AN EXPLORATORY STUDY TO IDENTIFY BARRIERS TO IMPLEMENTATION OF COMPUTATIONAL THINKING

by

Thomas Lloyd Stokke
Bachelor of Science, University of North Dakota, 1987
Master of Science, University of North Dakota, 1991

A Dissertation

Submitted to the Graduate Faculty

of the

University of North Dakota

in partial fulfillment of the requirements

for the degree of

Doctor of Philosophy

Grand Forks, North Dakota

August
2019

This dissertation, submitted by Thomas Lloyd Stokke, in partial fulfillment of the requirements for the Degree of Doctor of Philosophy from the University of North Dakota, has been read by the Faculty Advisory Committee under whom the work has been done and is hereby approved.

Gail Ingwalson

Dr. Gail Ingwalson, Chairperson

Myrna Olson

Dr. Myrna Olson

[illegible]

Dr. Emanuel Grant

Ryan Summers

Dr. Ryan Summers

This dissertation is being submitted by the appointed advisory committee as having met all of the requirements of the School of Graduate Studies at the University of North Dakota and is hereby approved.

[illegible]

Chris Nelson
Associate Dean of the School of Graduate Studies

7/30/19

Date

PERMISSION

Title	An Exploratory Study to Identify Barriers to Implementation of Computational Thinking
Department	Teaching and Learning
Degree	Doctor of Philosophy

Thomas Lloyd Stokke
August 1, 2019

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGMENTS

I want to thank my chair, Dr. Gail Ingwalson, and the rest of my committee, past and present. I have no doubt that working with me has been trying at times. I truly appreciate your patience, effort, and support. It’s been a journey, and I thank each of you for being a part of it.

I want to thank the Computer Science Department for their continued support, specifically Dr. O’Neil, Dr. Marsh, and Dr. Grant for playing along with whatever crazy idea or outreach activity I wanted to try. Directly or indirectly, this support helped fuel the passion which lead me to pursue this degree.

I want to thank my parents, who have always been supportive through my entire educational process, even when there had to be serious concerns about the outcome. You were right, if I “just apply myself” good things can happen. I would also want to thank Ginger, who was always been supportive in every way possible.

And most importantly, to Nick, Elle, Erik, and especially to my wife Angie, so much of the time spent on this endeavor was time spent away from family. So, thank you, I can’t express in words my appreciation for your love, support, and understanding as I pursued this goal.

ABSTRACT

Computational thinking, or learning to address problems in a systematic manner such that the problem *could* be solved using a computer, is a skill that can be applied not just to computer science, but to other areas of study. Given the ubiquitous nature of computing as a critical component of innovation in many fields, learning to use a computer as a problem-solving tool could be viewed as a required skill for today's students. There have been many initiatives promoting the introduction of computational thinking into a classroom, such as the "Hour of Code" (Wilson, 2015), yet the inclusion of such content into curricula at the time of this study was limited. Part of the problem is likely to be that learning to use a computer as a problem-solving device is something that is rarely taught to educators, whether as pre-service teachers or in professional development classes for teachers already working in the field. Nonetheless, it is likely that there are other deterrents as well. The purpose of this study was to use a cross-sectional survey employing both closed ended (quantitative) and open-ended (qualitative) questions to collect data from K-5 teachers regarding perceived barriers to implementation of computational thinking content into their curricula.

CHAPTER I

INTRODUCTION

The demands of a global economy require that our future workforce be prepared with problem-solving and critical thinking skills. Thomas Friedman, in his book *The World is Flat*, stated, we must prepare students for jobs not yet created (Friedman, 2005). As cliché as that may sound, consider that the first text-based web browser was created in 1990, and the first graphical web browser was created in 1993 (McPeak, 2018). It is safe to say that almost all students graduating from college at the time of this study must have been proficient using a technology that did not exist when they were born. To go a step further, consider that the Apple iPhone was first introduced in 2007 (Sanford, 1996-2019) and the Android smart phone was first introduced in 2008 (Chitu, 2007; German, 2011). Neither of these devices was in existence when those same college students started high school, yet smart phones were seemingly ubiquitous in society at the time of this report; Android alone had over 2 billion activations (Popper, 2017).

Freidman (2005) argued that the world needs "versatilists"; individuals with expertise in some field and in technology who are "adaptable and versatile" (p. 289). Freidman posited that the world needs people with diverse skills able to integrate computing skills with skills specific to a domain to solve big problems the world will face in the 21st century (Friedman, 2005).

Computing is at the heart of how we work and communicate. It has had a dramatic effect on the world around us, changing how we view science, engineering, business, and the medical profession. Advancements in literally every field are made possible through the use of computing technologies, yet few of those making the advancements will be formally trained computer scientists. In order to advance this trend, students must be computer-literate not only in the consumption of content, but in the creation of content. Today's students must learn to use the computer as a problem-solving tool for preparation in whatever field they choose (Chiazzese, Fulantelli, Pipitone, Taibi, 2017; Furst, Isbell, Guzdial, 2007; Guzdial, 2008; Guzdial & Soloway, 2003; Kafai & Burke, 2013; Kafai & Burke, 2014; Resnick, Bruckman, & Martin, 1996). Mitchel Resnick, LEGO Papert Professor of Learning Research at the MIT Media Lab and founder and director of the MIT Lifelong Kindergarten research group, stated, "Would you rather that your children learn to play the piano, or learn to play the stereo?" (Resnick et al., 1996, p. 40). Learning to use a computer to create content will allow a student to express themselves in ways far beyond what they could do otherwise. When students are computer literate, they are generally thought of as being competent to use tools created by someone else, restraining a student to using ideas and limitations of someone else's tools. Once a student has learned to create with a computer, they have the means to develop their own voice and the skills to express their own ideas. Advancements made in any field, and not just computer science, require development of new tools, implementation of new and innovative ideas, and not just use of whatever tools we have available at a given time.

