and student actions in the moment. Also, observers conducted their observations independently, limiting the amount of classroom action they could focus on while writing their notes. To mitigate some of the limitations with conducting observations in this manner, I introduced processes to help observers capture their notes systematically.

Observers divided classroom lessons into six-minute segments. During each segment, they spent the first three minutes watching the classroom and the remaining three minutes recording their notes. In the recording phase, they focused on their jottings and not on completing the checkbox portion of the observation protocol. Observers filled out the checkbox portion of their protocols after each visit. Also, after testing the observation protocol during the 2014-2015 school year, I narrowed the number of categories of classroom discourse moves and activity types to focus on actions that required little interpretation on the part of the observer (e.g., observers found it easier to note when a teacher was asking a question than to decide if a discourse move should be categorized as a metaphor or a justification).

This approach to reducing classroom observations into a standardized protocol form was not without its limits. First, given observers' varying levels of computer science knowledge and the fast pace of classroom dialogue, we did not focus on the accuracy of teachers' remarks during the observations. So, our observations provide more information about the frequency of classroom activities and less detail about the quality of those activities. Second, with only one observer present in the classroom it was not possible to record every interaction that occurred, especially during lab sessions where students worked at individual computers and instructors circulated the room to answer questions. So, each observation can only be viewed as an approximation of the activities that occurred during our visits.

3.7 Data Analysis

3.7.1. Interviews and Open-ended Questionnaire Items

Each interview and open-ended questionnaire item was analyzed to identify instances when participants discussed pedagogical content knowledge, content knowledge, instructional responsibilities, or their own growth as a teacher. I identified types of knowledge and responsibilities to use in my coding scheme through a review of literature on PCK drawing heavily on the work of Shulman (1986) and Ball, Thames, and Phelps (2008). The coding scheme went through multiple iterations, particularly around ways to capture teachers' actions in the classroom. The final coding scheme contains three main categories: CS teaching knowledge, instructional responsibilities, and other. CS teaching knowledge describes knowledge of CS content and problem solving, knowledge of students, and knowledge of instructional practices. Instructional responsibilities describes some of the activities teachers engage in when planning and delivering their lessons. Lastly, the other category captures instances where teachers either talk about their own development or do not mention ideas captured by other codes. The coding scheme is listed in Table 3.6. Teachers often discussed multiple ideas in one unit and since I was interested in capturing the breadth of ideas shared, each unit could receive multiple codes. Code counts were tabulated per teacher to identify the percentage of units containing each subcategory, each major category, and a co-occurrence of a knowledge category and a responsibility category. The coded interview and questionnaire data were used to supplement other analyses and to explore the relationship between PCK and instructional responsibilities.

Table 3.6

Coding Scheme Used with Interviews and Open-ended Questionnaire Items

Category	Definition
CS Teaching Knowledge	
Content Knowledge	
Computing Topics	Teacher's understanding of specific CS topics (not knowledge of the field)
Methods for Solving Problems	Teacher's ability to solve CS problems
Pedagogical Content Knowledge	
Student Understanding and Difficulties	Knowledge of students' ideas and misconceptions about CS
Student Interest and Motivation	Knowledge of student interest and motivation related to CS
Representations and Methods	Knowledge of how topics are represented and how topics can be presented to learners
Timing, Pacing, and Sequencing	Knowledge of how topics should be organized to support learners
Instructional Responsibilities	
Plan and Organize Lessons	
Find materials	Search for and evaluating instructional materials
Create materials	Create instructional materials
Modify materials	Modify instructional materials created by others
Review materials	Look over existing instructional materials
Practice materials	Complete tasks that will be assigned to students
Organize lesson	Decide on the timing, pacing, sequencing of lessons; Decide on student grouping for group work
Implement and Monitor Instruction	
Assist students	Provide help to students
Evaluate learning	Assess students, assign grades, review student progress
Present ideas	Present CS information, give CS explanations, whole class instruction
Other	
Self	Discussing one's effectiveness as a teacher
None	No evidence of other coding categories is present

3.7.2. Closed-ended Questionnaire Items

Two closed-ended items were analyzed to determine teachers' feelings of comfort and preparedness with the units covered in their classes during case study observations. Teachers responded to these questions on each pre-lesson questionnaire selecting among three levels of comfort or preparedness (i.e., not at all, somewhat, and completely). I converted these levels into numeric values (i.e., -1, 0, and 1) in order to average feelings of comfort and preparedness over the school year. Six closed-ended items were also analyzed to determine who engaged in instructional responsibilities on each teaching team. Teachers responded to these items on each post-lesson questionnaire selecting among four degrees of involvement (i.e., no one, mostly volunteers, both volunteers and teachers, and mostly teacher). I converted these levels into numeric values in order to average engagement in instructional responsibilities over time. Since I was most interested in the degree to which teachers assumed instructional responsibilities, I collapsed the levels of *no one* and *mostly volunteers* to a value of 0, assigned *both volunteers and teachers* a value of 1, and assigned *mostly teacher* a value of 2.

3.7.3. Observations

Observation data was used to provide another perspective on the instructional responsibilities assumed by teachers and to paint a portrait of the types of activities implemented in the teachers' classrooms. The observation protocol data was used to calculate the percentage of class time teachers and volunteers devoted to delivering instruction, assisting students, and various discourse moves. Due to changes in the observation protocol between the first and second semesters, I only included the observation protocol categories gathered across the entire school year. Percentages of delivering instruction and assisting students were compared against teachers' self-reported questionnaire items. Questionnaire data were deemed in agreement with observation data if (a) a teacher indicated mostly volunteers and the observer noticed volunteers performing the responsibility a greater percentage of class time than the teacher, (b) a teacher indicated both teacher and volunteers and the observer noticed both members of the instructional team performing the responsibility during class time, or (c) a teacher indicated mostly teacher and the observer noticed the teacher performing the responsibility a greater percentage of class time than volunteers.

3.7.4. PCK Questionnaire

The PCK questionnaire was used to analyze teaching knowledge about the concept of linear data structures and the most difficult topics in teachers’ courses. I analyzed the quantity and type of ideas participants shared about teaching linear data structures. Since little research exists describing expert computer science teaching knowledge, I also compared these responses against publicly available data sources gathered from experienced computer science educators who completed similar tasks. This comparison allows for a heuristic analysis of participants’ teaching knowledge proficiency by exploring its similarity to the knowledge of experienced teachers. These comparison sources were an online repository of computer science teaching tips (http://csteachingtips.org) and workshop discussions in response to Loughran, Mulhall, and Berry’s CoRe instrument for multiple computing topics (Saeli et al., 2010).

3.7.5. Teaching Beliefs Questionnaire

To better understand the beliefs of the case study participants, I asked each teacher to answer four open-ended items about their epistemological beliefs drawn from Luft and Roehrig’s Teacher Beliefs Interview (2007). Luft and Roehrig identified five categories of beliefs (i.e., traditional, instructive, transitional, responsive, and reform-based) that were used to code participants’ responses. Given ambiguity in the terms *responsive* and *reform-based* as defined by Luft and Roehrig and as defined by other teacher learning literature, I replaced the terms with the categories of *interactive* and *responsive*. I drew on sample responses provided in Luft and Roehrig (2007) and the Salish I Research Project (1997) to determine the most appropriate category to apply to each response. Responses could receive multiple codes if teachers expressed more than one idea in their comments. Interview data were used to supplement their responses and to glean the epistemological beliefs of Mr. Edwards and Mr. Perez who did not complete the questionnaire.

3.7.6. Content Assessment

Teachers were also asked to complete a multiple choice content assessment as part of the main study. The assessment was used as a measure of the teachers' CS content understanding. I calculated the percentage of items teachers answered correctly and compared their performance against the average score of students in their classes who also agreed to share their assessment results with our research team.

3.8 Trustworthiness

Reviewing the procedures and analytic methods of a study can help readers evaluate the quality of the research and determine the utility of the work for advancing knowledge and impacting the lives of others. Lincoln and Guba (1985) described four components of trustworthiness: applicability, neutrality, truth value, and consistency. Applicability concerns the transferability of findings to contexts beyond the study and can be demonstrated by providing details about participants and their contexts, threats to generalizability, the scope and bounds of the study, thick description of the data, and congruency with prior theory (Miles & Huberman, 1994). Neutrality captures the degree of objectivity in a study or how much results are influenced by research bias. Miles and Huberman (1994) suggest researchers establish neutrality by describing study methods and results in such detail as to allow for an audit of procedures, considering alternative hypotheses, explicating their assumptions and biases, and making data available for others to analyze. In other sections of this chapter I provided details about the participants and their locales, study procedures, study limitations, and my background that allow the reader to judge the neutrality of this work and its applicability to CS teacher preparation in other settings. In this section, I focus on the truth value and consistency of this work.

Truth value relates to the credibility of study findings and consistency relates to the reliability of the study procedures. Creswell (2006) identified eight methods to validate qualitative research and he recommends studies make use of at least two of these methods: triangulation, prolonged engagement and persistent observation in the field, clarifying research bias, rich and thick description, peer review, negative case analysis, member checking, and external audits. I used triangulation and prolonged engagement in the field to ensure the credibility and consistency of this study.

3.8.1. Data Source and Method Triangulation

Triangulation is a strategy of corroborating evidence gathered from multiple sources and methods (Creswell, 2012). Of the four approaches to triangulation (i.e., data source triangulation, analyst triangulation, theory or perspective triangulation, and methods triangulation), the validity of case study research can best be strengthened with the triangulation of data sources and methods (Yin, 2013). As described in the Data Collection Procedures and Data Sources sections above, I engaged in data source and method triangulation by using different methods (i.e., interviews, questionnaires, observations, and assessments) to gather evidence over multiple months from participating teachers who worked in different school contexts. While these methods of triangulation helped me to substantiate the patterns that emerged from different data sources, one case study participant, Ms. Jones, found the redundancy excessive.

Although Ms. Jones completed all requests we made during the study, she once commented on a questionnaire in December 2015: “I feel like a lot of these questions are repetitive between the pre-observation survey, post-observation survey, and post-observation interview." Unlike other teachers, Ms. Jones was a terse participant, so responding to similar questions across instruments did not tend to elicit more information from her. Also, Ms. Jones completed the case study tasks promptly; she often completed the post-lesson questionnaire just before her lesson reflection interview. So, the time between completing the lesson questionnaires and interviews was much shorter for her than for other participants. I intended the pre-lesson questionnaire to capture teachers' intentions related to their upcoming lessons, the lesson reflection interview to capture teachers' immediate response to their lesson, and the post-lesson questionnaire to capture thoughts after having some time to reflect on their day. This redundancy did not appear problematic for other participants.

3.8.2. Analyst Triangulation

I also incorporated analyst triangulation into this study by involving multiple researchers in classroom observations and in coding data from interviews and open-ended questionnaire items. Einsenhardt argues that the use of multiple researchers is advantageous for building trustworthiness in the study because "their different perspectives increase the likelihood of capitalizing on any novel insights which may be in the data…[and] convergent perceptions add to the empirical grounding of the hypotheses, while conflicting perceptions keep the group from premature closure" (1989, p. 538). However, involving multiple researchers in this study also required aligning our approaches to gather and analyze data reliably. Next, I describe how this reliability was established with observers and with coders.

Reliability of Observation Data. Three research staff helped me conduct observations for this study. All three researchers also participated in the project during the 2014-2015 school year. Given the diversity of the researchers' backgrounds relative to teaching and to computer science, I held a two-day training session in early September 2014 the introduced observers to the larger WestEd study, case study logistics, observation protocol, the PCK framework, and the two programming languages used in the TEALS courses. Due to delays in study recruitment, most observers began visiting participating teachers in January 2015, so I held a second training that reviewed the September 2014 training and also included time for practicing the observation protocol using a classroom video obtained from TEALS, listening to and discussing excerpts from two interviews conducted in fall 2014, and a tour of the database used to store study data.

In fall 2015, observers continued classroom observations using a revised version of the Year 1 observation protocol. Once observers completed a visit and uploaded their notes to our study database, I reviewed their write-ups to check for missing information, unclear comments, and observer concerns. Then I met with the observer to discuss and resolve any identified issues. If the observer and I made any decisions about how to interpret classroom events, I shared these decisions with other observers. In December 2015, two changes occurred in the study. The observation protocol was revised (see Observation Protocol II above) and one observer left our team. So, I held another training session in January 2016 to review the protocol changes and establish interrater reliability (IRR).

During this training session, we watched nine-minute excerpts of video collected by TEALS in two different classrooms, completed the observation protocol for these segments, clarified elements of the protocol that were confusing, and completed the observation protocol a second time for both nine-minute excerpts. I calculated IRR for our ratings of both the first and second viewing of the classroom videos using Krippendorff's alpha reliability coefficient. I selected Krippendorff's reliability coefficient because it allows for more than two observers, any number of categories, any level of measurement, and both large and small sample sizes (Krippendorff, 2011). While there is no definitive agreement on the acceptable ranges of reliability coefficients, Krippendorff (2012) cautiously suggested a value of at least .67 to draw reliable conclusions. Krippendorff alpha coefficients typically range from 0 (no correlation in codes) to 1 (complete agreement), but they can assume negative values "when coders consistently agree to disagree" (Krippendorff, 2008, p. 7). To simulate our normal observation conditions, I divided the videos into one-minute segments for a total of 20 segments. After each minute passed, I paused the video and then we completed the observation protocol form. Table 3.7 presents the alpha coefficients for each category of the observation protocol.

Fourteen categories passed Krippendorf's suggested threshold value of .67. However, eleven categories fell below the suggested threshold. Four categories related to *voices in the room* and *interactions* each with α values between -0.01 and 0.45. These categories are rather straight forward and easy to interpret, so I reviewed each coding choice to identify possible reasons for the discrepancies in our agreement. Discrepancies related to (a) identifying when we heard the volunteer's voice and (b) interactions that occurred during lab time. We watched these video segments again and identified all the instances where volunteers were speaking or where students, teachers, and volunteers were interacting. Some of the discrepancies occurred because some of the audio on the video was unclear. However, this review also highlighted how it is impossible for one person to focus on multiple interactions happening simultaneously in a classroom. Disagreement also occurred with categories related to classroom discourse. The α values for students posing questions, teachers responding to questions, and volunteers

Table 3.7

Interrater Reliability for Observer Training

Category	Krippendorf's α
Instructional Segment Type[1]	
Direct Instruction	1.00
IRE	**0.63**
Instructional Conversation	0.72
Non-content activities	0.76
Classroom Segment Type	
Student work time (individual)	0.87
Student work time (groups)[2]	1.00
Students take assessment[2]	1.00
Lab time at computers	0.87
AP exam prep[2]	1.00
Voices in Room	
Student	1.00
Teacher	0.83
Volunteer	**0.44**
Interactions	
Student-student	**0.41**
Student-teacher	**0.45**
Student-volunteer	0.87
Teacher-volunteer	**-0.01**
Classroom Discourse	
Poses question: student	**0.59**
Poses question: teacher	1.00
Poses question: volunteer	0.82
Responds to question: student	0.82
Responds to question: teacher	**0.27**
Responds to question: volunteer	**-0.02**
Provides explanation: student	**0.47**
Provides explanation: teacher	**0.58**
Provides explanation: volunteer	**0.59**

[1]The observation protocol allows observers to indicate whether the teacher or volunteer performed the specified instructional activity. During IRR, we condensed these options and simply indicated if any instructor performed the activity.
[2]These activities were not present during the video segments used for IRR, so Krippendorf's α indicates perfect agreement because all observers agreed these activities did not occur.
Note: **Bolded items** indicate categories that do not pass Krippendorf's suggested threshold for acceptable values of agreement.

responding to questions were all much lower than the related categories of other actors performing the same discourse moves. For each of these categories, there were only one or two segments of disagreement. We reviewed the videos as a team and realized that for each category, one member on our team miscoded a segment that they thought should have been coded differently. At the end of our IRR meeting, we felt confident that we could identify the types of actions that should be coded as containing the volunteer’s voice, interactions between students, posing questions, and responding to questions. However, we also acknowledged that we might miss some of these instances, especially when classes engaged in lab time.

All categories of providing explanations received low agreement scores, but for different reasons. First, for most segments, our team agreed that no student provided an explanation. However, in one segment, two of us agreed that a student provided an explanation. This one segment of disagreement amongst a mostly absent category resulted in a low agreement score, although we felt confident we had a shared understanding of the category. Second, we disagreed on coding segments where teachers provided explanations because either (a) one observer thought an explanation spanned across units while others thought it started slightly later in the video clip or (b) one observer indicated this discourse move during a lab time segment where we could not hear what the teacher said to the student. Similarly, we disagreed on coding segments where volunteers provided explanations because either (a) one observer thought an explanation lasted one unit while others thought it finished in a following unit or (b) one observer thought an introduction to a lab assignment was an explanation while the others thought it was not an explanation because it lacked discussion of content. We thought that overlapping segments was a non-issue because in our real observations, segments would be separated into larger time spans (i.e., 6 mins) where transitions would be more distinct. I also reminded observers not to make assumptions about actions in the classroom and to only code activities they felt certain they observed.

One category where we appeared to have a disagreement on the understanding of the code was instructional segments containing IRE sequences ($\alpha = .63$). There were two segments where two observers coded the presence of IRE sequences while the third observer did not. This discrepancy seemed related to distinguishing IRE sequences from direct instruction. Often, IRE sequences occurred in the middle of a direct instruction segment. For example, a teacher might launch a lesson explaining the different ways to initialize arrays, insert an IRE sequence to ask students how to write each initialization type in Java, and then continue with a live coding demonstration of creating and using an array. The first observer would only code this instructional sequence as direct instruction. However, I decided that I also wanted to capture IRE sequences that occurred within direct instruction sequences and so we agreed as a team to do this in our observations.

Our classroom observations were limited by the fact that we did not video record lessons and had to make decisions in the moment about which activities occurred during our visits. I acknowledge that by using this approach we may not have captured all the activities that occurred in a classroom, but by observing teachers across multiple lessons we captured a range of instructional activities and discourse moves used in their classrooms. Our IRR process also highlighted drawbacks to conducting observations independently and dividing our observations into segments where activities occurring on segment boundaries could alter frequency counts.

Reliability of Coding. One observer also assisted me in coding interview data and questionnaire data. I invited this observer to join me in the coding phase of the project given our prior experience coding data together for another study. We began our coding process in April 2016 where we first discussed articles that influenced my initial coding scheme and then we went through several rounds of trying out coding schemes, calculating IRR, and revising the coding scheme based on our discussions. The final coding scheme used is shown in Table 3.6 above. Beginning in August 2016, we went through three steps to establish IRR with the final coding scheme: training, agreement, and reliability.

The goal of the training phase was to review the coding scheme and provide the second coder with examples of coded units to illustrate the categories of the coding scheme. I selected six visits from Ms. King, Ms. Jones, and Mr. Miller to use during this phase. Half of the selected visits occurred during the first semester and the other half occurred during the second semester. I chose data from these three participants to include data that was both familiar and unfamiliar to each coder. I worked directly with Ms. King, the second coder worked directly with Ms. Jones, and neither of us worked directly with Mr. Miller. The training set consisted of 132 units. The second coder agreed with 69 units (52%) without reservation, but wanted to discuss the remaining 63 units (48%). She agreed with the original codes applied to 42 of these 63 units, and disagreed with codes applied to the other 21 units. After discussing these 21 units together, we decided to maintain the original codes applied to 16 units and to either partially or completely change codes applied to 5 units. In sum, we agreed with the original codes applied to 111 units (84%); for units of disagreement, we deferred to my coding choice for 16 units (12%), and we deferred to the second coder’s choice for 5 units (4%).

The goal of the agreement phase was to review data files together and identify how we would code them, resolve discrepancies in our decisions, and make decisions on how to handle similar cases in the future. This form of negotiated agreement is useful when conducting exploratory work and can help to increase IRR (Campbell, Quincy, Osserman, & Pedersen, 2013). I selected two visits from Mr. Edwards to use during this phase, one from September 2015 and one from February 2016. The agreement set consisted of 29 units. We agreed on 24 units (83%) and we reconciled our disagreements with 5 units (17%).

The goal of the reliability phase was to show consistency in our application of the coding scheme by separately coding a subset of the data and comparing our coding choices using Krippendorff's alpha reliability measure. I selected two visits from Ms. Jones and Mr. Perez to use during this phase. As in the prior two phases, half of the selected visits occurred during the first semester and the other half occurred during the second semester. I chose data from these two participants so that each teacher was included in one of the three phases. The reliability set consisted of 81 units. Fifteen categories passed Krippendorff's suggested threshold value of .67, one category fell just below the threshold at $\alpha = .66$, and one category was never used in the coding set (see Table 3.8).

Table 3.8

Interrater Reliability for Coder Training

Category	Krippendorf's α Round 1	Krippendorf's α Round 2
KNOWLEDGE		
Common Content Knowledge		
Computing topics	0.75	0.92
Methods for solving problems	0.85	1.00
Pedagogical Content Knowledge		
Student understanding and difficulties	1.00	0.91+
Student interest and motivation	**0.66**	1.00
Representations and methods	0.86	0.93
Timing, pacing, and sequencing	1.00	**0.66+**
RESPONSIBILITIES		
Plan and Organize Lessons		
Find materials	0.85	1.00
Create materials	0.71	0.90
Modify materials	1.00	**0.66+**
Review materials	---	---
Practice materials	1.00	1.00
Organize lesson	1.00	1.00
Implement and Monitor Instruction		
Assist students	1.00	1.00
Evaluate learning	1.00	0.79+
Present ideas	0.75	0.82
OTHER		
Focus on self as a teacher	0.90	1.00
None	0.87	0.92

+ indicates categories where the α value in round 2 was less than the α value in round 1.
Note: Bolded items indicate categories that do not pass Krippendorf's suggested threshold for acceptable values of agreement.

We then met to review all units of disagreement and agreed how we wanted to code them and similar units in the future. We then independently recoded the units of disagreement and recalculated our IRR score. It is important to remember that each unit could receive multiple codes. Units of disagreement included units where there was no agreement between any of the applied codes as well as units where we agreed on some codes and disagreed on others. So, during the second round of IRR, it was possible for us to disagree on codes that we agreed upon during the first round and lower our IRR values. While alpha values generally improved during the second round of the reliability phase, there were four categories that saw a decrease in agreement and two of these categories fell slightly below Krippendorff's suggested threshold value of .67. Accepting .67 as the minimum value of acceptable agreement, we felt confident we could consistently code fourteen of the coding categories independently, less confident that we would consistently code the timing and modify categories, and unsure about our consistency for the review code since it did not appear in the reliability set.

After establishing IRR, I divided the remaining data files amongst myself and the second coder, balancing across teacher, time of year, and course, and we independently coded different sets of data files. After completing our independent coding, I noticed I applied the *none* code more often to the subset of data files I coded than the second coder applied to the subset of data files she coded (see Figure 3.10). Results from z-tests of equal proportions suggest our coding differed for a subset of the data, particularly with Mr. Edwards, Ms. Jones, and Mr. Miller (see Table 3.9). Since my set of data files and the second coder's set of data files were balanced across time (i.e., we each coded first semester and second semester interviews for each participant), I expected some comparability across our separate coding. It is also possible that teachers discussed vastly different ideas at the visits I coded than at the visits coded by the second coder, which would explain my overuse of the none code. However, these low p-values made me hesitant to make strong claims about the frequency of codes across teachers.

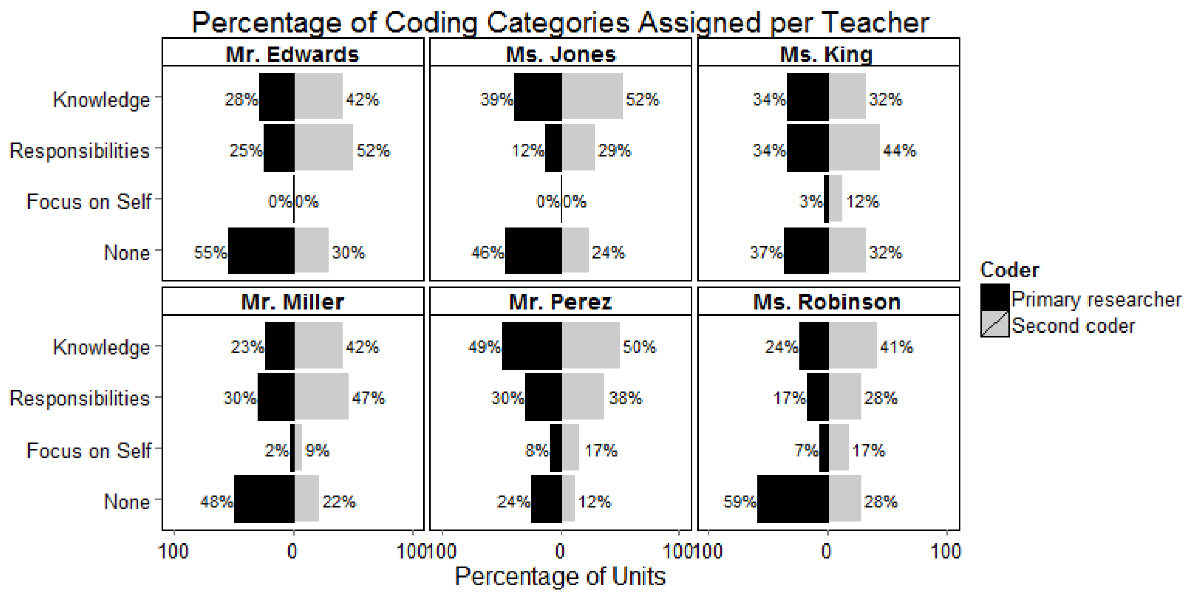


Figure 3.10. Percentage of codes applied to interview and open-ended questionnaire data.

3.8.3. Prolonged Engagement and Persistent Observation

Prolonged engagement in the field and persistent observation of sites is important for the trustworthiness of a study because it helps to develop trust between the researcher and the participants, overcome distortions in behavior due to the presence of a newcomer in the classroom, and distinguish irrelevant events from atypical events of importance (Lincoln & Guba, 1985). I find these strategies particularly important when working in K-12 environments because teachers sometimes associate the presence of a researcher in their classroom with performance evaluations conducted by their administrators and district coaches. Another reason these strategies were important for this study is because I asked teachers to reveal the weaknesses of their teaching knowledge and practice. Exposing one's weaknesses is unlikely to occur after one encounter with a researcher.

Table 3.9

Test of Equal Proportions for Post-interrater Reliability Coding

Coding Category		*z*	*p*
Mr. Edwards			
	Knowledge	1.81	0.18
	Responsibilities*	6.52	0.01
	Focus on Self	---	---
	None*	4.89	0.03
Ms. Jones			
	Knowledge	1.96	0.16
	Responsibilities*	4.56	0.03
	Focus on Self	---	---
	None*	5.48	0.02
Ms. King			
	Knowledge	0.07	0.79
	Responsibilities	0.88	0.35
	Focus on Self	2.75	0.10
	None	0.28	0.59
Mr. Miller			
	Knowledge*	4.34	0.04
	Responsibilities	3.28	0.07
	Focus on Self	3.24	0.07
	None*	7.74	0.01
Mr. Perez			
	Knowledge	0.01	0.92
	Responsibilities	0.40	0.53
	Focus on Self	1.05	0.31
	None	1.29	0.26
Ms. Robinson			
	Knowledge	3.07	0.08
	Responsibilities	1.49	0.22
	Focus on Self	2.07	0.15
	None*	9.56	<0.01

*Indicates results of the *z*-test were statistically significant indicating the percentage of codes applied by the primary researcher and the percentage of codes applied by the second coder may not be comparable.

The observation team achieved prolonged engagement and persistent observation by working with most participants during the 2014-2015 school year before this dissertation work began, except in the case of Mr. Perez who joined the study during the 2015-2016 school year. Furthermore, we visited the participants multiple times across the school year which allowed us to observe activity around the ebb and flow of high school calendars (i.e., regular class time, winter breaks, and end-of-school exams). By working with teachers over time, we developed a rapport that made the participants comfortable talking to observers about their craft. This rapport was reflected in comments made by a couple of participants:

> "I do want to say because you have been listening to me for two years that I am feeling much more confident." (Ms. King, April 2016)
>
> "OK, those were fun. I actually like this. It gets me thinking." (Mr. Miller, October 2015)

3.9 Researcher's Role

Schwandt defines researcher reflexivity as "the process of critical self-reflection on one's biases, theoretical predispositions, preferences" (2007, p. 260). Highlighting one's background experiences allows readers to understand the perspective of the researcher and their interpretation of study data (Creswell, 2012). Reflexivity also serves as a means of establishing validity in qualitative research (Creswell & Miller, 2000). Below I provide a short account of my prior experiences related to computer science, teaching, and the locales within which participants worked to provide context for the perspective I bring to studying CS PCK.

My interest in computer science traces back to my childhood in the 1980s when I dabbled with the computers in my family home. Once in high school, I enrolled in my first formal computer science courses which included an introduction to programming in a language similar to Karel and the AP CS AB course which focused on the C++ programming language. I continued to pursue computing studies in college where I earned a bachelor’s degree in the field. While in college, I interned at two software companies and served as a teaching assistant for a course on databases while studying abroad in Europe. During my graduate career, I gained further experience in helping others learn to program as a teaching assistant in courses focused on computer-based learning environments and multi-agent computational modelling. I also conducted small studies exploring the use of pair programming and peer instruction to support novice programmers enrolled in college computing courses. In the middle of my graduate career, I took a leave of absence to spend a year developing and leading courses on databases and software development for young adults in West Africa. More recently I have conducted evaluations of initiatives focused on introducing high school boys of color and Latinx rural youth to programming and the tech industry.

As someone who identifies as a Black American woman, I was often the *only one* in the environments where I learned and applied my CS knowledge. Despite the notorious female-unfriendly culture of the computing field (e.g., Margolis & Fisher, 2003), my experiences in CS have been mostly positive and included supportive peers, family members, professors, and mentors. I enjoy computing and continue to incorporate it into my professional work as an educational researcher and in pursuit of my personal interests. At the same time, I understand the ordeals computing can present to learners. During my studies, I spent many sleepless nights reviewing concepts that required new ways of thinking, deciphering lecture slides of lengthy programs that professors sped through in class, and attempting to solve assignments that often felt beyond my capacity. I watched friends enter the CS program with great hopes, but then leave the department after reaching the limit of their patience or their endurance with a competitive learning environment that did not lead them towards their goals. In my roles as a teaching assistant and instructor, I saw many students struggle with course topics, compare themselves negatively to peers who progressed more quickly, and, for some, eventually give up on computing. Computer science can be challenging to learn, but I do believe that more learners, especially novice learners, can experience success in the field if provided with the right pedagogical and motivational supports.

WestEd's receipt of an NSF award to study CS PCK of high school teachers presented me with a propitious opportunity to address problems in computer science education through research. However, while I had experiences as a CS student and instructor that could inform my exploration of CS PCK, I have never been a high school teacher. I felt my inexperience with secondary classroom teaching would limit my perspective on the study data because I did not understand the daily realities of the participants' teaching lives. Furthermore, I was never educated in California and I had limited knowledge of the peculiarities of the educational systems within the state. Lastly, I had been living in the region for only three years when the WestEd study began and so I had little historical knowledge about the communities within which participants worked. I worried that this lack of experience with the K-12 schooling culture might impede my exploration of CS teachers working within this space.

During this study and the writing of this dissertation, I pursued multiple opportunities to bolster my understanding of the lives of high school CS teachers in California. First, I joined a local CSTA chapter where I attended monthly meetings of K-12 teachers and educational nonprofit staff who discussed issues of pedagogy and advocacy. Second, I attended a two-day professional development workshop at the University of California at Berkeley that introduced the popular computing BJC course upon which the TEALS Intro curriculum is based. Third, I brainstormed with district staff who support CS teachers around their ideas for researching CS initiatives within their school systems. And fourth, I attended a public focus group hosted by the California State Board of Education to gather feedback from teachers in northern California about the content of CS standards for the state. These opportunities allowed me to interact with educators across the state who varied in their CS teaching experience, courses taught, and student populations. My discussions with these educators provided insight into their most pressing problems related to CS. I also had the chance to see how teachers, both novice and experienced, responded to professional learning opportunities that occurred outside of the classroom. So, although I entered this study as an outsider researcher, I believe these opportunities provided me a better understanding of the lives of high school computer science teachers which supported my interpretation of the data gathered in this study.

CHAPTER 4. CS TEACHING KNOWLEDGE

4.1 Introduction

Teacher knowledge develops with experience and requires strong content knowledge (Borko & Livingston, 1989; Hashweh, 2005). Prior research suggests it takes at least five to seven years to attain teaching expertise (Berliner, 2004) and ten years to develop programming expertise (Robins et al., 2003). Furthermore, the field of CS is constantly changing, requiring teachers to stay one step ahead of their students to learn new material, programming languages, and paradigms (Gal-Ezer & Stephenson, 2010). So, what teaching knowledge can an experienced educator new to CS realistically develop within a three-year, on-the-job professional development program?

To investigate the development of CS PCK, I asked participants to complete a set of activities centered around the topics covered in their courses. During these activities, teachers (a) reflected on student difficulties and instructional methods related to linear data structures, (b) identified the most difficult topics in their courses and described how they teach those topics, and (c) completed an assessment and multiple think-aloud interviews focused on their content knowledge for teaching. Given that teaching knowledge depends on the contexts within which teachers work (Berliner, 2004), some of these activities related directly to participants' recent lessons. Other activities were identical for all participants to allow for comparisons across cases. The specific question I explored with these activities was: *what knowledge of CS content and student thinking do in-service CS teachers develop while participating in an on-the-job PD program?* Before describing these activities and their results in more detail, I will provide a summary of research related to learning and teaching CS. This review is intended to provide context for the study results described later in the chapter.

4.2 Research on Learning and Teaching CS

Computer science is notoriously challenging for novice learners. Du Boulay (1986), concentrating on programming, attributed this challenge to five areas learners need to confront simultaneously: recognizing what problems programming can solve (orientation), modelling how computers execute commands (the notional machine), learning the syntax and semantics of programming languages (notation), acquiring standard templates for common tasks (structures), and planning, implementing, and testing programs (pragmatics). While attempting to master these five areas, learners manifest misconceptions and less efficient practices. Effective teaching requires knowledge of these difficulties and a repertoire of strategies to address them.

4.2.1. Knowledge of Student Understanding

In contrast to literature on CS PCK, a wealth of research exists on student understanding of CS dating back to the 1970s. For example, du Boulay and O'Shea (1976) created a primer to explain Logo programming to middle school students using analogical modelling of a Logo machine. Since that time, work in this area has focused on mental models, perceptions and misconceptions, and differences between novices and experts (Fincher & Petre, 2004b).

Robins, Rountree, and Rountree (2003) and Clancy (2004) provided reviews of research on student understanding of CS focused specifically on novice programming. Their reviews describe literature in this area as focused on expert-novice differences, programming knowledge versus strategies, program comprehension versus program generation, procedural paradigms versus object-oriented paradigms, how novices learn, misconceptions, and attitudes. Several patterns of student understanding summarized in their reviews are presented in Table 4.1.

Table 4.1

Summary of novice programmer understanding from Robins, Rountree, and Rountree (2003) *and Clancy* (2004)

Topics Related to Novice Programming	Summary
Structure of knowledge	The knowledge of novices is organized around superficial similarities, context specific, and lacks detailed mental models. Novices avoid, rather than manage, complexity.
Planning	Novices do not often plan their programs. Issues related to planning programs and arranging pieces of code, and not language features, explains most of their difficulties.
Debugging	Novices do not often test their programs. Novices repair programs by attempting small fixes instead of larger reformulations of their code.
Code execution	Novices have difficulty tracing code, understanding the sequential nature of program execution, and understanding flow of control in programs.
Programming constructs	Novices understand updating and testing variables better than initialization, which they understand better than assignment. Understanding of recursion is supported by learning about iterative functions first.
Common bugs	Novices have misconceptions and produce bugs related to loops, conditionals, recursion, and arrays.
Sources of confusion	Ambiguity between programming terms, English definitions (e.g., *while* indicating continuous activity in English versus a once-per-iteration test in programming), and mathematical notation (e.g., = as a test of equivalence versus as an assignment operator) is a source of confusion for novice programmers. Overgeneralizing patterns learned in one context is another source of confusion (e.g., only seeing methods with number arguments can lead some students to think all methods must take numbers as arguments).

Another area of research providing insight into student CS knowledge is programming languages and environments designed for beginning learners. These tools are designed to simplify aspects of programming found difficult by novices. Kelleher and Pausch (2005) created a taxonomy of 80 such tools organized around (a) the goal of either teaching programming or using programming for other goals and (b) the aspect of programming made easier for novices. Their review describes the hypotheses underlying the creation of these tools which include:

- General-purpose languages have unnecessary syntax and features, commands ambiguous with English, and inconsistent uses for syntax
- Novices have difficulty with syntax (e.g., recalling commands and order of parameters, knowing whether and where to use braces)
- Novices have difficulty understanding program execution and debugging
- Novices struggle to figure out what to build, do not know what they can build, and cannot gauge project difficulty
- Mechanical difficulties of creating programs is a major barrier for novices

When learning CS, students face difficulties with specific concepts, expressing solutions with programming, and problem solving. Educators need to be aware of student misconceptions and of strategies to support them in overcoming common difficulties.

4.2.2. Instructional Strategies

Instructional strategies are techniques used to make subject matter understandable to students. Strategies can be domain independent such as lectures, group projects, and simulations or they can be subject-specific like inquiry practices in science. Examples of instructional strategies used in computer science include pair programming (Hanks, McDowell, Draper, & Krnjajic, 2004; McDowell et al., 2003; L. A. Williams & Kessler, 2001), peer instruction (Cutts, Carbone, & van Haaster, 2004; Pargas & Shah, 2006; Simon, Kohanfars, Lee, Tamayo, & Cutts, 2010), media computation (Guzdial, 2003), and equity practices (Margolis et al., 2014). Prior research on instructional strategies in CS has focused on the impact of different approaches on student outcomes. For example, in a meta-analysis of 32 studies of strategies used in introductory collegiate courses, Vihavainen, Airaksinen, and Watson (2014) found evidence that courses with relatable content or cooperative elements had more impact on student pass rates. Very little research has explored the relationship between use of instructional strategies and teacher knowledge in CS.

4.2.3. Linear Data Structures

A common topic in introductory computing courses is linear data structures. Linear data structures, such as lists and arrays, store a sequential collection of elements. Figure 4.1 presents a schematic of an array called myList that stores ten values. Common learner difficulties related to arrays include confusing indices and element values, confusing rows and columns of multidimensional arrays, and mistaking the length of an array for the last index in zero-based structures (du Boulay, 1986; Kaczmarczyk et al., 2010). Researchers have explored different strategies for introducing students to linear data structures including computer games (e.g., Eagle & Barnes, 2008), test-driven learning methods (Hilton & Janzen, 2012), and media computation approaches (e.g., Guzdial, 2003).

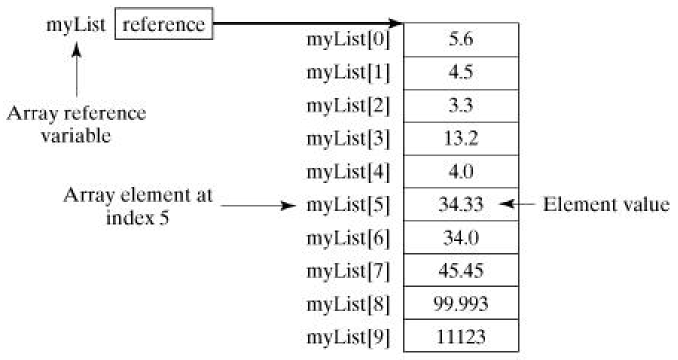


Figure 4.1. A visual representation of the array called myList which stores ten values. Source: https://www.tutorialspoint.com/java/java_arrays.htm

4.3 Method

4.3.1. Participants

Four teachers who varied in their prior CS teaching experiences participated in this component of the case study: Ms. Robinson, Mr. Miller, Ms. Jones, and Ms. King. Ms. Robinson and Mr. Miller had less prior CS teaching experience; both teachers were in their second year with the TEALS program and neither taught CS prior to joining TEALS. Ms. Robinson taught one section of the AP course. She also taught an introductory CS course outside of the TEALS program. Mr. Miller taught two sections of the Intro course, one collaboratively with TEALS volunteers and one independently. He followed the TEALS curriculum during the first semester and then spent the second semester teaching HTML and Python. Ms. King and Ms. Jones had more prior CS teaching experience. While Ms. King was also in her second year with TEALS, she taught an introductory Java course independently in the past. During the study, she taught three sections of the AP course. Ms. Jones was in her third year with TEALS and she taught multiple sections of the Intro course. She followed the TEALS curriculum during the first semester and spent the second semester teaching Java. If experience improves teaching knowledge, one might expect Ms. King and Ms. Jones to display greater PCK than Ms. Robinson and Mr. Miller.

4.3.2. Data Collection

Three types of data were collected to explore teachers' PCK. The data sources included think-aloud interviews conducted during the first semester of the study (see Appendices D, E, and F), a PCK questionnaire administered at the end of the school year (see Appendix I), and a content assessment completed at the end of the school year as part of the main CSPCK study. These data sources are detailed in the Methods chapter.

4.3.3. Data Analysis

Linear data structures. Using a synthesis of responses gathered from Ms. Jones and Mr. Miller's first semester think-aloud interviews and all four teachers' responses to the PCK questionnaire, I analyzed the quantity and type of ideas participants shared about teaching linear data structures. First, I compared these responses against publicly available data sources gathered from experienced computer science educators who completed similar tasks. This comparison allowed for a heuristic analysis of participants' teaching knowledge proficiency by exploring its similarity to the knowledge of experienced teachers. These comparison sources, henceforth named the *expert educator list*, were an online repository of computer science teaching tips (http://csteachingtips.org) and workshop discussions amongst experienced CS teachers in response to Loughran, Mulhall, and Berry's CoRe instrument (Saeli et al., 2010). Second, I categorized responses to determine if participants were providing qualitatively different responses than those on the expert educator list. I used du Boulay's (1986) five areas of programming difficulty to categorize the nineteen student difficulties identified by participants and expert educators. These categories are described above. I also used a grounded coding approach to categorize methods of addressing student difficulties into one of six categories: assigning problem solving tasks, providing real-life examples, relating concepts to other subject domains, using representations, sequencing content, and presenting information.

Most difficult computing topics. Teachers were asked to identify the most difficult topics in their courses and how they address the topics differently than other subject matter. First, I categorized participant responses using the K-12 Computer Science Framework (“K–12 Computer Science Framework,” 2016) which provides curricular guidelines for computing at the primary and secondary levels. Produced by a committee of educators, researchers, and other advocates, the guidelines were released in fall 2016. The framework includes five core concepts, each with its own subconcepts, presented as learning progressions that identify what students should learn by the end of grades 2, 5, 8, and 12. Second, I compared participant responses against two ranked lists of difficult CS topics gathered in prior research. While there is agreement in the computer science education community on which topics are difficult for students, there is less consensus on the relative difficulty of these topics. Researchers have explored the relative difficulty of computing topics by asking educators to rank lists of concepts (e.g., Lahtinen, Ala-Mutka, & Järvinen, 2005; Milne & Rowe, 2002; Schulte & Bennedsen, 2006) or to provide open-ended responses about difficult topics (e.g., Dale, 2006), and by analyzing student solutions to programming problems (e.g., Cherenkova, Zingaro, & Petersen, 2014). I selected questionnaire data gathered by Schulte and Bennedsen (2006) and Dale (2006) as comparison sources (see Appendix J). Schulte and Bennedsen (2006) asked 349 computing educators to rank the difficulty of twenty-eight computing topics on a five-point scale. The topics were selected from prior research on difficult introductory programming topics. Dale (2006) asked 347 computer science educators to describe the most difficult topic to teach in their introductory courses. Dale categorized her response set into sixteen categories and reported on the number of times each category was mentioned. I selected both data sets because they included responses from high school teachers, educators in the United States, and educators who mostly taught Java or C++. I also selected these two sources because of their differences in identifying difficult topics; one list used topics identified by the literature while the other list used topics identified by participants.

Content Knowledge. Participant responses to think-aloud interviews and a multiple-choice assessment provided insight into teachers' understanding of the content covered in their courses. In analyzing data on teachers' content knowledge, I focused on the correctness and the level of knowledge (i.e., partial or complete) expressed in their responses. Audio transcripts of think-aloud interviews were transcribed and unitized around interview prompts. Responses were coded as either *correct* (i.e., explanations were accurate), *partial* (i.e., explanations contained both accurate and inaccurate claims), or *incorrect* (i.e., explanations were completely inaccurate). For the content assessment, I calculated the percentage of items teachers answered correctly and compared their performance against the average score of students in their classes who agreed to share their assessment results. Participants completed content assessments aligned to either the Intro or AP curricula.

4.4 Results

4.4.1. Linear Data Structures

Participants described thirteen difficulties students have with linear data structures, four of which were also included on the expert educator list (see Table 4.2). Four difficulties on the expert educator list were not mentioned by any participants, one of which was specific to the Python programming language that only Mr. Miller used towards the end of the school year (see Table 4.3). Difficulties provided by participants focused almost equally on issues with notation and issues with pragmatics or structures. None of the difficulties provided by participants related to issues with the notional machine. The most commonly reported difficulties were confusing indices in arrays, confusing Java Array methods and ArrayList methods, and rearranging values in an array. In contrast, most of the difficulties provided on the expert educator list related to issues with notation. The other difficulties on the expert educator list related either to issues with the notional machine or to issues with pragmatics or structures. None of the difficulties from participants or on the expert educator list related to issues with orientation, which may arise less when discussing specific computing topics since issues related to orientation can occur across multiple topics.

Participants described six methods for supporting student difficulties with linear data structures, only one of which was also mentioned on the expert educator list (see Table 4.4). Methods provided by participants related mostly to presenting information. Other methods provided by participants related to sequencing content, representations, and one generic problem-solving task. Only two instructional strategies were reported by multiple teachers: generic problem-solving tasks and generic descriptions of presenting information. Fourteen methods on the expert educator list were not mentioned by any participants (see Table 4.5). Most methods provided on the expert educator list described specific problem-solving tasks. Other methods on the expert educator list related to representations, real-life examples, relating content to other domains, sequencing content, and presenting information.

Table 4.2

Student Difficulties with Linear Data Structures

ID	Difficulty	Difficulty Type
DE1	Students think that arrays start at index 1 instead of 0. Students assume that the element at index 3 and the 3rd element always refer to the same element.	Notation
DE2	A common misconception students have with arrays is getting rows and columns backwards. When looping over arrays, mixing up these parameters can cause some very strange and confusing bugs.	Pragmatics or Structures
DE3	Putting parenthesis after the length method for Arrays in Java, it's just length without parenthesis. Students get really confused here because the Java method size() for ArrayList has parentheses.	Notation
DE7	Students think you can use the 'item (any) of (list)' block in Snap! and Scratch to check for every item in a list, but this block actually returns a random item.	Notation
DP1	Length of 2D array vs length of a row.	Pragmatics or Structures
DP2	Null pointer exceptions.	Pragmatics or Structures
DP3	Closing the gaps when removing an item from a list. Moving values but preserving order in a list (adding/removing items to list).	Pragmatics or Structures
DP4	How to initialize array with desired length, with pre-set elements.	Notation
DP5	Discerning between the index number and the corresponding data value. Students try to modify the number of times an event occurs by adding 1, but they code the addition to the value instead of to the frequency of that value's occurrences.	Pragmatics or Structures
DP6	Not realizing that a new list has no elements.	Notation
DP7	Not realizing the change-by-one block in Snap! means add 1.	Notation
DP8	When students are coding they often forget about the contains block in Snap!	Notation
DP9	If asked to rewrite a script with repetitive blocks using lists, some students will create a list but not make use of the list in a loop.	Pragmatics or Structures

Note: Difficulties prefaced with DE were also included on the expert comparison data source. Difficulties prefaced with DP were only mentioned by case study participants.

Table 4.3

Student Difficulties with Linear Data Structures Not Mentioned by Participants

ID	Difficulty	Difficulty Type
DE5	Students may think that assigning one array to point to another array makes a copy of that array, failing to make a distinction between shallow and deep copies.	Notional
DE6	Students have difficulty working with temporary variables in arrays.	Pragmatics or Structures
DE8	Reinforce that certain types in Python such as lists are mutable while others (e.g., strings, tuples) are not. Students have difficulty tracing code that uses mutable types because they often forget this.	Notional
DE9	One of the first obstacles students need to overcome when learning about arrays is that there is only one name for several places where to store values.	Notation

Table 4.4

Methods of Addressing Student Difficulties with Linear Data Structures

ID	Method	Method Type
ME1	When teaching null dereferences, show a call that dereferences a null pointer in a debugger to give students a snapshot of what is happening.	Present information
MP1	More practice.	Problem solving task
MP2	Labeling 0, 1, 2, etc. on diagrams on the board. While tracing, labeling the arrays. I draw diagrams and encourage them to draw too.	Representations
MP3	I teach arrays first, so they can see the point of the List, but some books/teachers do it the other way.	Sequencing
MP4	Offering choices of methods available to amend a list.	Present information
MP5	I try to keep mentioning that [a 2D array] is just an array of arrays.	Present information
MP7	Present information to students (without details on what is presented).	Present information

Note: Methods prefaced with ME were also included on the expert comparison data source. Methods prefaced with MP were only mentioned by case study participants.

Table 4.5

Methods of Addressing Student Difficulties with Linear Data Structures Not Mentioned by Participants

ID	Method	Method Type
ME2	Have students translate between Java Array and ArrayList to highlight the differences between the two.	Problem solving task
ME3	Use a visual proof to demonstrate that the better strategy for resizing arrays is doubling the size to make it easier for students to understand.	Representations
ME4	Tell students that the data structure linked lists were used for the human genome sequencing project to motivate the value of this structure and increase interest.	Real life example
ME5	Have students brainstorm about the data structure Pandora uses for playlists to help motivate ArrayLists or linked lists through comparison to everyday life.	Real life example
ME6	Have students answer questions like: arr[2][3] + arr[1][4]. This is important because students may have difficulty using the values they get from array references in other calculations.	Problem solving task
ME7	Starting from a natural model, such as a text string. Students are given a text string and asked to do something with it with many letters. Then reaching lists of strings, as for example list of names, which means an array of strings.	Problem solving task
ME8	Providing challenging examples where the index is an array element, implying several brackets, like: a[[[1]]]	Problem solving task
ME9	Finding the maximum value of the array or the place of the maximum value. This implies asking the students the two different aspects of an array: what is the index and what is the value.	Problem solving task
ME10	It could be possible to collaborate with a mathematics teacher to synchronize the teaching of arrays with the teaching of coordinates (coordinate x).	Relate to other domains
ME11	Give commands to an imaginary person sitting in an empty desk to model what happens when you give commands to an object that hasn't been instantiated to help students understand null pointer exceptions. This activity helps students realize one of the common situations where a null pointer exception might incur.	Representations
ME12	Use tangible examples to demonstrate arrays (e.g., egg carton, seed sorter, linked carabiners, mail postboxes, Dixie cups, pill organizers, dresser of clothing).	Representations
ME13	Do a problem requiring students to loop through arrays/lists.	Problem solving task
ME14	Have students implement a larger program that requires arrays/lists (e.g., nxn Magic Square, mazes, seam carving, plot pictures on grid, Four rotations, time counters).	Problem solving task
ME15	Do loops before arrays/lists.	Sequencing

Next, responses were reviewed by teacher to discern any similarities or differences across participants based on their prior CS teaching experiences. The Intro teachers identified student difficulties during think-aloud interviews not reported on the PCK questionnaire (Mr. Miller discussed DP6 and DP7; Ms. Jones discussed DP8, DP9, and DE7). Since the AP teachers did not complete similar think-aloud interviews, this cross-teacher comparison focused only on ideas shared on the PCK questionnaire that all four participants completed. Table 4.6 shows the number of unique difficulties and methods each teacher reported on the PCK questionnaire and the number of their responses also included on the expert educator list. Table 4.7 lists the specific difficulties and methods mentioned by each teacher. Ms. King, a more experienced AP CS teacher, provided the most unique student difficulties, the most methods, and the most responses also mentioned on the expert educator list. Ms. Robinson, a less experienced AP CS teacher, provided one unique student difficulty, one unique method, and one response that was also on the expert educator list. In contrast, the two Intro teachers, despite having different levels of CS teaching experience, produced a similar number of responses. Ms. Jones provided two student difficulties and two methods; Mr. Miller provided three student difficulties and three methods.

Table 4.6

Student Difficulties and Methods for Linear Data Structures Mentioned by Participants

Teacher	Unique Difficulties	Difficulties on Expert List	Unique Methods	Methods on Expert List
Ms. Robinson	1	1	1	0
Mr. Miller	3	1	3	0
Ms. Jones	2	1	2	0
Ms. King	6	3	5	1

Table 4.7

Student Difficulties and Methods for Linear Data Structures Mentioned by Participants

Teacher	Difficulties	Methods
Ms. Robinson	---	MP7
	DE3	---
Mr. Miller	DP3, DP5	MP1, MP4, MP7
	DE1	---
Ms. Jones	DP4	MP1, MP7
	DE1	---
Ms. King	DP1, DP2, DP3	MP1, MP2, MP3, MP5
	DE1, DE2, DE3	ME1

Note: Within each cell, items listed on the top row were only mentioned by study participants. Items on the bottom row were also mentioned on the expert lists.

Lastly, teachers described the types of lessons and activities they would plan around linear data structures and a rationale for those lessons. Mr. Miller and Ms. King each described three to four materials from published sources (e.g., worksheets from APlus, TEALS lessons, exercises from Cay Horstmann's Big Java textbook, Udacity videos). Ms. Jones listed four generic lessons and activities (e.g., worksheets on manipulating arrays, lesson on 2D arrays) and five lab projects that involved manipulating arrays (e.g., Array Statistics to calculate descriptive statistics for an array of integers; Index Product to create a 2D array where elements are the products of their indices). Instead of describing lessons or activities, Ms. Robinson offered three examples of what arrays can represent (i.e., concert seating, store warehouse, student information). Participants offered three reasons for their lessons and activities (see Table 4.8). All four teachers said students need to gain practice using linear data structures. Mr. Miller, Ms. Jones, and Ms. King made references to vertical content knowledge and how students need to learn about arrays and lists because they relate to other projects or course topics. Ms. Jones also provided direct instruction as a rationale for one of her lessons.

Table 4.8

Rationale for Lessons/Activities on Linear Data Structures

Rationale	Examples	Number of Participants
Student practice	• Practice for learning the notation and the difference between arrays and ArrayList • Additional Practice and application. • Students need further practice with iteration. The worksheet allows for targeted problems where students can focus on one idea at a time. • Again students need to repeat setting up looping through lists-- practice makes perfect!	4
Direct instruction	• In order to use arrays, students need to have an introduction to it.	1
Vertical content knowledge	• Students need to learn about navigation and manipulation of 2D arrays in order to do some larger projects. • It's a common use of traversing a list.	3

Summary. Data gathered from the PCK questionnaire prompts related to student difficulties showed participants had greater knowledge of student difficulties than instructional methods and that their ideas differed from the expert educator list. Participant ideas about student difficulties focused equally on issues with notation and issues with pragmatics or structures. In contrast, ideas on the expert educator list focused mostly on issues with notation and also included issues with the notional machine. While participant ideas about methods for addressing student difficulties focused mostly on presenting information, ideas on the expert educator list focused mostly on problem solving tasks. The expert educator list also included sharing real-life examples as a method of supporting students, which participants did not mention in their responses. Participants offered ideas about student difficulties and instructional methods that did not appear on the expert educator list.

When describing their rationales for using lessons or activities to teach linear data structures, participants explained that students need practice with the topic and that the topic was related to other course content. Comparing their rationales to the K-12 Computer Science Framework ("K–12 Computer Science Framework," 2016), participants' ideas focused exclusively on developing student content knowledge and not on core practices that learners should use when engaging with CS (e.g., collaborating, communicating about computing).

Across participants, the data show a slight correlation between the knowledge shared on the questionnaire and their amount of CS teaching experience. Ms. Robinson, an AP teacher who had less CS teaching experience, offered the fewest ideas related to student difficulties and instructional methods. She was also the only participant who did not discuss vertical content knowledge when explaining her rationale for lessons on linear data structures. Ms. King, an AP teacher who had more CS teaching experience, offered the most ideas related to student difficulties and instructional methods and the most ideas common to the expert educator list. However, the Intro teachers, Mr. Miller who had less CS experience and Ms. Jones who had more CS teaching experience, offered similar numbers of ideas that were greater than those offered by Ms. Robinson and less than those offered by Ms. King.

4.4.2. Most Difficult Computing Topics

Teachers were asked to identify the most difficult topics in their courses and how they address the topics differently (see Table 4.9). Topics listed by participants all fall within two of the five K-12 Computer Science Framework concepts: (a) computing systems and (b) algorithms and programming. This is expected given the focus on programming in both the Intro and AP

Table 4.9

Most Difficult Computing Topics Covered in Teachers' Courses

Teacher	Topics	Ways of Addressing Topics	Related K12CS Topics*
Ms. Robinson	• Arrays • Binary • Boolean logic • Hex conversion • Decimal • Sorting • Strings	Additional practice and time	• 9-12.AP.Algorithms • 9-12.AP.Variables • 3-5.AP.Variables • 3-5.AP.Control
Mr. Miller	• Methods • Objects	Nothing different from how other topics are taught	• 9-12.AP.Modularity • 6-8.AP.Modularity
Ms. Jones	• Recursion	Varied instructional strategies	• 9-12.AP.Control
Ms. King	• Inheritance • Recursion	Additional practice and time	• 9-12.AP.Modularity • 9-12.AP.Control

*AP is an abbreviation for Algorithms and Programming

courses. Responses from Mr. Miller, Ms. Jones, and Ms. King relate to two subconcepts students should master by either eighth grade or twelfth grade: control and modularity. The topics listed by Ms. Robinson cover three different subconcepts that students should master by either fifth grade or twelfth grade: algorithms, variables, and control. Ms. Robinson also mentioned binary and hexadecimal conversion as a difficult topic. The K-12 Computer Science Framework only mentions this as a computing systems topic that should not be prioritized at the elementary level. Noticeably absent from the teachers' responses was the subtopic of program development which covers concepts such as planning, testing, and debugging. Researchers have identified program planning as an area of difficulty for students who struggle to put pieces of code together, struggle tracing code, and spend little time planning and testing their programs (Robins et al., 2003). While teachers did not mention program planning as a most difficult topic, they did comment on this challenge during their interviews. For example, Ms. King once said, “So we've done pseudocoding and even a short exercise it is hard for them to think in pseudocode. What they do is they code it and then they write the pseudocode. Or they write the pseudocode very close to code” (Ms. King, December 4, 2015).

Participant responses were compared against Schulte and Bennedsen’s (2006) list of twenty-eight difficult topics in introductory programming courses. All of the topics listed by Mr. Miller, Ms. Jones, and Ms. King were included on the Schulte and Bennedsen list; only two of the seven topics listed by Ms. Robinson appear on the list (i.e., arrays, strings). The topics listed by Ms. Jones and Ms. King are ranked as the first and fifth most difficult topics on the Schulte and Bennedesen list. Topics listed by Mr. Miller were ranked as the fourteenth and twentieth most difficult topics. Ms. Robinson’s topics were ranked at position twenty-two. Participant responses were also compared against Dale’s (2006) list of sixteen difficult topics in introductory programming courses. Only one of Ms. Robinson’s topics – arrays – appeared on Dale’s list, but it was the fifth most commonly reported topic. The topics listed by Mr. Miller, Ms. Jones, and Ms. King were ranked at positions seven, nine, ten, eleven, and fifteen. For the most part, participant responses showed a similar pattern when compared against Schulte and Bennedesen’s ranking and Dale’s ranking. However, the responses provided by Ms. Robinson shifted from one of the less difficult topics on Schulte and Bennedesen’s list to one of the more difficult topics on Dale’s list. Another noticeable pattern related to the most difficult computing topics was the number of responses provided. While most teachers listed one or two topics related to the concepts of modularity and control that all appeared on comparison lists, Ms. Robinson listed seven topics spanning multiple concepts, only two of which were included on comparison lists.

Participants were also asked how they address the most difficult topics of their courses differently than other subject matter. Ms. Jones discussed instructional approaches that were specific to the topic of recursion, for example:

> We show interesting recursive images and develop some analogies for recursion based on these images. We have a culture day/“CS unplugged day” on fractals. We give very structured and well guided projects for students to make that involve recursion, and we offer a lot of one-on-one support. There is less freedom on these projects, and more support. Also, we do not go too in depth into recursion as we know the topic is difficult and we just want to give the students an introduction.

Both AP teachers, Ms. Robinson and Ms. King, wrote that they would devote more time and student practice to the most difficult topics in their courses. Lastly, Mr. Miller said he did not treat objects and methods any differently than other topics. Ms. Robinson and Ms. King mentioned generic approaches that could apply to any topic while Ms. Jones provided ideas that were specific to the topic of recursion. This may suggest that Ms. Jones had a greater understanding of instructional strategies for the topic she identified as most difficult and that Ms. Robinson and Ms. King were still developing computing specific instructional strategies for the topics they identified. Lastly, Mr. Miller did not do anything differently when teaching what he considers the most difficult course topics. This might suggest that he could identify which topics are challenging for students but he was still learning how those topics can best be presented to support student understanding.

Summary. Data gathered from the PCK questionnaire prompt about the most difficult course topics showed a mixed picture of the participants' teaching knowledge. When considering the ranking of most difficult topics, the knowledge of Ms. Robinson, the AP teacher with less CS teaching experience, differed from the knowledge of the other teachers. Ms. Robinson listed several topics as the most difficult topics in her course, while the other participants listed either one or two topics. Only a subset of Ms. Robinson's topics were included on the comparison lists, while all of Mr. Miller, Ms. Jones, and Ms. King's ideas were included on both comparison lists. When looking at methods used to teach the most difficult topics, Ms. Jones, an Intro teacher with more CS teaching experience, offered strategies specific to CS. In contrast, Ms. Robinson and Ms. King offered generic strategies and Mr. Miller offered none. Lastly, the participants focused on CS concepts when identifying the most difficult topics in their courses and not on strategies used to work with concepts.

4.4.3. Content Knowledge

Effective teaching requires knowledge of concepts and procedures and the ability to explain that knowledge to others and so content knowledge can serve as a gauge of preparedness for teaching. Teachers' content knowledge was examined through their performance on the content assessment and their responses to think-aloud interviews.

Ms. Robinson correctly answered 41% of the items on the AP assessment at the end of the school year. She incorrectly answered ten items related to variables, loops, program logic, arrays, input and data types, objects, classes, sorting algorithms, and recursion. While most teachers performed better than their students, Ms. Robinson scored slightly lower than her students' average score of 42%. Her responses to six items across two think-aloud prompts related to loops and conditionals showed a stronger understanding of the AP course topics. She correctly reasoned through four items. Her responses were partially correct for two assessment items where she made two notational errors related to Java syntax. First, when critiquing an assessment item about for-loops, she correctly identified the number of times three of the four loops executed but she made an off-by-one error for one of the loops. Second, when critiquing an assessment item about conditionals, she identified all the conceptual errors in the problem but overlooked a syntax error in the proposed solution that incorrectly used an equal sign (=) instead of a double equal sign (==) to test equality.

Mr. Miller correctly answered all items on the Intro assessment at the end of the school year, performing better than his students who averaged a score of 45%. His responses to nine items across three think-aloud prompts related to loops, conditionals, and arrays showed a partial understanding of some Intro course topics. He correctly reasoned through six items related to loops, conditionals, and lists. His responses were partially correct for three items. First, in evaluating assessment items related to conditionals, he misinterpreted one of three English statements that were to be translated into conditional statements in Snap! code. He thought the statement ‘x-position and y-position are of opposite signs’ meant make each value have the opposite sign; the statement is asking students to test if the position values are of opposite sign. Second, while evaluating another assessment item where students were asked to identify and correct errors in a coding solution that reported if an age was even and between 10 and 100, he correctly identified all the conceptual errors of the program but overlooked a syntax error that used the less than (<) and greater than (>) operators instead of less than or equal (≤) and greater than or equal (≥) operators to test for inclusion. Third, in reviewing a student solution to the task of drawing a square where opposite sides were the same color, he correctly identified what the student’s first loop should accomplish but he did not explain how the two loops could be revised together to produce the desired result.

Ms. King correctly answered 82% of the items on the AP assessment. She incorrectly answered three items related to variables, data types, and objects. She performed better than her students who averaged a score 58% on the exam. Her responses to seven items across two think-aloud prompts showed a partial understanding of conditionals, loops, and variables. She correctly reasoned through four items including one assessment item about loops and three student solutions related to a task about variables. Her responses to three items were partially correct. First, she incorrectly solved a problem where students were asked to replace three assignment statements with one line of code, but she did provide rationales for incorrect student solutions to the problem. Second, when critiquing an assessment item about conditional statements, she correctly identified that the code returned the maximum of three numbers but then discussed how the code might not execute as expected if the list of three items contained duplicates; this issue would not occur in the coding solution she reviewed. Third, when critiquing another assessment item about conditionals, she identified all the conceptual errors in the problem but overlooked a syntax error in the proposed solution that used an equal sign (=) instead of a double equal sign (==) to test equality.

Ms. Jones correctly answered 93% of the items on the Intro assessment. She answered one item related to recursion partially correctly (i.e., she selected one correct response out of three correct responses). She performed better on the assessment than her students who averaged

60% on the assessment. Ms. Jones responded to six items across two think-aloud prompts related to arrays and variables and answered all items correctly.

Summary. As with responses related to the most difficult course topics, data gathered from the content assessments and think-aloud interviews showed a mixed picture of the participants' teaching knowledge. Regarding the content assessment, Ms. Robinson, the AP teacher with less CS teaching experience, scored significantly lower than the other three participants. Mr. Miller, an Intro teacher with less CS teaching experience, received a perfect score on the content assessment. Ms. King and Ms. Jones, both teachers with more CS teaching experience, made some errors on the content assessment. Regarding the think-aloud interviews, Ms. Jones, the Intro teacher with more CS teaching experience, demonstrated the most complete content knowledge while the other three teachers showed partial understanding.

Across participants, discussions of student solution prompts were more complete than discussions of assessment item prompts. While think-aloud interviews provided opportunities for richer explanations, they may not be well suited for assessing knowledge of syntax. Three teachers – Ms. Robinson, Ms. King, and Mr. Miller – discussed assessment items that asked students to identify all the errors in a coding solution. All three teachers identified the conceptual errors in the code, but they each overlooked purely syntactical errors related to comparison operators. The think-aloud interview format may be less conducive to noticing syntactical issues because such errors can be easily overlooked when reading code outside of a programming environment.

4.5 Discussion

Results from the PCK questionnaire, think-aloud interviews, and content assessments were used to explore one research question: what knowledge of CS content and student thinking do in-service CS teachers develop while participating in an on-the-job PD program? Results show a range of content knowledge and PCK across participants, with certain instruments eliciting more evidence of teacher knowledge than others. The results of this section show that teachers seem to know more about student difficulties than instructional strategies specific to computer science and that they are still developing their content knowledge. Also, the data suggest that participant knowledge may not simply be a subset of expert educator knowledge, because participants expressed ideas about student difficulties and teaching methods that did not appear on the expert educator list. It is possible that expert educators are aware of these ideas and did not mention them, have integrated them with other ideas, or no longer consider them useful in their teaching. To explore this more, future studies might investigate how ideas about student conceptions and instructional methods change as teachers gain more experience in CS classrooms, a better understanding of the discipline, and mastery of the subject-matter and related practices.

Regarding the PCK questionnaire, one noticeable difference between the participant responses and the expert educator list was the use of real world examples (e.g., linear data structures were used in the human genome sequencing project) and metaphors (e.g., linear data structures are like egg cartons). Participants did not mention these examples when discussing how to support students' understanding of linear data structures. Examples from the real world and metaphors play an important role in CS teaching as they help students form mental models of CS concepts (Hazzan et al., 2015; Woollard, 2005). Teachers new to CS may not have knowledge of real world examples and useful metaphors to draw upon if they do not have much experience with CS outside of their classrooms. Unlike knowledge of student conceptions that teachers can gather through their interactions with learners, the classroom may not provide new CS teachers with the experiences necessary to accumulate a stock of these examples or metaphors that are specific to course content. Furthermore, it is possible that transitioning teachers will develop their own metaphors of CS concepts, but that the metaphors may be flawed if they do not deeply understand the concepts they want to represent. Thus, the acquisition of real world examples and metaphors related to CS concepts may need to be a focal point of training for transitioning teachers.

Although only four teachers participated in this component of the study, the data they provided suggests there may be at least two stages of teacher knowledge development through which transitioning CS educators progress. Ms. Robinson's teaching knowledge seemed different than other teachers. She listed fewer student difficulties, identified several topics as the most difficult course topic, and received a low score on the content assessment although she explained many items correctly during think-aloud interviews. Mr. Miller, Ms. Jones, and Ms. King appeared to be at a different developmental stage because they possessed greater knowledge of student conceptions and course content. Although these three teachers appeared to be at a similar stage, their content knowledge and PCK varied in different ways. Teaching knowledge looks different across teachers and depends on the specific contexts within which teachers work, so I would not expect the results of this section to be uniform or show a progression towards the same point. However, future work might investigate a larger pool of transitioning CS teachers to determine if there are different profiles of CS PCK development that could distinguish teachers like Mr. Miller, Ms. Jones, and Ms. King and identify more targeted areas of need related to their individual PCK development.

Limitations. I used three different data sources to investigate teachers' content knowledge and PCK, but they did not all provide the same portrait of CS PCK within or across teachers. Consider the responses of Ms. King and Ms. Jones. While Ms. King displayed the strongest CS PCK when identifying student difficulties and instructional strategies related to linear data structures, she only provided generic pedagogical strategies when discussing the most difficult computing topics. In contrast, Ms. Jones identified computing specific instructional strategies when discussing the most difficult computing topics, but she provided fewer student difficulties and instructional strategies than Ms. King when discuss linear data structures. As another example, consider the results of Ms. Robinson and Mr. Miller on the content knowledge instruments. Ms. Robinson performed poorly on her content assessment, but she could articulate more PCK ideas during think-aloud interviews. Mr. Miller, on the other hand, received a perfect score on his assessment, but displayed partial understanding in the think-aloud interviews. Lastly, both Ms. Jones and Mr. Miller identified student difficulties with linear data structures during their first semester interviews that they did not include on their end-of-year PCK questionnaire.

Information gathered from these sources might suggest, as echoed in the literature (e.g., Baxter & Lederman, 1999), that there are limitations to relying on one instrument to measure teaching knowledge. However, it may also be the case that the instruments I created, and used for the first time in this study, suffer from measurement validity issues which can be addressed.

While there may not be one perfect instrument to capture a complete picture of CS PCK at different stages of development, it may be possible to improve the think-aloud interviews, PCK questionnaire, and content assessment to provide more consistent results across instruments.

CHAPTER 5. INSTRUCTIONAL RESPONSIBILITIES

5.1 Introduction

A teacher's role is multifaceted and involves many responsibilities ranging from subject specific tasks (e.g., preparing instructional materials), general pedagogical activities (e.g., classroom management), and other professional obligations (e.g., coaching extracurricular teams). New teachers are often overwhelmed when confronted with all these responsibilities once they enter the classroom and find it difficult to attend to relevant events (Feiman-Nemser, 2003; Feldon, 2007). Focusing on responsibilities during training can better equip educators to implement instructional tasks once they begin new teaching assignments (D. L. Ball & Forzani, 2009). But which responsibilities should teachers focus on during their training to best support their professional development?

In the TEALS program, participants distribute responsibilities across their team to facilitate teacher learning. This co-teaching arrangement provides the opportunity to explore how teaching tasks relate to teacher development within authentic classroom environments. While TEALS offers guidance on the distribution of responsibilities for their PD program, teams implement their daily teaching tasks differently depending on factors such as teacher readiness, volunteer availability, and the type of learning environment used in the classroom. One goal of this dissertation was to identify the specific responsibilities teachers assumed while participating in the PD and explore how these responsibilities supported PCK development. Drawing on self-reported data of instructional responsibilities, observations, interviews, and questionnaire responses, I explore the following questions: *what instructional responsibilities do teachers undertake when planning and implementing their lessons within the conditions of their co-*

teaching partnerships? and *how does teacher knowledge support the implementation of instructional responsibilities?*

5.2 Research on Instructional Responsibilities

In the past few years, some researchers have renewed interest in bringing instructional responsibilities to the fore of teacher education and providing teachers with opportunities to engage in elements of interactive teaching (Lampert, 2010). Ball and Cohen (1999) proposed a collaborative, inquiry-oriented model of professional learning that is centered around the tasks and artifacts of instruction providing educators with opportunities to analyze and improve their craft. Grossman and McDonald (2008) suggested the field focus on *pedagogies of enactment* where teachers develop skilled practice through performing tasks, receiving feedback, and attempting the tasks again. Kazemi, Lampert, and Ghousseini (2007) suggested decomposing teaching into *routines of practice* that both novice and expert educators could perform to become what they term ambitious teachers who support the success of all students.

Essential to a focus on practice in teacher training are frameworks of instructional routines that provide a common vocabulary to describe tasks specific to particular subject matter and tasks common across grade levels and subjects (Grossman & McDonald, 2008). Some of these tasks include leading a discussion, assessing student knowledge, presenting ideas, finding examples to make a specific point, connecting a topic to topics taught in prior or future years, appraising content on instructional materials, modifying tasks to be easier or harder, and selecting representations (D. L. Ball et al., 2008; Kazemi et al., 2007). Several groups of researchers have worked towards dividing instructional responsibilities into high-leverage practices that can improve student achievement, occur regularly in teaching, and support teacher understanding of students and their craft (Grossman et al., 2009). However, there is still no consensus around which teaching tasks should form the core of practice-based professional development (Forzani, 2014).

5.3 Method

5.3.1. Participants

Six teachers who varied in their professional development stages participated in this component of the case study. Ms. Robinson, was in the volunteer-led stage. She relied heavily on volunteers to lead her AP course during the first term and she began transitioning into a lead role on her teaching team during the second term. Two teachers, Ms. Jones and Mr. Miller, were in the collaborative stage. They worked with volunteers to plan, implement, and revise their courses throughout the year. The remaining three teachers, Ms. King, Mr. Edwards, and Mr. Perez, were in the teacher-led stage for the entire school year. They directed their courses receiving support from volunteers to assist students, grade assignments, and explain topics unfamiliar to the teachers. It is important to consider each participant’s stage in the PD program when examining their responsibilities, because their progress in the program will influence how they share teaching tasks with their volunteers.

5.3.2. Data Collection and Analysis

At the end of each visit, teachers were asked to identify who on their teaching team contributed to instructional responsibilities related to lesson preparation (i.e., developing lessons, creating assignments), instructional delivery (i.e., delivering lessons, managing the classroom), and evaluation of learning (i.e., assisting students, grading student work). Option choices were: mostly teacher, both volunteers and teacher, mostly volunteer, and no one. I converted these levels into numeric values in order to average engagement in instructional responsibilities over the school year. Since I was most interested in the degree to which teachers assumed instructional responsibilities, I collapsed the levels of *no one* and *mostly volunteers* to a value of 0, assigned *both volunteers and teachers* a value of 1, and assigned *mostly teacher* a value of 2. Ms. Robinson, Mr. Edwards, and Mr. Perez only completed five of their six post-lesson questionnaires; Ms. Jones, Ms. King, and Mr. Miller completed all of their post-lesson questionnaires.

Observation data were compared against participants' self-reported information in order to determine the degree to which observers and participants shared a common understanding of instructional responsibilities. Observation data was compared against self-reported data for the thirty-three visits where teachers completed the questionnaire item about responsibilities. Due to changes in the observation protocol between the first and second semesters, I only included the observation protocol categories gathered across the entire school year. Percentages of delivering instruction and assisting students gathered from the observation data were compared against the closed-ended, post-lesson questionnaire items. Questionnaire data were deemed in agreement with observation data if (a) a teacher indicated mostly volunteers completed a task and the observer noticed volunteers performing the responsibility a greater percentage of class time than the teacher, (b) a teacher indicated both teacher and volunteers completed a task and the observer noticed both members of the instructional team performing the responsibility during class time, or (c) a teacher indicated mostly the teacher completed a task and the observer noticed the teacher performing the responsibility a greater percentage of class time than volunteers.

To explore responsibilities in more depth and to identify other teaching tasks assumed by participants, I analyzed reflections offered during interviews and on open-ended questionnaire items. Participant comments were coded into one of nine responsibilities grouped into two major categories: (a) planning and organizing lessons and (b) implementing and monitoring instruction. (see Table 5.1). The coding scheme was based on Ball, Thames, and Phelps' (2008) list of mathematical tasks of teaching, TEALS documentation describing the tasks expected of teaching teams, and responsibilities discussed by case study participants.

Table 5.1

Coding Scheme of Instructional Responsibilities

Category	Definition
Plan and Organize Lessons	
Find materials	Search for and evaluate instructional materials
Create materials	Create instructional materials
Modify materials	Modify instructional materials created by others
Review materials	Look over existing instructional materials
Practice materials	Complete tasks that will be assigned to students
Organize lesson	Decide on the timing, pacing, sequencing of lessons
	Decide on student grouping for group work
Implement and Monitor Instruction	
Assist students	Provide help to students
Evaluate learning	Assess students, assign grades, review student progress
Present ideas	Present CS information, give CS explanations, whole class instruction

I used Ball and Cohen's (1999) model of professional learning as an interpretive guide to conduct an exploratory analysis into the relationship between responsibilities and teaching knowledge. Their model highlights how many teaching tasks can support an examination of practice because "in the course of these tasks, teachers may puzzle, weigh alternatives, draw on what they know or can access as resources for judgments and decisions" (D. L. Ball & Cohen, 1999, p. 14). I examined how different teaching tasks allowed for active noticing, interpretation, and working with artifacts of practice. For example, when searching for assessment items (i.e., find materials), did participants compare multiple sources or consider how well the assessments fit with material covered in their course? Although teachers were asked about their instructional responsibilities and their PCK, they were not asked to explicitly discuss how their responsibilities supported their teaching knowledge development. So, I looked through the coded questionnaire and interview data for instances where teachers discussed their responsibilities to see if they also made comments about their PCK. While just a small number of comments were found, they give ideas for factors to consider when looking at the relationship between specific responsibilities and PCK development.