Many benefits may be gained from learning to create content with a computer, benefits that extend far beyond merely being able to work with a computer. Students need to learn "computational thinking," the ability to break down and analyze problems in a fashion that can be addressed using a computer (Papert, 1996; Wing, 2006). Learning to create with a computer requires a student learn skills useful in all academic areas; students should learn to think algorithmically and develop critical thinking and problem-solving skills. It has been said that the best way to learn something is to teach it to someone else. A computer can be that "someone else." What better way is there to learn than to explain the solution to a problem to a computer? A computer has no prior knowledge of information (material input) and makes no assumptions. To explain a solution to a computer requires a student to break down all parts of an algorithm to its most basic state and then explain the solution to that algorithm to a computer in such a way that there can be no confusion or ambiguity regarding execution of the solution.

The need for STEM (science, technology, engineering, and mathematics) education in the U.S. education system is at an all-time high (National Governors Association, 2007). While not a formal part of the STEM acronym, computer science plays a critical role in STEM education. Including computer science in K-12 education is paramount to ensuring today's students are well prepared for their future role in society (President's Council of Advisors on Science and Technology, 2010; Wilson et. al., 2010). In an interview with Stephen J. Dubner of the Freakonomics podcast, Gina Raimondo, the governor of Rhode Island, stated why she felt computing skills are critical for all K-12 students:

> I think of it [teaching computer science in every district and every grade, starting in kindergarten] as access and exposure, and also just providing people with a basic level of essential skills. . . . No matter what job you have, you have to have some basic familiarity with computer skills and digital skills. And so it is as essential in this economy as any other skill that we teach. (Dubner, 2018, time into podcast 32:47)

Yet, formal computer science education in K-12 environments is struggling (Ericson et al., 2008). It is hard to find and retain computer science teachers. High paying jobs in industry make it difficult to attract educators and the constant evolution of the field makes it a daunting task for teachers to keep their knowledge of technology current (Denning, 1981; Guzdial, 2012). Add in the realities of limited capabilities to offer elective classes, tightening budgets, difficulties in retaining teachers, and falling enrollments in K-12 education in rural settings and overall difficulties in finding qualified educators in STEM fields, and the likelihood of a school maintaining a dedicated computer science curriculum is highly unlikely (The College Board, 2019b; Briggs & Snyder, 2012; American Association for Employment in Education, 2016; Maio, 2016; Marder, 2016). To further complicate the issue, since computer science is not a formal component of any set of educational standards, teachers and administrators are less inclined to devote time, energy, and resources to non-required material.

Given these obstacles, regardless of how ingrained computers are into our social fabric, school systems universally fail to provide the education students need in this area. If we expect today's students to be proficient with future technology, using computing devices that will be found in literally every industry and field, it is a necessity students have a solid background in developing content using computers and technology. Yet, regardless of need, given the aforementioned obstacles, few schools in the United States are willing or able to hire educators dedicated to teaching computational thinking. Therefore, if computational thinking content is not to be delivered in subject-specific classes (i.e. in classes focused specifically on computational thinking methods, including use of technology), then any computational thinking education students acquire must come from teachers working within existing classes in existing fields.

Given most schools' limited capacity to offer electives in all but the largest school districts, if advances are to be made in learning to use computers to solve problems in North Dakota, as well as literally any other rural area, those advances will have to be accomplished through standard curricula, and not in a class dedicated to just computing concepts, at least for a time. Since computing skills will be required in literally every field, it seems reasonable for computing skills to be introduced in context within current curricula. It might be in mathematics, science, English, or any other area where a computer might typically be used. Educators in K-12 schools, those teachers who never trained in computer science, or even computational thinking, will be some of the ones who will have to introduce computational thinking to their students.

There are numerous tools available for learning programming within the context of a standard K-12 curriculum. Programs such as Alice (Carnegie Mellon University, 1999-2017), Scratch (Lifelong Kindergarten Group, n.d.), Squeak Etoys ("Squeakland: Home of Squeak Etoys," n.d.), Kodu (Microsoft, 2019), and Logo (Harvey, n.d.) are all free and have been successfully used in schools around the world. All of these software programs work with the language of a host operating system. Given the internationalization and localization of modern operating systems, allowing today's computers to work with literally every language in the world, these programs work well with ELL learners or in a world language class. Since all these programs allow a user to input mathematical functions to control movement of characters or objects, it is easy to create an opportunity and need to correctly implement a formula. Whatever the need, there is age-appropriate software available to supplement required material.

While the majority of research in computer science education has focused on K-12 education (Franke et al., 2013; Hubwieser et al., 2014; Randolph, 2007), the focus of this study will be on K-5 educators. Educators in middle school and high school teach in specific subject areas. Before being able to receive an educators' license, a teacher generally must earn a bachelor's degree in the fields they wish to teach in along with taking required education classes (Education Standards and Practices Board, 2017). K-5 educators are usually "generalists" in the sense that they typically teach all curricular content except for a few "specialists" who teach in specific areas (e.g., physical education, music, or foreign languages). Given that the addressed problem is students' lack of exposure to general computational thinking skills and not a lack of specific computer programming skills, this study will address K-5 educators rather than 7-12 educators who would teach a class in computer programming. The population for this study included teachers with the potential to reach *all* students within the K-5 grade range, and not just teachers who teach older students, middle school or high school students, who have chosen to take a computer science elective.