5.4 Results

5.4.1. Instructional Responsibilities

Teachers' self-reported instructional responsibilities, averaged across the school year, are displayed in Figure 5.1. All teachers reported undertaking each of the six responsibilities in their CS classes to some degree. As expected, participants in the teacher-led PD stage reported greater responsibility of both lesson preparation tasks than participants in the volunteer-led or collaborative PD stages. However, the assumption of responsibilities related to instructional delivery varied across PD stages. Ms. Robinson, Ms. Jones, and Mr. Edwards, each in a different PD stage, reported sharing the responsibility of delivering lessons with their volunteers; Mr. Miller, Ms. King, and Mr. Perez reported greater independent responsibility of this activity. Apart from Mr. Edwards, all teachers bore the brunt of managing their classrooms, which is recommended in the TEALS model. The assumption of responsibilities related to evaluation of learning also varied across PD stages. The task of assisting students was reported more frequently by participants in the volunteer-led and teacher-led PD stages and less frequently by participants in the collaborative PD stage. Mr. Perez and Ms. Jones reported less responsibility for the task of grading student work than the other four teachers.

This first step in the analysis served to identify the frequency with which participants undertook instructional tasks and differences in these frequencies based on PD stage. Two tasks did not relate directly to PD stage: assisting students and grading student work. The self-reported data suggests that participants followed the ideal TEALS implementation regarding lesson preparation tasks and, for the most part, delivering lessons (i.e., teachers with a greater lead role in their classrooms assumed these tasks more frequently). The one exception to this pattern was Mr. Edwards whose pattern of response for all instructional tasks seemed different than other teachers. He was the only participant to report the same average frequency of responsibility for every instructional task across the school year. In the next section, I explore this anomaly and others by comparing the participants' self-reported data with observational data.

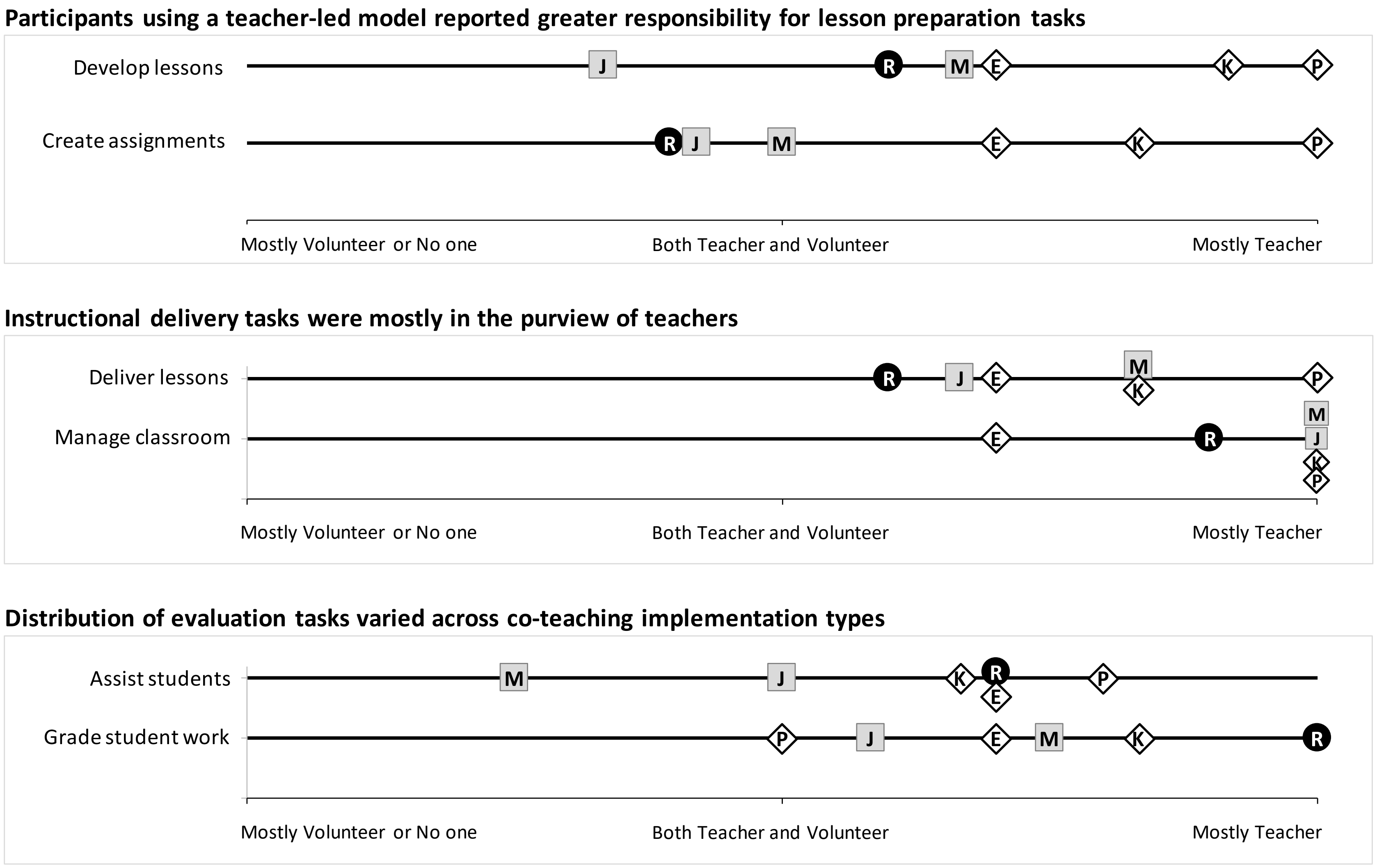


Figure 5.1. Self-reported distribution of instructional tasks averaged across the school year. Rating options: mostly volunteer (0), no one (0), both teacher and volunteer (1), and mostly teacher (2). Co-teaching model: Volunteer-led (●), Collaborative (■), Teacher-led (◊). Letters within symbols indicate the first letter of the teacher's last name.

Observed instructional responsibilities. Since multiple researchers have cautioned against relying on teacher self-report data (e.g., Randolph et al., 2008), I compared the agreement between self-reported data gathered on the post-questionnaire and observation data (see Table 5.2). For the task of delivering lessons, agreement of teacher ratings and observation data was low when teachers indicated that both they and their volunteers shared the task. For the task of assisting students, agreement of teacher ratings and observation data was low when teachers indicated volunteers were primarily responsible for the task.

Table 5.2

Visits Where Teacher Ratings Agree with Observation Data on Distribution of Instructional Tasks

	Delivering Lessons		Assisting Students	
Teacher Rating	Agree With Observer	Disagree With Observer	Agree With Observer	Disagree With Observer
Mostly Volunteer	3	0	1*	3
Both Teacher and Volunteer	1*	6	14	5
Mostly Teacher	19	4	8	2
Totals	23	10	23	10

*Cases where agreement between self-reported teacher ratings and observation data were less than the number of disagreements.

Three types of discrepancies occurred in how observers and teachers rated instructional responsibilities for the same lesson. First, Mr. Edwards and Ms. Jones indicated on multiple questionnaires that they shared a responsibility with volunteers when no volunteers were present in the classroom. While I asked teachers to focus on the class observed this study, it may be that when Mr. Edwards and Ms. Jones reported on responsibilities, they also considered other class sessions with no volunteers that occurred the same day as case study observations where volunteers were present. Also, Mr. Edwards recruited student helpers for his course who he may have considered as volunteers when completing the surveys. These student helpers were present in two of the three discrepant lessons.

Second, Ms. King, Ms. Jones, and Mr. Perez reported that they were mostly responsible for delivering lessons during class periods that were devoted entirely to lab time or student presentations. Observers did not count teachers' actions during these lessons as leading the class or direct instruction. So, it seems teachers and observers interpreted lesson delivery differently.

Lastly, Ms. Jones and Mr. Miller sometimes disagreed with observers about who delivered lessons and who assisted students, respectively. On three occasions, Ms. Jones reported that both she and her volunteers delivered the lesson. However, the observer visiting Ms. Jones' classroom saw volunteers involved mostly in assisting students. The observer only noted volunteers clarifying an error in Ms. Jones' lecture during one visit. As noted above, Ms. Jones may have considered other class sessions or her volunteers' role in creating the lessons when reporting on this responsibility. On three occasions, Mr. Miller reported that volunteers mostly assisted students when the observer recorded Mr. Miller supporting students more often. On another occasion, Mr. Miller reported that he mostly assisted students when the observer recorded equal effort between the teacher and the volunteers for this responsibility. During two of these visit, volunteers conducted code reviews where they spent time with individual students evaluating a past project. During the other two visits, the observer recorded Mr. Miller supporting students for more time than the volunteers. Mr. Miller may have distinguished the ways in which he and volunteers were providing support to students (e.g., formal code reviews versus answering individual questions) and he may have concentrated on a subset of these ways when reporting on the distribution of this responsibility.

While the data showed that participants' self-reported data about the frequency of instructional responsibilities aligned mostly with observation data, there were occasional discrepancies. These discrepancies may have stemmed from teachers looking across classes on observation days or receiving support from students. Discrepancies related to lesson delivery may be explained by an overloaded term. Lesson delivery seemed to encompass both traditional lectures and computer lab time. Computer labs are an integral component of CS courses (Hazzan et al., 2015) and they usually differ in format from direct instruction. However, both are teacher-led activities which should be distinguished when investigating instructional responsibilities in CS.

Examples of instructional responsibilities. In this phase of the analysis, I wanted to better understand the work involved in participants' everyday teaching. During interviews and on open-ended questionnaire items, teachers provided more detail on the activities involved in planning and organizing lessons (see Table 5.3) and in implementing and monitoring instruction. (see Table 5.4). Teachers talked about (a) finding materials through Internet searches and on teaching forums; (b) creating materials like videos or presentations to supplement the TEALS curriculum or address specific student difficulties; (c) modifying materials like test questions or projects to better align with their curricula or to narrow students' focus; and (d) organizing lessons to fit within their course schedule or to better scaffold student understanding. When discussing how they familiarized themselves with instructional materials, teachers talked about (a) reviewing materials by looking over past assignments assigned to students and (b) practicing the projects assigned to students or writing test solutions. Teachers also provided concrete examples of how they presented ideas, assisted students in class, and used different methods of

Table 5.3

Teacher Quotations About Responsibilities Related to Planning and Organizing Lessons

Responsibility	Quotation
Find materials	I've actually been wracking my brain for the last few weeks looking at lots of examples on the internet [about interfaces, abstract classes], and I didn't find any that really went beyond just the very basics. (Mr. Edwards, 9/29/2015)
	TEALS gave us this Python curriculum that they are testing this year for the second semester, and it didn't have these scaffolds. But I found this one, for instance, on the forum that TEALS has. (Mr. Miller, 5/13/2016)
Create materials	[I] wrote some code ahead of time to demo the student. (Ms. Robinson, 2/24/2016)
	We are meeting every Tuesday evening to make the tests and the quizzes and the rubrics and everything, and they are basically making most of the material that I am teaching in class. (Ms. Jones, 2/10/2016)
Modify materials	The Space Invaders [assignment] was from the Berkeley site, and then we just modified it a little bit so that the bricks, or aliens, aren't moving, and so that [the students] could focus on just getting the clones on the screen. (Ms. Jones, 2/6/2016)
	The questions I have chosen from the test bank are ones that I feel they understand because we practiced them and that they are similar to the AP. I will probably rewrite a few questions to make sure it is a little more closely aligned with the AP. (Ms. King, 2/5/2016)
Organize lesson	Yeah, I am like, 'OK, we have a deadline. I want to give them at least two weeks to study, so we need to be done - we won't cover all of the Elevens [project], we'll have to wrap up this Wednesday. Then we need to give the test on unit seven.' (Ms. Robinson, 3/28/2016)
	There is a number guessing game project, drawing a tic-tac-toe board, the brick wall problem. [Also] they have this whole page of script variables projects that also involve conditionals, boolean and conjunctions... So I would want to think about what order I want these things to go in, and where I can give students some choice as to what they are doing and where to slow them down. (Mr. Perez, 9/25/2015)
Practice materials	[My volunteer] had written the code, so I copied the code onto my notes. And so I used that. I try to takes as much notes as possible while the students are maybe typing the code…I try to write the code. (Ms. Robinson, 12/9/2015)
	I found sample scripts with bugs through BJC and other CS teachers, and solved each problem on my own. (Mr. Perez, 9/23/2015)
Review materials	I once again watched all the Khan Academy lessons I've assigned students to watch (Mr. Miller, 2/12/2016)
	I looked at the assignments from last year and reassigned them. (Ms. King, 2/5/2016)

Table 5.4

Teacher Quotations About Responsibilities Related to Implementing and Monitoring Instruction

Assist students	And then, as I walk around, I say 'okay this is what the instructor said'. Then we see the code, 'this is why he did it', or 'this is what I wrote down'. And I try to explain what is going on. So I mean I have an example I am showing it some. (Ms. Robinson, 12/9/2015)
	And that allows myself and the TEALS volunteers to kind of float around the room and answer questions where needed or check in and be like, you know, you guys have been working on this for a long time, what is going on. (Mr. Perez, 9/25/2015)
	I know that different students need more or less scaffolding, so I usually increase the hints as time passes. The first wave just makes it through the benchmark. The next day I'll start out with some tips. The next day, I may show some actual code. (Ms. King, 12/4/2015)
Evaluate learning	We grade it based on the rubric… So while we are looking to see if they capitalized everything correctly and indented and put the braces in the right places and all that, we are definitely seeing the code too, and we can catch things like if they are using arrays or not. Or like, just being inefficient or putting things in the wrong place. (Ms. Jones, 2/10/2016)
	After grading the tests last night I realized we have some real sort of mechanical issues we've covered. In project one we worked with ArrayLists, in project two we switched to arrays, and there is some confusion with about 30% of the kids, and so I wanted to take a step back today (Mr. Edwards, 10/21/2015)
	More than half the students finished their Practice-It, which is a nice aid to assess their knowledge. (Ms. Robinson, 2/24/2016)
	Students who finished early were helping other students. And we are slowly training the early finishers the right way to help. In terms of not just saying, 'you did this and this, you need to do this' or 'type that'. Or grabbing the mouse and typing it for them. We are trying to get the early finishers to ask like 'explain the code to me', 'what do you think you should do here' - try to make the actual people figure it out themselves. (Ms. Jones, 4/4/2016)
	What I announced to second period is I will spot check, I will just randomly pick some projects and call you over, but I would hope that I can get in the semester to everybody at some point. (Mr. Miller, 10/2/2015)
	The classroom teacher met with the TEALS volunteers and we discussed what the students were having trouble with. (Ms. Jones, 2/10/2016)
	I have a collection of exit tickets that assess students' understandings of map and keep that I will look through after completing this survey. (Mr. Perez, 10/23/2015)
Present information	I did some live coding to explain something that they weren't getting. (Ms. King, 2/5/2016)
	I introduce the topic lightly, have the kids try it, and then go over it more in depth once I find out where they need it, because I feel like they listen a lot better after they have already tried it. (Mr. Edwards, 9/23/2015)
	I try to do hands on things, and so even when we were talking about searching stuff, I took the kids outside and we sorted people…because I think some kids needs more of a tactile kind of experience. (Mr. Edwards, 3/9/2016)

formative and summative evaluation of student learning (e.g., code reviews, student peer review, entry tickets).

Coded interview data and questionnaire data were reviewed to identify the number of times teachers talked about each of the nine instructional responsibilities included in the coding scheme to discern patterns in the focus of teachers' comments and their PD stage. Treemaps displaying coding frequencies for each teacher are presented in Figure 5.2. In these maps, the most frequently mentioned tasks are in the upper left corner and shading is used to indicate type of task (i.e., either a planning task or an implementation task). These diagrams can be used to easily see if a participant's coded interview units focused on one type of instructional task more than the other. First, Ms. Robinson, Ms. Jones, and Mr. Edwards, each in a different PD stage, did not discuss all the teaching tasks. Ms. Robinson did not discuss finding materials or modifying materials. She was in her second year of the TEALS program and uncomfortable with many course topics, so she likely relied heavily on the TEALS curricular materials in their original form. During her first case study visit, she said: "We are following along the TEALS lesson plan, a guide. So we are using the work - pretty much following it very closely. Sometimes we don't, but most of the time we do." She also expressed relief during interviews in December, February, and March at having access to pseudocode and problem solutions from her volunteers and from TEALS.

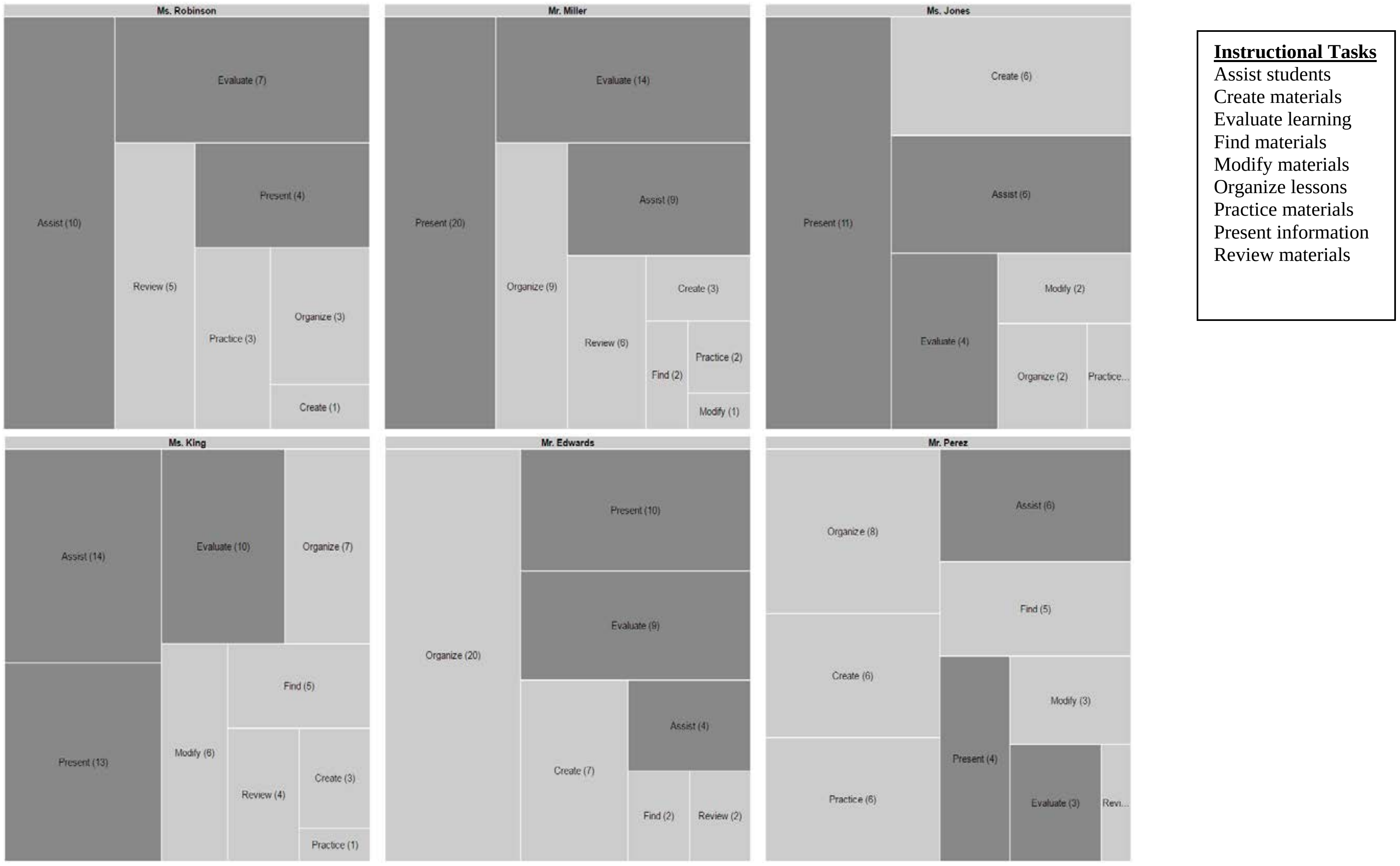


Figure 5.2. Number of interview and questionnaire units where teachers discussed their instructional responsibilities. Tasks related to planning and organizing lessons are highlighted in light gray; tasks related to implementing and monitoring instruction are highlighted in dark gray. Tasks with higher frequency lie towards the top left corner of each treemap.

Ms. Jones did not discuss finding materials. During the study, she was in her third year of using materials created by her volunteers in the 2013-2014 school year. A remark offered in the fall semester suggests she may have been satisfied with the materials and did not need to find other materials for her course:

> This was prepared about three years ago, so the volunteers came up with the projects and the rubrics, and it was also following the structure of CS 10 at Berkeley. And then they have prepared most of the materials - like the paper notes, and then I would tweak it…It has reached a point now where it is the fifth time we are doing it, so we have all the materials set and we have tweaked it enough where I just do everything, and the volunteers just show up. We have one common agenda where I will write what we are doing that day, and if they want to see, they can look at it, otherwise they just show up and then they help. (Ms. Jones, October 26, 2015)

Ms. Jones did not discuss reviewing materials, but she did discuss the related task of practicing materials. Similarly, Mr. Edwards did not explicitly discuss modifying materials, although he did discuss the related task of finding materials. He also did not discuss practicing materials, but he did discuss the similar task of reviewing materials. So, while Mr. Edwards and Ms. Jones did not discuss all the teaching tasks covered by the coding scheme, they did discuss other tasks that involved interacting with pedagogical materials in similar ways.

Second, when looking at the top two most frequently discussed tasks for each teacher, which accounted for between 33% to 56% of units, three patterns emerged. These patterns, a focus on (a) lesson planning, (b) lesson delivery, or (c) both types of responsibilities, seem to relate to the number of years teachers have used the TEALS curriculum. Mr. Perez' most frequently discussed tasks related to lesson planning and he was in his first year teaching the Intro curriculum. Ms. Robinson, Mr. Miller, and Ms. King's most frequently discussed tasks related to lesson delivery, and they were using the same TEALS curriculum they used the previous school year. Lastly, Mr. Edwards and Ms. Jones' most frequently discussed tasks included one lesson delivery task and one lesson planning task. Mr. Edwards and Ms. Jones were the furthest along in the TEALS program, at years 4 and 3 respectively. This suggests there may be a learning progression related to curricular knowledge where teachers first focus on lesson planning when they use a curriculum for the first time, then shift to a focus on lesson delivery as they gain some experience with the curriculum, and eventually focus on both tasks equally when they have used the curriculum multiple times.

Summary. The TEALS program provided participants with broad categories of instructional responsibilities to distribute across their teaching team. In this part of the analysis, I wanted to understand the nuances of instructional responsibilities teachers undertook. According to participants' self-reported data, teachers performed a variety of tasks related to lesson preparation, lesson delivery, and evaluation of learning. In accordance with the suggested TEALS co-teaching model, all teachers were heavily involved in managing their classrooms and partook in lesson delivery. The distribution of other responsibilities differed across teams and this variation was not simply a factor of each team's stage in the TEALS program. Participants in teacher-led teams were more involved in lesson preparation tasks, but responsibility for evaluation tasks did not correlate with PD stage.

The results suggest that, within a CS context, directing a computer lab needs to be considered as an instructional responsibility distinct from lesson delivery because it involves different instructional methods. Also, the task of assisting students might be refined to highlight different ways of supporting students around CS content. For example, most teachers assisted students by visiting them at their computers and addressing issues they had with an assignment. But, in Mr. Miller's course I observed one instance of volunteers using an authentic disciplinary practice (i.e., code reviews) to assist students in revising their assignments.

5.4.2. Relationship Between Responsibilities and Teaching Knowledge

The CS PCK development framework I presented in the literature review (see Figure 2.8) suggests that a reciprocal relationship exists between instructional responsibilities and PCK. That is, performing instructional responsibilities can support PCK development, and PCK can influence how instructional responsibilities are implemented. In this section, I explore this relationship by examining comments participants offered during their interviews.

Creating and modifying instructional materials. Creating and modifying instructional materials allows participants to work with authentic artifacts of practice, evaluate their utility to support student understanding, and reflect on how best to present content to students. However, some teachers found these responsibilities challenging when they had not mastered the content themselves. For example, Mr. Miller once described a syntax error he made when modifying a quiz:

> I took last year's quiz, and it was half on HTML and half on JavaScript. So I just took away all the JavaScript questions and added my own CSS questions…Thursday night as I was correcting it, I realized I made a mistake in writing the code. I forgot the semi-colon in one place… I kind of wish I had a little more help on that from the volunteers, that they had taken a look and told me 'you missed something'. (Mr. Miller, March 28, 2016)

While Mr. Miller caught his mistake, he did so only after presenting his modified quiz to students. Teachers with weaker content knowledge might need additional support around evaluating the accuracy of their instructional materials. The working style of Ms. Jones and her volunteers provides an example of how teachers can receive such support in a fashion more aligned with Ball and Cohen's model that includes professional discussion centered around artifacts of practice. This team often reviewed the instructional materials they created or modified in their weekly Tuesday evening meetings and discussed how to introduce them to students. Regular conversations around instructional materials could provide a venue for teachers to critique the adequacy of their resources for supporting student learning and to strengthen their subject matter knowledge with the support of content experts.

Another factor related to these two responsibilities is time. As Mr. Edwards once said, "I have been making videos for the kids to watch at home so I can get some of the lecture done that way…Of course that takes me more time, because it takes me an hour to make a twenty-minute video. But at least it gives the kids time in class to work together". Time was a rare commodity for case study participants who in addition to learning to teach a new discipline were still involved in teaching other courses in their main disciplines. Mr. Edwards was the furthest along in the TEALS program and he taught an AP CS course about 20 years ago. Teachers with less experience might require more time and effort than Mr. Edwards to create instructional materials from scratch. So, independently creating materials that require intense amounts of preparation may be less advantageous for transitioning CS teachers, who instead might benefit more from working collaboratively with volunteers to create or modify resources. While creating and modifying instructional materials might be difficult for any teacher with limited content knowledge, regardless of their discipline, it may be of greater concern amongst transitioning CS teachers who have had few prior opportunities to study CS. For these teachers, it may be more efficacious to rely on existing curricular materials until they become more comfortable with CS content.

Reviewing instructional materials. Prior to their lessons, participants often looked over or completed materials they later assigned to students. During a March 2016 interview, Ms. Robinson discussed how reviewing completed solutions helped her identify salient aspects of the lesson content that guided her formative assessment of students during lab time:

> I had the solution, so I can compare, like ‘OK, well this is what you are supposed to do’. I mean, I used it as my guide. So I kind of leaned on it, which I really appreciated that TEALS had the solution. Because sometimes we don't always have solutions to like their tests, so I have to do the solutions, but this was nice…And I am glad I did look at the code beforehand because I did notice that they had to use math.random and that gives it more of a random mix of shuffle. So if they got at least half of activity three which included the random, I would just move on. (Ms. Robinson, March 28, 2016)

As another example, Mr. Perez discussed how completing solutions himself helped him think through how to organize and select tasks for his students:

> So there's a whole bunch of projects, and part of the reason I didn't introduce them is I want to go through and do the ones I haven't done yet myself before I give them to students…So I want to spend some time this weekend figuring out which ones would be valuable to the students, which ones would be accessible, and then what kind of sequencing I would want to put on them. Because I know the script variables - they are

> focused on script variables. The drawing a tic-tac-toe board and the brick wall one are very focused on abstraction. And then the number guessing game is kind of both. Less abstraction, but it would be good if they used that. But it is also kind of an encompassing project. So I would want to think about what order I want these things to go in, and where I can give students some choice as to what they are doing and where to slow them down. (Mr. Perez, September 25, 2015)

While reviewing instructional materials seemed to support both Ms. Robinson and Mr. Perez in critically analyzing project assignments and student work, their level of content knowledge seemed influential in how they approached this responsibility. Ms. Robinson, who had less content knowledge, relied on completed solutions. In an interview conducted in December, she described another instance where she copied code written by a volunteer and used that as a guide when assisting students during lab time. In contrast, Mr. Perez, who had greater content knowledge, wanted to complete solutions himself to inform his planning, which he also mentioned on questionnaires completed in October, February, and March.

A transitioning teacher with a low level of content knowledge might increase their own content knowledge by completing solutions. However, reviewing solutions completed by more knowledgeable others might support PCK development by helping teachers identify important features of problem solutions, which they themselves may not have the content knowledge to identify. In a co-teaching context like TEALS, volunteers could be especially beneficial in helping transitioning teachers to review materials and understand the salient aspects of those materials. Reviewing completed solutions also provides transitioning teachers with the opportunity to engage in an activity similar to the disciplinary practice of peer review (Hazzan et al., 2015) that they can later use with their own students. Transitioning teachers with a higher level of content knowledge might benefit in other ways from reviewing instructional materials. For these teachers, reviewing materials might support PCK development by sparking reflection around the most appropriate pacing, scaffolding, and organization of their lessons.

Finding materials. TEALS provided complete curricula for the AP and Intro courses, however four participants, Ms. King, Mr. Edwards, Mr. Miller, and Mr. Perez, supplemented their courses by finding additional problems and scaffolds online. Sometimes participants found materials in resources recommended by TEALS, the webpages of other CS teachers, and online teaching forums. The value of the finding instructional materials for PCK development seemed variable. For example, Ms. King once discussed how she used a web application designed to support AP CS A courses to easily identify extra credit assignments for students who were ahead in her class: “PracticeIt is nice because I don't have to prepare it, I just have to pick the problems out”. In contrast, Mr. Edwards spent more effort searching for examples of abstract classes: “And so I've actually been wracking my brain for the last few weeks looking at lots of examples on the Internet, and I didn't find any that really went beyond just the very, very basics”. Drawing from materials designated for the course might save teachers time in selecting appropriate assignments, but it might also require less active noticing and evaluation of those resources. Mr. Perez and Mr. Miller each discussed materials they found on teacher created sources. For example, on a questionnaire completed in September 2015 Mr. Perez wrote, “I found sample scripts with bugs through BJC and other CS teachers, and solved each problem on my own. I also organized these scripts into a presentation that had them in manageable chunks…These resources provided the scripts that guide students' thinking to most effectively practice [debugging] Boolean operators, script variables, and for loops." Their comments suggest that finding materials through teacher created sources might support PCK development when these sources also discuss how the materials support student learning and challenges.

Instructional delivery. Presenting ideas in front of the entire class offered multiple opportunities for all participants to reflect on their content knowledge, pacing, and student engagement. It seems the immediate feedback teachers received during lesson delivery encouraged reflection on how to improve their instruction. While delivering lessons, teachers identified their own errors or received responses from students indicating the need to adjust instruction. Consider the following excerpts highlighting different ways in which teachers became aware of errors in their content knowledge while presenting a lesson:

> I gave an example [about static binding and dynamic binding] that seemed to contradict something that I had told the kids the week before, and I didn't even spot that. But one of my super sharp kids in second period said 'hey, this seems to contradict'. And I said I would have to get back to him. Which I am okay with, as long as I have the resources to find that out. So I am still learning. (Mr. Edwards, February 4, 2016)

> I saw as I was working through [the String problems] that I could have made the Rumpelstiltskin example a little bit better. You saw me change his name to R to illustrate that, so I should've had a word that had whatever I was searching for. Just the first letter, so they could've seen that. (Ms. King, September 29, 2016)

> But what I actually presented, I think I would do it the same kind of way again. There was that one thing I didn't get to in first period, and I forgot to do it because it just threw me… OK, so in that very top block there I changed from item 1 equal to 1. I had somebody tell me how could I make it so that it goes to the else, and he said, oh you change it to equals 2 or something, but I didn't show [the item-any-of-list block] because probably that it is just if any of those items is equal to 2 it is going to be true and say 'eureka'. But it is not that. It is a random choice. The any is not 'if any of them', it is a randomly picked. That word any, I mean, I got thrown by that. I thought it would be always true. But, oh no it isn't. Oh I see, it is going to randomly pick one of them (Mr. Miller, November 11, 2015)

Also, delivering lessons in real time gave teachers opportunities to think about pacing. The actual time spent on different lesson sections served as a comparison against teacher expectations, providing immediate feedback on how to revise pacing for future lessons and what additional supports might need to be included for students. Ms. Robinson and Mr. Perez discussed these topics during second semester interviews:

> Well because last week I did cover a topic, but I took the whole period. I didn't want to take up the whole period. I want to spend 15 minutes and then have them practice. Maybe I didn't give them enough examples because I wanted to make sure that they had enough time to practice. So it is like trying to find a medium. And you have to be prepared, and so I didn't have enough time to quite come up with more examples to show them so they could understand the concept. (Ms. Robinson, February 24, 2016)

> The debrief at the end was better than no debrief, and I think it will kind of help students go back and hear again how do we look at things [at a] table and make sense of the recursive relationship there, or look at it as a recursive relationship. Like I said, I think it would have been more effective if I had been able to put up student work rather than just kind of filling time on my own. (Mr. Perez, March 14, 2016)

Lastly, lesson delivery also provided teachers with immediate feedback from students. Student feedback helped teachers think of how to present content differently to either keep students engaged or to think of what supports students needed. Consider the following quotes from Ms. Jones and Mr. Miller during first semester interviews:

> And last year - the first year we taught it, we just took notes straight, and they just fell asleep. So we had them guessing last year first what shapes the blocks were, help them stay awake a little bit. And we had it on white boards. People [were using] white boards in three locations in the room. But I forgot to bring the white boards this year, so I wrote it on the board, which seemed just as good, so that was fine. (Ms. Jones, September 16, 2015)

> I did it a little differently in second period, I made it a working game. So I just kept clicking on it and showed that it worked. And it was just three lines - three blocks or something. And then I said, 'start with this basic framework and you can create your guessing game'. Because in second period I saw there were quite a few that were a little lost as to how to begin. (Mr. Miller, October 27, 2015)

Evaluation of learning. All teachers were involved with assisting students during class time and evaluating student learning. These responsibilities offered teachers the opportunity to see multiple student solutions which helped in identifying misconceptions and more elegant solutions. For example, Ms. King gave the following example during a December interview, “And we sort of all as a class realized that we had to find the index, and looking for the index for the operator should work, and then later on I realized by looking at some student's code that you could just find the index of the space, and that would work better to break it up.” In CS, problem solving tasks usually allow for multiple problem-solving approaches. As Ms. King’s quote demonstrates, the task of evaluating student learning can support teachers in recognizing those different approaches (not just misconceptions) and developing ways to judge the quality of the solutions.

Another factor related to this responsibility was the amount of time teachers spent with individual students during class time. Consider the following comments from Mr. Miller and Mr. Perez:

> And there are times like that where I am walking through the class, and they will say I don't get it, this isn't working. And I don't always catch it right away. I feel great when I do. ‘Oh, well, all you have got to do is this and this’, but sometimes it is more complex. I don't have five minutes to stand there and go, ‘wait, let's see, let me think out’ - so I have to say, ‘you should go block by block through it and do the same sort of debugging techniques’. (Mr. Miller, October 2, 2015)

> It was nice to be able to go and work with the struggling groups knowing that he could go around and answer and support the other groups in doing things. And that is kind of, ideally, that is the most important thing for me, in terms of the volunteers. Is that when I need to focus in and spend a little more time with a group that they can still be there for the rest. (Mr. Perez, October 27, 2015)

Like most other participants, Mr. Miller spent less time with individual students in class so that he could circulate to more students and attend to other tasks. Mr. Perez, on the other hand, preferred to spend more time with individual students, especially those struggling with content, and let his volunteers circulate around to the rest of the students. The comments from Mr. Miller and Mr. Perez suggest that extended interaction could allow teachers to explore student work more deeply and find solutions to support student difficulties. Both styles of interaction can be useful in CS classrooms. The comment from Mr. Perez showed how volunteers in a co-teaching model can provide teachers with the flexibility to try the longer interaction style without worrying about the rest of their class. This may be important for helping transitioning CS teachers explore and develop mastery using different styles of evaluating student learning in class without negative side effects to other students.

Other responsibilities. Lastly, while I focused on instructional responsibilities related to preparing and presenting content to students, teachers also discussed other responsibilities that seemed to reduce opportunities for engaging in tasks that supported PCK development. As shown in Figure 5.1 above, teachers were mostly responsible for managing their classrooms.

Sometimes while teachers took attendance or passed out TEALS swag, they relegated more content-focused responsibilities to their volunteers. As Mr. Miller said,

> I had a little trouble with timing. I was very happy to have the volunteers there because I tried to walk around and let them help students, and then I wanted to deal with the raffle, deal with the warmups, and the one girl who lost her password on the survey. There was another student who asked to use the printer. So all these little interruptions that I try and comply and answer to, it takes away from class time. (Mr. Miller, May 13, 2016)

In addition to classroom responsibilities, Ms. Robinson was heavily engaged in the expansion of CS education in her district. She acknowledged that attending to these advocacy responsibilities reduced the amount of time she had to focus on learning course content:

> So I am like marketing, advertising, promoting, teaching, and getting people to help because I don't know all of this computer science. I mean I am so fortunate to have my four engineers. And I am very fortunate to have some kids who are very good at programming who can help. And so I am pretty spread thin…I am just trying to get the word out. That is why I don't have that much time to you know even sometimes do the curriculum. But I am taking advantage of the great volunteers that I have, because they can deliver. (Ms. Robinson, December 9, 2015)

While she had little time to learn content, advocacy work seemed to keep Ms. Robinson motivated in her teaching assignment. Other factors, such as motivation, may be as important to developing and sustaining effective teachers as PCK. Another way in which advocacy work might support Ms. Robinson is by helping her develop a community of peers also vested in CS education. Many CS teachers are isolated and belonging to a professional community can help them overcome that isolation (Ni & Guzdial, 2012). Through her advocacy work, Ms. Robinson connected with other CS teachers in her district which may offer the connection she needs as a teacher new to CS.

Summary. The exploratory analysis of the relationship between responsibilities and teaching knowledge suggests that instructional responsibilities assumed while teaching can support teacher knowledge development in different ways (see Table 5.5). By working with instructional materials and supporting students, teachers have multiple opportunities to work with content, confront student understandings, and consider their instructional strategies. Teacher reflections also suggest that instructional responsibilities support PCK development differentially depending on content knowledge expertise. Teachers with lower content knowledge can make errors in their work, may be less able to discern what is important to focus on, and may require additional time to complete their duties. These teachers may need more guidance in the responsibilities that relate to choices around content knowledge (e.g., in creating or finding materials), which can easily be accomplished through a co-teaching relationship in a program like TEALS. Teachers with greater content knowledge may not need to focus on the accuracy of content as much and can shift their attention to other areas. They may benefit from working more closely with content to think about pacing and instructional strategies. Also, they may be able to engage in responsibilities that are time intensive. These instructional tasks, depending on how they are implemented, can offer teachers experience with some disciplinary practices important in CS. While still exploratory, these results may have heuristic value for thinking more about the relationship between instructional responsibilities and PCK development.

Table 5.5

Relationship Between Instructional Responsibilities and Teaching Knowledge

Responsibility	Relationship to Teaching Knowledge
Create/modify materials	• Weaker content knowledge can lead to mistakes when creating or modifying materials. Teachers creating or modifying materials might benefit from a content expert review to bolster content knowledge. • Creating materials might require more time and content knowledge than modifying materials.
Review materials	• Reviewing completed solution sets can help teachers to identify salient aspects of lesson content. • Completing materials can give teachers ideas about organizing tasks but may require strong content knowledge.
Find materials	• Selecting from materials designed for a course can save teachers time, but might require less evaluation of how those materials support student learning. • Teacher created sources might support teacher knowledge development because they sometimes include discussions on how materials support student learning and on common student challenges.
Deliver lessons	• Delivering lessons helps teachers identify errors in their content knowledge and to evaluate their expectations for pacing. • Immediate student feedback helps teachers understand what strategies are engaging for students.
Evaluate learning	• Evaluating learning exposes teachers to many student solutions and helps them to identify student misconceptions. • Spending more time with fewer individual students might allow teachers to probe student understanding more deeply; spending less time with more individual students might allow teachers to identify the frequency of student misconceptions.
Non-content duties	• Other duties such as classroom management and advocacy take time away from responsibilities to that develop PCK, but may motivate teachers to persist in their assignments.

5.5 Discussion

Case study participants were involved in a professional development program where much of their learning happened during real time classroom teaching as they enacted their instructional responsibilities. These responsibilities did not simply increase as teachers progressed through the PD stages of the TEALS program. Instead, the ways in which teachers shared responsibilities with their volunteers varied across stages and across responsibility type. The data also suggest that (a) instructional responsibilities might vary in the opportunities they provide for developing PCK expertise and (b) the usefulness of responsibilities might vary based on a teacher's content knowledge. The data also highlighted some instructional responsibilities important to CS teaching (i.e., managing a computer lab, code reviews) that may be new to transitioning teachers. Delegating these tasks to teachers, drawing on volunteer support where necessary, may not only help teachers improve their PCK but may also expose teachers to authentic CS practices that will increase their understanding of the discipline.

Limitations. Given the semi-structured format of the interviews and the focus on individual lessons, caution should be exercised in interpreting the coded data. First, while some interview items and open-ended questionnaire items asked teachers explicitly about their lesson preparation and their role on their TEALS team, teachers also shared examples of their responsibilities while responding to various other questions making it difficult to compare responses consistently across teachers. Second, the total number of units where teachers discussed instructional responsibilities were few, accounting for less than 25% of each teacher's total units. Lastly, case study visits focused on individual lessons and not entire units; it is possible that teachers performed other responsibilities outside of the study visits that were not discussed during interviews. Designing more systematic ways of eliciting the ways instructional responsibilities supported teaching knowledge and the ways teacher knowledge constrained instructional responsibilities is an obvious next step for this work.

CHAPTER 6. CONFIDENCE, EPISTEMOLOGIES, AND TEACHING

6.1 Introduction

An understanding of teacher knowledge development must consider factors beyond PCK and instructional responsibilities that also influence what teachers can or will do to improve their craft. Individual traits such as orientations to teaching, prior experiences, and personal characteristics influence the instructional strategies teachers use (Magnusson et al., 1999; S. Park & Oliver, 2008). Confidence determines educators' willingness to try new teaching approaches and allow students more authority in their classrooms (Goldsmith et al., 2014). Teaching also differs by discipline and the teaching subculture of each discipline relates to educators' beliefs, norms, approaches to teaching, collaboration, and leadership within schools (Grossman & Stodolsky, 1995; Spillane, 2005; van Veen, Sleegers, Bergen, & Klaassen, 2001). These factors raise questions for teacher knowledge development of experienced educators who are between disciplines. Do their existing beliefs about teaching and learning carry into their new courses or do they assume new epistemologies? How are instructional decisions made when educators are confident about certain aspects of their teaching but hesitant about others?

Although epistemological beliefs and confidence were not an initial focus of this study, they appeared prominently in comments teachers made during the study when they explained their selection of instructional tasks and the focus of their professional activities. To investigate epistemological beliefs and confidence more systematically, I asked participants to rate their confidence at the beginning of each case study visit and to complete four, open-ended items on a teaching beliefs questionnaire at the end of the school year. The specific question I explored with these data was: *how do confidence and epistemological beliefs influence instructional responsibilities?*

6.2 Research on Confidence and Epistemologies

6.2.1. Confidence

Teaching confidence describes a "teacher's belief in his or her capability to organize and execute courses of action required to successfully accomplish a specific teaching task in a particular context" (Tschannen-Moran, Hoy, & Hoy, 1998, p. 233). Educators teaching courses outside their area of expertise, like the participants in this study, may have lower confidence in their ability to perform instructional responsibilities than they do in their regular courses (Ross et al., 1999). Lack of confidence can lead teachers to avoid teaching, rely on prepared instructional materials, and minimize teacher-student discourse (Harlen & Holroyd, 1997; Schneider & Plasman, 2011). Some ways that teaching confidence can improve are through strengthening content knowledge and having successful teaching experiences (as evidenced in Swackhamer et al., 2009; Tschannen-Moran et al., 1998). Ni and colleagues (Morrison et al., 2012; Ni, 2009; Ni & Guzdial, 2012) have explored the role of confidence in CS teacher development. Similar to findings from other disciplines, they found that lack of confidence prevented some teachers from implementing contextualized computing curricula, confidence was a factor in developing an identity as a CS teacher, and confidence can be improved through participation in a professional learning community. The co-teaching model used in the TEALS program provides opportunities for teachers to strengthen their confidence in teaching CS. Through collaborations with one or more volunteers, transitioning CS teachers in the TEALS program have access to peers with whom they can discuss issues of practice on a regular basis and build relationships around their CS teaching.

6.2.2. Epistemological Beliefs

The beliefs teachers bring into the classroom about the nature and acquisition of knowledge, or their epistemic beliefs (Chinn et al., 2011), will influence the decisions and actions they make. Scholars studying epistemological beliefs have categorized teaching viewpoints on a spectrum ranging from more teacher-focused, didactic beliefs to more student-focused, constructivist beliefs (e.g., Brockmeyer, 1998; Hashweh, 1996; Luft & Roehrig, 2007; Peterson, Fennema, Carpenter, & Loef, 1989; Simmons et al., 1999). Teacher-focused, didactic beliefs tend to view teachers as conveyors of knowledge and students as receptacles of that knowledge. Student-focused, constructivist beliefs view students as constructors of their knowledge and teachers as facilitators of this process. Some researchers have found that teachers espousing more constructivist views of teaching and learning notice students' conceptions more and use more varied instructional strategies (Hashweh, 1996; Peterson et al., 1989).

Epistemological beliefs of computing educators is an understudied topic. Some prior work has identified the ideas teachers possess about their discipline (e.g., Carbone, Mannila, & Fitzgerald, 2007; Lewis, Jackson, & Waite, 2010). One study conducted by Kordaki (2013) explored the relationship between beliefs and teaching practices of CS educators. Using results from a multiple case study of twenty-five high school computing teachers in Greece, she identified two prominent belief types: empowering beliefs more aligned with constructivist views and constraining beliefs more aligned with behaviorist views. An example of an empowering belief Kordaki observed was "development of algorithmic thinking and high cognitive skills are appropriate competences for students as learners in computing" (2013, p. 150). An example of a constraining belief she observed was "direct teaching is appropriate when one has to teach a specific curriculum, with specific learning aims and within specific time limits" (2013, p. 151). She also identified five instructional approaches computing teachers used that varied in their amounts of direct teaching, student participation, and project work. Her analysis showed that teachers often held multiple yet conflicting beliefs, teachers' beliefs aligned with their practices, and teachers' self-described practices misaligned with observation data. Kordaki posited these conflicting patterns resulted from contextual factors (e.g., institutional barriers such as the elective status of courses, the rapidly evolving nature of computing) that also influenced their instructional actions.

6.3 Methods

6.3.1. Participants

Six teachers participated in this component of the case study. As described in earlier chapters, these teachers varied in their PD stages, CS PCK, CS content knowledge, and non-CS teaching assignments. Ms. Robinson was in the volunteer-led PD stage. She seemed to have lower levels of PCK and content knowledge compared to other teachers, and she taught geometry. Mr. Miller was in the collaborative PD stage. He demonstrated some PCK and appeared to have a partial understanding of CS content, and he taught algebra. Ms. Jones was also in the collaborative PD stage. She demonstrated some PCK and a strong understanding of CS content, and she taught trigonometry. Ms. King was in the teacher-led PD stage. She demonstrated strong PCK and a partial understanding of CS content, and she taught support mathematics. Mr. Perez and Mr. Edwards were both in the teacher-led PD stage. Neither participant completed the study tasks related to teacher knowledge, but both expressed confidence in their ability to independently lead their CS courses. Mr. Perez previously tutored college students in CS, was involved in the BJC community, and taught algebra. Mr. Edwards taught the AP course about 25 years ago, was in his fourth year with TEALS, and taught digital arts and animation.

6.3.2. Data Collection and Analysis

Two types of data were collected to explore teachers' epistemological beliefs and their teaching confidence. The data sources included close-ended questionnaire items collected at the start of each case study visit and an epistemological beliefs questionnaire administered at the end of the school year.

Confidence ratings. As a proxy for measuring teaching confidence, participants were asked to rate their comfort with course topics and their preparedness to guide student learning of those topics on a questionnaire administered before each visit. First, teachers identified the units their class worked on during the visits and then answered: (1) how comfortable are you with the unit? and (2) how well prepared do you feel to guide student learning of this content? Rating options were not at all, somewhat, and completely. I converted these levels into numeric values (i.e., -1, 0, and 1) in order to average feelings of comfort and preparedness over the school year. Teachers were also asked to explain their interpretation of the rating levels during interviews.

Epistemological beliefs questionnaire. I asked teachers to answer four open-ended items about their epistemological beliefs drawn from the Teacher Beliefs Interview (Luft & Roehrig, 2007): how do you maximize student learning in your classroom?; how do your students learn computer science best?; how do you describe your role as a teacher?; and in the school setting, how do you decide what to teach and what not to teach? Luft and Roehrig identified five categories of beliefs (see Table 6.1). The definitions of two terms in their categorization scheme, *responsive* and *reform-based*, differ somewhat from the usage of these

Table 6.1

Teaching Belief Categories (based on Wong & Luft, 2015, p. 627)

Category	Orientation	Description
Traditional	Teacher-centered	Focus is on teacher providing information and resources in a structured manner and environment
Instructive	Teacher-centered	Teacher decides experiences and reacts based on subjective evaluation of student actions and performance
Transitional	Both teacher-centered and student-centered	Emphasis on teacher–student relationship that includes subjective and affective components that does not necessarily focus on teaching or learning of [computer] science
Interactive[1]	Student-centered	Centers on opportunities and value of collaboration between students and teacher, as well as between students as peers. Focus is on development of [computer] science learning and content knowledge
Responsive[2]	Student-centered	Focus on individualized and student-centered methods of learning that considers student responses, interests, and abilities. Promotes a collaborative environment in which students apply skills and knowledge to novel situations

Wong and Luft originally used the terms *responsive*[1] (instead of interactive) and *reform-based*[2] (instead of responsive) in their categorization scheme.

terms in other teacher learning literature. In the broader literature, responsive teaching "foregrounds the substance of students' disciplinary ideas, recognize[s] the disciplinary connections within students' ideas, and take[s] up and pursue the substance of students' ideas" (Robertson, Atkins, & Levin, 2015, p. 27). This definition aligns more with Luft and Roehrig's description of reform-based beliefs that focus on student responses, interests, and abilities and less with their description of responsive beliefs that center on interactions amongst students and between teachers and students. The key distinction is that one category (i.e., responsive, or what Luft and Roehrig term reform-based) goes beyond a simple focus on ensuring students share their ideas to a focus on pursuing those ideas. To distinguish the nuances of these terms, I offer the categories of *interactive* to describe beliefs centered on providing opportunities of students to exchange ideas and *responsive* to describe beliefs centered on taking up students' ideas in instruction. This modified coding scheme was used to code participants' questionnaire responses. Based on Kordaki's (2013) finding that CS teachers carried mixed beliefs, I decided that participant responses could receive multiple codes if teachers expressed more than one idea in their comments. Interview data were used to supplement their responses and to glean the epistemological beliefs of Mr. Edwards and Mr. Perez who did not complete the questionnaire. Before presenting the results, it is worth noting that within the CS education community today, student-centered beliefs are encouraged and reflected in both teacher training materials (e.g., active learning promoted by Hazzan et al., 2015) and course frameworks (e.g., inquiry practices in the ECS curriculum; Margolis et al., 2014).

6.4 Results

6.4.1. Feelings of Confidence

Averages of teachers' self-reported ratings are depicted in Figure 6.1. Below I describe how teachers made sense of comfort and preparedness, trends across professional development stage and across semesters, and factors influencing self-reported ratings of comfort and preparedness.

Meanings of comfort and preparedness. Teacher responses to the confidence items suggest that comfort and preparedness were interpreted as distinct but related constructs. If teachers perceived these constructs as the same, levels of comfort and preparedness would be identical at each visit. While teachers frequently selected the same ratings for comfort and preparedness at each visit, only Ms. King did so for all visits. During interviews at the beginning of the second semester, we asked each teacher to explain how they interpreted comfort, preparedness, and their corresponding response scales. Comfort seemed to capture confidence in one's own content knowledge and experience making use of content knowledge to solve problems, while preparedness seemed to relate to prior experience teaching the content. For some teachers, the distinction between comfort and preparedness was less delineated and confidence seemed to describe their assuredness with applying content knowledge specifically to teaching. Table 6.2 provides examples of ways teachers interpreted comfort and preparedness.

Patterns of comfort and preparedness. When looking at the data in Figure 6.1 with respect to teachers' PD stage, two interesting patterns emerge. First, teachers in the teacher-led phase rated their comfort and preparedness higher than teachers in the volunteer-led and collaborative phases. Second, teachers in the teacher-led phase rated their comfort higher than or equal to their preparedness; teachers in the volunteer-led and collaborative phases rated their preparedness higher than their comfort. So, participants in the teacher-led group seem more confident about teaching the topics covered during their case study visits.

Ratings were also disaggregated by semester to determine the impact of changes in curriculum and teaching roles on participants' confidence levels (see Figure 6.2). Ms. King and Mr. Miller rated their comfort and preparedness consistently across both semesters. The courses they delivered during this study were similar to the courses they delivered the prior school year with minor modifications. Both Ms. Jones and Mr. Edwards rated their confidence in the second semester lower than their confidence in the first semester. The decrease reported by Ms. Jones can be explained by her introduction of a new curriculum during the second semester. In the past, Ms. Jones and her volunteers delivered two semesters of the Intro course each year. During this study, she taught a Java curriculum during the second semester for the first time. In describing one of her Java lessons, she commented “I would say I definitely feel uncomfortable because I am just one or two days ahead of the students…Basically if I haven’t taught it yet, I am uncomfortable”.

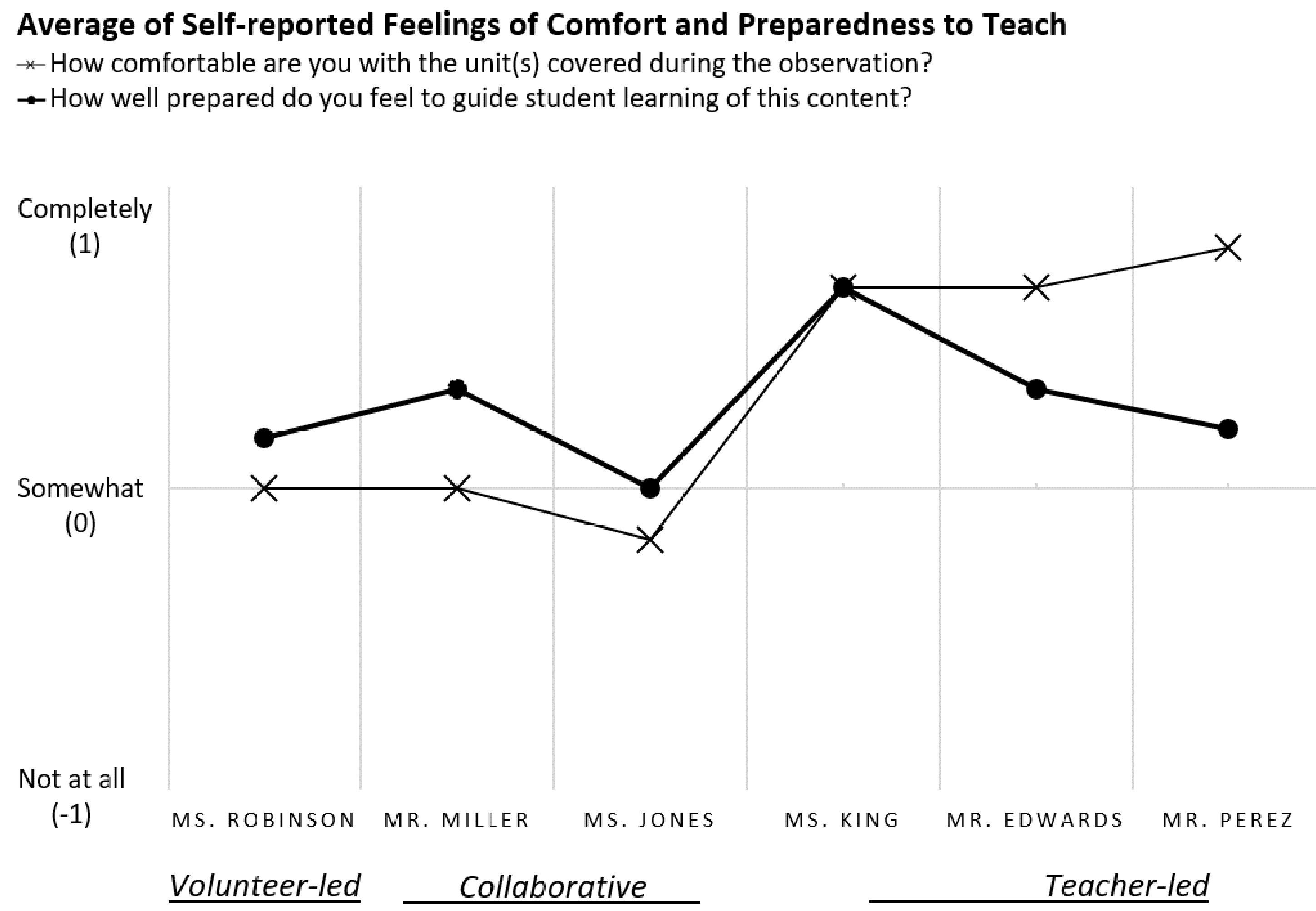


Figure 6.1. Participants' self-reported ratings of comfort and preparedness averaged across the school year. Rating levels are not at all (-1), somewhat (0), and completely (1).

Table 6.2

Teacher Explanations of Comfort and Preparedness

Confidence Rating	Explanation
Somewhat comfortable Completely prepared	Interviewer: What do you mean by somewhat comfortable? Mr. Miller: I am afraid somebody is going to ask me a question and I won't remember. So it happened yesterday [me and the students] found what they did wrong. It was a silly thing of course, a capital letter that I didn't notice. So, if I was really comfortable I would just catch it all the time and remember everything, but since I haven't taught it for a year, I forgot some things. Interviewer: And then for you said completely prepared, so how is that? Mr. Miller: I felt like I knew what I was going to be doing with them…I was thinking that was more to follow the curriculum.
Completely comfortable Somewhat prepared	Interviewer: You said you are completely comfortable but somewhat prepared. Mr. Perez: I've done the Explore task myself, but having not led students through it before, I didn't feel like I had a sense of what they were going to get stuck on and what supports they might need versus where it would be good to let them struggle. So, that's why I put somewhat prepared. First time I have done it.
Somewhat comfortable Somewhat prepared	Interviewer: You said you were somewhat comfortable and prepared for the topic. What else would make very comfortable and prepared for it? Ms. King: There are a couple of those holes like that, where if I try really hard, I can get it right, but I am not sure I totally understand why it is working, and that is where I do go to the volunteers, either in front of the kids or privately…There is nothing that I am not comfortable enough to teach, but there are some things that I feel less on top off, so those I give them middle.

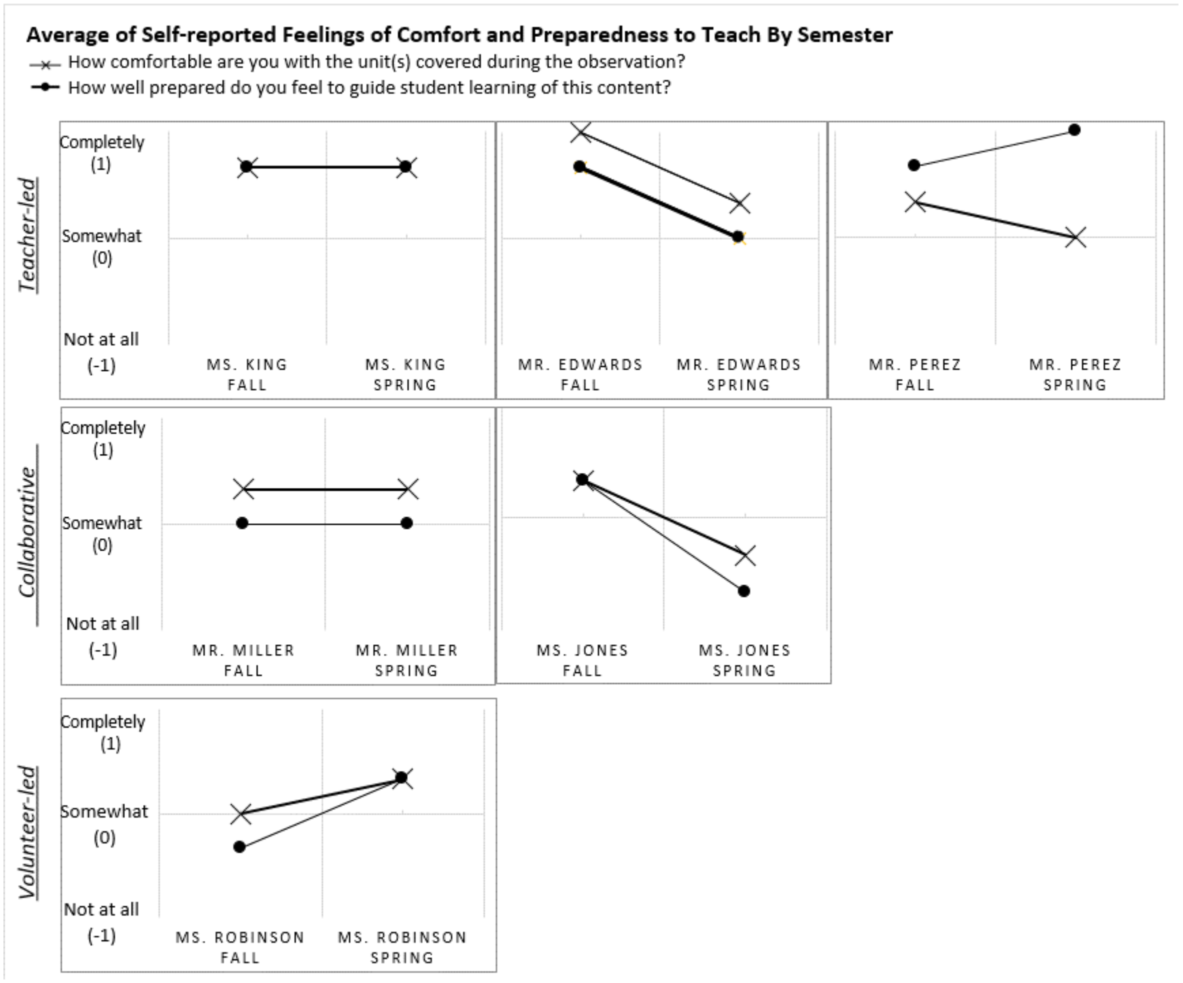


Figure 6.2. Participants' self-reported ratings of comfort and preparedness averaged across fall and spring semesters. Rating levels are not at all (-1), somewhat (0), and completely (1).

Mr. Edwards's lower ratings in the second semester seem to contradict comments he offered during interviews about his preparedness to teach the AP course independently. For example, when asked how he defined comfortable and prepared he said, "this is my fourth year [teaching AP], so I don't think there is any part of the course that I would say not at all. So, I would say for most of the course I feel completely prepared, just based on past experiences". Also, he rated himself somewhat prepared and comfortable with topics he covered in prior visits (i.e., arrays and ArrayLists; inheritance) and with a lesson centered around reviewing for the AP exam, a review he had done the prior three years. It is possible that Mr. Edwards considered other factors besides content knowledge and previous teaching experience when rating his level of confidence.

Lastly, Ms. Robinson's confidence rating increased during the spring semester and Mr. Perez' feelings of comfort increased while his feelings of preparedness decreased. These changes seem to reflect more of a sensitivity to averaging ratings across a small number of visits than to actual changes in the teachers' feelings of comfort and preparedness. For example, Ms. Robinson rated herself as somewhat prepared and as somewhat or not at all comfortable for five of her six lessons. She rated herself completely prepared and completely comfortable during her last visit where students worked on final projects. This class session, however, was qualitatively different than prior sessions. It did not focus on the AP curriculum, students explored projects of their own choosing, and "some advanced students were doing [projects] that are beyond [her] right now, i.e. String Theory or programming in Java to make some really cool designs". Without considering this visit, Ms. Robinson's fall and spring ratings would be similar and hover around the rating level of somewhat. Mr. Perez only completed five of his pre-visit questionnaires and ratings from his sixth visit could shift the average ratings shown on Figure 6.2. For example, if he rated himself as somewhat comfortable and completely prepared for the last visit, ratings across fall and spring semester would look identical.

Factors influencing confidence. Teachers in the volunteer-led and collaborative PD stages said that more experience completing student tasks and more experience presenting ideas to students would increase their feelings of comfort and preparedness. However, division of instructional responsibilities sometimes prevented teachers from having these experiences, especially when teachers offloaded confidence building activities to their volunteers to attend to other responsibilities or due to insufficient preparation time. The following comment made by Mr. Miller illustrates this idea:

> [I would feel more comfortable or prepared] if I actually did the lessons myself. I haven't written the program, or, I should sit down and, maybe this summer, actually go through and…But if I had more time, ideally I would do all this stuff myself. (Mr. Miller, May 13, 2016)

Participants in the teacher-led PD stage generally felt confident in their teaching abilities and did not discuss what would increase their feelings of comfort and preparedness. Instead, they provided examples of how assisting students, presenting ideas, and modifying materials provided opportunities that helped them to reflect on and improve upon their craft. The following quotes from Ms. King highlight this idea:

> Then when I checked their answers, one of the things I was checking was that they had used the limit correctly and in the way that it was most readable. Last year I would have just checked that it was correct, this year I am a little more precise about 'well, it works fine that way, but it would be better if you did it this way' (Ms. King, October 30, 2015)

> I do want to say because you have been listening to me for two years that I am feeling much more confident. I certainly knew the answers to those questions and why. And I know how to do the [free response questions on the AP exam], so I don't feel quite as much like I am winging it anymore, but more definitive…I am sure that when I talk about things I am less hesitant, and [the students] can tell that. So it is going well. (Ms. King, April 15, 2016)

So, while instructional responsibilities like completing student tasks, modifying materials, assisting students, and presenting ideas can provide fruitful opportunities for teachers to gain experience and increase their confidence, teachers may lose out on these learning opportunities if these responsibilities are given to volunteers. TEALS encourages teaching teams to decide for themselves how they will work together in their CS classrooms, and data gathered during this study show teams vary in how they distribute instructional responsibilities between teachers and volunteers. These findings suggest that teachers may benefit more if teaching teams are encouraged to delegate confidence-building tasks to teachers and not to volunteers.

Summary. It appears that self-reported ratings of comfort and preparedness can be used to understand participants' feelings of confidence of their content knowledge and their ability to teach content to students. Participants in the teacher-led PD stage tended to report higher levels of confidence than teachers in the volunteer-led or collaborative PD stages, which might be explained by them taking on more responsibilities that support confidence building. Making use of teacher knowledge supported teachers in increasing their feelings of confidence, however, these opportunities were sometimes missed because they were delegated to volunteers.

6.4.2. Beliefs about Teaching and Learning

Beliefs about teaching. All teachers expressed contrasting ideas when commenting on beliefs about teaching and there were no noticeable patterns based on the type of co-teaching model implemented (see Figure 6.3). When asked how they make decisions about what to teach, all respondents expressed traditional beliefs saying they used their curriculum as a guidepost. As Ms. King wrote, "I stick pretty closely to the AP curriculum, since that's the measuring stick. This is similar to using the California standards for algebra or other math courses". Several teachers had opportunities to teach material beyond their curriculum, and here they expressed different ideas about deciding what to teach. Ms. Jones and her volunteers created a semester-long introduction to Java to follow their first semester Intro course. In describing the development of this course, she remarked that “decisions were mostly based on the volunteer’s wisdom and what students would need to be prepared for taking AP CS in the future”, an instructive belief. Ms. King and Ms. Robinson expressed interactive beliefs when discussing how they structured the weeks after the AP exam; both gave students time to identify and explore computing topics of interest. In contrast, when asked to describe their roles as teachers, participants offered more student-centered aligned beliefs. Ms. Jones saw herself as responsible

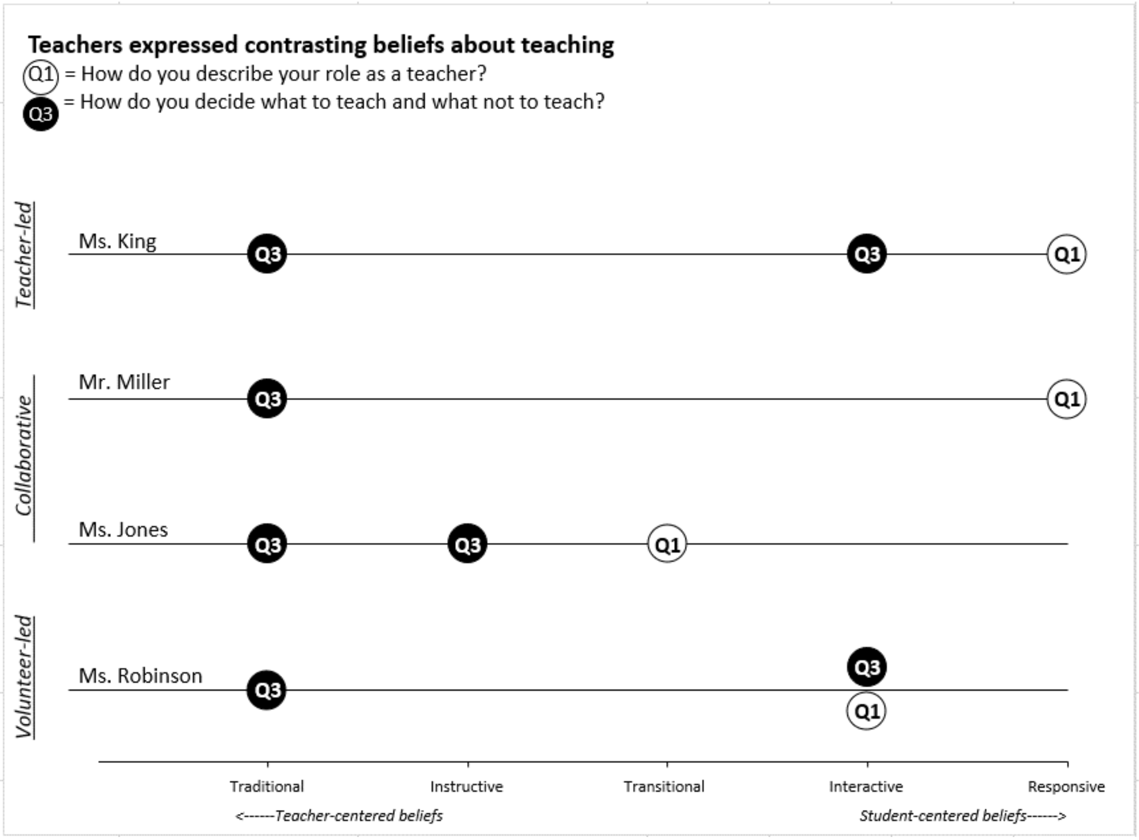


Figure 6.3. Participants’ beliefs about teaching categorized using modified version of Luft and Roehrig’s (2007) five-category belief coding scheme. Each response could receive multiple belief codes.for providing a safe and structured learning environment for her students, a

transitional belief. Ms. Robinson saw herself as a collaborator who helps and is helped by her students, an interactive belief. She said, “students and I worked together to help each other. There are a handful of students who are very proficient in CS, so I tap into their knowledge for assistance to help the rest of the class”. Lastly, both Ms. King and Mr. Miller saw themselves as guides to facilitate students’ own learning, a responsive belief. As Mr. Miller commented, “I describe my role as a resource guide. Students really only learn and retain what they learn by doing, by working to complete a task. I believe my role is to be a facilitator of this productive struggle”.

Curriculum goals appeared to create tension between participants’ beliefs about teaching and their instructional practices. This friction was notable in Ms. King who commented about her struggles to balance her preferred style of teaching with the structure of the AP curriculum:

> As I become more familiar with the AP, I am thinking more about [how the AP asks questions], and less about the way I learned to code, which is, “I have to do something, how do I do it, oh this works”. So that was how I taught the first couple years. I am not a trained engineer, I was self-taught, so I used those skills, but they don’t give you the best written and most efficient code…What I don't like about teaching the AP is that we stop having fun while we review. What I do like about the AP is that it structures the curriculum, emphasizes what I should be emphasizing. So I may not always agree on what it emphasizes, but maybe 8 out of 10 [times] I do…If I knew a little more, I'd be pass the AP, and I'd be wanting to teach it my own way. (Ms. King, February 5, 2016)

Mr. Edwards, who did not complete the TBI questionnaire, made similar comments in his interviews, expressing discontent with the lack of creative expression in the AP curriculum. He responded to this conflict by selecting programming tools he thought would be more engaging for students and interspersing the AP curriculum with projects:

> They can take all the code they have written in class so far, drop it in Processing, and it will run because Processing really is Java. It just makes graphics easier…I think that is an effective way to teach recursion…I think it is [more fun] to sort red squares on the checkerboard than it is to sort just numbers on the console. I am very visual. I find that more interesting, so I think some of the kids will find that more interesting. I think it is a real weakness in the AP course. The entire thing is just done in the console. And so this is just one of the few places that we can really break out. (Mr. Edwards, March 9, 2016)

So, the two TBI items related to beliefs about teaching evoked statements reflecting different belief categories. These differences seem to reflect concomitant ideas stemming from teachers' own beliefs and from external factors to which teachers are accountable that influenced what teachers did in the classroom. Teachers sometimes bent their practices to accommodate external factors and other times they tried to weave their own beliefs into classroom activities, what Fang (1996) refers to as the inconsistency thesis. This need to accommodate external factors might also extend into the decisions teachers made regarding their co-teaching responsibilities and interactions. For example, Mr. Miller reflected on an instance where he followed his volunteer's decision despite wanting to start a lesson with a warm-up activity:

> I like to start the class with my warm-up, my do-now. And my volunteer said to me last night when we were planning, he said we are really going to need a lot of time to explain and to demonstrate and we won't have time for a warm-up. And I kind of agree, although I feel like those words up on the board, I would have rather said take 3 minutes and

> define them on your - or, yeah I had it - that is exactly what I was planning to do was use those words and say which blocks involve user input, instead of just call on somebody and say it - ask it. So that everybody would be doing it…But that way everybody is on task doing things. I like the whole idea of starting with something that everyone has to do. So, that is the change I would have made…we could have had a warm up. (Mr. Miller, October 2, 2015)

This is all to say that in interpreting the data gathered from teachers participating in this study, one must consider that teachers did not always take actions aligned with their beliefs, they were also responding to the multiple demands of their students, volunteers, and curriculum.

Beliefs about learning. Teachers expressed diverse beliefs about student learning that were mostly transitional and interactive (see Figure 6.4). Again, no noticeable patterns were observed based on PD stage. When asked how students learn computer science best, all teachers said “by doing” or “by practice”, a transitional belief. However, for most teachers, this was not sufficient. For example, Mr. Miller distinguished programming as a part of computer science and offered a comment reflecting more interactive beliefs about learning computer science:

> My students learn "Computer Programming" best by trial and error performance. They must be able to write code, recognize errors and then debug them, making corrections to their code. "Computer Science", however, incorporates much more than programming. The best way to learn CS is to research aspects of interest to students and to recognize how what they learn can apply to their daily life. To help students recognize such useful applications, a class "discussion" is often helpful.

Ms. Robinson also expressed an interactive belief that centered around students making sense of computer science by sharing their ideas with others. She thought students also learned computer science by simulating programs with their peers and giving short presentations about topics they researched. In contrast, Ms. Jones expressed a more traditional belief saying that in addition to practice, students "need some guidance and direct teaching. After that, working on projects in lab time and generating solutions on their own seems to work."

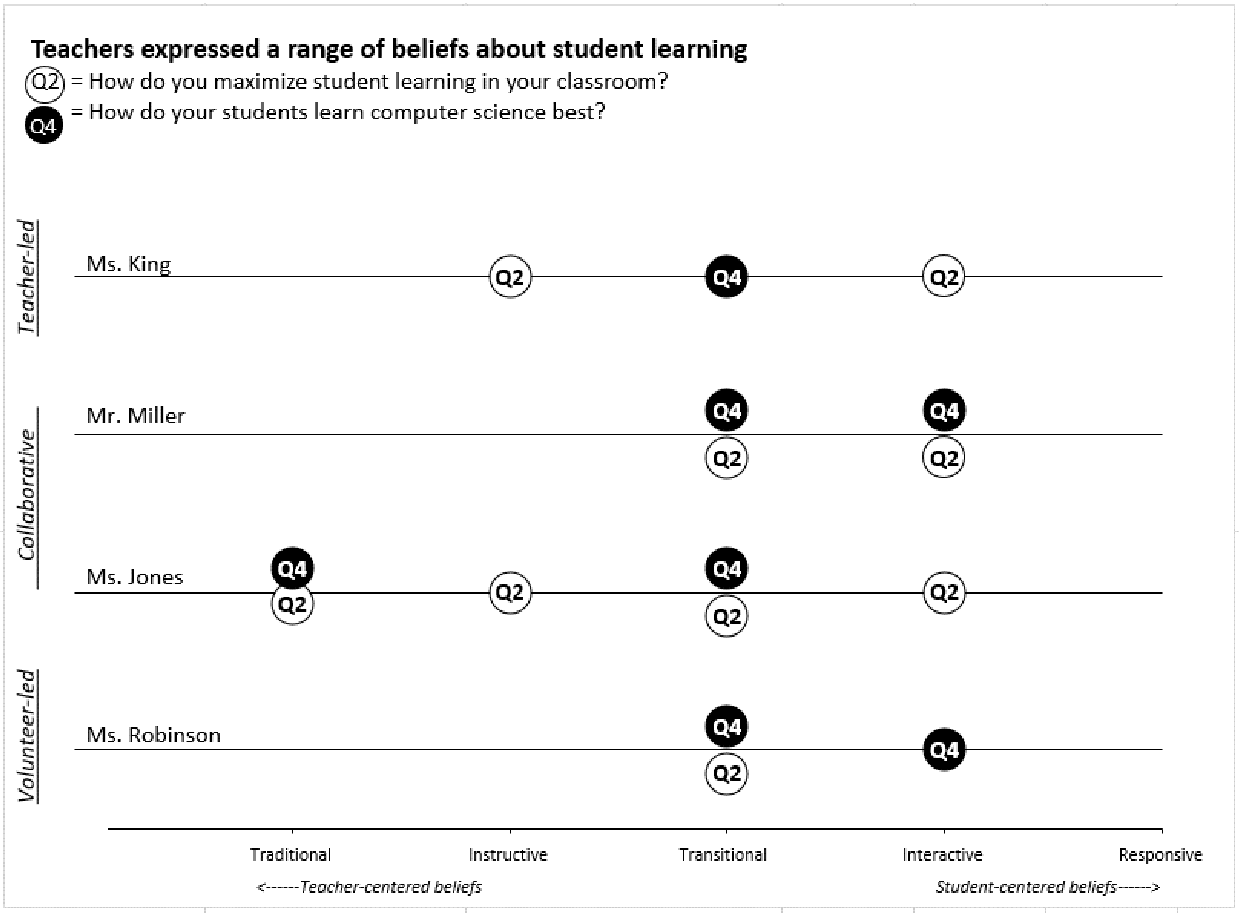


Figure 6.4. Participants' beliefs about student learning categorized using a modified version of Luft and Roehrig's *(2007)* five-category belief coding scheme. Each response could receive multiple belief codes.