Research Questions

The purpose of this study was to gain insight and understanding into K-5 educators' attitudes and beliefs regarding implementation of computational thinking content in their classrooms. Based upon prior experiences, there are a variety of possible explanations for why teachers may be unwilling or unable to introduce computational thinking content to their students. These may include among other possibilities, in no particular order: lack of interest, lack of comfort level with computational thinking material, lack of comfort level with needed technology, limited time to prepare for teaching new material, limited peer support, lack of administrative support, and little perceived utility for the students. Research questions include delving into details regarding why educators feel as they do regarding computational thinking, and what steps, if any, might be done to alleviate concerns, improve interest, and increase the likelihood of implementing computational thinking content into their curricula.

Based upon previous research, the intent of this study was to answer the following research questions:

1. What are educators' levels of understanding in regard to the definition of and opportunities to utilize computational thinking?

2. What are educators' levels of interest and comfort surrounding computational thinking?
3. What are educators' beliefs in regard to the importance of computational thinking in the classroom and their careers?
4. What are educators' beliefs regarding the use of computational thinking in specific K-5 content areas of English, mathematics, social studies, and science?

Conceptual Framework

Teachers play a fundamental role in formal instructional systems. If a meaningful change in a curriculum is going to be effective, the teachers, not technology, will have to be the agent of change (Fisher, 2006). Teachers are viewed as a "cornerstone" or "the most influential factor" in educational innovations (van Driel, Beijaard, & Verloop, 2001; Fishman & Davis, 2006). Harris stated that "using technology as a 'Trojan horse' for educational reform has succeeded in only a minority of K-12 contexts" (Brinkerhoff, 2006, pp. 39-40). Yet, it has been shown that teachers are uncertain or reluctant about adopting new curricular and/or instructional practices (Ponticell, 2003). There are a number of challenges teachers face when making changes in their classrooms, including but not limited to: being asked to teach outside their area, feeling isolated, classroom management concerns, lack of pedagogical knowledge, and limited time for planning (Yadav, Gretter, Hambrusch, & Sands, 2016). Given teachers importance in the educational system, it is critical to examine teachers' attitudes and perceptions regarding innovation in their classroom curriculum, and the factors that affect their willingness to accept such changes.

Theoretical Framework - Decomposed Theory of Planned Behavior

The Theory of Planned Behavior (TPB), proposed by Ajzen and Fishbein (Ajzen, 1985), is based upon the principle that behavior is directed by three factors: a) attitudes, b) subjective norms, and c) perceived behavioral control, or degree of control a person believes they have in a situation. The TPB evaluates attitudes (an individual's feelings regarding certain behaviors), subjective norms (degree of influence others' have on a person's behavior), and perceived behavior control or the degree in which a persons' internal factors (i.e. talents or skills) and external factors (i.e., opportunities, resources, or support) predict one's behavior (Smarkola, 2008).

The decomposed theory of planned behavior (DTPB) extends the TPB by breaking it down into belief-based indirect measures. It was suggested by Taylor and Todd (1995) that the DTPB increases "explanatory power" and provides a better understanding of behavior (Sadaf, Newby, & Ertmer, 2012a) than the TPB. The DTPB model is a widely used and validated model, useful for predicting behavior intentions in information technology and education studies. Figure 1 provides a pictorial view of the decomposed theory of planned behavior.

Attitude

In this study, *attitude* refers to teachers' feelings about introducing computational thinking into classrooms. Past research has found a strong positive relationship between teachers' attitudes and their intentions to use computer technology in their classrooms (Sadaf et al., 2012b; Teo, 2009; Teo, Lee, & Chai, 2008). It is expected that a similar positive relationship will be found between teachers' attitudes regarding computational thinking and their intentions to introduce computational thinking into their curricula.