When asked how they maximize student learning, teachers expressed a range of beliefs from traditional viewpoints to interactive viewpoints. Ms. King, Mr. Miller, and Ms. Jones maximize learning by providing student opportunities to interact with and learn from each other, an interactive belief. For example, Mr. Miller said, “to maximize student learning, I encourage students to work together with others in completing a task. Unfortunately, this is easier to do in my math classes than in my CS classes. I believe that working with peers allows students to understand concepts well enough to explain them to their peers.” Both Mr. Edwards and Mr. Perez, who did not complete the TBI, expressed similar comments in their interviews:

> I have been using pair programming throughout the year. I use complex instruction group work in my other classes, and I think the ability to have another person to bounce ideas off of…and to be forced to explain things to is really powerful for students working through their ideas and developing a deep understanding. (Mr. Perez, March 14, 2016)

> I like having kids come up and try to put answers on the board. I thought Student A and Student B did a good job on that. However, sometimes that takes more time. It is just quicker for me to throw up an answer. I think it is valuable, though, because sometimes having the kids come up not only helps those two, but it also, you know, kids might be paying a little bit more attention. (Mr. Edwards, November 30, 2015)

Mr. Miller, Ms. Jones, and Ms. Robinson also maximize student learning by creating classroom environments for students that are safe or involve multiple activities, transitional beliefs. Both Ms. King and Ms. Jones also expressed instructive ideas related to monitoring students to maximize learning (e.g., using short deadlines, prodding students who are off task, giving assessments). Ms. Jones also discussed traditional ideas related to creating a structured environment for learning by using a routine every day and providing prepared lessons.

Responses to TBI items related to beliefs about student learning show that case study teachers believed students learn computer science best with a variety of activities that include some form of peer exchange and the space to pursue topics of interest. These student-centered beliefs were reflected in the setup of the teachers' classrooms and the structure of their instructional activities. For example, Ms. King implemented a flipped classroom where students watched lectures from an online course at home and spent the majority of class time working on projects collaboratively and receiving support from the teacher and volunteers. Ms. Jones frequently incorporated peer grading in her labs where students had to evaluate each other's programs against a rubric. Mr. Edwards purposely arranged the physical space of his classroom to facilitate different types of activities:

> So the reason I have the room set up [with desks in the middle and computers on the perimeter] is …so I can have two groups, and I can talk with one group about one thing while the other group is at the computer, and then I switch back…It makes for a much, much more chaotic teaching environment, but I feel like it gives all the kids - they can pursue what interests them most…I hope this has been the most effective year we have had in dealing with this issue. The reason this is an issue is because unlike where everybody who is in pre-calc just finished algebra II…everybody comes in from whatever background…I think that is going to be likely in most computer programming AP classes nationally, until - if we had a series of pre-classes that were required, that would cut down on that a lot. (Mr. Edwards, March 9, 2016)

While the use of various activities aligned with teachers' beliefs of student learning, it did present teachers with a management challenge. Using various activities allowed students to move at their own pace, which also meant students needed a lot of individualized support. As Mr. Perez once said, "there was only one of me and nine different groups of students with different needs." This is where teachers benefitted from volunteer support, distributing the task of assisting students across the instructional team. So, while teachers arranged their classrooms to align with their beliefs about student learning, it created a demand for attention from students that was fulfilled with volunteer support.

Epistemological beliefs across domains. An underlying premise of the TEALS model is that teachers will draw on their existing pedagogical knowledge when they begin teaching computer science. Does this extend to their epistemological beliefs as well? Each teacher was visited once in a non-CS class and during these visits several teachers compared their teaching across disciplines. Ms. King taught support mathematics classes in addition to the AP CS course. Her teaching in the AP CS course seemed more aligned with her transitional and student-centered beliefs than her math teaching. She preferred to "demonstrate how to do something, and then they work independently while I come around and help them", but found this was easier to accomplish in her CS classes because students chose to enroll in the class. She believed students in her mathematics course lacked motivation which led her to use more direct instruction. Mr. Edwards, who also taught animation and web design courses, appeared to have responsive beliefs and he designed his classes so that "kids are doing what they want to do". For him, his AP CS course was less aligned with his beliefs because "the AP curriculum is pretty dry…it is almost like you are studying for this test, you are learning a specific set of skills, and there is not a lot of room for creativity, for getting outside the bounds of this very narrow scope of the course. I don't know, some teachers might consider that more fun, but I really enjoy the other classes more.” Both Ms. Jones and Ms. Robinson discussed how the nature of computer science and mathematics at the high school level differed, creating a need for different styles of teaching. For example, Ms. Jones commented, “I would say in computer science, it is probably more big picture problem solving, whereas in math, it is more procedural and less creative…The strategies are different because in computer class it is mostly lab time, so it is a lot of one-on-one [time] or they have to talk to their neighbor. It is just learning through doing. Whereas like in math, a lot of it is just like teacher guided practice.” These comments suggest that epistemological beliefs vary across disciplines and that the nature of disciplines and students’ abilities and motivation strongly influence instructional choices implemented inside classrooms.

Summary. Teachers expressed mostly student-centered beliefs about teaching and learning. Participant comments showed some alignment with the core practices of the K-12 Computer Science Framework, especially the practices of collaborating around computing and communicating about computing. Ms. Jones differed slightly from the other teachers with a mix of both teacher-centered and student-centered beliefs. Her teacher-centered beliefs appeared more strongly in her mathematics teaching. There were no noticeable differences between beliefs held by teachers based on their PD stage. Comments on the TBI and during interviews suggest that external factors such as the curriculum, students’ ability levels and interest, volunteer preferences, and the physical space of their classrooms sometimes influenced teachers to make instructional decisions that conflicted with their preferred beliefs. This was particularly apparent when teachers compared their computer science teaching to their teaching in other disciplines.

Both Ms. King and Ms. Jones expressed discipline dependent beliefs, while Mr. Edwards appeared to carry consistent beliefs across all his courses. He was the only teacher to say his CS course felt more restrictive than his non-CS courses. The data gathered about teachers' epistemological beliefs and their influence on instructional practices have implications for current efforts to expand CS across the nation. While teachers possessed epistemological beliefs aligned with the student-centered pedagogies promoted by the CS education community, contextual factors still influenced instructional decisions that conflicted with these beliefs (e.g., participants modifying their typical teaching practices when reviewing for the AP exam). Transitioning CS teachers who are placed in more restrictive courses like the AP course may need additional supports to successfully navigate tensions between the demands of course and student-centered CS teaching practices.

6.5 Discussion

In this chapter I explored participants' teacher confidence and epistemological beliefs and their relationship to instructional responsibilities. Self-reported participant data revealed at least two sources of teacher confidence: content knowledge and teaching experience. Data gathered from the beliefs questionnaire showed participants held mostly student-centered beliefs about learning but disparate beliefs about teaching that reflected tension between their ideas and external pressures. Context also influenced confidence and beliefs. For example, the results from Ms. Jones showed that confidence decreased when she entered a new teaching context (e.g., teaching Java). Also, while sharing instructional responsibilities with volunteers supported participants in balancing their teaching tasks, this distribution of responsibilities sometimes took away opportunities that could support teacher confidence. Lastly, when visited in their non-CS courses, some teachers discussed different epistemic beliefs that influenced their instructional decisions.**Limitations.** I measured teaching confidence using two closed-ended questionnaire items focused on comfort with content and ability to guide student learning. Confidence, however, also involves teachers' feelings of efficacy towards other and more fine grained teaching responsibilities such as motivating students, gauging student understanding, and providing alternative explanations (Tschannen-Moran & Hoy, 2001). Other factors involved in confidence might explain Mr. Edwards' discrepant results. So, the findings here might present a narrow and incomplete view of participants' teaching confidence. Future work in this area might explore the other sources of confidence within CS teaching.

Regarding the teacher belief data, teachers were asked about their beliefs once at the end of the school year. Capturing teachers' beliefs across multiple months could provide more insight into the consistency, or shifts, of the viewpoints observed in this study. Here, there is no comparison against which to judge the teachers' responses. Second, teachers provided their responses on a questionnaire and a range of beliefs were reported. The questionnaire format did not allow me to probe if teachers held certain beliefs more strongly than others. However, the teachers' responses do suggest that their beliefs influenced choices they made in their classroom and they highlighted instances where external factors prevented them from enacting instruction aligned with their beliefs.

CHAPTER 7. DISCUSSION

The goal of this study was to provide insight into how transitioning CS teachers improve their practice by exploring the relationship between PCK and instructional responsibilities. In most studies of computer science PCK, the focus is either on pre-service teachers or experienced CS teachers. Studies of teachers outside this dichotomy are rare. Similar to Liberman et al. (2012), I focused on individuals who were neither novice CS educators nor expert CS educators, but rather experienced educators who were learning to teach a new discipline. In the United States, a focus on transitioning teachers is important because many CS educators come from the ranks of in-service teachers trained in other disciplines. In the following paragraphs, I discuss each research question explored in this project, how the study results relate to larger bodies of literature on CS PCK and PCK in other disciplines, and implications for research and for practice.

7.1 Summary of Findings

What knowledge of computer science content, student thinking, and instructional strategies do teachers develop? Three findings emerged related to participants' content knowledge and PCK. First, teachers displayed more knowledge about student difficulties than instructional strategies. Within the context of on-the-job teacher learning, this finding might be expected. A classroom of twenty or so pupils might provide teachers with more opportunities to interact with student ideas than with the instructional strategies observed in their volunteer content experts (many of whom are not trained pedagogically as teachers) or gleaned from instructional materials. However, this finding aligns with the results of Schneider and Plasman's learning progression of science educators that indicate "it is helpful for teachers to think about learners first, then to focus on teaching" (Schneider & Plasman, 2011, p. 27). So, the transitioning CS teachers in this study may be following a similar progression in their PCK development. This distinction may suggest that transitioning CS teachers may benefit from concrete experiences around student learning before they begin to consider how best to support that learning.

Second, teachers displayed some knowledge expressed by expert CS educators when identifying student difficulties, instructional strategies, and difficult topics, but their knowledge was not completely identical to the expert educator list. It may be expected that transitioning CS teachers possess a subset of the PCK held by expert educators, but that does not explain why participants offered ideas that were not mentioned on the expert educator list. This leads me to ask, who should be the expert comparison group for transitioning CS teachers? Zendler and colleagues (Zendler & Hubwieser, 2013; Zendler & Klaudt, 2012) found that high school teachers attached different importance to content concepts of CS than university professors. Similarly, Schulte and Bennedsen (2006) found high school teachers, college instructors, and university professors provided different rankings for the difficulty, relevancy, and cognitive level of CS topics. Baxter's (1987) case study of two experienced secondary CS teachers, one formally trained in CS and the other mostly self-taught, revealed differences in the structure of their programming knowledge, the ways they communicated their knowledge to students, and the ways they structured their units. The results of these studies suggest that CS PCK will depend on the teaching context within which one works (i.e., secondary vs. tertiary) and the source of one's subject matter knowledge. Differences in teaching knowledge across grade levels may be explained by the aims of CS learning at each level. Within the U.S., for example, many secondary courses cover foundational and introductory CS topics while many tertiary courses provide greater depth and coverage of the CS discipline. What one considers essential knowledge at the introductory level probably differs from what one considers essential knowledge at an advanced level. Differences in teacher knowledge based on teachers' training might stem from epistemological differences about the nature and methods of CS represented in one's training course (e.g., technocratic versus scientific views of computing). So, it remains unclear if the results gathered in this study simply confirm that transitioning CS teachers have less PCK than experienced teachers or if the expert comparison sources were insufficient for judging the knowledge of secondary teachers who are developing their craft mostly through on-the-job experiences.

Lastly, while Ms. Robinson displayed both low content knowledge and low PCK, the relationship between content knowledge and PCK for other participants varied. Mr. Miller and Ms. Jones identified similar numbers of student difficulties and teaching strategies on the PCK questionnaire but performed differently on their think-aloud tasks, with Ms. Jones showing a more complete understanding of CS topics. Ms. King identified the most student difficulties and teaching strategies on the PCK questionnaire, but performed less well on the content assessment than Mr. Miller and Ms. Jones. Admittedly, Mr. Miller and Ms. Jones taught a different course than Ms. King. However, one might expect a teacher who performed better on the PCK questionnaire to also perform better on assessments of content knowledge, but the findings do not support this hypothesis. One explanation for these observations is that content knowledge and PCK might still be distinct knowledge areas for these transitioning teachers, each developing at its own rate. Many models of teaching knowledge depict content knowledge as related to but separate from PCK (e.g., D. L. Ball et al., 2008). However, Baxter (1987) posited that this distinction would be less clear in more expert teachers, because as teachers acquire more experience, teaching become the lens through which they interpret content knowledge. Another interesting finding is that the participants did not all offer the same student difficulties or teaching strategies on the PCK questionnaire. As stated above, Baxter (1987) found that the PCK of two experienced CS teachers looked different, so we might also expect the developing PCK of transitioning teachers to look different across individuals. Differing development pathways has methodological implications for the study of CS PCK. Namely, one instrument may not be sufficient to capture a complete view of developing teacher knowledge. And, as suggested by Baxter and Lederman in a review of methods to study PCK, simply assessing teacher knowledge without considering how teachers transform that knowledge into practice "incurs a significant risk of distorted meaning and interpretation" (Baxter & Lederman, 1999, p. 159).

A second explanation for these observations is that the instruments used to gather evidence of PCK were methodologically flawed. The PCK questionnaire, content assessment, and think-aloud interviews were based off instruments used in prior studies and materials used in CS teaching practice. However, these instruments did not undergo empirical validation. Such a process could refine the instruments so that they gather more consistent and confirming evidence of CS PCK.

What instructional responsibilities do teachers undertake when planning and implementing their lessons within the conditions of their co-teaching partnerships? Teachers discussed or were observed implementing several responsibilities related to planning and implementing instruction. The responsibilities focused on in this study overlap with several *fields* *of pedagogical operation* and *aspects of teaching and learning* identified in KUI's literature-based model of CS PCK that was validated with interviews of experienced CS educators (Hubwieser, Magenheim, et al., 2013; Margaritis et al., 2015). Two responsibilities discussed by study participants are not explicit in the KUI model: reviewing materials and practicing materials. In addition to helping teachers plan their classes, these responsibilities provided teachers with opportunities to practice the content covered in their lessons and to practice methods used in the CS discipline (e.g., peer review of coding scripts). Indeed, these tasks may be more pertinent to teachers new to course content than to experienced teachers, whose responsibilities are reflected in the KUI model. When teachers have limited content knowledge, they devote more of their planning time to learning content than to preparing instruction (Borko & Livingston, 1989). Another responsibility that supported CS PCK development was assisting students. Helping students work through assignments in class gave teachers a view into students' problem-solving processes and not just their resultant coding scripts. Problem solving is a core activity of CS and one's process for approaching a problem is just as important as the outcome of the solution. By assisting students in class, transitioning CS teachers had multiple opportunities to observe this disciplinary activity and began to develop heuristics for judging the quality of student work.

Self-reported data of the instructional responsibilities assumed during each classroom observation revealed that teachers were involved in all aspects of teaching, but to varying degrees depending on how they implemented their co-teaching relationship. Participants in teacher-led teams had greater responsibility for developing lessons and creating assignments. Participants in collaborative teams had less responsibility for assisting students and grading student work. These different approaches to co-teaching are not unlike models used in special education, English language learning, and gifted education (e.g., Hughes & Murawski, 2001; Pardini, 2006; Scruggs et al., 2007). The co-teaching approaches used by Ms. King, Mr. Edwards, Mr. Perez, Mr. Miller, and Ms. Robinson fall into the category of *one teach, one assist* where “one teacher leads instruction while the other circulates among the students offering individual assistance” (Friend, Cook, Hurley-Chamberlain, & Shamberger, 2010, p. 12). The approach used by Ms. Jones can be described as a hybrid mixture of *team teaching* where “both teachers are responsible for planning, and they share the instruction of all students” (Perl, Maughmer, & McQueen, 1999, p. 12) and *one teach, one assist*. Other models of co-teaching such as parallel teaching, alternative teaching, and station teaching where not used by the case study teachers. This is understandable because the TEALS program encouraged the one teach, one assist model. Some scholars have encouraged the use of co-teaching to aid student teachers during their practicums (e.g., Perl et al., 1999), and the results of this study provide support for the utility of co-teaching with transitioning teachers as well. It seems extremely difficult for one teacher to balance the demands of learning new content, understanding how to transform that content into ways that support student learning, and attending to all their other course responsibilities. Distributing instructional responsibilities across multiple instructors seems to benefit teachers by allowing them to offload some of the tasks required in running a classroom and concentrating on a smaller number of responsibilities.

A comparison of participants’ self-reported data of instructional responsibilities and observation data gathered by the research team highlighted discrepancies in how teachers and observers defined some teaching tasks. When reporting on the frequency of delivering lessons, participants included both activities that occurred during lectures and during lab time. However, observers only focused on the former. The ways teachers engaged in leading their class during lectures differed from lab time and should be more clearly distinguished in future instruments used to measure instructional responsibilities.

How does teacher knowledge support the implementation of teaching responsibilities? The study results provide some preliminary insights into the relationship between teacher knowledge and teaching responsibilities. First, it seems that responsibilities which require teachers to critically analyze artifacts of their practice (e.g., assessing the appropriateness of a quiz found on the Internet) support teachers in making instructional decisions if they have the appropriate content knowledge. It also seems that responsibilities which provide immediate, external feedback (e.g., students highlighting an error presented in a lesson) can encourage teachers to refine their teaching strategies. The cognitive mechanism behind this decision making and refinement appears to be reflection, which many scholars have highlighted as important in transforming teaching practice (e.g., Rodgers, 2002). By reflecting on their practice either during teaching or while preparing for instruction, what Schön (1983) called reflection-in-action and reflection-on-action, teachers revise their PCK to incorporate what they have learned while implementing their responsibilities. Second, the data suggest that content knowledge mediates how teaching responsibilities support PCK development, particularly when creating or reviewing materials. For example, a teacher with high content knowledge might be better equipped to create a quiz from scratch, while a teacher with low content knowledge might benefit from using an existing quiz with an answer key to more easily identify salient features of the assessment. So, when deciding if a responsibility will support a particular teacher's PCK development, one should consider both the amount of reflection required as well as the level of content knowledge needed. Third, some of the observed instructional responsibilities were unique to CS teaching (e.g., code reviews, managing the computer lab as students worked through CS tasks). These responsibilities have the potential to provide teachers with a better understanding of the discipline of CS and its authentic practices. Teachers encountering these responsibilities for the first time might require additional support from content experts to identify salient aspects of those responsibilities. Lastly, some responsibilities did not appear to support CS PCK development (e.g., managing the classroom, advocacy activities) but proved useful for motivating teachers. This leads me to suggest that in selecting tasks to best support PCK development, one cannot ignore the motivational and affective factors that encourage teacher learning and persistence. Teachers may also need to engage in less reflective activities that support the growth of other areas of their professional identities such as their identity as a CS teacher.

How do confidence and epistemological beliefs influence instructional responsibilities? While not a primary focus of this dissertation, it became clear from participants' interview comments that attending to confidence and epistemological beliefs might provide more insight into teachers' developing PCK and instructional responsibilities. The patterns I observed in this study give more credence to prior research in these areas, namely that (a) teaching confidence and beliefs influence instructional decisions and (b) confidence increases with greater content knowledge (Kordaki, 2013; Morrison et al., 2012; Ni, 2009; Ni & Guzdial, 2012; Ross et al., 1999). Participants in this study associated their confidence with both their comfort with subject matter and their prior experiences teaching topics. The data suggest that as teachers gained more of a lead role in their classrooms and assumed more instructional responsibilities, their confidence also increased. Results from participants who used either a volunteer-led or collaborative co-teaching approach suggest that confidence stemming from prior teaching experiences might appear before confidence stemming from understanding of subject matter content. Ms. Jones showed how confidence can decrease when teaching a new curriculum for the first time. She is the only participant who switched to a new curriculum during the study. Her decline in confidence differed from Liberman et al.'s (2012) case study of an experienced CS teacher who maintained her confidence when she began to teach a new programming paradigm. It may be that prior confidence influences a teacher's response to new curricula, with teachers possessing lower levels of confidence more susceptible to a decrease in confidence when facing new courses. It is important to explore the effect of curriculum changes on teacher confidence within CS because most teachers who persist in their careers as CS educators will eventually need to learn new programming languages or concepts as the field continues to evolve. Responses to the epistemological beliefs questionnaire showed teachers held mostly student-centered beliefs, which align with dominate ideas about K-12 CS education in the U.S., but that external demands, such as the restrictive nature of the AP course, led teachers to take actions counter to their beliefs. Lastly, participants' beliefs and teaching looked different across disciplines, suggesting that the nature and culture of a discipline can influence instructional practices.

7.2 Implications for Research

This study was an initial attempt to understand the PCK of transitioning CS teachers. The study results provide a portrait of CS teaching knowledge, insights into the enactment of teaching knowledge, and insights into the influence of beliefs and confidence on this knowledge. Given the limited amount of prior research on this topic, I assumed an exploratory focus for this case study which led to some methodological limitations as I borrowed, revised, and created new instruments to elicit evidence of PCK at different phases of the project. In any research endeavor, investigators must delimit their work to particular participants and settings. For this study, I focused on a small group of math certified secondary teachers participating in the TEALS program at schools located within the San Francisco Bay Area. Obviously, these bounds limit the generalizability of this work and raise questions about PCK development of teachers working in other contexts. Below I offer suggestions for future work to address these limitations and further enhance our understanding of CS PCK development.

Documenting CS PCK Development. The PCK questionnaire proved useful in eliciting examples of teacher knowledge related to the concept of lists and distinguishing teachers based on their content knowledge and experience independently leading CS courses. A natural extension of this work would be to ask teachers to complete the questionnaire for other big topics in introductory secondary computing courses. Lists were selected because all participants had experience teaching the topic. However, just as prior research has shown that topics vary in their difficulty for students, they might also vary in their difficulty for transitioning teachers. It is possible that Ms. King provided more student misconceptions because she understood lists better than other teachers. If asked to complete the same exercise for a more difficult topic such as recursion, would she provide a similar number of student difficulties? Also, my analysis focused more on the quantity of responses given and less on the quality of those ideas. Future work should investigate the effectiveness of the teaching methods provided and the prevalence of the student misconceptions identified. This may require new techniques to identify those methods and misconceptions that do not only rely on teacher opinion, such as data mining or learning analytics (e.g., Blikstein et al., 2014; Cherenkova et al., 2014). Another limitation is that the PCK questionnaire was administered once at the end of the school year. Examining responses to the questionnaire at different points in a teacher's career can provide insight into how PCK changes over time. This could be accomplished with longer longitudinal studies or with novice-transitioning-expert teacher comparisons.

Looking Inside the CS Classroom. Lesson observations provided information on the types of activities covered in classrooms and the distribution of teaching responsibilities between teachers and volunteers. However, our inability to record lessons made it difficult to focus on other events. It would be interesting to compare a teacher's PCK with the enactment of that knowledge in the classroom. For example, one could examine the teaching strategies used for particular topics and how those strategies vary based on a teacher's level of PCK for given topics. Recordings of classroom video could also be useful in spawning new lines of research on CS teacher learning. For example, video has been used to help mathematics teachers notice, reflect on, and capture critical teaching moments (e.g., Seago, 2003; van Es & Sherin, 2008; van Es, Stockero, Sherin, Zoest, & Dyer, 2015). Within CS, video studies could help researchers understand what transitioning educators focus on in their classrooms and how focal events provide, or prevent, opportunity of PCK development. Similar to the TIMSS Video Study (Stigler & Hiebert, 1999), video studies in CS classrooms could contribute to a repository of recordings to use in teacher education, evaluating the impact of professional development on CS teaching practice for transitioning teachers, and informing educational policy.

Distributing Instructional Responsibilities. The distribution of instructional responsibilities between teachers and volunteers was influential in providing teachers with learning opportunities. However, TEALS encouraged a one teach, one assist model so I was not able to explore the utility of other co-teaching models. Do other models support PCK development differently? Are other models more beneficial for different levels of content knowledge? Also, TEALS recruits volunteers from the tech industry which, according to Eden (2007), is dominated by a technocratic view of computing. Would the supports provided through the co-teaching model differ if volunteers were recruited from research settings where rationalist and scientific views of computing may be more common? Not all teachers have the luxury of support from volunteer content experts nor is a model like TEALS easy to implement in more isolated regions like rural America. Even within a program such as TEALS, volunteers can sometimes withdraw from their duties. For teachers without access to content experts, can a co-teaching model be simulated through other means and do these alternatives provide teachers with the same learning opportunities? For example, Ms. King implemented a flipped classroom where students watched lectures on a popular MOOC. Could a teacher working alone use a similar approach in place of volunteer content experts, thereby allowing her more time to focus on assisting students during lab time?

Crossing Disciplinary Boundaries. There seems to be a popular belief that mathematics and computer science are closely related disciplines so it is easier for a mathematics teacher to learn CS content than a teacher trained in another subject. But how do transitioning teachers develop CS PCK if they come from other, supposedly less similar, subjects? In the future, I would like to investigate the CS PCK development of in-service teachers trained in other disciplines, including non-STEM fields. Such a study would provide more insights into (a) the relationship between content knowledge and PCK, (b) the ways teachers make use of their existing teacher knowledge when transitioning to a new course, and (c) the unique challenges each discipline brings to transitioning CS teachers. Another interesting area of research in this avenue is to explore how transitioning to CS impacts instructional knowledge, beliefs, and practice in teachers' primary disciplines. For example, current CS courses promote a vision of education aligned with reform teaching. Does this encourage teachers to bring reform practices back to their other courses? For mathematics teachers, who are encouraged to implement reform practices in their math teaching but do not, do their CS experiences provide new insights into how they can practically bring reform practices to their math classrooms?

Supporting Students. Teachers are not the only ones crossing disciplinary boarders, their students are as well. How do teachers support students in learning the epistemic beliefs and inquiry practices of CS, which may differ from other subjects they have studied? Also, the goal of teaching is to presumably improve student outcomes. So, while it is important to identify what experiences might best support teacher development, we need to also consider which of those experiences will also lead to desired student outcomes.

Culturally Relevant PD. Lastly, the computing field suffers from a lack of diversity. I do not know if this disparity extends to the secondary CS teaching force, nor did I attend to the role of equity and culture in teacher knowledge development. Other scholars have investigated equity-based and culturally relevant practices to support students in computing as well as professional development programs to disseminate these practices (e.g., Eglash et al., 2006; Margolis et al., 2014; Pinkard et al., 2017; Scott, Clark, Hayes, Mruczek, & Sheridan, 2010).

However, I wonder if these ideas also extend to teacher learning. Results from the work of Hu, Heiner, and McCarthy's (2016) to introduce ECS across the state of Utah suggest that culture does play a role in teacher learning and acceptance of professional development. So, future research on teacher learning in CS should also consider issues of equity and culture.

7.3 Implications for Policy and Practice

Given the demand for more CS learning opportunities in American schools, educational agencies are turning to in-service teachers to increase their CS teaching force. Out-of-field teaching assignments are not new and may lead to unexpected instructional repercussions. For example, experienced teachers working in new domains rely more on procedural rules, implement less risky student-focused strategies, fail to anticipate student ideas, and have lower teacher efficacy (Ross et al., 1999). This dissertation highlighted specific challenges faced by in-service mathematics teachers transitioning to CS that can inform the decisions of administrators and teacher educators when expanding their CS programs.

First, in addition to content and curricular knowledge, professional development opportunities should include a focus on several additional areas to support experienced teachers transitioning into CS classrooms. As argued by Armoni (2011), two of these areas are PCK and the nature of CS. Teachers need to be aware of common student conceptions about CS ideas, types of problem solving approaches students use with CS tasks, and effective instructional methods specific to CS. Such knowledge is needed when teachers are making decisions around lesson design, identifying issues in student work, and evaluating the utility of instructional materials. Teachers also need exposure to the nature of CS, particularly to see that CS is not limited to programming and to understand how their courses relate to the broader discipline.

Without an accurate view of the discipline, teachers may transmit negative and limited preconceptions of the field to students and be less equipped to evaluate the relevance of new material, ideas, and behaviors they encounter in their classrooms. Teachers also need exposure to authentic disciplinary experiences not only to learn CS content but to also develop a stock of metaphors and real-world examples they can bring back into their classrooms. Related to these knowledge areas, professional development for transitioning teachers also needs to highlight the pluralism of problem solving in CS. Learners can and will use multiple problem-solving approaches, all of which cannot be identified a priori. Teachers will need to develop comfort with working through unexpected approaches in the moment. They will also need to consider how to support a classroom of students who may all need individualized supports to address unique problems within the timeframe of a class period. Professional development opportunities should devote attention to disciplinary differences between CS and teachers' primary subject area. This can help teachers identify which instructional practices may be more or less effective to carry into CS and the epistemic cognitions students might bring with them into CS classrooms. These learning opportunities cannot be confined to one professional development experience. CS teachers will need opportunities for continual learning throughout their careers to stay current with advances in the discipline.

Second, transitioning teachers should be given extra time to balance the demands placed upon them in taking on a CS teaching assignment. Although experienced teachers may not need to worry about their general pedagogical skills, they do need to learn new content, curricula, and instructional tasks (e.g., managing a computer lab). In addition, they must also continue teaching their primary courses and fulfilling other professional obligations (e.g., school committees, extracurricular teams). Asking teachers to handle existing responsibility along with their new CS responsibility can be overwhelming and sometimes make it impossible for teachers to fulfill all their obligations. This can be an issue if, for example, teachers minimize tasks that would support them in becoming more effective CS teachers. Reducing the number of courses teachers are assigned as they first take on CS is one way to alleviate this burden. Assigning teachers multiple sections of their CS courses can provide more opportunities for them to practice and refine their CS teaching.

Third, stakeholders should also attend to the alignment between curricula selected for their schools and their educational aims for CS, not only for students, but also for teachers. CS courses vary in their demands. For example, AP courses, which are tied to high-stake exams, may place demands on teachers to cover the curriculum in a certain way that is less productive for their teacher knowledge development. For teachers who still need to experiment with their CS instruction, this leaves little room for failure. Teachers very new to CS may fare better with less high-stake courses. Also, study participants teaching the AP CS A course expressed conflict between the goals of the AP curriculum and their desire to present CS as a fun, exploratory topic to their students. This conflict sometimes led participants to teach counter to their epistemological beliefs. Teachers who continually find themselves in the position of teaching CS counter to their beliefs may lose interest and motivation in persisting in their new roles.

Fourth, supports should be made available to help teachers establish professional learning communities, especially for isolated teachers who have no CS colleagues within their schools or districts. Teachers can benefit from interacting with colleagues to identify projects for students, discuss the advantages and disadvantages of programming environments, get feedback on newly created curricular materials, and learn about different instructional approaches. Also, interacting with teachers coming from similar contexts (e.g., other mathematics teachers, other teachers delivering the AP CS Principles course) can provide opportunities for teachers to discuss challenges and find solutions common to their contexts. These communities can also provide the sense of identity needed to help teachers persist through their transitions and continue with their computer science teaching assignments. Supports for professional learning communities can take the form of time given to teachers for such meetings, logistical support (e.g., a meeting space within the district), or the creation of online networks to facilitate exchanges between teachers.

7.4 Conclusion

Much remains to be learned about the PCK development of experienced secondary teachers who are transitioning into computing classrooms. While computer science education research may be considered a young field relative to other areas, this study and the work of other researchers are building an empirically validated knowledge base of computer science teacher learning. It is my hope that these research efforts will continue and that the results of this work will equip teachers with the knowledge and skills to become better computer science educators. The importance of computing in our lives continues to grow and I believe that it will, like mathematics and literacy, become a gateway to educational opportunities and economic livelihood. Teachers, well prepared teachers, will play a vital role in helping youth pass through the computing gateway.

REFERENCES

Abell, S. K. (2008). Twenty Years Later: Does pedagogical content knowledge remain a useful idea? *International Journal of Science Education*, *30*(10), 1405–1416. https://doi.org/10.1080/09500690802187041

Abelson, H., & DiSessa, A. A. (1986). *Turtle Geometry: The Computer as a Medium for Exploring Mathematics*. MIT Press.

Abrahamson, D., & Wilensky, U. (2007). Learning Axes and Bridging Tools in a Technology-Based Design for Statistics. *International Journal of Computers for Mathematical Learning*, *12*, 23–55.

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge. *Journal of Educational Technology & Society*, *19*(3), 47–57.

Arkansas Code Annotated, 6-16–146 A.C.A § (2015).

Armoni, M. (2011). Looking at Secondary Teacher Preparation Through the Lens of Computer Science. *Trans. Comput. Educ.*, *11*(4), 23:1–23:38. https://doi.org/10.1145/2048931.2048934

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From Scratch to “Real” Programming. *Trans. Comput. Educ.*, *14*(4), 25:1–25:15. https://doi.org/10.1145/2677087

Avalos, B. (2011). Teacher Professional Development in Teaching and Teacher Education over ten years. *Teaching and Teacher Education: An International Journal of Research and Studies*, *27*(1), 10–20. https://doi.org/10.1016/j.tate.2010.08.007

Ball, A. (2009). Toward a Theory of Generative Change in Culturally and Linguistically Complex Classrooms. *American Educational Research Journal*, *46*(1), 45–72. https://doi.org/10.3102/0002831208323277

Ball, D. L. (1988). *Knowledge and reasoning in mathematical pedagogy: Examining what prospective teachers bring to teacher education*. Michigan State University.

Ball, D. L., & Cohen, D. (1999). Developing Practice, Developing Practitioners: Toward a practice-based theory of professional education. *Teaching as the Learning Profession San Francisco: Jossey-Bass*.

Ball, D. L., & Forzani, F. M. (2009). The Work of Teaching and the Challenge for Teacher Education. *Journal of Teacher Education*, *60*(5), 497–511. https://doi.org/10.1177/0022487109348479

Ball, D. L., Thames, M. H., & Phelps, G. (2008). Content Knowledge for Teaching What Makes It Special? *Journal of Teacher Education*, *59*(5), 389–407. https://doi.org/10.1177/0022487108324554

Bamberger, J., & diSessa, A. (2003). Music as Embodied Mathematics: A Study of a Mutually Informing Affinity. *International Journal of Computers for Mathematical Learning*, *8*(2), 123–160. https://doi.org/10.1023/B:IJCO.0000003872.84260.96

Bandura, A. (1977). Self-efficacy: Toward a unifying theory of behavioral change. *Psychological Review*, *84*(2), 191–215. https://doi.org/10.1037/0033-295X.84.2.191

Barker, L. J., McDowell, C., & Kalahar, K. (2009). Exploring Factors That Influence Computer Science Introductory Course Students to Persist in the Major. In *Proceedings of the 40th*

ACM Technical Symposium on Computer Science Education (pp. 153–157). New York, NY, USA: ACM. https://doi.org/10.1145/1508865.1508923

Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010). Using Scalable Game Design to Teach Computer Science from Middle School to Graduate School. In *Proceedings of the Fifteenth Annual Conference on Innovation and Technology in Computer Science Education* (pp. 224–228). New York, NY, USA: ACM. https://doi.org/10.1145/1822090.1822154

Baumfield, V. (2006). Tools for pedagogical inquiry: the impact of teaching thinking skills on teachers. *Oxford Review of Education*, *32*(2), 185–196. https://doi.org/10.1080/03054980600645362

Baxter, J. A. (1987). *Teacher Explanations in Computer Programming: A Study of Knowledge Transformation* (Doctoral Dissertation). Stanford University, CA, United States.

Baxter, J. A., & Lederman, N. G. (1999). Assessment and measurement of pedagogical content knowledge. In *Examining pedagogical content knowledge* (pp. 147–161). Springer.

Bay Area Council Economic Institute. (2012). *The Bay Area: A Regional Economic Assessment*. Retrieved from http://www.bayareaeconomy.org/media/files/pdf/BAEconAssessment.pdf

Ben-Ari, M. (1998). Constructivism in computer science education. In *Proceedings of the twenty-ninth SIGCSE technical symposium on Computer science education* (pp. 257–261). Atlanta, Georgia, United States: ACM. https://doi.org/http://doi.acm.org/10.1145/273133.274308

Bender, E., Hubwieser, P., Schaper, N., Margaritis, M., Berges, M., Ohrndorf, L., … Schubert, S. (2015). Towards a Competency Model for Teaching Computer Science. *Peabody Journal of Education (0161956X)*, *90*(4), 519–532.

Bentley, J. L. (1982). *Writing Efficient Programs*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc.

Berglund, A., Daniels, M., & Pears, A. (2006). Qualitative research projects in computing education research: an overview. In *Proceedings of the 8th Australasian Conference on Computing Education-Volume 52* (pp. 25–33). Australian Computer Society, Inc.

Berliner, D. C. (2001). Learning about and learning from expert teachers. *International Journal of Educational Research*, *35*(5), 463–482. https://doi.org/10.1016/S0883-0355(02)00004-6

Berliner, D. C. (2004). Describing the Behavior and Documenting the Accomplishments of Expert Teachers. *Bulletin of Science, Technology & Society*, *24*(3), 200–212. https://doi.org/10.1177/0270467604265535

Bernier, D. (2012). *In Need of Repair: The State of K-12 Computer Science Education in California*.

Biggers, M., Brauer, A., & Yilmaz, T. (2008). Student perceptions of computer science: a retention study comparing graduating seniors with cs leavers. *SIGCSE Bull.*, *40*(1), 402–406. https://doi.org/10.1145/1352322.1352274

Blikstein, P., & Wilensky, U. (2009). An Atom is Known by the Company it Keeps: A Constructionist Learning Environment for Materials Science Using Agent-Based

Modeling. *International Journal of Computers for Mathematical Learning*, *14*(2), 81–119. https://doi.org/10.1007/s10758-009-9148-8

Blikstein, P., & Wilensky, U. (2010). MaterialSim: A Constructionist Agent-Based Modeling Approach to Engineering Education. In M. J. Jacobson & P. Reimann (Eds.), *Designs for Learning Environments of the Future* (pp. 17–60). Springer US. https://doi.org/10.1007/978-0-387-88279-6_2

Blikstein, P., Worsley, M., Piech, C., Sahami, M., Cooper, S., & Koller, D. (2014). Programming Pluralism: Using Learning Analytics to Detect Patterns in the Learning of Computer Programming. *Journal of the Learning Sciences*, *23*(4), 561–599. https://doi.org/10.1080/10508406.2014.954750

Blömeke, S., & Delaney, S. (2012). Assessment of teacher knowledge across countries: a review of the state of research. *ZDM Mathematics Education*, *44*(3), 223–247.