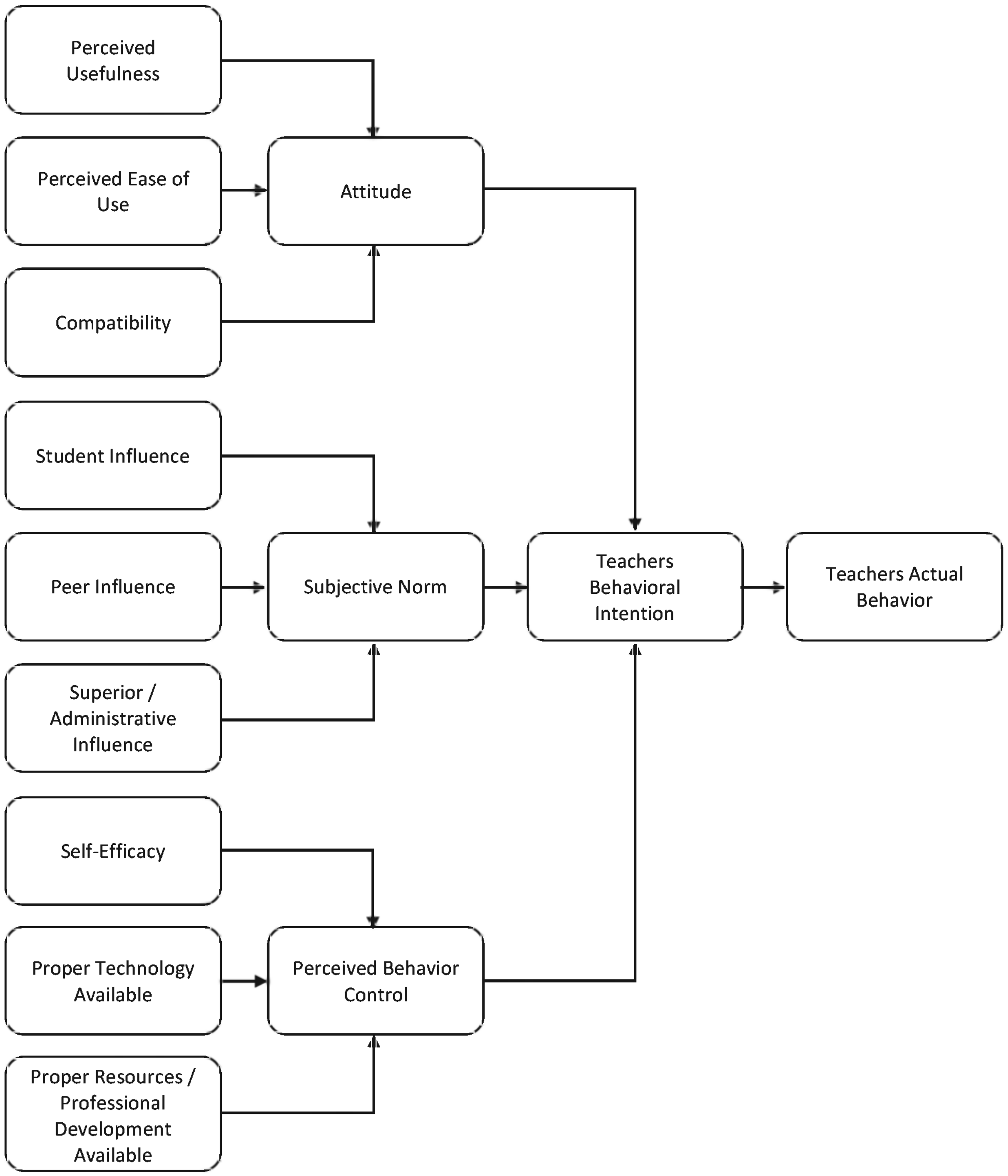
Perceived Usefulness
Perceived Ease of Use
Compatibility
Attitude
Student Influence
Peer Influence
Superior / Administrative Influence
Subjective Norm
Teachers Behavioral Intention
Teachers Actual Behavior
Self-Efficacy
Proper Technology Available
Proper Resources / Professional Development Available
Perceived Behavior Control

Figure 1. Pictorial View of the Decomposed Theory of Planned Behavior. Adapted from "Understanding Information Technology Usage: A Test of Competing Models," by S. Taylor and P. A. Todd, 1995, *Information Systems Research*, *6*(2), p. 146. Copyright 1995 by the Institute for Operations Research and the Management Sciences.

The DTPB model breaks attitude down into three components:

- perceived usefulness
- perceived ease of use
- compatibility

Perceived usefulness is the extent to which teachers believe that teaching computational thinking in their classroom will help improve students' overall learning and capabilities. *Perceived ease of use* is the amount of effort teachers believe they will have to exert in order to successfully teach computational thinking in their classroom. *Compatibility* gauges if teachers feel the introduction of computational thinking is compatible with how they work, or with how teachers feel the changes fit in with their needs within their classrooms (Taylor & Todd, 1995).

Subjective Norms

The second element of DTPB is *subjective norms*. Subjective norms refer to the influence others have on a person's behavior, on whether or not they feel they should perform or include a behavior (Ajzen, 1991). Previous work has shown that subjective norms can be a key factor affecting teachers' intentions to use technology (Sugar, Crawley, & Fine, 2004; Teo, 2009). Subjective norms are influences, or social pressures, intended or otherwise, that may have an effect on what teachers believe is important. This includes how suggestions or opinions from other teachers or administrators may affect beliefs.

The DTPB models breaks subjective norms down into three components:

- student influence
- peer influence
- superiors' influence. (Ajzen, 1991)

Student influence includes the positive effect teachers feel introduction of computational thinking will have on students' learning. It also includes how comfortable a teacher feels students will be with respect to computational thinking content or the expectation level of students that computational thinking material should be integrated into their coursework. *Peer influence* is the positive effect peers (colleagues) have on a teacher's feelings toward including computational thinking into their curriculum. *Superiors' influence* is the positive effect that teachers feel from their administration, whether it be on a personal level from principal or vice-principal, or whether the influence comes indirectly from a district or even state level.

Perceived Behavioral Control

The final element of DTPB is *perceived behavioral control*. Perceived behavioral control refers to teachers' perceptions of the ease or difficulty in implementing computational thinking into their curricula. Previous studies have shown that teachers who use self-assured skills and necessary resources are likely to adopt new technologies; it seems reasonable to think that those same skills sets would impact a teacher's decision to teach their students to work with computational thinking (Teo, 2009; Yushau, 2006).

The DTPB models breaks perceived behavioral control down into three components:

- self-efficacy
- proper technology available
- proper resources and professional development available

Self-efficacy is defined as the perception of how well one can perform a behavior (Bandura, 1982). In this study, self-efficacy will refer to a teacher's perceived ability to introduce computational thinking content into their curriculum. *Proper technology available* refers to having appropriate hardware and software for computational thinking exercises in the classroom. *Proper resources and professional development available* refers to having appropriate support systems in place to address any potential problem, whether the problem is curricular in nature or a hardware or software issue requiring technical support. It also means having the time required to adequately learn to use hardware and software for computational thinking exercises.

It is important to recognize that perceived behavioral control depends upon one having some level of actual behavioral control. As the opportunity to perform a behavior increases, the likelihood of an individual performing the behavior increases as well. Individuals who feel they have neither the resources nor opportunities to successfully implement or perform a behavior, even if it is a desirable behavior, are unlikely to report an intention to implement said behavior (Kan & Fabrigar, 2017; Martinez & Lewis, 2016). Ajzen (2015) stated, "Actual behavioral control is expected to moderate the effect of intention on behavior" (p. 125). If teachers do not feel they will be provided sufficient professional development opportunities to fully embrace computational thinking concepts, or if they feel they will not receive appropriate administrative support, it is possible their reported intentions will not be positive, even if they personally feel strongly about the introduction of computational thinking into their classrooms.

There are other limitations to the DTPB to consider (LaMorte, 2018; Tang, 2016):

- Demographics and personality are not addressed in the framework;
- It is difficult to measure perceived behavior control;
- As time between intent and action increases it becomes less likely the action will happen;
- The framework assumes people will make rational decisions and ignores unconscious motives.

The Decomposed Theory of Planned Behavior is a widely-used and validated model for predicting intention and one's behavior based upon attitudes, subjective norms, and perceived behavior control measures. By decomposing the measures into belief-based indirect measures, the Decomposed Theory of Planned Behavior provides a comprehensive means to understand how one's attitudes, subjective norms, and perceived behavior control can affect his or her intentions regarding the use of computational thinking in the classroom.

Purpose Statement

A purpose statement "identifies the variables, their relationship, and the participants and site for the research" (Creswell, 2012, p. 110). The purpose statement should be a single sentence, starting with words such as "the purpose of this study" or "the intent of this study." It should relate directly to the research problem, convey the overall purpose of the study, and tell what the study will accomplish (Locke, Spirduso, & Silverman, 2014). According to Creswell and Plano Clark (2011), independent and dependent variables are typically listed in order from left to right along with the theory that will be used in the study. The purpose of this cross-sectional survey was to gather information from K-5 educators working at the time of this study to determine what factors influence or affect their attitudes or interests in using computational thinking coursework in their classrooms.

Significance of the Study

The findings from this study may increase the likelihood of computational thinking content being introduced into classrooms. By addressing the needs of teachers, quality of instruction may be improved where computational thinking is already part of the curriculum. Findings from this study may also assist those designing professional development workshops to be more responsive to the needs of teachers. Findings may also help guide administrators interested in adding computational thinking into their overall K-5 curricula.

Limitations of the Study

There are unavoidable limitations in this proposed study. First, all the K-5 teachers who take the survey will be self-selected, such that those who wish to participate will do so. Second, it seems likely that most of the participants will feel strongly either for or against computational thinking. Additionally, given the general absence of the introduction of formal computational thinking skills at any level, it maybe be difficult to verify the level at which the participants understand the definition of computational thinking.

On a personal level, the author of this dissertation has been an educator, a faculty member in a computer science department at the college level, for over 15 years. He has also been actively involved in outreach activities in K-8 schools in addition to starting and directing youth computer science-based summer camps for over 10 years. Therefore, it is possible the author could be recognized by potential participants, which could have an influence on any responses. Finally, as an educator in the field of computer science, the author has observed there is value in using computational thinking in many areas and may be biased in favor of using computational thinking methods in our educational system.

Definitions

Abstraction – Removing unnecessary details from a problem so that a general solution can be found that will work for similar problems.

Attitude – For purposes of this study, attitude refers to how a teacher feels about introducing computational thinking into their classroom.

Compatibility – In regards to computational thinking, how Teachers believe computational thinking fits into their current classroom curriculum.

Computational Thinking – An approach to problem solving such that the solution *could* be implemented with a computer, but does not require the use of a computer or involve programming. The basic steps in computational thinking include problem solving, decomposition, abstraction, and pattern matching.

Computer Programming or Programming – The act of creating or writing programs to be used on computers. Computer programs include a sequence of statements in a programming language that can be translated into a form a computer can understand and execute.

Computer Science – The study of computers and computational systems. This includes an understanding of hardware and the ability to design and implement software. The field of computer science spans from the theories behind computational systems to the practical and efficient implementation of applications.

Decomposed Theory of Planned Behavior – A theoretical framework that attempts to explain and predict one's intentions.

Decomposition – Breaking a problem down into a series of smaller, simpler steps.

Likert Scale – An ordinal scale in which respondents are asked to provide a response of "strongly agree," "agree," "disagree," or "strongly disagree." A "neutral," or "neither agree nor disagree" option is commonly included at the midpoint. A true Likert scale has five options. A "Likert-type scale" may give respondents more or less than five options (may alter the number of options available to three, seven, nine, or some other number or may change the wording of the options or both).

Pattern Matching – During decomposition of a problem, determining if a solution already exists for each smaller, simpler step.

Peer Influence – In regards to computational thinking, how much influence a teacher perceives other teachers will have on their decision to include computational thinking into their classroom.

Perceived Behavior Control – In regards to computational thinking, how easy a teacher believes it will be to implement computational thinking into their classroom curriculum.

Perceived Ease of Use – In regards to computational thinking, how much effort teachers feel they need to exert to add computational thinking into their classroom teaching methods.

Perceived Usefulness – In regards to computational thinking, how teachers perceive introducing computational thinking into their lesson plans will help their students.

Problem Solving – The ability to identify a problem, collect and analyze data in order to fully understand the problem, and using that data solve the problem.

Proper Resource and Professional Development Available – In regards to computational thinking, a teacher's perception of resources available for teaching computational thinking. Will they have enough time to develop curricula, gain needed knowledge, and find appropriate support for implementing a new and unknown subject like computational thinking, and will professional development opportunities be available?

Proper Technology Available – In regards to computational thinking, a teacher's perception of the adequacy of hardware and software available in their classroom for teaching computational thinking?