Blum, L., & Cortina, T. J. (2007). CS4HS: An Outreach Program for High School CS Teachers. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (pp. 19–23). New York, NY, USA: ACM. https://doi.org/10.1145/1227310.1227320

Borko, H., & Livingston, C. (1989). Cognition and Improvisation: Differences in Mathematics Instruction by Expert and Novice Teachers. *American Educational Research Journal*, *26*(4), 473–498. https://doi.org/10.3102/00028312026004473

Bort, H., & Brylow, D. (2013). CS4Impact: Measuring Computational Thinking Concepts Present in CS4HS Participant Lesson Plans. In *Proceeding of the 44th ACM Technical*

Symposium on Computer Science Education (pp. 427–432). New York, NY, USA: ACM. https://doi.org/10.1145/2445196.2445323

Brandes, A., & Wilensky, U. (1991). Treasureworld: An Environment for the Study and Exploration of Feedback. In *Constructionism*. Norwood N.J.: Ablex Publishing Corp.

Brockmeyer, M. A. (1998). *The impact of an extended inquiry-based in-service programme on the beliefs and practices of beginning secondary science teachers*. The University of Iowa, Iowa Cita, IA.

Buchholz, M., Saeli, M., & Schulte, C. (2013). PCK and Reflection in Computer Science Teacher Education. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (pp. 8–16). New York, NY, USA: ACM. https://doi.org/10.1145/2532748.2532752

Buchman, A. L. (1956). Computer Programming and Coding at the High School Level. In *Proceedings of the 1956 11th ACM National Meeting* (pp. 118–121). New York, NY, USA: ACM. https://doi.org/10.1145/800258.808964

Bucholtz, M. (2000). The politics of transcription. *Journal of Pragmatics*, *32*(10), 1439–1465. https://doi.org/10.1016/S0378-2166(99)00094-6

Buechley, L., Eisenberg, M., & Elumeze, N. (2007). Towards a Curriculum for Electronic Textiles in the High School Classroom. In *Proceedings of the 12th Annual SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 28–32). New York, NY, USA: ACM. https://doi.org/10.1145/1268784.1268795

Buehl, M. M., & Fives, H. (2016). The Role of Epistemic Cognition in Teacher Learning and Praxis. In J. A. Greene, W. A. Sandoval, & I. Bråten (Eds.), *Handbook of Epistemic Cognition* (pp. 247–264). New York, NY: Routledge.

Campbell, J. L., Quincy, C., Osserman, J., & Pedersen, O. K. (2013). Coding In-depth Semistructured Interviews Problems of Unitization and Intercoder Reliability and Agreement. *Sociological Methods & Research*, 0049124113500475. https://doi.org/10.1177/0049124113500475

Carbone, A., & Kaasbøll, J. J. (1998). A Survey of Methods Used to Evaluate Computer Science Teaching. In *Proceedings of the 6th Annual Conference on the Teaching of Computing and the 3rd Annual Conference on Integrating Technology into Computer Science Education: Changing the Delivery of Computer Science Education* (pp. 41–45). New York, NY, USA: ACM. https://doi.org/10.1145/282991.283014

Carbone, A., Mannila, L., & Fitzgerald, S. (2007). Computer science and IT teachers' conceptions of successful and unsuccessful teaching: A phenomenographic study. *Computer Science Education*, *17*(4), 275–299. https://doi.org/10.1080/08993400701706586

Carlsen, W. S. (1987). Why Do You Ask? The Effects of Science Teacher Subject-Matter Knowledge on Teacher Questioning and Classroom Discourse.

Carpenter, T. P., Fennema, E., Peterson, P. L., & Carey, D. A. (1988). Teachers' Pedagogical Content Knowledge of Students' Problem Solving in Elementary Arithmetic. *Journal for Research in Mathematics Education*, *19*(5), 385–401. https://doi.org/10.2307/749173

Carter, L. (2006). Why Students with an Apparent Aptitude for Computer Science Don’T Choose to Major in Computer Science. In *Proceedings of the 37th SIGCSE Technical Symposium on Computer Science Education* (pp. 27–31). New York, NY, USA: ACM. https://doi.org/10.1145/1121341.1121352

Century, J., Lach, M., King, H., Rand, S., Heppner, C., Franke, B., & Westrick, J. (2013). *Building an Operating System for Computer Science*. Chicago, IL: CEMSE, University of Chicago with UEI, University of Chicago. Retrieved from http://cemse.uchicago.edu/computerscience/OS4CS/

Cherenkova, Y., Zingaro, D., & Petersen, A. (2014). Identifying Challenging CS1 Concepts in a Large Problem Dataset. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 695–700). New York, NY, USA: ACM. https://doi.org/10.1145/2538862.2538966

Chicago Public Schools. (2014). CPS Announces First Schools to Implement District’s Comprehensive K-12 Curriculum [Press Release]. Retrieved from http://cps.edu/News/Press_releases/Pages/PR1_03_19_2014.aspx

Chinn, C. A., Buckland, L. A., & Samarapungavan, A. (2011). Expanding the Dimensions of Epistemic Cognition: Arguments From Philosophy and Psychology. *Educational Psychologist, 46*(3), 141–167. https://doi.org/10.1080/00461520.2011.587722

Chinn, C. A., & Malhotra, B. A. (2002). Epistemologically authentic inquiry in schools: A theoretical framework for evaluating inquiry tasks. *Science Education, 86*(2), 175–218. https://doi.org/10.1002/sce.10001

Clancy, M. (2004). Misconceptions and Attitudes that Interfere with Learning to Program. In *Computer Science Education Research* (pp. 85–100). Taylor and Francis.

Common Core of Data. (2016). *U.S. Department of Education. Institute of Education Sciences, National Center for Education Statistics.* Retrieved from https://nces.ed.gov/ccd/schoolsearch/

Computational Thinking in Science and Math. (2016). Retrieved April 24, 2017, from http://ct-stem.northwestern.edu/

Cooper, S., Grover, S., & Simon, B. (2014). Building a Virtual Community of Practice for K-12 CS Teachers. *Commun. ACM*, *57*(5), 39–41. https://doi.org/10.1145/2594456

Creswell, J. W. (2006). *Qualitative Inquiry and Research Design: Choosing among Five Approaches* (2nd edition). Thousand Oaks: SAGE Publications, Inc.

Creswell, J. W. (2008). *Research Design: Qualitative, Quantitative, and Mixed Methods Approaches, 3rd Edition* (3rd edition). Thousand Oaks, Calif: SAGE Publications, Inc.

Creswell, J. W. (2012). *Educational Research: Planning, Conducting, and Evaluating Quantitative and Qualitative Research*. Addison Wesley.

Creswell, J. W., & Miller, D. L. (2000). Determining Validity in Qualitative Inquiry. *Theory Into Practice*, *39*(3), 124–130. https://doi.org/10.1207/s15430421tip3903_2

CSTA. (2014). Results from the CSTA-Oracle Academy 2014 U.S. High School CS Survey: The State of Computer Science in U.S. High Schools: an Administrator’s Perspective. Retrieved from http://csta.acm.org/Research/sub/Projects/OracleSurvey2014.html

CSTA. (2015). *CSTA National Secondary School Computer Science Survey 2015*. Retrieved from http://csta.acm.org/Research/sub/Projects/ResearchFiles/CSTA_NATIONAL_SECONDARY_SCHOOL_CS_SURVEY_2015.pdf

Cuny, J. (2012). Transforming High School Computing: A Call to Action. *ACM Inroads*, *3*(2), 32–36. https://doi.org/10.1145/2189835.2189848

Cuny, J. (2015). Transforming K-12 Computing Education: An Update and a Call to Action. *ACM Inroads*, *6*(3), 54–57. https://doi.org/10.1145/2809795

Cutts, Q., Carbone, A., & van Haaster, K. (2004). Using an Electronic Voting System to Promote Active Reflection on Coursework Feedback. In *Proceedings of the International Conference on Computers in Education*. Melbourne, Australia.

Daehler, K. R., Heller, J. I., & Wong, N. (2015). Supporting Growth of Pedagogical Content Knowledge in Science. In J. Loughran, P. Friedrichsen, & A. Berry (Eds.), *Re-examining Pedagogical Content Knowledge in Science Education* (pp. 45–59). Routledge.

Dagdilelis, V., & Xinogalos, S. (2012). Preparing Teachers for Teaching Informatics: Theoretical Considerations and Practical Implications. In *Proceedings of the 7th Workshop in Primary and Secondary Computing Education* (pp. 78–81). New York, NY, USA: ACM. https://doi.org/10.1145/2481449.2481468

Dale, N. B. (2006). Most Difficult Topics in CS1: Results of an Online Survey of Educators. *SIGCSE Bull.*, *38*(2), 49–53. https://doi.org/10.1145/1138403.1138432

Darling-Hammond, L. (2000). Teacher Quality and Student Achievement. *Education Policy Analysis Archives*, *8*(0), 1. https://doi.org/10.14507/epaa.v8n1.2000

Davis, E. A. (2004). Knowledge integration in science teaching: Analysing teachers’ knowledge development. *Research in Science Education*, *34*(1), 21–53.

Denning, P. J. (2003). Great Principles of Computing. *Commun. ACM*, *46*(11), 15–20. https://doi.org/10.1145/948383.948400

Denning, P. J., Comer, D. E., Gries, D., Mulder, M. C., Tucker, A., Turner, A. J., & Young, P. R. (1989). Computing As a Discipline. *Commun. ACM*, *32*(1), 9–23. https://doi.org/10.1145/63238.63239

Depaepe, F., De Corte, E., & Verschaffel, L. (2016). Mathematical Epistemological Beliefs. In J. A. Greene, W. A. Sandoval, & I. Bråten (Eds.), *Handbook of Epistemic Cognition* (pp. 147–164). New York, NY: Routledge.

Depaepe, F., Verschaffel, L., & Kelchtermans, G. (2013). Pedagogical content knowledge: A systematic review of the way in which the concept has pervaded mathematics educational research. *Teaching and Teacher Education*, *34*, 12–25. https://doi.org/10.1016/j.tate.2013.03.001

Desimone, L. M. (2009). Improving Impact Studies of Teachers' Professional Development: Toward Better Conceptualizations and Measures. *Educational Researcher*, *38*(3), 181–199. https://doi.org/10.3102/0013189X08331140

Desimone, L. M., Porter, A. C., Garet, M. S., Yoon, K. S., & Birman, B. F. (2002). Effects of Professional Development on Teachers' Instruction: Results from a Three-year Longitudinal Study. *Educational Evaluation and Policy Analysis*, *24*(2), 81–112. https://doi.org/10.3102/01623737024002081

diSessa, A. A., & Abelson, H. (1986). Boxer: A Reconstructible Computational Medium. *Commun. ACM*, *29*(9), 859–868. https://doi.org/10.1145/6592.6595

Dogan, S., Pringle, R., & Mesa, J. (2016). The impacts of professional learning communities on science teachers' knowledge, practice and student learning: a review. *Professional Development in Education*, *42*(4), 569–588.

du Boulay, B. (1986). Some Difficulties of Learning to Program. In E. Soloway & J. C. Spohrer (Eds.), *Studying the Novice Programmer* (pp. 283–299). Lawrence Elbaum Associates.

du Boulay, B., & O'Shea, T. (1976). *How to work the LOGO machine: a primer for ELOGO*. University of Edinburgh, Department of Artificial Intelligence.

Dunst, C. J., Bruder, M. B., & Hamby, D. W. (2015). Metasynthesis of In-Service Professional Development Research: Features Associated with Positive Educator and Student Outcomes. *Educational Research and Reviews*, *10*(12), 1731–1744.

Eagle, M., & Barnes, T. (2008). Wu's Castle: Teaching Arrays and Loops in a Game. In *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education* (pp. 245–249). New York, NY, USA: ACM. https://doi.org/10.1145/1384271.1384337

Eden, A. H. (2007). Three Paradigms of Computer Science. *Minds and Machines*, *17*(2), 135–167. https://doi.org/10.1007/s11023-007-9060-8

Edwards, L. D. (1998). Embodying mathematics and science: Microworlds as representations. *The Journal of Mathematical Behavior*, *17*(1), 53–78. https://doi.org/10.1016/S0732-3123(99)80061-3

Eglash, R., Bennett, A., O'donnell, C., Jennings, S., & Cintorino, M. (2006). Culturally situated design tools: Ethnocomputing from field site to classroom. *American Anthropologist*, *108*(2), 347–362.

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *Academy of Management Review*, *14*(4), 532–550. https://doi.org/10.5465/AMR.1989.4308385

Eisenhardt, K. M., & Graebner, M. E. (2007). Theory Building From Cases: Opportunities And Challenges. *Academy of Management Journal*, *50*(1), 25–32. https://doi.org/10.5465/AMJ.2007.24160888

Ericson, B., Armoni, M., Gal-Ezer, J., Seehorn, D., Stephenson, C., & Trees, F. (2008). *Ensuring exemplary teaching in an essential discipline: Addressing the crisis in computer science teacher certification*. New York: The Computer Science Teachers Association.

Ericson, B., Rogers, K., Parker, M., Morrison, B., & Guzdial, M. (2016). Identifying Design Principles for CS Teacher Ebooks Through Design-Based Research. In *Proceedings of the 2016 ACM Conference on International Computing Education Research* (pp. 191–200). New York, NY, USA: ACM. https://doi.org/10.1145/2960310.2960335

Fang, Z. (1996). A review of research on teacher beliefs and practices. *Educational Research*, *38*(1), 47–65. https://doi.org/10.1080/0013188960380104

Feiman-Nemser, S. (2003). What New Teachers Need To Learn. *Educational Leadership*, *60*(8), 25–29.

Feldon, D. F. (2007). Cognitive Load and Classroom Teaching: The Double-Edged Sword of Automaticity. *Educational Psychologist*, *42*(3), 123–137. https://doi.org/10.1080/00461520701416173

Fields, D. A., Searle, K. A., & Kafai, Y. B. (2016). Deconstruction Kits for Learning: Students' Collaborative Debugging of Electronic Textile Designs. In *Proceedings of the 6th Annual*

Conference on Creativity and Fabrication in Education (pp. 82–85). New York, NY, USA: ACM. https://doi.org/10.1145/3003397.3003410

Fincher, S., & Petre, M. (Eds.). (2004a). *Computer Science Education Research*. Taylor & Francis.

Fincher, S., & Petre, M. (2004b). Part One: the field and the endeavor. In *Computer Science Education Research* (pp. 1–81). Taylor and Francis.

Fincher, S., Tenenberg, J., & Robins, A. (2011). Research Design: Necessary Bricolage. In *Proceedings of the seventh international workshop on Computing education research (ICER '11)* (pp. 27–32). Providence, RI, USA: ACM. https://doi.org/http://dx.doi.org/10.1145/2016911.2016919

Forsythe, G. (1963). Educational Implications of the Computer Revolution. In W. F. Freiberger & W. Prager (Eds.), *Applications of Digital Computers* (pp. 166–178). Boston, MA. Retrieved from https://exhibits.stanford.edu/feigenbaum/catalog/qx857vs2697

Forzani, F. M. (2014). Understanding "Core Practices" and "Practice-Based" Teacher Education: Learning From the Past. *Journal of Teacher Education*, *65*(4), 357–368. https://doi.org/10.1177/0022487114533800

Franke, M. L., Carpenter, T., Fennema, E., Ansell, E., & Behrend, J. (1998). Understanding teachers' self-sustaining, generative change in the context of professional development1. *Teaching and Teacher Education*, *14*(1), 67–80. https://doi.org/10.1016/S0742-051X(97)00061-9

Franke, M. L., Carpenter, T. P., Levi, L., & Fennema, E. (2001). Capturing teachers' generative change: A follow-up study of professional development in mathematics. *American Educational Research Journal*, *38*, 653–689.

Freeman, J., Magerko, B., McKlin, T., Reilly, M., Permar, J., Summers, C., & Fruchter, E. (2014). Engaging Underrepresented Groups in High School Introductory Computing Through Computational Remixing with EarSketch. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 85–90). New York, NY, USA: ACM. https://doi.org/10.1145/2538862.2538906

Friend, M., Cook, L., Hurley-Chamberlain, D., & Shamberger, C. (2010). Co-teaching: An illustration of the complexity of collaboration in special education. *Journal of Educational and Psychological Consultation*, *20*(1), 9–27.

Frieze, C. (2007). *The critical role of culture and environment as determinants of women's participation in computer science*. Carnegie Mellon University. Retrieved from file://C:/northwestern/compSci_endnote.Data/PDF/frieze_thesis_CMU-CS-07-118-0996839169/frieze_thesis_CMU-CS-07-118.pdf

Gal-Ezer, J., & Stephenson, C. (2010). Computer science teacher preparation is critical. *ACM Inroads*, *1*, 61–66.

Garcia, D. D., Franke, B., Hoeppner, S., & Paley, J. (2014). Teaching Tips We Wish They'D Told Us Before We Started: High School Edition. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 463–464). New York, NY, USA: ACM. https://doi.org/10.1145/2538862.2538870

Garet, M. S., Porter, A. C., Desimone, L., Birman, B. F., & Yoon, K. S. (2001). What Makes Professional Development Effective? Results From a National Sample of Teachers. *American Educational Research Journal*, *38*(4), 915–945. https://doi.org/10.3102/00028312038004915

Gay, G. (2002). Preparing for Culturally Responsive Teaching. *Journal of Teacher Education*, *53*(2), 106–16.

Giannakos, M. N., Doukakis, S., Crompton, H., Chrisochoides, N., Adamopoulos, N., & Giannopoulou, P. (2014). Examining and mapping CS teachers' technological, pedagogical and content knowledge (TPACK) in K-12 schools. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings* (pp. 1–7). https://doi.org/10.1109/FIE.2014.7044406

Ginat, D. (2008). Learning from wrong and creative algorithm design. In *Proceedings of the 39th SIGCSE technical symposium on Computer science education* (pp. 26–30). Portland, OR, USA: ACM. https://doi.org/http://doi.acm.org/10.1145/1352135.1352148

Goldman, S. R., Britt, M. A., Brown, W., Cribb, G., George, M., Greenleaf, C., … READI, P. (2016). Disciplinary Literacies and Learning to Read for Understanding: A Conceptual Framework for Disciplinary Literacy. *Educational Psychologist*, *51*(2), 219–246. https://doi.org/10.1080/00461520.2016.1168741

Goldsmith, L., Doerr, H., & Lewis, C. (2014). Mathematics teachers' learning: a conceptual framework and synthesis of research. *Journal of Mathematics Teacher Education*, *17*(1), 5–36.

Goode, J., & Margolis, J. (2011). Exploring Computer Science: A Case Study of School Reform. *Trans. Comput. Educ.*, *11*(2), 12:1–12:16. https://doi.org/10.1145/1993069.1993076

Goode, J., Margolis, J., & Chapman, G. (2014). Curriculum is Not Enough: The Educational Theory and Research Foundation of the Exploring Computer Science Professional Development Model. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (pp. 493–498). New York, NY, USA: ACM. https://doi.org/10.1145/2538862.2538948

Google CS4HS. (n.d.). Retrieved March 5, 2017, from https://www.cs4hs.com/

Google, & Gallup. (2015). *Searching for Computer Science: Access and Barriers in U.S. K-12 Education*.

Google Inc., & Gallup Inc. (2016a). *Diversity Gaps in Computer Science: Exploring the Underrepresentation of Girls, Blacks and Hispanics*. Retrieved from http://goo.gl/PG34aH

Google Inc., & Gallup Inc. (2016b). *Trends in the State of Computer Science in U.S. K-12 Schools*. Retrieved from http://goo.gl/j291E0

Goss, M., Powers, R., & Hauk, S. (2013). Identifying Change in Secondary Mathematics Teachers' Pedagogical Content Knowledge. In *Proceedings for the 16th conference on Research in Undergraduate Mathematics Education, Denver, CO.*

Götschi, T., Sanders, I., & Galpin, V. (2003). Mental Models of Recursion. In *Proceedings of the 34th SIGCSE Technical Symposium on Computer Science Education* (pp. 346–350). New York, NY, USA: ACM. https://doi.org/10.1145/611892.612004

Gray, J., Haynie, K., Packman, S., Boehm, M., Crawford, C., & Muralidhar, D. (2015). A Mid-Project Report on a Statewide Professional Development Model for CS Principles. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 380–385). New York, NY, USA: ACM. https://doi.org/10.1145/2676723.2677306

Grgurina, N., Barendsen, E., Zwaneveld, B., van Veen, K., & Stoker, I. (2014). Computational Thinking Skills in Dutch Secondary Education: Exploring Pedagogical Content Knowledge. In *Proceedings of the 14th Koli Calling International Conference on Computing Education Research* (pp. 173–174). New York, NY, USA: ACM. https://doi.org/10.1145/2674683.2674704

Griffin, J., Pirmann, T., & Gray, B. (2016). Two Teachers, Two Perspectives on CS Principles. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 461–466). New York, NY, USA: ACM. https://doi.org/10.1145/2839509.2844630

Grobe, T., Curnan, S., & Melchior, A. (1990). Synthesis of Existing Knowledge and Practice in the Field of Educational Partnerships.

Grossman, P. L., Hammerness, K., & McDonald, M. (2009). Redefining teaching, re-imagining teacher education. *Teachers and Teaching*, *15*(2), 273–289. https://doi.org/10.1080/13540600902875340

Grossman, P. L., & McDonald, M. (2008). Back to the Future: Directions for Research in Teaching and Teacher Education. *American Educational Research Journal*, *45*(1), 184–205. https://doi.org/10.3102/0002831207312906

Grossman, P. L., & Stodolsky, S. S. (1995). Content as Context: The Role of School Subjects in Secondary School Teaching. *Educational Researcher*, *24*(8), 5–23. https://doi.org/10.3102/0013189X024008005

Guzdial, M. (2003). A media computation course for non-majors (Vol. 35, pp. 104–108). ACM.

Guzdial, M. (2004). Programming Environments for Novices. In *Computer Science Education Research* (pp. 127–154). Taylor and Francis. Retrieved from file://C:/northwestern/compSci_endnote.Data/PDF/guzdial_programming environments for novices-2702252544/guzdial_programming environments for novices.pdf

Guzdial, M. (2010). Does contextualized computing education help? *ACM Inroads*, *1*, 4–6. https://doi.org/10.1145/1869746.1869747

Guzdial, M. (2011). Learning How to Prepare Computer Science High School Teachers. *Computer*, *44*(10), 95–97. https://doi.org/10.1109/MC.2011.316

Guzdial, M., Ericson, B., Mcklin, T., & Engelman, S. (2014). Georgia Computes! An Intervention in a US State, with Formal and Informal Education in a Policy Context. *Trans. Comput. Educ.*, *14*(2), 13:1–13:29. https://doi.org/10.1145/2602488

Handal, B. (2003). Teachers' Mathematical Beliefs: A Review. *The Mathematics Educator*, *13*(2). Retrieved from http://tme.journals.libs.uga.edu/index.php/tme/article/view/131

Hanks, B., & Brandt, M. (2009). Successful and unsuccessful problem solving approaches of novice programmers. In *Proceedings of the 40th ACM technical symposium on Computer science education* (pp. 24–28). Chattanooga, TN, USA: ACM. https://doi.org/http://doi.acm.org/10.1145/1508865.1508876

Hanks, B., McDowell, C., Draper, D., & Krnjajic, M. (2004). Program quality with pair programming in CS1. In *Proceedings of the 9th annual SIGCSE conference on Innovation and technology in computer science education* (pp. 176–180). New York, NY, USA: ACM. https://doi.org/10.1145/1007996.1008043

Harlen, W., & Holroyd, C. (1997). Primary teachers' understanding of concepts of science: impact on confidence and teaching. *International Journal of Science Education*, *19*(1), 93–105. https://doi.org/10.1080/0950069970190107

Hashweh, M. (1996). Effects of science teachers' epistemological beliefs in teaching. *Journal of Research in Science Teaching*, *33*(1), 47–63. https://doi.org/10.1002/(SICI)1098-2736(199601)33:1<47::AID-TEA3>3.0.CO;2-P

Hashweh, M. (2005). Teacher pedagogical constructions: a reconfiguration of pedagogical content knowledge. *Teachers and Teaching*, *11*(3), 273–292.

Hashweh, M. (2013). PEDAGOGICAL CONTENT KNOWLEDGE: TWENTY-FIVE YEARS LATER. *Advances in Research on Teaching*, *19*, 115–140.

Hazzan, O., Lapidot, T., & Ragonis, N. (2011). *Guide to Teaching Computer Science: An Activity-Based Approach*. Springer.

Hazzan, O., Lapidot, T., & Ragonis, N. (2015). *Guide to Teaching Computer Science: An Activity-Based Approach* (2nd ed. 2014 edition). London: Springer.

Heller, J. I., Daehler, K. R., Wong, N., Shinohara, M., & Miratrix, L. W. (2012). Differential effects of three professional development models on teacher knowledge and student achievement in elementary science. *Journal of Research in Science Teaching*, *49*(3), 333–362. https://doi.org/10.1002/tea.21004

Hilton, M., & Janzen, D. S. (2012). On Teaching Arrays with Test-driven Learning in WebIDE. In *Proceedings of the 17th ACM Annual Conference on Innovation and Technology in Computer Science Education* (pp. 93–98). New York, NY, USA: ACM. https://doi.org/10.1145/2325296.2325322

Holbert, N. R., & Wilensky, U. (2011). Racing Games for Exploring Kinematics: A Computational Thinking Approach. In *Proceedings of the 7th International Conference on Games + Learning + Society Conference* (pp. 109–118). Pittsburgh, PA, USA: ETC Press. Retrieved from http://dl.acm.org/citation.cfm?id=2206376.2206390

Holmboe, C., McIver, L., & George, C. (2001). Research agenda for computer science education. In *13th Workshop of the Psychology of Programming Interest Group* (Vol. 207223).

Hooper, P. K. (1996). They Have Their Own Thoughts. In Y. B. Kafai & M. Resnick (Eds.), *Constructionism in Practice: Designing, Thinking, and Learning in a Digital World* (pp. 241–254).

Hu, H. H., Heiner, C., & McCarthy, J. (2016). Deploying Exploring Computer Science Statewide. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (pp. 72–77). New York, NY, USA: ACM. https://doi.org/10.1145/2839509.2844622

Hubwieser, P., Berges, M., Magenheim, J., Schaper, N., Bröker, K., Margaritis, M., … Ohrndorf, L. (2013). Pedagogical Content Knowledge for Computer Science in German Teacher Education Curricula. In *Proceedings of the 8th Workshop in Primary and Secondary*

Computing Education (pp. 95–103). New York, NY, USA: ACM. https://doi.org/10.1145/2532748.2532753

Hubwieser, P., Magenheim, J., Mühling, A., & Ruf, A. (2013). Towards a conceptualization of pedagogical content knowledge for computer science. In *Proceedings of the ninth annual international ACM conference on International computing education research* (pp. 1–8). ACM.

Hughes, C. E., & Murawski, W. A. (2001). Lessons from another field: Applying coteaching strategies to gifted education. *Gifted Child Quarterly*, *45*(3), 195–204.

Jenkins, J. T., Jerkins, J. A., & Stenger, C. L. (2012). A Plan for Immediate Immersion of Computational Thinking into the High School Math Classroom Through a Partnership with the Alabama Math, Science, and Technology Initiative. In *Proceedings of the 50th Annual Southeast Regional Conference* (pp. 148–152). New York, NY, USA: ACM. https://doi.org/10.1145/2184512.2184547

Johns Hopkins University Center for Social Organization of Schools. (1983). *School Uses of Microcomputers. Reports from a National Survey. Issue No. 1.*

Joint Task Force on Computing Curricula, A. for C. M. (ACM), & Society, I. C. (2013). *Computer Science Curricula 2013: Curriculum Guidelines for Undergraduate Degree Programs in Computer Science*. New York, NY, USA: ACM.

Joy, M., Sinclair, J., Sun, S., Sitthiworachart, J., & López-González, J. (2008). Categorising computer science education research. *Education and Information Technologies*, *14*(2), 105–126. https://doi.org/10.1007/s10639-008-9078-4

K–12 Computer Science Framework. (2016). Retrieved November 27, 2016, from http://k12cs.org

Kaczmarczyk, L. C., Petrick, E. R., East, J. P., & Herman, G. L. (2010). Identifying student misconceptions of programming. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 107–111). Milwaukee, Wisconsin, USA: ACM. https://doi.org/http://doi.acm.org/10.1145/1734263.1734299

Kafai, Y. B. (2006). Constructionism. In R. K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (1 edition, pp. 35–46). Cambridge ; New York: Cambridge University Press.

Kagan, D. M. (1990). Ways of Evaluating Teacher Cognition: Inferences Concerning the Goldilocks Principle. *Review of Educational Research*, *60*(3), 419–469. https://doi.org/10.3102/00346543060003419

Katz, S., Allbritton, D., Aronis, J., Wilson, C., & Soffa, M. (2006). Gender, achievement, and persistence in an undergraduate computer science program. *ACM SIGMIS Database*, *37*, 57.

Kay, A. C. (1972). A Personal Computer for Children of All Ages. In *Proceedings of the ACM Annual Conference - Volume 1*. New York, NY, USA: ACM. https://doi.org/10.1145/800193.1971922

Kay, A. C., & Goldberg, A. (1977). Personal dynamic media. *IEEE Computer*, 31–41.

Kazemi, E., Lampert, M., & Ghousseini, H. (2007). *Conceptualizing and Using Routines of Practice in Mathematics Teaching to Advance Professional Education | Spencer*. Chicago, IL: Spencer Foundation. Retrieved from http://www.spencer.org/conceptualizing-and-using-routines-practice-mathematics-teaching-advance-professional-education

Kelleher, C., & Pausch, R. (2005). Lowering the Barriers to Programming: a taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys*, *37*, 83–1137.

Kick, R., & Trees, F. P. (2015). AP CS Principles: Engaging, Challenging, and Rewarding. *ACM Inroads*, *6*(1), 42–45. https://doi.org/10.1145/2710672

Kinnunen, P., & Malmi, L. (2006). Why students drop out CS1 course? In *Proceedings of the second international workshop on Computing education research* (pp. 97–108). New York, NY, USA: ACM. https://doi.org/10.1145/1151588.1151604

Kordaki, M. (2013). High school computing teachers' beliefs and practices: A case study. *Computers & Education*, *68*, 141–152. https://doi.org/10.1016/j.compedu.2013.04.020

Korotkin, A. L., Darby, Jr., C., & Romashko, T. (1970). *A Survey of Computing Activities in Secondary Schools. Final Report.* (No. AIR-852-10/70-FR).

Krippendorff, K. (2008). Systematic and Random Disagreement and the Reliability of Nominal Data. *Communication Methods and Measures*, *2*(4), 323–338. https://doi.org/10.1080/19312450802467134

Krippendorff, K. (2011). Computing Krippendorff's Alpha-Reliability. *Departmental Papers (ASC)*. Retrieved from http://repository.upenn.edu/asc_papers/43

Krippendorff, K. (2012). *Content Analysis: An Introduction to Its Methodology*. SAGE.

Lahtinen, E., Ala-Mutka, K., & Järvinen, H.-M. (2005). A Study of the Difficulties of Novice Programmers. In *Proceedings of the 10th Annual SIGCSE Conference on Innovation and*

Technology in Computer Science Education (pp. 14–18). New York, NY, USA: ACM. https://doi.org/10.1145/1067445.1067453

Lampert, M. (2010). Learning Teaching in, from, and for Practice: What Do We Mean? *Journal of Teacher Education*, *61*, 21–34. https://doi.org/10.1177/0022487109347321

Lang, K., Galanos, R., Goode, J., Seehorn, D., & Trees, F. (2013). *Bugs in the System: Computer Science Teacher Certification in the U.S.* Computer Science Teachers Association. Retrieved from http://csta.acm.org/ComputerScienceTeacherCertification/sub/CSTA_BugsInTheSystem.pdf

Lapidot, T. (2005). *Computer Science Teachers' Learning during their Everyday Work* (Doctoral Dissertation). Technion University, Israel.

Leake, M., & Lewis, C. (2016). Designing a New System for Sharing Computer Science Teaching Resources. In *Proceedings of the 19th ACM Conference on Computer Supported Cooperative Work and Social Computing Companion* (pp. 321–324). New York, NY, USA: ACM. https://doi.org/10.1145/2818052.2869109

Lee, C. D., Goldman, S. R., Levine, S., & Magliano, J. (2016). Epistemic cognition in literary reasoning. In J. A. Greene, W. A. Sandoval, & I. Bråten (Eds.), *Handbook of Epistemic Cognition* (pp. 165–183). New York, NY: Routledge.

Lee, C. D., Spencer, M. B., & Harpalani, V. (2003). "Every Shut Eye Ain't Sleep": Studying How People Live Culturally. *Educational Researcher*, *32*, 6–13.

Lewis, C., Jackson, M., & Waite, W. (2010). Student and faculty attitudes and beliefs about computer science. *Communications of the ACM*, *53*, 78–85.

Liberman, N., Kolikant, Y. B.-D., & Beeri, C. (2012). “Regressed Experts” as a New State in Teachers’ Professional Development: Lessons from Computer Science Teachers’ Adjustments to Substantial Changes in the Curriculum. *Computer Science Education, 22*(3), 257–283.

Lincoln, Y. S., & Guba, E. G. (1985). *Naturalistic Inquiry*. SAGE.

Loughran, J., Mulhall, P., & Berry, A. (2004). In search of pedagogical content knowledge in science: Developing ways of articulating and documenting professional practice. *Journal of Research in Science Teaching*, *41*(4), 370–391. https://doi.org/10.1002/tea.20007

Loyd, B. H., & Gressard, C. (1984). Reliability and Factorial Validity of Computer Attitude Scales. *Educational and Psychological Measurement*, *44*(2), 501–505. https://doi.org/10.1177/0013164484442033

Luft, J. A., & Roehrig, G. H. (2007). Capturing Science Teachers’ Epistemological Beliefs: The Development of the Teacher Beliefs Interview. *Electronic Journal of Science Education, 11*(2). Retrieved from http://ejse.southwestern.edu/article/view/7794

Magnusson, S., Krajcik, J., & Borko, H. (1999). Nature, Sources, and Development of Pedagogical Content Knowledge for Science Teaching. In J. Gess-Newsome & N. G. Lederman (Eds.), *Examining Pedagogical Content Knowledge* (pp. 95–132). Springer Netherlands. Retrieved from http://link.springer.com/chapter/10.1007/0-306-47217-1_4

Maloney, J., Resnick, M., Rusk, N., Silverman, B., & Eastmond, E. (2010). The Scratch Programming Language and Environment. *Trans. Comput. Educ.*, *10*, 1–15. https://doi.org/10.1145/1868358.1868363

Margaritis, M., Magenheim, J., Hubwieser, P., Berges, M., Ohrndorf, L., & Schubert, S. (2015). Development of a competency model for computer science teachers at secondary school level. In *2015 IEEE Global Engineering Education Conference (EDUCON)* (pp. 211–220). https://doi.org/10.1109/EDUCON.2015.7095973

Margolis, J., & Fisher, A. (2003). *Unlocking the Clubhouse: Women in Computing*. MIT Press.

Margolis, J., Goode, J., Chapman, G., & Ryoo, J. J. (2014). That Classroom "Magic." *Commun. ACM*, *57*(7), 31–33. https://doi.org/10.1145/2618107

Martinez, M. C., Gomez, M. J., Moresi, M., & Benotti, L. (2016). Lessons Learned on Computer Science Teachers Professional Development. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 77–82). New York, NY, USA: ACM. https://doi.org/10.1145/2899415.2899460

Matthews, M. E. (2013). The Influence of the Pedagogical Content Knowledge Framework on Research in Mathematics Education: A Review across Grade Bands. *Journal of Education*, *193*(3), 29–37.

McDonald, M., Kazemi, E., & Kavanagh, S. S. (2013). Core Practices and Pedagogies of Teacher Education: A Call for a Common Language and Collective Activity. *Journal of Teacher Education*, *64*(5), 378–386. https://doi.org/10.1177/0022487113493807

McDowell, C., Werner, L., Bullock, H. E., & Fernald, J. (2003). The impact of pair programming on student performance, perception and persistence.

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

Miles, M. B., & Huberman, A. M. (1994). *Qualitative Data Analysis: An Expanded Sourcebook*. Sage.

Miller, L. A. (1974). Programming by non-programmers. *International Journal of Man-Machine Studies*, *6*, 237–260.

Milne, I., & Rowe, G. (2002). Difficulties in Learning and Teaching Programming—Views of Students and Tutors. *Education and Information Technologies*, *7*(1), 55–66. https://doi.org/10.1023/A:1015362608943

Mizzi, D. (2013). The Challenges Faced by Science Teachers When Teaching Outside Their Specific Science Specialism. *Acta Didactica Napocensia*, *6*(4), 1–6.

Morgan, D. L. (2007). Paradigms Lost and Pragmatism Regained: Methodological Implications of Combining Qualitative and Quantitative Methods. *Journal of Mixed Methods Research*, *1*(1), 48–76. https://doi.org/10.1177/2345678906292462

Morrison, B. B., Ni, L., & Guzdial, M. (2012). Adapting the disciplinary commons model for high school teachers: improving recruitment, creating community. In *Proceedings of the ninth annual international conference on International computing education research* (pp. 47–54). New York, NY, USA: ACM. https://doi.org/10.1145/2361276.2361287

National Research Council. (2011). *Report of a Workshop of Pedagogical Aspects of Computational Thinking*. Washington, D.C.: The National Academies Press. Retrieved from http://www.nap.edu/catalog.php?record_id=13170

Nelson, T. H. (2005). Knowledge Interactions in Teacher-Scientist Partnerships: Negotiation, Consultation, and Rejection. *Journal of Teacher Education*, *56*(4), 382–395. https://doi.org/10.1177/0022487105279938

Ni, L. (2009). What Makes CS Teachers Change?: Factors Influencing CS Teachers' Adoption of Curriculum Innovations. In *Proceedings of the 40th ACM Technical Symposium on Computer Science Education* (pp. 544–548). New York, NY, USA: ACM. https://doi.org/10.1145/1508865.1509051

Ni, L., & Guzdial, M. (2012). Who AM I?: Understanding High School Computer Science Teachers' Professional Identity. In *Proceedings of the 43rd ACM Technical Symposium on Computer Science Education* (pp. 499–504). New York, NY, USA: ACM. https://doi.org/10.1145/2157136.2157283

Niess, M., Suharwoto, G., Lee, K., & Sadri, P. (2006). Guiding inservice mathematics teachers in developing technology pedagogical content knowledge (TPCK). In *Society for Information Technology and Teacher Education Annual Conference* (pp. 20–24).

Ohrndorf, L., & Schubert, S. (2013). Measurement of Pedagogical Content Knowledge: Students' Knowledge and Conceptions. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (pp. 104–107). New York, NY, USA: ACM. https://doi.org/10.1145/2532748.2532758

Ohrndorf, L., & Schubert, S. (2014). Students' Cognition: Outcomes from an Initial Study with Student Teachers. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education* (pp. 112–115). New York, NY, USA: ACM. https://doi.org/10.1145/2670757.2670758

Oliver, D. G., Serovich, J. M., & Mason, T. L. (2005). Constraints and Opportunities with Interview Transcription: Towards Reflection in Qualitative Research. *Social Forces, 84*(2), 1273–1289. https://doi.org/10.1353/sof.2006.0023

Opfer, V. D., & Pedder, D. (2011). Conceptualizing Teacher Professional Learning. *Review of Educational Research*, *81*(3), 376–407.