Self-Efficacy – In regards to computational thinking, a teacher's perception of their ability to implement computational thinking into their classroom curriculum.

Student Influence – In regards to computational thinking, how teachers feel computational thinking will affect their students.

Subjective Norms – In regards to computational thinking, the influence other people have on a teacher's decision to implement computational thinking into their classroom.

Superiors' Influence – In regards to computational thinking, how much influence a teacher perceives their immediate superiors (principal, superintendent, etc.) will have on their decision to include computational thinking into their classroom curriculum.

Summary

The need for STEM education, and specifically computer science education, is at an all-time high (National Governors Association, 2007; President's Council of Advisors on Science and Technology, 2010; Wilson et al., 2010). In the world at the time of this study, the use of computers was ubiquitous, helping to drive advancements in literally every field. Yet, while computers have been seemingly everywhere, students at the time of this report generally limited their use of computing devices to consuming content and using programs created by others, limiting their use to products created by others. To become effective problem-solvers, students need to learn to use their computers as problem-solving tools to create their own content for use on their computers (Papert, 1996; Wing, 2006). Current educational curricula do little to help students learn to create content to create their own solutions to today's problems (The College Board, 2019b; Briggs & Snyder, 2012; American Association for Employment in Education, 2016; Ericson et. al., 2008; Maio, 2016; Marder, 2016,).

Computational thinking, a skill set which addresses problems with the idea that they can be solved by a computer, can be useful in effectively addressing problems in all disciplines, with or without the use of a computer. Computational thinking is a skill that can be effectively introduced when working with or without a computer.

In this chapter the Decomposed Theory of Planned Behavior, a theoretical framework for guiding the research process, was introduced. In Chapter II the concept of computational thinking will be explored, and a definition appropriate for this study will be provided. Chapter III will describe the methods used in this study. Chapter IV will provide the results of the study, and Chapter IV will include a discussion of results.

CHAPTER II

LITERATURE REVIEW

What is Computational Thinking?

In a 1996 *Communications of the ACM* article, Jeanette Wing used the phrase "computational thinking" to advance the concept that everyone can benefit from learning to think like a computer scientist (Wing, 2006). However, there was, and continues to be, a common misconception that computer science equals computational thinking. Learning to think like a computer scientist is not the same as *being* a computer scientist. Computational thinking is not merely the act of programming a computer, or even working with a computer. Computer science and computational thinking involve so much more than just programming a computer. Hal Abelson (Abelson & Sussman, 1986), M.I.T. Professor and SIGCSE Outstanding Contribution to Computer Science Education winner, stated:

> [Computer science is] not really very much about computers—and it's not about computers in the same sense that physics is not really about particle accelerators, and biology is not really about microscopes and Petri dishes . . . and geometry is not really about using surveying instruments. . . . Now the reason that we think computer science is about computers is pretty much the same reason that the Egyptians thought geometry was about surveying instruments; and that is when some field is just getting started and you don't really understand it very well, it's

very easy to confuse the essence of what you're doing with the tools that you use.

(time into lecture – 00:49)

Computational thinking *may* use a computer to implement and test solutions to problems, but it does not have to. At the heart of computational thinking is designing a solution to a problem. Computational thinking involves skills required to analyze, design, and implement solutions that could maybe use a computer, and not necessarily skills used to implement said solutions that require using a computer.

The concept of "computational thinking" is not easily defined, or universally agreed upon. Even the phrase computational thinking is commonly referred to as procedural literacy (Bogost, 2008, p. 137; Mateas, 2005, p 101) or computational literacy (diSessa, 2001, p. 6). During a "Scratch Conference" in 2010, Mark Guzdial and Barbara Erikson, both leaders in computer science education, were "as confused as anyone else what 'Computational Thinking' means" (Guzdial, 2010, para. 1). Even Wing has provided multiple definitions. In 2006, in her *Communications of the ACM* article, she suggested that computational thinking, the notion of learning to think like a computer scientist, would benefit everyone and not just those in the field of computer science (Wing, 2006). In 2010, Wing, along with Jan Cuny and Larry Snyder, defined computational thinking as "the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (Cuny, Snyder, & Wing, 2010, p. 1). Sussman defined computational thinking as a means of determining a specific set of instructions for doing things, which includes the means for managing complexity and finding ways to automate tasks (National Research Council, 2010). Later, Wing and Sussman suggested that computational thinking is a "bridge between computer science and engineering," a means of thinking about solutions for problems that encompass multiple disciplines (National Research Council, 2010; p. 12). Bogost stated that computational thinking is about the understanding of the processes and procedures used to solve problems and not how they might actually be programmed into a computer (Vee, 2013). Lee et al. (2011) argued that computational thinking broadens the capabilities of people through the use of abstract tools for managing complexity and the automation of tasks. Owen Astrachan offered a more profound viewpoint in how he perceives the goals of computational thinking. He argued:

> Computational literacy will allow civilization to think and do things that will be new to us in the way that the modern literate society would be almost incomprehensible to preliterate cultures, but it's a different kind of literacy than what it means to be familiar. By computational literacy, I do not mean a casual familiarity with a machine that computes. (National Research Council, 2010, p. 14)

In 2009, the Computer Science Teachers Association (CSTA) and the International Society for Technology in Education (ISTE), together with funding from the National Science Foundation, started a project to create an operational definition of computational thinking for K-12 education. They concluded that computational thinking is a problem-solving process that includes the ability: (a) to express solutions in a manner that allows for the use of computers to solve them, (b) to logically organize and analyze data, (c) to define relevant data to a process using abstractions, and (d) to identify an ordered sequence of steps that can be used to solve a problem (Computer Science Teachers Association and the International Society for Technology in Education, 2011). Virginia Tech (VT) believed that computational thinking offered a skill set valuable to all of its students and recently added a computational thinking component to their general education requirements. As part of this process, VT defined computational thinking as

"The intellectual skills rooted in the ability to conceive of meaningful information-based representations that can be effectively manipulated using an automated agent (e.g. a computer)" (Kafura, Bart, & Chowdhury, 2015).

For the purpose of this study, computational thinking will be defined as the ability to address a problem with the idea that it *could* be solved using a computer, but not that it necessarily will be solved using a computer. The basic skills used in computational thinking are the ability to accurately and fully describe a problem, breaking the overall problem down into a sequence of smaller problems (decomposition), removing unnecessary details (abstraction), recognizing where a solution already exists for a part of the overall solution (pattern matching), and constructing, and possibly testing/debugging, a solution for the overall problem.

Goals of Computational Thinking

Given the range of definitions for computational thinking, it seems reasonable that there are numerous sets of expectations or goals students take away from their experiences with computational thinking. Authors of the report *Computer Science K-8: Building a Strong Foundation* (Stephenson & Barr, 2012) suggested that computational thinking may improve or enhance a student's ability in the following areas:

- confidence in dealing with complexity,
- persistence in working with difficult problems,
- tolerance for ambiguity,
- the ability to deal with open ended problems, and
- the ability to communicate and work with others to achieve a common goal or solution. (p. 5)

The report also identifies five reasons why computational thinking may be critical for K-5 students:

- Improves thinking capabilities – the ability to understand the problem and systematically describe a solution.
- Helps to develop future creators and innovators – the next generation of scientists and engineers will need to be adaptive and use the computer as a problem-solving device, regardless of their field of expertise.
- Empowers students – Early success in creating problem solving tools can help promote the notion that the computer can be used in relevant and meaningful ways.
- Lays the foundation for future use – the earlier students are exposed to basic computational thinking concepts, the more likely they are to include these skills in future challenges.
- Collaboration and teamwork – As the world's problems become increasingly complex, they will require creative solutions incorporating multiple perspectives. This will demand that students are capable of working in teams, or in group projects. The skills introduced in computational thinking include allowing for the division of labor, allowing numerous students to work in parallel toward a solution (CSTA, 2012).

The CSTA report "Running on Empty: The Failure to Teach K-12 Computer Science in the Digital Age" (Wilson, Sudol, Stephenson, & Stehlik, 2010) suggested standards or a framework for teaching designed to meet the following goals:

- Students should understand the nature of computer science.
- Students are exposed to the basics of computer science.
- Students should be able to apply computer science skills in other subjects.
- K-8 students should learn basic computer science skills and develop the ability to integrate those skills with algorithmic thinking.

In 2016, the Advanced Placement (AP) Program administered by The College Board launched a new AP® offering, the AP® Computer Science Principles (AP CSP) course. During its initial year, over 2,500 schools offered the class, with over 50,000 secondary students taking the final exam. The AP CSP course has goals that include an introduction to the concept of computer science, and it exposes students to how computer and technology can be used to change the world (The College Board, 2019a). Specifically, the AP Computer Science Principles course focuses on how to apply the "seven big ideas," the "core computer science knowledge and capabilities" ("Seven Big Ideas of Computer Science," n.d., para. 1) that an educated citizen should know and understand:

1. Computing is a creative human activity that engenders innovation and promotes exploration . . .
2. Abstraction reduces information and detail to focus on concepts relevant to understanding and solving problems . . .
3. Data and information facilitate the creation of knowledge . . .
4. Algorithms are tools for developing and expressing solutions to computational problems . . .
5. Programming is a creative process that produces computational artifacts . . .

6. Digital devices, systems, and the networks that interconnect them enable and foster computational approaches to solving problems . . .
7. Computing enables innovation in other fields including science, social science, humanities, arts, medicine, engineering business. ("Seven Big Ideas of Computer Science," n.d., paras. 2-8)

Google for Education (2015) has offered its own set of concepts introduced and developed through training computational thinking:

- **Abstraction:** Identifying and extracting relevant information to define main idea(s)
- **Algorithm Design:** Creating an ordered series of instructions for solving similar problems or for doing a task
- **Automation:** Having computers or machines do repetitive tasks
- **Data Analysis:** Making sense of data by finding patterns or developing insights
- **Data Collection:** Gathering information
- **Data Representation:** Depicting and organizing data in appropriate graphs, charts, words, or images
- **Decomposition:** Breaking down data, processes, or problems into smaller, manageable parts
- **Parallelization:** Simultaneous processing of smaller tasks from a larger task to more efficiently reach a common goal
- **Pattern Generalization:** Creating models, rules, principles, or theories of observed patterns to test predicted outcomes

- **Pattern Recognition:** Observing patterns, trends, and regularities in data
- **Simulation:** Developing a model to imitate real-world processes (para. 5)

People at the Center for Computational Thinking at Carnegie Mellon, the institution where Jeannette Wing was a professor when she wrote her seminal paper on computational thinking, expanded their description of computational thinking to go beyond generalized computing skills to include many human elements (Center for Computational Thinking, n.d.). People at the Center described computational thinking as:

- "A way of solving problems, designing systems, and understanding human behavior" (para. 2).
- Creating and using "different levels of **abstraction**" (para. 3).
- Thinking **algorithmically** such that mathematics concepts can be appropriately applied to design secure, efficient, and fair solutions.
- "Understanding the consequences of **scale**" (para. 5) from the viewpoint of efficiency as well as economic and social perspectives.