Orton, K., Weintrop, D., Beheshti, E., Horn, M. S., Jona, K., & Wilensky, U. (2016). Bringing Computational Thinking into High School Mathematics and Science Classrooms. In *Proocedings of the International Conference of the Learning Sciences (ICLS) 2016.* Singapore.

Papert, S. (1972). Teaching Children Thinking. *Innovations in Education & Training International*, *9*(5), 245–255. https://doi.org/10.1080/1355800720090503

Papert, S., & Harel, I. (1991). Situating Constructionism. In *Constructionism*. Ablex Publishing Corporation. Retrieved from http://www.papert.org/articles/SituatingConstructionism.html

Pardini, P. (2006). In One Voice: Mainstream and ELL Teachers Work Side-by-Side in the Classroom, Teaching Language through Content. *Journal of Staff Development*, *27*(4), 20–25.

Pargas, R. P., & Shah, D. M. (2006). Things are clicking in computer science courses. *SIGCSE Bull.*, *38*(1), 474–478. https://doi.org/10.1145/1124706.1121489

Park, K., Hurt, A., Fisher, T., & Rost, L. C. (2016, April 20). Map: How Per-Pupil Spending Compares Across U.S. School Districts - Education Week. *Education Week*. Retrieved from http://www.edweek.org/ew/section/multimedia/map-how-per-pupil-spending-compares-across-us.html

Park, S., & Oliver, J. S. (2008). Revisiting the conceptualisation of pedagogical content knowledge (PCK): PCK as a conceptual tool to understand teachers as professionals. *Research in Science Education, 38*(3), 261–284.

Pea, R. (1986). Language-independent conceptual “bugs” in novice programming. *Journal Educational Computing Research*, *2*, 25–36.

Perl, M., Maughmer, B., & McQueen, C. (1999). Co-teaching: A different approach for cooperating teachers and student teachers. In *Association of Teacher Educators Annual Conference, Chicago, IL*.

Petersen, A., Craig, M., Campbell, J., & Tafliovich, A. (2016). Revisiting Why Students Drop CS1. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (pp. 71–80). New York, NY, USA: ACM. https://doi.org/10.1145/2999541.2999552

Peterson, P. L., Fennema, E., Carpenter, T. P., & Loef, M. (1989). Teacher’s Pedagogical Content Beliefs in Mathematics. *Cognition and Instruction*, *6*(1), 1–40. https://doi.org/10.1207/s1532690xci0601_1

Petre, M., & Price, B. (2004). Using Robotics to Motivate “Back Door” Learning. *Education and Information Technologies*, *9*(2), 147–158. https://doi.org/10.1023/B:EAIT.0000027927.78380.60

Pinkard, N., Erete, S., Martin, C. K., & Royston, M. M. de. (2017). Digital Youth Divas: Exploring Narrative-Driven Curriculum to Spark Middle School Girls’ Interest in Computational Activities. *Journal of the Learning Sciences, 26*. https://doi.org/10.1080/10508406.2017.1307199

Poirot, J. L. (1979). Computer Education in the Secondary School: Problems and Solutions. In *Proceedings of the Tenth SIGCSE Technical Symposium on Computer Science Education* (pp. 101–104). New York, NY, USA: ACM. https://doi.org/10.1145/800126.809562

Poplin, M. S., Drew, D. E., & Gable, R. S. (1984). *CALIP, Computer Aptitude, Literacy, and Interest Profile*. Pro-Ed.

Postholm, M. B. (2012). Teachers' Professional Development: A Theoretical Review. *Educational Research*, *54*(4), 405–429.

Powell-Moman, A. D., & Brown-Schild, V. B. (2011). The Influence of a Two-Year Professional Development Institute on Teacher Self-Efficacy and Use of Inquiry-Based Instruction. *Science Educator*, *20*(2), 47–53.

Powers, K., Ecott, S., & Hirshfield, L. M. (2007). Through the Looking Glass: Teaching CS0 with Alice. In *Proceedings of the 38th SIGCSE Technical Symposium on Computer Science Education* (pp. 213–217). New York, NY, USA: ACM. https://doi.org/10.1145/1227310.1227386

Professional Development. (n.d.). Retrieved December 15, 2015, from https://code.org/educate/professional-development

Ragonis, N. (2009). Computing Pre-University: Secondary Level Computing Curricula. In *Wiley Encyclopedia of Computer Science and Engineering*. John Wiley & Sons, Inc. https://doi.org/10.1002/9780470050118.ecse974

Ragonis, N., & Hazzan, O. (2009). Integrating a Tutoring Model into the Training of Prospective Computer Science Teachers. *Journal of Computers in Mathematics and Science Teaching*, *28*(3), 309–339.

Ragonis, N., & Shilo, G. (2013). What is It We Are Asking: Interpreting Problem-solving Questions in Computer Science and Linguistics. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (pp. 189–194). New York, NY, USA: ACM. https://doi.org/10.1145/2445196.2445253

Randi, J., & Zeichner, K. M. (2004). New Visions of Teacher Professional Development. *Yearbook of the National Society for the Study of Education*, *103*(1), 180–227.

Randolph, J., Julnes, G., Sutinen, E., & Lehman, S. (2008). A methodological review of computer science education research. *Journal of Information Technology Education*, *7*, 135–162.

Rist, R. S. (2004). Learning to Program: Schema Creation, Application, and Evaluation. In *Computer Science Education Research* (pp. 175–197). Taylor and Francis.

Roberts, E. (2004). The Dream of a Common Language: The Search for Simplicity and Stability in Computer Science Education. In *Proceedings of the 35th SIGCSE Technical Symposium on Computer Science Education* (pp. 115–119). New York, NY, USA: ACM. https://doi.org/10.1145/971300.971343

Robertson, A. D., Atkins, L. J., & Levin, D. M. (2015). What is Responsive Teaching? In A. D. Robertson, R. E. Scherr, & D. Hammer (Eds.), *Responsive Teaching in Science and Mathematics* (pp. 1–35). New York, NY: Routledge.

Robins, A., Rountree, J., & Rountree, N. (2003). Learning and teaching programming: A review and discussion. *Computer Science Education*, *13*, 137–172.

Rodgers, C. R. (2002). Seeing Student Learning: Teacher Change and the Role of Reflection. Voices inside Schools. *Harvard Educational Review*, *72*(2), 230–53.

Rodriguez, B., Rader, C., & Camp, T. (2016). Using Student Performance to Assess CS Unplugged Activities in a Classroom Environment. In *Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 95–100). New York, NY, USA: ACM. https://doi.org/10.1145/2899415.2899465

Ross, J. A., Cousins, J. B., Gadalla, T., & Hannay, L. (1999). Administrative Assignment of Teachers in Restructuring Secondary Schools: The Effect of Out-of-Field Course Responsibility on Teacher Efficacy. *Educational Administration Quarterly*, *35*(5), 782–805. https://doi.org/10.1177/00131619921968824

Ryoo, J., Goode, J., & Margolis, J. (2015). It takes a village: supporting inquiry- and equity-oriented computer science pedagogy through a professional learning community. *Computer Science Education*, *25*(4), 351–370.

Saeli, M., Perrenet, J., Jochems, W. M., & Zwaneveld, B. (2012). Pedagogical Content Knowledge in Teaching Material. *Journal of Educational Computing Research*, *46*(3), 267–293.

Saeli, M., Perrenet, J., Jochems, W., & Zwaneveld, B. (2010). *Portraying the pedagogical content knowledge of programming—The technical report*. Retrieved from https://www.tue.nl/fileadmin/content/universiteit/Over_de_universiteit/Eindhoven_School_of_Education/Onderzoek/Projecten_promovendi/Mara_Saeli_SPJZ_Technical_Report.pdf

Salish Research Consortium. (1997, June). Secondary Science and Mathematics Teacher Preparation Programs: Influences on New Teachers and Their Students. Instrument Package & User's Guide. Retrieved from http://archive.org/details/ERIC_ED463974

San Francisco Unified School District. (2015). Board Approves Plans to Expand Computer Science Curriculum to All Grades [Press Release]. Retrieved from http://www.sfusd.edu/en/news/current-news/2015-news-archive/06/board-approves-plans-to-expand-computer-science-curriculum-to-all-grades.html

Schanzer, E., Fisler, K., Krishnamurthi, S., & Felleisen, M. (2015). Transferring Skills at Solving Word Problems from Computing to Algebra Through Bootstrap. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (pp. 616–621). New York, NY, USA: ACM. https://doi.org/10.1145/2676723.2677238

Schmidt, D. A., Baran, E., Thompson, A. D., Mishra, P., Koehler, M. J., & Shin, T. S. (2009). Technological Pedagogical Content Knowledge (TPACK). *Journal of Research on Technology in Education*, *42*(2), 123–149. https://doi.org/10.1080/15391523.2009.10782544

Schneider, R. M., & Plasman, K. (2011). Science Teacher Learning Progressions: A Review of Science Teachers' Pedagogical Content Knowledge Development. *Review of Educational Research*, 0034654311423382. https://doi.org/10.3102/0034654311423382

Schon, D. A. (1983). *The Reflective Practitioner: How Professionals Think In Action*. Basic Books.

Schulte, C. (2008). Block Model: An Educational Model of Program Comprehension As a Tool for a Scholarly Approach to Teaching. In *Proceedings of the Fourth International Workshop on Computing Education Research* (pp. 149–160). New York, NY, USA: ACM. https://doi.org/10.1145/1404520.1404535

Schulte, C., & Bennedsen, J. (2006). What Do Teachers Teach in Introductory Programming? In *Proceedings of the Second International Workshop on Computing Education Research* (pp. 17–28). New York, NY, USA: ACM. https://doi.org/10.1145/1151588.1151593

Schwandt, T. A. (2007). *The SAGE Dictionary of Qualitative Inquiry. Third Edition*. SAGE Publications.

Scott, K. A., Clark, K., Hayes, E., Mruczek, C., & Sheridan, K. (2010). Culturally Relevant Computing Programs: Two examples to Inform Teacher Professional Development (Vol. 2010, pp. 1269–1277). Presented at the Society for Information Technology & Teacher Education International Conference. Retrieved from /p/33532/

Scott, K. A., & White, M. A. (2013). COMPUGIRLS' Standpoint: Culturally Responsive Computing and Its Effect on Girls of Color. *Urban Education*, *48*(5), 657–681. https://doi.org/10.1177/0042085913491219

Scruggs, T. E., Mastropieri, M. A., & McDuffie, K. A. (2007). Co-Teaching in Inclusive Classrooms: A Metasynthesis of Qualitative Research. *Exceptional Children*, *73*(4), 392–416. https://doi.org/10.1177/001440290707300401

Seago, N. (2003). Using video as an object of inquiry for mathematics teaching and learning. In *Using Video in Teacher Education* (Vol. 10, pp. 259–286). Emerald Group Publishing Limited. https://doi.org/10.1016/S1479-3687(03)10010-7

Searle, K. A., & Kafai, Y. B. (2015). Boys' Needlework: Understanding Gendered and Indigenous Perspectives on Computing and Crafting with Electronic Textiles. In *Proceedings of the Eleventh Annual International Conference on International*

Computing Education Research (pp. 31–39). New York, NY, USA: ACM. https://doi.org/10.1145/2787622.2787724

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., … Verno, A. (2011). CSTA K-12 Computer Science Standards. Retrieved from http://csta.acm.org/Curriculum/sub/CurrFiles/CSTA_K-12_CSS.pdf

Seehorn, D., Pirmann, T., Lash, T., Batista, L., Ryder, D., Sedwick, V., … Uche, C. (2016). [Interim] CSTA K-12 Computer Science Standards. Retrieved from http://www.csteachers.org/resource/resmgr/Docs/Standards/2016StandardsRevision/INTERIM_StandardsFINAL_07222.pdf

Sengupta, P., & Wilensky, U. (2009). Learning Electricity with NIELS: Thinking with Electrons and Thinking in Levels. *International Journal of Computers for Mathematical Learning*, *14*(1), 21–50. https://doi.org/10.1007/s10758-009-9144-z

Shaw, A. C. (1995). *Social construction and the inner city : design environments for social development and urban renewal* (Thesis). Massachusetts Institute of Technology. Retrieved from http://dspace.mit.edu/handle/1721.1/29095

Shulman, L. (1986). Those who understand: Knowledge growth in teaching. *Educational Researcher*, *15*, 4–14.

Shulman, L. (1987). Knowledge and Teaching: Foundations of the New Reform. *Harvard Educational Review*, *57*(1), 1–23. https://doi.org/10.17763/haer.57.1.j463w79r56455411

Silverstein, S. C., Dubner, J., Miller, J., Glied, S., & Loike, J. D. (2009). Teachers' Participation in Research Programs Improves Their Students' Achievement in Science. *Science*, *326*(5951), 440–442. https://doi.org/10.1126/science.1177344

Simmons, P. E., Emory, A., Carter, T., Coker, T., Finnegan, B., Crockett, D., … Labuda, K. (1999). Beginning Teachers: Beliefs and Classroom Actions. *Journal of Research in Science Teaching*, *36*(8), 930–954. https://doi.org/10.1002/(SICI)1098-2736(199910)36:8<930::AID-TEA3>3.0.CO;2-N

Simon, B., Kohanfars, M., Lee, J., Tamayo, K., & Cutts, Q. (2010). Experience report: peer instruction in introductory computing. In *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 341–345). New York, NY, USA: ACM. https://doi.org/10.1145/1734263.1734381

Smith, J. P. (1996). Efficacy and Teaching Mathematics by Telling: A Challenge for Reform. *Journal for Research in Mathematics Education*, *27*(4), 387–402. https://doi.org/10.2307/749874

Snelbecker, G. E., & Bhote, N. P. (1995). Elementary Versus Secondary School Teachers Retraining to Teach Computer Science. *Journal of Research on Computing in Education, 27*(3), 336.

Solomon, C. (1986). *Computer Environments for Children: A Reflection on Theories of Learning and Education*. MIT Press.

Sorva, J. (2012). *Visual Programming Simulation in Introductory Programming Education*. Aalto University.

Spillane, J. P. (2005). Primary school leadership practice: how the subject matters. *School Leadership & Management*, *25*(4), 383–397. https://doi.org/10.1080/13634230500197231

Spohrer, J. G., & Soloway, E. (1986). Analyzing the high frequency bugs in novice programs. In *Papers presented at the first workshop on empirical studies of programmers* (pp. 230–251). Norwood, NJ, USA: Ablex Publishing Corp. Retrieved from http://dl.acm.org/citation.cfm?id=21842.28897

Stake, R. E. (2000). Case Studies. In N. K. Denzin & Y. S. Lincoln (Eds.), *Handbook of Qualitative Research* (2 edition, pp. 435–454). Thousand Oaks, Calif: Sage Publications, Inc.

Stanton, J., Goldsmith, L., Adrion, W. R., Dunton, S., Hendrickson, K., Peterfreund, A., … Zinth, J. D. (2017). *State of the States Landscape Report: State-Level Policies Supporting Equitable K-12 Computer Science Education*. BYN Mellon.

STEM + Computing Partnerships (STEM+C). (2016, December 30). National Science Foundation. Retrieved from https://nsf.gov/pubs/2017/nsf17535/nsf17535.htm

Stephenson, C. (2000). *A report on high school computer science education in five US states* (p. 2003). Retrieved from http://www.holtsoft.com/chris/HSSurveyArt.pdf

Stigler, J. W., & Hiebert, J. (1999). *The Teaching Gap: Best Ideas from the World's Teachers for Improving Education in the Classroom*. New York: Free Press.

Strohecker, C. (1991). Elucidating styles of thinking about topology through thinking about knots. In *Constructionism*. Norwood N.J.: Ablex Publishing Corp.

Swackhamer, L. E., Koellner, K., Basile, C., & Kimbrough, D. (2009). Increasing the Self-Efficacy of Inservice Teachers through Content Knowledge. *Teacher Education Quarterly*, *36*(2), 63–78.

Taub, R., Ben-Ari, M., & Armoni, M. (2009). The Effect of CS Unplugged on Middle-school Students' Views of CS. In *Proceedings of the 14th Annual ACM SIGCSE Conference on Innovation and Technology in Computer Science Education* (pp. 99–103). New York, NY, USA: ACM. https://doi.org/10.1145/1562877.1562912

Tedre, M., & Sutinen, E. (2008). Three traditions of computing: what educators should know. *Computer Science Education*, *18*(3), 153–170. https://doi.org/10.1080/08993400802332332

The City of New York. (2016). Equity and Excellence: Mayor de Blasio Announces Reforms to Raise Achievement Across all Public Schools [Press Release]. Retrieved from http://www1.nyc.gov/office-of-the-mayor/news/618-15/equity-excellence-mayor-de-blasio-reforms-raise-achievement-across-all-public

The College Board. (2014). Computer Science A: Course Description. Retrieved from https://secure-media.collegeboard.org/digitalServices/pdf/ap/ap-computer-science-a-course-description.pdf

The College Board. (2015). *AP Program Participation and Performance Data 2015: Program Summary Report*. Retrieved from https://secure-media.collegeboard.org/digitalServices/pdf/research/2015/Program-Summary-Report-2015.pdf

The Joint Task Force for Computing Curricula 2005. (2005). *Computing Curricula 2005: The Overview Report*. New York, NY, USA: ACM.

The White House. (2016). FACT SHEET: President Obama Announces Computer Science For All Initiative [Press Release]. Retrieved from https://www.whitehouse.gov/the-press-office/2016/01/30/fact-sheet-president-obama-announces-computer-science-all-initiative-0

Tschannen-Moran, M., & Hoy, A. W. (2001). Teacher efficacy: capturing an elusive construct. *Teaching and Teacher Education*, *17*(7), 783–805. https://doi.org/10.1016/S0742-051X(01)00036-1

Tschannen-Moran, M., Hoy, A. W., & Hoy, W. K. (1998). Teacher Efficacy: Its Meaning and Measure. *Review of Educational Research*, *68*(2), 202–248. https://doi.org/10.2307/1170754

Tucker, A., Deek, F., Jones, J., McCowan, D., Stephenson, C., & Verno, A. (2006). *A Model Curriculum for K–12 Computer Science: Final Report of the ACM K–12 Task Force Curriculum Committee Second Edition*. Computer Science Teachers Association.

Turkle, S., & Papert, S. (1992). Epistemological Pluralism and the Revaluation of the Concrete. *Journal of Mathematical Behavior*, *11*, 3–33.

U.S. Census Bureau. (2015, August 5). State and County QuickFacts [Data file]. Retrieved September 2, 2015, from http://quickfacts.census.gov/

van Driel, J. H., Verloop, N., & de Vos, W. (1998). Developing science teachers' pedagogical content knowledge. *Journal of Research in Science Teaching*, *35*(6), 673–695. https://doi.org/10.1002/(SICI)1098-2736(199808)35:6<673::AID-TEA5>3.0.CO;2-J

van Es, E. A., & Sherin, M. G. (2008). Mathematics teachers' "learning to notice" in the context of a video club. *Teaching and Teacher Education*, *24*(2), 244–276. https://doi.org/10.1016/j.tate.2006.11.005

van Es, E. A., Stockero, S. L., Sherin, M. G., Zoest, L. R. V., & Dyer, E. (2015). Making the Most of Teacher Self-Captured Video. *Mathematics Teacher Educator*, *4*(1), 6–19. https://doi.org/10.5951/mathteaceduc.4.1.0006

van Veen, K., Sleegers, P., Bergen, T., & Klaassen, C. (2001). Professional orientations of secondary school teachers towards their work. *Teaching and Teacher Education*, *17*(2), 175–194. https://doi.org/10.1016/S0742-051X(00)00050-0

Vermunt, J. D., & Endedijk, M. D. (2011). Patterns in Teacher Learning in Different Phases of the Professional Career. *Learning and Individual Differences*, *21*(3), 294–302.

Vihavainen, A., Airaksinen, J., & Watson, C. (2014). A Systematic Review of Approaches for Teaching Introductory Programming and Their Influence on Success. In *Proceedings of the Tenth Annual Conference on International Computing Education Research* (pp. 19–26). New York, NY, USA: ACM. https://doi.org/10.1145/2632320.2632349

Wayne, A. J., & Youngs, P. (2003). Teacher Characteristics and Student Achievement Gains: A Review. *Review of Educational Research*, *73*(1), 89–122. https://doi.org/10.3102/00346543073001089

Weinstein, M. (2006). TAMS Analyzer: Anthropology as Cultural Critique in a Digital Age. *Social Science Computer Review*, *24*(1), 68–77. https://doi.org/10.1177/0894439305281496

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining Computational Thinking for Mathematics and Science Classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147. https://doi.org/10.1007/s10956-015-9581-5

Weintrop, D., & Wilensky, U. (2015). To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-based Programming. In *Proceedings of the 14th International Conference on Interaction Design and Children* (pp. 199–208). New York, NY, USA: ACM. https://doi.org/10.1145/2771839.2771860

Weiss, I. R., Pasley, J. D., Smith, P. S., Banilower, E. R., & Heck, D. J. (2003). *Looking Inside the Classroom: A Study of K-12 Mathematics and Science Education in the United States*.

Where computer science counts. (n.d.). Retrieved December 21, 2015, from https://code.org/action

Whitworth, B., & Chiu, J. (2015). Professional Development and Teacher Change: The Missing Leadership Link. *Journal of Science Teacher Education*, *26*(2), 121–137.

Wilensky, U. (1995). Paradox, programming, and learning probability: A case study in a connected mathematics framework. *The Journal of Mathematical Behavior*, *14*(2), 253–280. https://doi.org/10.1016/0732-3123(95)90010-1

Wilensky, U. (1997a). *NetLogo Wolf Sheep Predation model*. Northwestern University, Evanston, IL.: Center for Connected Learning and Computer-Based Modeling. Retrieved from http://ccl.northwestern.edu/netlogo/models/WolfSheepPredation

Wilensky, U. (1997b). What is Normal Anyway? Therapy for Epistemological Anxiety. *Educational Studies in Mathematics*, *33*(2), 171–202. https://doi.org/10.1023/A:1002935313957

Wilensky, U. (1999a). GasLab—an Extensible Modeling Toolkit for Connecting Micro-and Macro-properties of Gases. In W. Feurzeig & N. Roberts (Eds.), *Modeling and*

Simulation in Science and Mathematics Education (pp. 151–178). Springer New York. https://doi.org/10.1007/978-1-4612-1414-4_7

Wilensky, U. (1999b). *NetLogo*. Northwestern University, Evanston, IL: Center for Connected Learning and Computer-Based Modeling. Retrieved from http://ccl.northwestern.edu/netlogo

Wilensky, U., Brady, C. E., & Horn, M. S. (2014). Fostering Computational Literacy in Science Classrooms. *Commun. ACM*, *57*(8), 24–28. https://doi.org/10.1145/2633031

Wilkerson-Jerde, M. H., & Wilensky, U. J. (2015). Patterns, Probabilities, and People: Making Sense of Quantitative Change in Complex Systems. *Journal of the Learning Sciences*, *24*(2), 204–251. https://doi.org/10.1080/10508406.2014.976647

Williams, L. (2007). Lessons Learned from Seven Years of Pair Programming at North Carolina State University. *SIGCSE Bull.*, *39*(4), 79–83. https://doi.org/10.1145/1345375.1345420

Williams, L. A., & Kessler, R. R. (2001). Experiments with Industry's "Pair-Programming" Model in the Computer Science Classroom. *Computer Science Education*, *11*(1), 7–20. https://doi.org/10.1076/csed.11.1.7.3846

Wilson, S. M., Rozelle, J. J., & Mikeska, J. N. (2011). Cacophony or Embarrassment of Riches: Building a System of Support for Quality Teaching. *Journal of Teacher Education*, *62*(4), 383–394.

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

Wing, J. (2008). Computational thinking and thinking about computing. *Philosophical Transactions A*, *366*, 3717.

Wong, S. S., & Luft, J. A. (2015). Secondary Science Teachers' Beliefs and Persistence: A Longitudinal Mixed-Methods Study. *Journal of Science Teacher Education*, *26*(7), 619–645. https://doi.org/10.1007/s10972-015-9441-4

Woollard, J. (2005). The Implications of the Pedagogic Metaphor for Teacher Education in Computing. *Technology, Pedagogy and Education*, *14*(2), 189–204.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational Thinking in Elementary and Secondary Teacher Education. *Trans. Comput. Educ.*, *14*(1), 5:1–5:16. https://doi.org/10.1145/2576872

Yin, R. (2013). *Case Study Research: Design and Methods* (Fifth Edition edition). Los Angeles: SAGE Publications, Inc.

Zendler, A., & Hubwieser, P. (2013). The influence of (research-based) teacher training programs on evaluations of central computer science concepts. *Teaching & Teacher Education*, *34*, 130–142.

Zendler, A., & Klaudt, D. (2012). Central Computer Science Concepts to Research-Based Teacher Training in Computer Science: An Experimental Study. *Journal of Educational Computing Research*, *46*(2), 153–172. https://doi.org/10.2190/EC.46.2.c

Zendler, A., McClung, O. W., & Klaudt, D. (2015). A Cross-Cultural Comparison of Concepts in Computer Science Education. *International Journal of Information & Learning Technology*, *32*(4), 235–256.

APPENDICES

APPENDIX A

Pre-lesson Questionnaire

Thank you for completing this pre-observation questionnaire. Your responses will help me better understand your goals and expectations for the upcoming lesson I will observe. It should take no more than 30 minutes to complete.

1. Date of lesson to be observed

	MM		DD		YYYY
Date:		/		/	

2. What is your role on your TEALS teaching team?

- ☐ High school teacher
- ☐ Volunteer

3. What unit(s) will your class be working on during the observation?

☐ Unit 1: Intro to Java and Static Methods	☐ Unit 0: Beginnings
☐ Unit 2: For Loops and Data Types	☐ Unit 1: SNAP Basics
☐ Unit 3: Conditionals, While Loops, String Parsing	☐ Unit 2: Loops
☐ Unit 4: Arrays and Arraylists	☐ Unit 3: Variables and Customization
☐ Unit 5: Objects	☐ Unit 4: Lists
☐ Unit 6: Inheritance, Polymorphism, Interfaces	☐ Unit 5: Cloning
☐ Unit 7: Searching and Sorting	☐ Unit 6: Final Project
☐ Unit 8: Recursion	☐ Culture Day
☐ Unit 9: AP Review	
☐ Unit 10: After the AP Exam	
☐ Other (please describe)	

4. How comfortable are you with the unit(s)?

- ☐ Completely comfortable
- ☐ Somewhat comfortable
- ☐ Not at all comfortable

5. How well prepared do you feel to guide student learning of this content?

- ☐ Completely prepared

- ☐ Somewhat prepared
- ☐ Not at all prepared

6. What is learning objective of this specific lesson?

7. Why is it important for students to know this?

8. How difficult is this learning objective compared to other topics covered in the course?

   - ☐ Less difficult
   - ☐ About the same difficulty
   - ☐ More difficult

9. Where does this lesson fit in the sequence of the unit you are working on? What have the students experienced prior to the lesson? What will they learn after the lesson?

10. What else do you know about this idea that you do not intend students to know yet?

11. What are the difficulties or limitations connected with teaching this topic?

12. What do you know about students' thinking that will influence your teaching of this topic?

13. What other factors will influence your teaching of this topic?

14. What teaching procedures will (would) you use to engage with this topic and why?

15. What technology will (would) you use to engage with this topic and why? (e.g., programming languages, programming environments, visualizations)

16. How will you know if this lesson is a success?

17. Is there anything else you would like to share about the lesson I will observe?

APPENDIX B

Post-lesson Questionnaire

Thank you for allowing me to observe your TEALS class this week. This questionnaire asks you to reflect on the lesson I observed and should take no more than 15 minutes to complete.

1. Date of lesson to be observed

	MM		DD		YYYY
Date:	☐	/	☐	/	☐

2. What is your role on your TEALS teaching team?

☐ High school teacher
☐ Volunteer teacher

3. What was the main topic of this lesson?

4. How did you prepare to teach this topic?

5. What additional preparation do you need to teach this topic again?

6. What advice would you offer someone teaching this topic for the first time?

7. What resources were used to plan this lesson?

8. How did these resources support instruction and learning?

9. How did these resources hinder instruction and learning?

10. Will you plan another lesson to revisit the topic(s) covered today?

☐ Yes
☐ No

Please explain why or why not:

11. Who on your TEALS team contributed to the following aspects of today’s lesson?

	Mostly volunteer(s)	Both volunteer(s) and teacher	Mostly teacher	No one
Developing the lesson	☐	☐	☐	☐
Delivering the lesson	☐	☐	☐	☐
Managing the classroom	☐	☐	☐	☐
Assisting students	☐	☐	☐	☐
Creating assignments/assessments	☐	☐	☐	☐
Grading student work	☐	☐	☐	☐

12. How did the co-teaching model used during today’s lesson support and/or hinder instruction and learning?

13. How did the co-teaching model used during today’s lesson support your development as a computer science high school teacher?

14. Is there anything else you would like to share about the lesson I observed?

APPENDIX C

Lesson Reflection Interview

1. How do you feel about how the lesson played out?

   Ask the following questions if not mentioned by the teacher:
   a. What aspects of the lesson supported student learning? student engagement?
   b. What did you expect students would take away from the lesson? Did they?
   c. What problems arose with instruction? with learning? How were the problems addressed?
   d. What would you revise?
   e. Did anything surprise you about how students responded to the lesson?
   f. Would you explain or present material differently than the volunteers did?

2. Can you describe the co-teaching model your team used today?

   Ask the following questions if not mentioned by the teacher:

   a. How well did the co-teaching model work?
   b. What role did you take? What role did your volunteer take?
   c. Did the co-teaching model prepare you to teach this lesson in the future? How?

APPENDIX D

Think-aloud Interviews: Assessment Prompts

Interview script*:*

- Imagine another computer science teacher has approached you for feedback on these test items she has put together *[lay out 3 items]*. She plans to give her students the test to see if they have some of the common difficulties with *[insert topic of interview task]*.
- For each assessment item, describe what difficulties students might have in answering it and if you would advise the teacher to include the item on her test.
- *[After the teacher finishes the task]* How might your students respond to these assessment items?

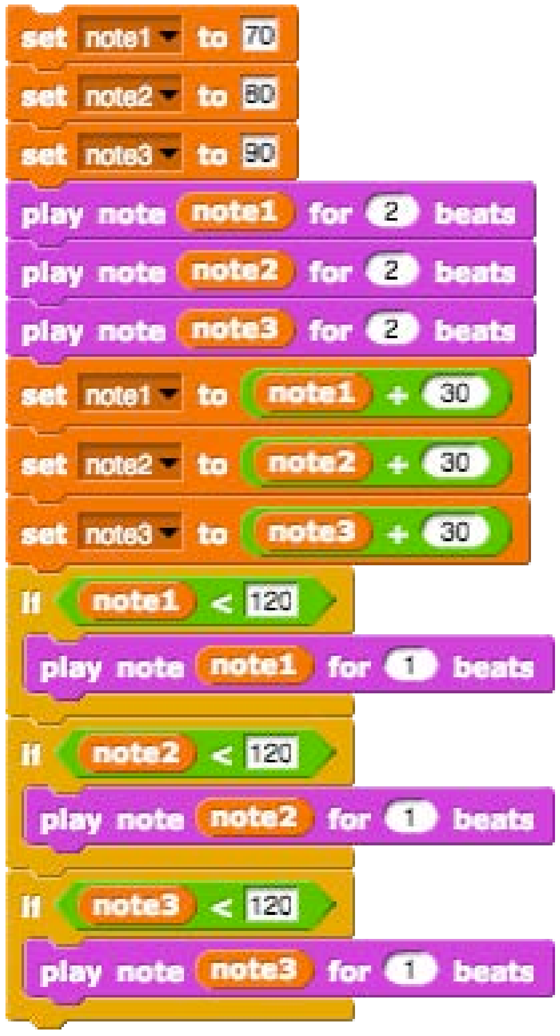
Arrays/lists (AP)

Item 1	Rewrite the following code to use arrays instead of ArrayLists:
	`public static void main(String[] args) {`
	
	`        ArrayList<Integer> scores = new ArrayList<>();`
	`        scores.add(50);`
	`        scores.add(72);`
	`        scores.add(34);`
	`        for(int i=0; i < scores.size(); i++){`
	`                System.out.println(scores.get(i));`
	`        }`
	`        boolean has50 = scores.contains(50);`
	`        System.out.println("Does the list have a score of 50?");`
	`        System.out.println(has50);`
	`}`	

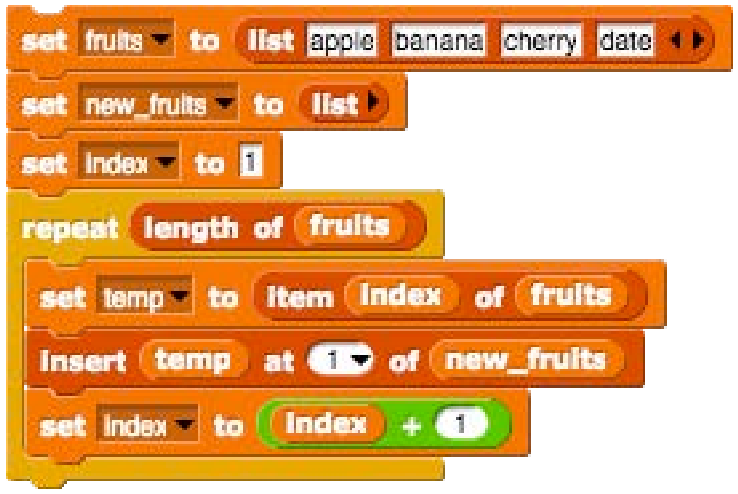
Item 2	Create a tracing table to show the value of fruits, temp, and i after each line of the for loop is called:
	`public static void main(String args[]){`
	`        String[] fruits = {"apple", "banana", "cherry", "date"};`
	`        int len = fruits.length;`
	
	`        for(int i=0; i < len/2; i++){`
	`                String temp = fruits[i];`
	`                fruits[i] = fruits[len-i-1];`
	`                fruits[len-i-1] = temp;`
	`        }`
	`}`	

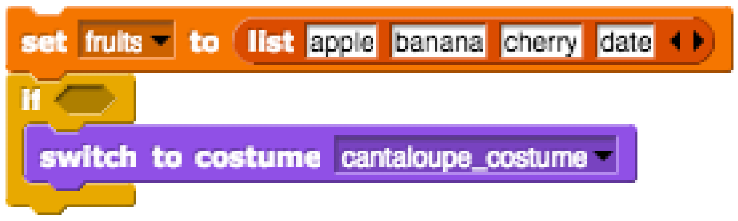
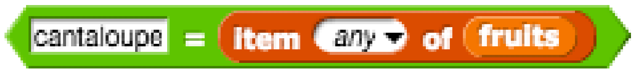
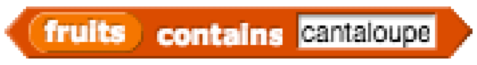
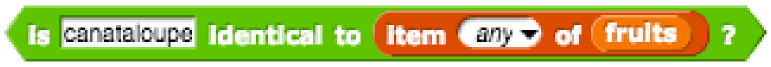
Item 3	`public static void main(String[] args) {`
	`    int[] arrayA = {10, 20, 30};`
	`    int[] arrayB = new int[3];`
	`    int[] arrayC = arrayA;`
	
	`    arrayA[0] = 3;`
	
	`    System.out.println(arrayC[0]);`
	
	`    arrayC = arrayB;`
	`    arrayB[0] = 9;`
	`    System.out.println(arrayC[0]);`
	`}`
		What does the code print?
		a) 3
		0
		b) 3
		9
		c) 3
		null
		d) 10
		0
		e) 10
		9
		f) 10
		null

Arrays/lists (Intro)

Item 1	Rewrite the following script using one list to represent the notes: 

```
set note1 to 70
set note2 to 80
set note3 to 90
play note (note1) for (2) beats
play note (note2) for (2) beats
play note (note3) for (2) beats
set note1 to ((note1) + (30))
set note2 to ((note2) + (30))
set note3 to ((note3) + (30))
if <(note1) < 120>
    play note (note1) for (1) beats
if <(note2) < 120>
    play note (note2) for (1) beats
if <(note3) < 120>
    play note (note3) for (1) beats
```

Item 2	Create a tracing table to show the value of fruits, new_fruits, and index after each block is called in the following script:
	
	`set fruits to list apple banana cherry date`
	`set new_fruits to list`
	`set index to 1`
	`repeat length of fruits`
	`    set temp to item index of fruits`
	`    insert temp at 1 of new_fruits`
	`    set index to index + 1`

Item 3	Your friend is trying to write a script that checks if the word 'cantaloupe' exists in a list of fruits, and, if so, changes the sprite's costume.
	
	`set fruits to list apple banana cherry date`
	`if ⬡`
	`    switch to costume cantaloupe_costume`
	
	Which of the following blocks could your friend use in the if block to make the script run correctly? Select all the blocks that could work.
	
	d)  `cantaloupe = item any of fruits`
	e)  `fruits contains cantaloupe`
	f)  `is canataloupe identical to item any of fruits ?`

Conditionals (AP)

Item 1	Consider the following for loops:
	
	I. `for(int i = 0; i < 10; i++){`
	`        System.out.println(i);`
	`}`
	
	II. `for(int i = 0; i < 20; i = i + 2){`
	`        System.out.println(i);`
	`}`
	
	III. `for (int i = 0; i <= 9; i++){`
	`        System.out.println(i);`
	`}`
	
	IV. `for (int i = 5; i <= 15; i++){`
	`        System.out.println(i);`
	`}`
	
	Which of these for loops is executed exactly 10 times? For those that are not executed 10 times, how many times are they executed?

Item 2	What does the following method do?
	
	`public static int ranking(int num1, int num2, int num3){`
	`    if (num1 < num2){`
	`        if (num2 < num3){`
	`            return num3;`
	`        }`
	`        else{`
	`            return num2;`
	`        }`
	`    }`
	`    else{`
	`        if (num1 < num3){`
	`            return num3;`
	`        }`
	`        else{`
	`            return num1;`
	`        }`
	`    }`
	`}`

Item 3

The following code is supposed to ask a user for their age and report if the age is an even number between 10 and 100. Rewrite this code to fix the four errors it contains.

```
Scanner console = new Scanner(System.in);
System.out.println("Enter your age:");
int age = console.nextInt();

if(age % 2 = 0){
		System.out.println("Your age is an even number.");
}
else{
		if (age < 10){
				if (age > 100){
						System.out.println("And you are between 10 and 100");
				}
		}
}
```

Conditionals (Intro)

Item 1

Translate the following English statements into Snap! blocks using operator blocks, x position , and y position:

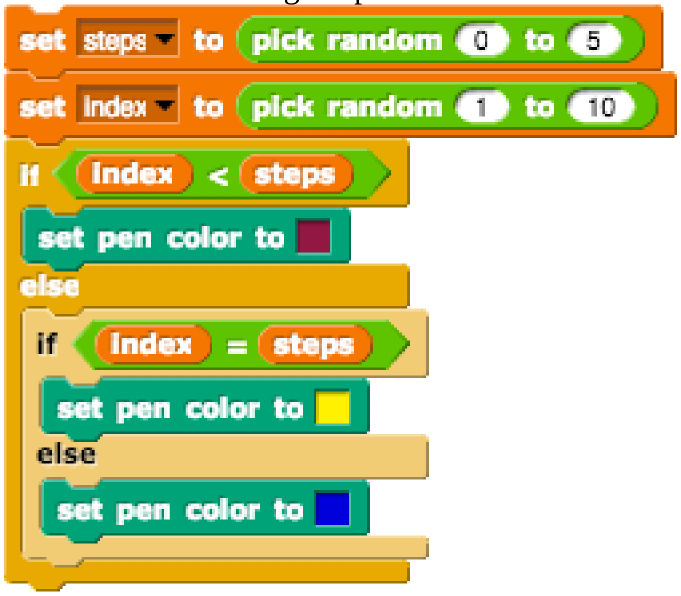
a. x position and y position are of opposite signs (e.g., -3 and 3)

b. x position is closer in value to 0 than y position

c. x position is more than twice as large as y position

Item 2

Consider the following script:

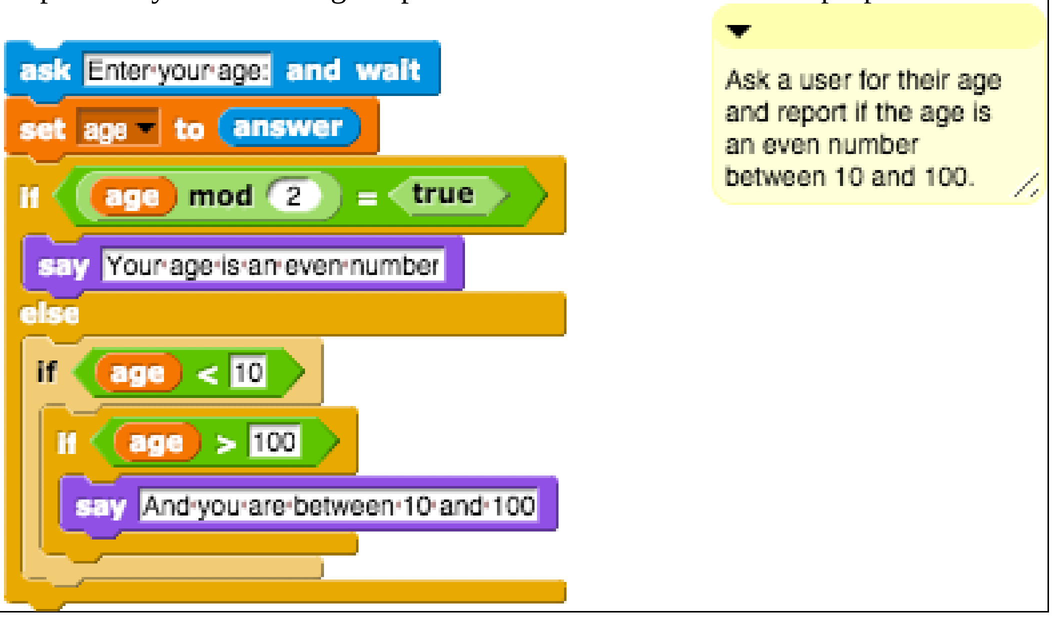


```
set steps to pick random 0 to 5
set index to pick random 1 to 10
if index < steps
    set pen color to ■
else
    if index = steps
        set pen color to ■
    else
        set pen color to ■
```

a) When will the sprite's pen be blue?

b) Complete the following table to show which values can turn your sprite a given color:

steps	index	Color of Sprite
3		Blue
	7	Yellow
4	4	

Item 3	Explain why the following script will not work as intended and propose a fix. 

```
ask [Enter your age:] and wait
set [age] to (answer)
if <((age) mod (2)) = <true>>
    say [Your age is an even number]
else
    if <(age) < [10]>
        if <(age) > [100]>
            say [And you are between 10 and 100]
```

Ask a user for their age and report if the age is an even number between 10 and 100.

APPENDIX E

Think-aloud Interviews: Student Work Prompts

Interview script:

- A group of students were presented with this task *[lay out the task]* and handed in these solutions *[lay out the 3 solutions]*.
- For each solution, discuss what problem that student was having and how you would help address their difficulty.
- *[After the teacher finishes the task]* How might your students respond to this task?

Looping (AP)

The following code simulates a counter that goes from 0 0 to 1 9. Rewrite this code using for loops.

```
int num1 = 0;
int num2 = 0;

System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);

num1 = 0;
num2 = num2 + 1;

System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
num1 = num1 + 1;
System.out.println(num2 + " " + num1);
```

Solution 1

```
for (int num2 = 0; num2 <= 1; num2++){
        System.out.println(num2);
}
for (int num1 = 0; num1 <= 9; num1++){
        System.out.println(" " + num1);
}
```

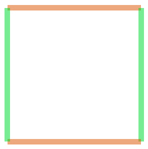
Solution 2

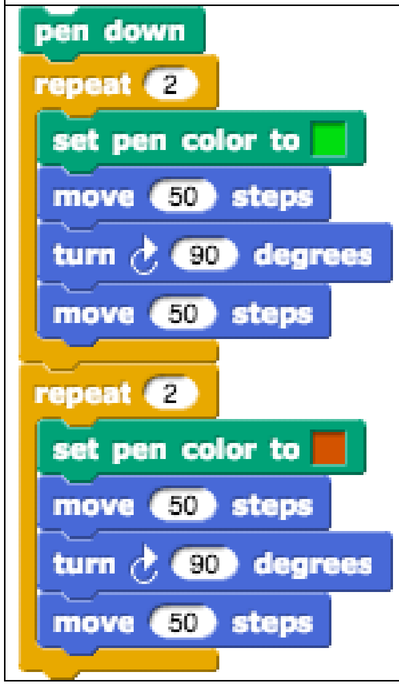
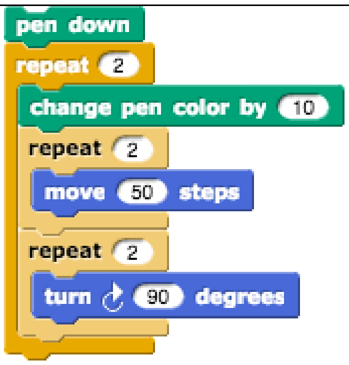
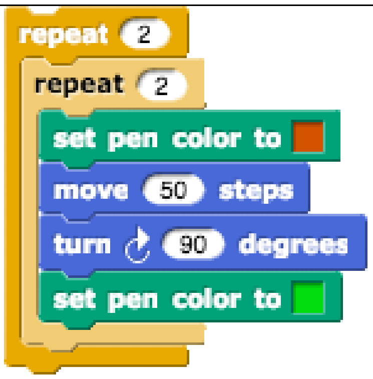
```
for (int num2 = 0; num2 <= 1; num2++){
        num2 = num2 - 1;
        for (int num1 = 0; num1 <= 9; num1++){
                num1 = num1 - 1;
                System.out.println(num2 + " " + num1);
        }
}
```

Solution 3

```
for (int num2 = 0; num2 < 1; num2++){
        for (int num1 = 0; num1 < 9; num1++){
                System.out.println(num2 + " " + num1);
        }
}
```

Looping (Intro)

Task	Write a script that creates a square like the one below where the opposite sides are the same color. 	
Sol. 1	Sol. 2	Sol. 3

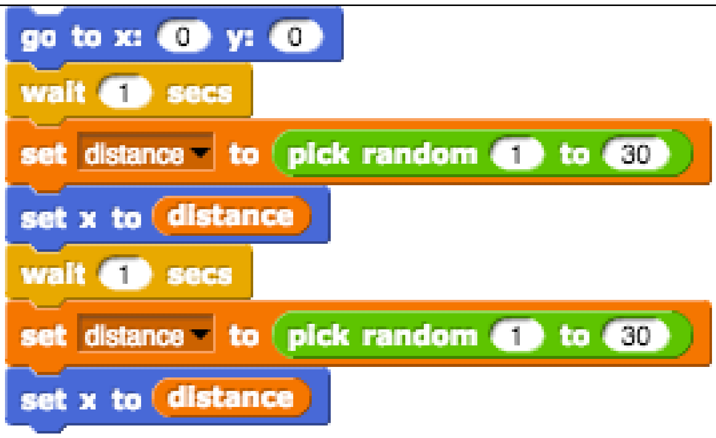
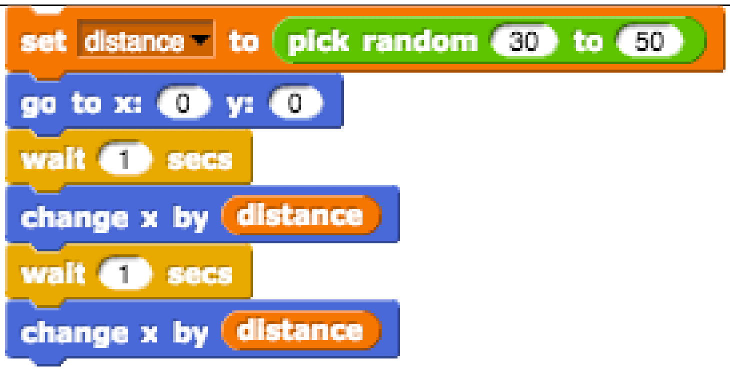
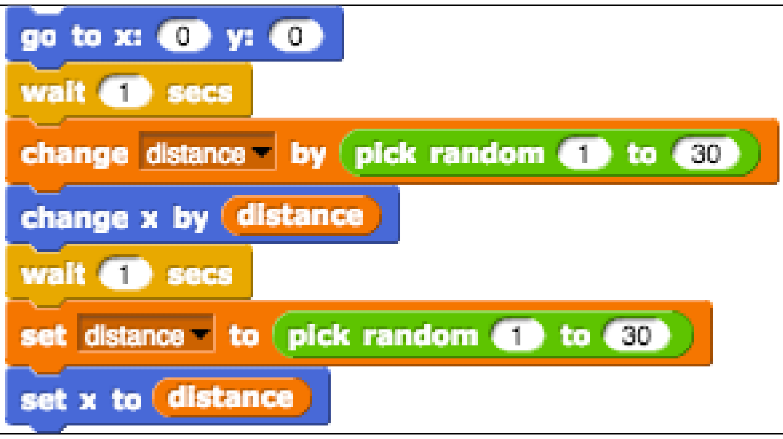

pen down
repeat 2
  set pen color to 🟩
  move 50 steps
  turn ↻ 90 degrees
  move 50 steps
repeat 2
  set pen color to 🟥
  move 50 steps
  turn ↻ 90 degrees
  move 50 steps
	
	pen down
	repeat 2
	  change pen color by 10
	  repeat 2
	    move 50 steps
	  repeat 2
	    turn ↻ 90 degrees
		
		repeat 2
		  repeat 2
		    set pen color to 🟥
		    move 50 steps
		    turn ↻ 90 degrees
		    set pen color to 🟩

Variables (AP)

Task	Consider the following static method:
	
	`public static int calculate(int x){`
	`        x = x + x;`
	`        x = x + x;`
	`        x = x + x;`
	
	`        return x;`
	`}`
	
	Replace the body of calculate with one line of code so that the modified version of calculate will return the same result as the original version for all x.

Sol. 1	`return 3 + x;`
Sol. 2	`return 3 * x;`
Sol. 3	`return 6 * x;`

Variables (Intro)

Task	Create a script that moves your sprite forward twice towards the right edge of the screen. Each time the sprite moves, it should move forward a new random number of times.
Soln. 1	 go to x: (0) y: (0) wait (1) secs set distance to (pick random (1) to (30)) set x to (distance) wait (1) secs set distance to (pick random (1) to (30)) set x to (distance)
Soln. 2	 set distance to (pick random (30) to (50)) go to x: (0) y: (0) wait (1) secs change x by (distance) wait (1) secs change x by (distance)
Soln. 3	 go to x: (0) y: (0) wait (1) secs change distance by (pick random (1) to (30)) change x by (distance) wait (1) secs set distance to (pick random (1) to (30)) set x to (distance)

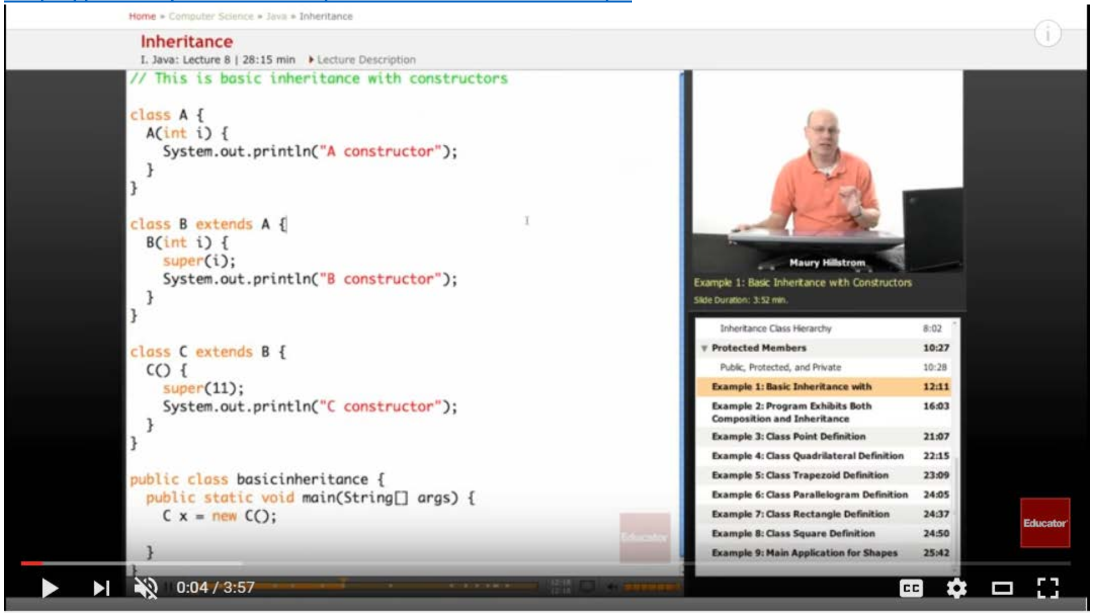
APPENDIX F

Think-aloud Interviews: Instructional Material Prompts

Interview Script:
I would like for you to watch these two short videos that explain *[insert selected topic]*. For each video, describe how well it presents the topic, if it needs any modification, and if you would use it in your instruction.

Inheritance/Cloning (AP)

a) https://www.youtube.com/watch?v=MkFP0vNddqw



Home » Computer Science » Java » Inheritance

Inheritance

I. Java: Lecture 8 | 28:15 min ▸ Lecture Description

```
// This is basic inheritance with constructors

class A {
  A(int i) {
    System.out.println("A constructor");
  }
}

class B extends A {
  B(int i) {
    super(i);
    System.out.println("B constructor");
  }
}

class C extends B {
  C() {
    super(11);
    System.out.println("C constructor");
  }
}

public class basicinheritance {
  public static void main(String[] args) {
    C x = new C();

  }
```

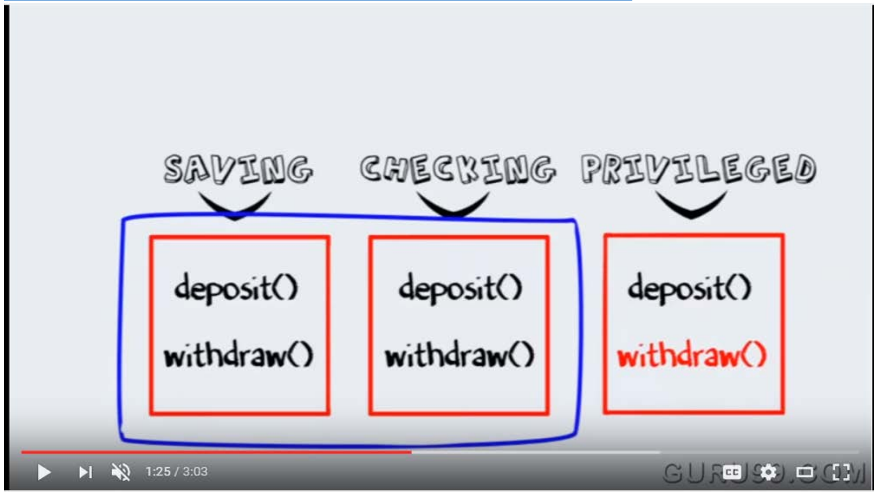
Maury Hillstrom

Example 1: Basic Inheritance with Constructors

Slide Duration: 3:52 min.

Inheritance Class Hierarchy	8:02
▾ Protected Members	10:27
Public, Protected, and Private	10:28
Example 1: Basic Inheritance with	12:11
Example 2: Program Exhibits Both Composition and Inheritance	16:03
Example 3: Class Point Definition	21:07
Example 4: Class Quadrilateral Definition	22:15
Example 5: Class Trapezoid Definition	23:09
Example 6: Class Parallelogram Definition	24:05
Example 7: Class Rectangle Definition	24:37
Example 8: Class Square Definition	24:50
Example 9: Main Application for Shapes	25:42

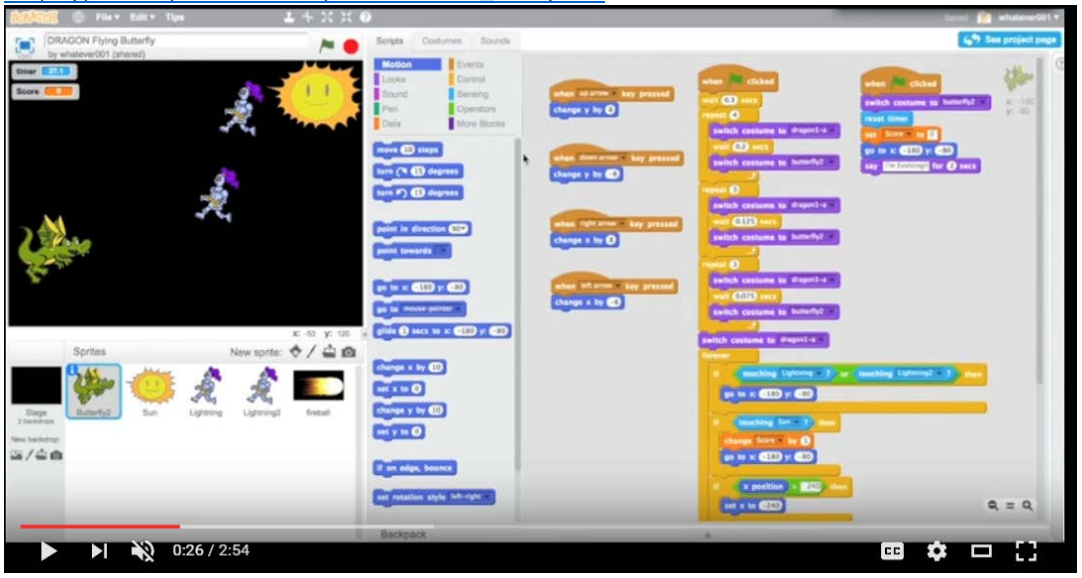
b) https://www.youtube.com/watch?v=gWpg3yMiL0M



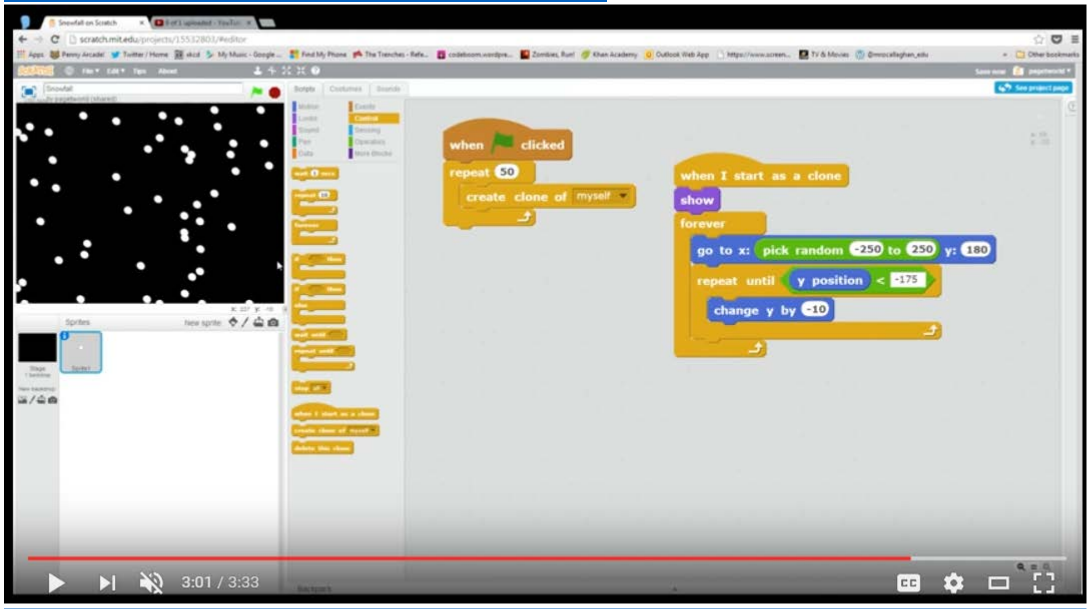
Inheritance/Cloning (Intro)

[Remind the teacher that these videos show cloning in Scratch, not in Snap!.]

a) https://www.youtube.com/watch?v=Zif_mYup7rA



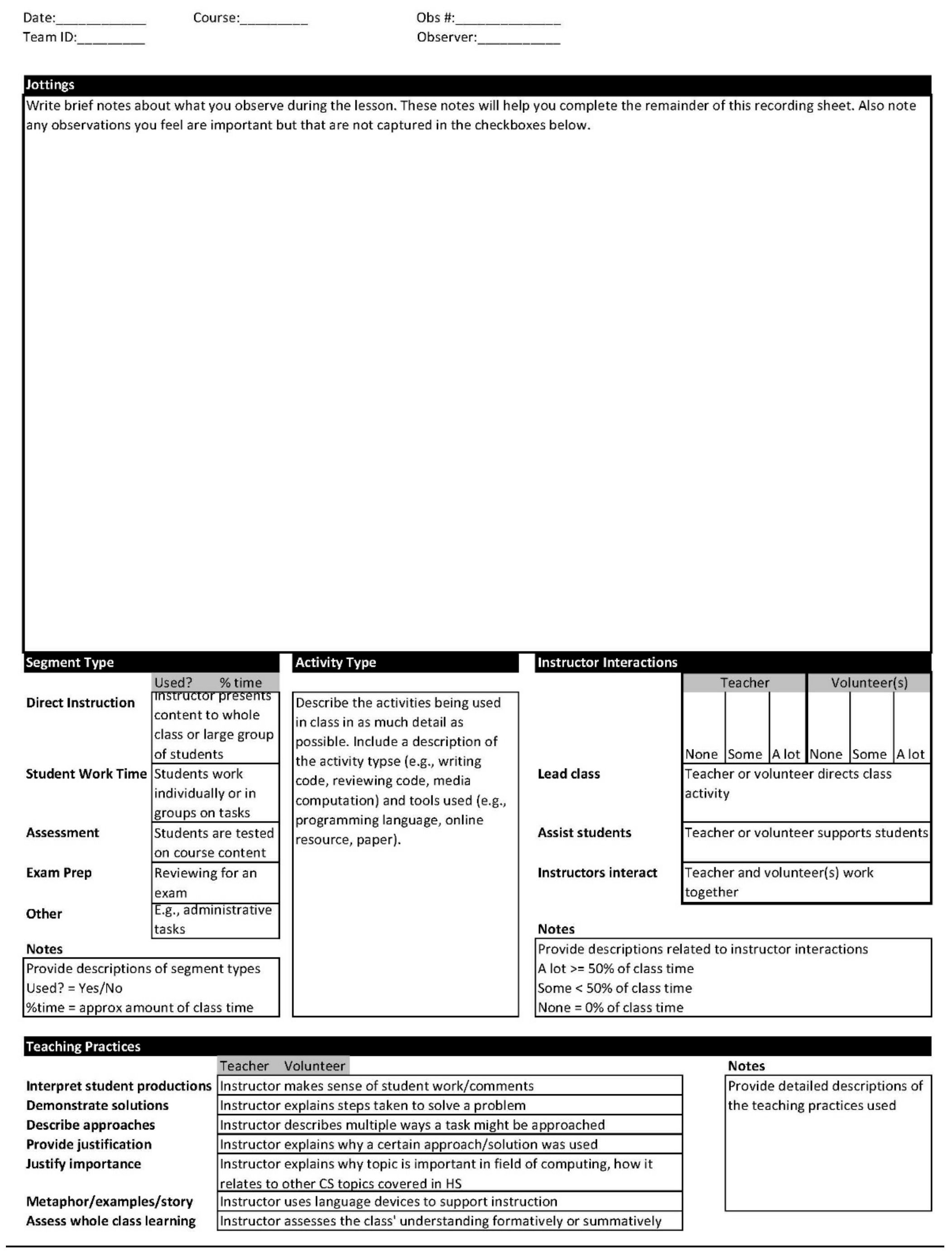
b) https://www.youtube.com/watch?v=pHrkUsrv9p4


```
when [flag] clicked
repeat 50
    create clone of myself

when I start as a clone
show
forever
    go to x: pick random -250 to 250 y: 180
    repeat until y position < -175
        change y by -10
```

APPENDIX G

Observation Protocol I



Date:____________ Course:_________ Obs #:______________
Team ID:_________ Observer:___________

Jottings

Write brief notes about what you observe during the lesson. These notes will help you complete the remainder of this recording sheet. Also note any observations you feel are important but that are not captured in the checkboxes below.

Segment Type

	Used? % time	
Direct Instruction	Instructor presents content to whole class or large group of students	
Student Work Time	Students work individually or in groups on tasks	
Assessment	Students are tested on course content	
Exam Prep	Reviewing for an exam	
Other	E.g., administrative tasks	

Notes

Provide descriptions of segment types
Used? = Yes/No
%time = approx amount of class time

Activity Type

Describe the activities being used in class in as much detail as possible. Include a description of the activity typse (e.g., writing code, reviewing code, media computation) and tools used (e.g., programming language, online resource, paper).

Instructor Interactions

		Teacher			Volunteer(s)		
		None	Some	A lot	None	Some	A lot
Lead class	Teacher or volunteer directs class activity						
Assist students	Teacher or volunteer supports students						
Instructors interact	Teacher and volunteer(s) work together						

Notes

Provide descriptions related to instructor interactions
A lot >= 50% of class time
Some < 50% of class time
None = 0% of class time

Teaching Practices

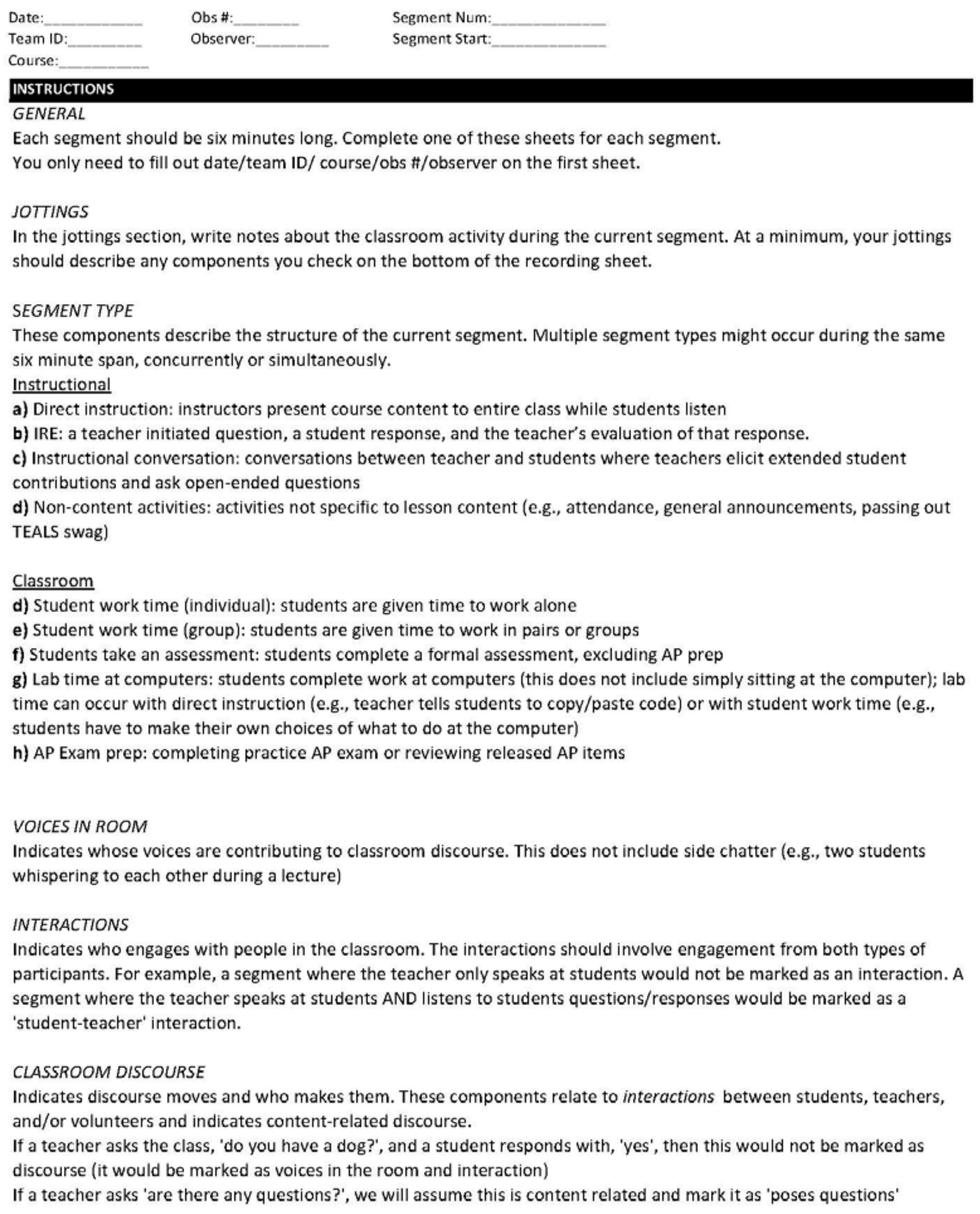
		Teacher	Volunteer
Interpret student productions	Instructor makes sense of student work/comments		
Demonstrate solutions	Instructor explains steps taken to solve a problem		
Describe approaches	Instructor describes multiple ways a task might be approached		
Provide justification	Instructor explains why a certain approach/solution was used		
Justify importance	Instructor explains why topic is important in field of computing, how it relates to other CS topics covered in HS		
Metaphor/examples/story	Instructor uses language devices to support instruction		
Assess whole class learning	Instructor assesses the class' understanding formatively or summatively		

Notes

Provide detailed descriptions of the teaching practices used

APPENDIX H

Observation Protocol II



Date:____________ Obs #:________ Segment Num:______________
Team ID:_________ Observer:_________ Segment Start:______________
Course:___________

INSTRUCTIONS

GENERAL

Each segment should be six minutes long. Complete one of these sheets for each segment.
You only need to fill out date/team ID/ course/obs #/observer on the first sheet.

JOTTINGS

In the jottings section, write notes about the classroom activity during the current segment. At a minimum, your jottings should describe any components you check on the bottom of the recording sheet.

SEGMENT TYPE

These components describe the structure of the current segment. Multiple segment types might occur during the same six minute span, concurrently or simultaneously.

Instructional

a) Direct instruction: instructors present course content to entire class while students listen
b) IRE: a teacher initiated question, a student response, and the teacher's evaluation of that response.
c) Instructional conversation: conversations between teacher and students where teachers elicit extended student contributions and ask open-ended questions
d) Non-content activities: activities not specific to lesson content (e.g., attendance, general announcements, passing out TEALS swag)

Classroom

d) Student work time (individual): students are given time to work alone
e) Student work time (group): students are given time to work in pairs or groups
f) Students take an assessment: students complete a formal assessment, excluding AP prep
g) Lab time at computers: students complete work at computers (this does not include simply sitting at the computer); lab time can occur with direct instruction (e.g., teacher tells students to copy/paste code) or with student work time (e.g., students have to make their own choices of what to do at the computer)
h) AP Exam prep: completing practice AP exam or reviewing released AP items

VOICES IN ROOM

Indicates whose voices are contributing to classroom discourse. This does not include side chatter (e.g., two students whispering to each other during a lecture)

INTERACTIONS

Indicates who engages with people in the classroom. The interactions should involve engagement from both types of participants. For example, a segment where the teacher only speaks at students would not be marked as an interaction. A segment where the teacher speaks at students AND listens to students questions/responses would be marked as a 'student-teacher' interaction.

CLASSROOM DISCOURSE

Indicates discourse moves and who makes them. These components relate to *interactions* between students, teachers, and/or volunteers and indicates content-related discourse.
If a teacher asks the class, 'do you have a dog?', and a student responds with, 'yes', then this would not be marked as discourse (it would be marked as voices in the room and interaction)
If a teacher asks 'are there any questions?', we will assume this is content related and mark it as 'poses questions'

Date:___________ Obs #:_______ Segment Num:_____________

Team ID:________ Observer:________ Segment Start:_____________

Course:__________

Jottings

Instructional Segment Type	Tea.	Vol.
Direct Instruction		
IRE		
Instructional conversation		
Non-content activities		

Voices in Room	Yes?
Student	
Teacher	
Volunteer	

Interactions	Yes?
Student-student	
Student-teacher	
Student-volunteer	
Teacher-volunteer	

Classroom Discourse	Stu.	Tea.	Vol.
Poses questions			
Responds to questions			
Provide explanations			

Classroom Segment Type	Used?
Student work time (individual)	
Student work time (groups)	
Students take assessment	
Lab time at computers	
AP Exam prep	

APPENDIX I

PCK Questionnaire

1. List the difficulties students have with linear data structures (i.e., arrays and lists in AP CS A, lists in Intro CS). For each difficulty:
    - provide a description of the difficulty
    - indicate the frequency with which you see the difficulty when teaching (always, sometimes, rarely, never – you are aware of the difficulty but haven't seen it with your students)
    - describe how you address the difficulty

    Please list as many difficulties as you can, adding additional rows if needed.

Difficulty	Frequency	How you address difficulty

2. List all of the types of lessons and activities you would plan around linear data structures. For each lesson/activity:
    - provide a description of the lesson/activity
    - describe your rationale for using the lesson/activity
    - describe all the ways you help students who have difficulty understanding the lesson/activity

    Please add additional rows if needed.

Lesson/activity	Rationale	How you help students who have difficulty

3. What are the most challenging topics covered in your course? For each topic, do you address these topics differently than other topics? If so, how?

APPENDIX J

Difficult Computing Topics for Introductory Computer Programming Courses

Ranking from Schulte & Bennedsen (2006)	Ranking from Dale (2006)
Recursion	Problem solving
Algorithm efficiency (Big-O)	Parameters
Generics (templates, type parameterisation)	Algorithm, functional decomposition, function
Advanced data-structures (Linked-lists, Trees)	Object-oriented problem solving and design
Polymorphism and inheritance	Arrays
Pointers and references	Test/testing, debugging
Design of classes[1]	Recursion
Problem solving strategies	Pointers
Responsibility driven design, CRC	Polymorphism
Algorithm design	Function, method
Static vs. non-static (methods and variables)	Inheritance
Debugging	Loops
Object communication	Class
Design of methods	Method
Scope of variables	Object-oriented concepts
Design of classes[1]	Oject interaction
Parameters	
Encapsulation	
Instance and other type of variables	
The concept of objects and classes	
The libraries associated with the language	
Simple Data structures (arrays, Strings, …)	
Mental model of the computer	
The programming language syntax	
Selection and Iteration	
Using the program development environment	
UML Class Diagrams (reading them)	
Ethics	

[1]One of these topics was mislabeled in Schulte & Bennedsen (2006) and should be *design of single classes.*

Note: Topics are listed from most difficult to least difficult.

VITA

Aleata Kimberly Hubbard

Education

2017	Ph.D., Learning Sciences, Northwestern University, Evanston, IL (expected 9/2017)
2012	M.A., Learning Sciences, Northwestern University, Evanston, IL
2006	B.S. (H&SS Honors), Computer Science and French and Francophone Studies, Carnegie Mellon University, Pittsburgh, PA

Related Professional Experience

2013 - present *Research Associate*, Science, Technology, Engineering, and Mathematics Program
WestEd, Redwood City, CA

Research: Investigating an on-the-job training program for high school teachers transitioning into computer science. Primary responsibilities include project management, case study design and implementation, site visits, observation team supervision, qualitative data analysis, and dissemination. (NSF-funded EHR Core Research study)

Program Evaluation: supporting clients in assessing and improving the effectiveness of their educational programs through logic model development, instrument creation, site visits, interviews, data analysis, and report writing. Evaluation projects include:

San Jose State University, a project to develop a new technology-focused minor for students in the social and behavioral sciences (NSF-funded IUSE project)

Stanford Global Studies, community outreach programs to support high school and community college faculty in internationalizing course curricula (U.S. DOE-funded National Resource Center)

ETR Associates, a pilot program to create ICT pathways for Latino and Latina youth through partnerships between high schools, community colleges, and industry (NSF-funded Advanced Technological Education project)

All Star Code Summer Intensive, a six-week summer program introducing high school boys of color to computing and the tech industry

MATHCOUNTS Reel Math Challenge, a national competition for middle school students to produce short videos introducing math concepts

Data Management: developing data architectures to support data cleaning and quality assurance for large-scale efficacy studies, supervising data management team, training staff in best practices for data management and R programming. Projects include:

Khan Academy Resources for Maximizing Mathematics Achievement: A Postsecondary Mathematics Efficacy Study (3-year IES-funded efficacy study)

National Research & Development Center on Cognition and Mathematics Instruction (5-year NSF-funded efficacy study)

2012-2013	*Research Assistant*, Science, Technology, Engineering, and Mathematics Program WestEd, Redwood City, CA Managed teams involved in research instrument integrity and gathering demographic and assessment data for educational research projects. Developed logic models, wrote literature reviews, and designed research plans to contribute to grant proposals in mathematics education and educational technology. Trained staff in developing databases to manage study data and participant information.
2008-2009	*Teaching Fellow,* Meltwater Entrepreneurial School of Technology Meltwater, Accra, Ghana Designed and taught courses on product development, databases, and web technologies to a cohort of 20 young adults.

Related Publications and Presentations

Hubbard, A., & Kao, Y. (2017, March). *Computer Science Teaching Knowledge: A Framework and Assessment*. In Proceedings of the 48th ACM Technical Symposium on Computing Science Education. New York, NY, USA: ACM.

Hubbard, A. K., Kao, Y., Brown, D. (2016). *Designing Think-aloud Interviews to Elicit Evidence of Computer Science Pedagogical Content Knowledge*. Paper presented at the 2016 Annual Meeting of the American Educational Research Association, Washington, D.C.

Kao, Y. S., DeLyser, L. A., & Hubbard, A. K. (2016, March). *Assessing the Development of Computer Science Pedagogical Content Knowledge in the TEALS Program*. In Proceedings of the 47th ACM Technical Symposium on Computing Science Education (pp. 695–695). New York, NY, USA: ACM.

Hubbard, A., & Kao, Y. (2014). *Industry partnerships to support computer science high school teachers’ pedagogical content knowledge*. In Proceedings of the 15th Annual Conference on Information technology education (pp. 89–90). Atlanta, Georgia, USA: ACM.