Common Ground and Context

While there are numerous perspectives on skills introduced through the use of computational thinking, they all share common ground. There also needs to be a context in which to view various definitions and descriptions of computational thinking. Given the context of students in Grades K-5, a goal is to identify traits useful and important in helping students learn how to succeed in the digital world.

For this K-5 context, the "big picture" goal of computational thinking is to help students learn to address problems with the notion that they *could* be solved using a computer, but do not necessarily *have* to be solved only by using a computer. It is critical that this study's working definition for computational thinking meet the following goals:

1. Must be generalizable into most, if not all subjects;
2. Must be able to integrate into current curricula; and
3. Must be understandable by K-5 teachers, the majority of whom have had little to no experience with computing or computation thinking.

Therefore, common traits found throughout most, if not all, of the various perspectives considered most practical and useful in meeting this "big picture" goal include:

- Understanding a problem – Collecting data and then analyzing the data to fully understand the problem.
- Decomposition – Breaking the overall problem down into a sequence of smaller, simpler tasks.
- Abstraction – Removing unnecessary details such that a solution can be used to solve all tasks of the problem of a given type. Abstraction generalizes a problem, removing unnecessary complexity, allowing one solution to be used to solve many similar issues.
- Pattern matching - Looking for patterns, to see if any of the smaller, simpler tasks of our problem have already been solved.
- Algorithmic thinking – Constructing a solution, a series of parallel and/or sequential steps that can be used to solve the problem. If possible, building a model to test, debug and refine the solution.

Not a New Concept in K-5 Education

The notion of introducing computational thinking in K-5 education is not new. Seymour Papert, in the foreword of his book *Mindstorms: Children, Computers, and Powerful Ideas*, stated:

> The computer is the Proteus of machines. Its essence is its universality, its power to simulate. Because it can take on a thousand forms and can serve a thousand functions, it can appeal to a thousand tastes. This book is the result of my own attempts over the past decade to turn computers into instruments flexible enough so that many children can each create for themselves something like what the gears were for me. (Papert, 1980; p. viii)

Papert saw a computer as a tool that would allow students to take a constructivist approach to learning. Papert was a pioneer in teaching children computational thinking through the use of LOGO programming to control a small, turtle-looking robot (Papert, 1980).

Even before Papert, Alan Perlis, a founding father of the field of computer science and the first recipient of the ACM A. M. Turing Award, argued the need for students to learn programming and a "theory of computation" (Guzdial, 2008, p. 25). In a sarcastic reference to the state of computer science in public education, Perlis stated, "It goes against the grain of modern education to teach children to program. What fun is there in making plans, acquiring discipline in organizing thoughts, devoting attention to detail and learning to be self-critical?" (Perlis, 1982, p. 13).

Using Computer Science to Learn Computational Thinking

Computational thinking is a strategy that can be used to address, understand, and then identify a solution for a problem. Computational thinking approaches a problem with the idea that it could be solved using a computer, but not that it must, or even will, be solved using a computer. Computational thinking is about the ideas used to address a problem, and not about the software and hardware that ultimately *could* be used to implement a solution. It is a means of ensuring that there is a clear understanding of a problem, breaking the problem into a sequence of smaller problems (decomposition), removing unnecessary details (abstraction), determining if any of the smaller problems in the sequence already have a solution (pattern matching), and building a series of solutions to each of the smaller problems that together will solve the larger problem (algorithms).

While this sequence of actions is common place when designing computer programs, these actions are by no means limited to use within the field of computer science. This overall sequence of actions, while designed with computers in mind, bears similarities to steps taken to construct a Gantt chart for project scheduling, or the PERT (project evaluation and review technique) method, commonly used in project management. The concept behind each individual step is not unique to the field of computer science. Decomposing problems, commonly known as divide-and-conquer in computer circles, is a concept likely to be taught in an economic, government, or history class. Steps taught to implement long division constitute an algorithm, a series of steps that may be repeated as necessary until the solution has been determined. Anytime a mathematical function is used more than once in a formula, the user has broken down the overall problem into several smaller problems, and then identified steps where there is reuse of functionality.

Computational thinking and computer programming are quite often confused. In practice, the phrases are often interchanged, and while not correct, often either phrase can be substituted for the other without effecting any discernible confusion in the meaning of a sentence. However, in reality they have very different meanings.

While computational thinking does not require working a computer, it is generally manifested in the act of computer programming. Computer programming, at least initially, is a most convenient means of implementing the act of computational thinking. Given that computational thinking is an idea or concept for approaching a problem, there must be a concrete means of testing and refining the process. The best way to adequately implement and test computational thinking skills is to physically apply those skills to an actual problem that requires a concrete solution.

Computational Thinking Versus Computer Programming, or Coding

The process used in programming a computer allows for a means of applying, testing, verifying, and refining one's computational thinking skills (Krauss & Pottsman, 2016; Kynigos & Grizioti, 2018). When using the agile software development framework, a popular set of methods and practices used to guide software development, programmers work through a cycle of steps: planning, building (programming), implementing (testing), and feedback (analyzing the results). Once a cycle is complete, anything learned from the previous cycle is used in the next cycle to improve the process. The critical first step in every iteration is planning, where decisions are made that dictate the direction of the building process. Implementation and feedback loops are means of using and testing a program. The significant "first step" has direct bearing on all future actions in the program. Incorrectly identifying an underlying problem all but guarantees remaining steps will fail. Improper analysis of a problem is sure to lead to an improper solution for the given problem. Lack of effort in the planning phase is so common place there is a reoccurring joke in computer science that states, "Weeks of programming can save you hours of planning." Computational thinking is what comes before programming. Computational thinking is the skill set that should be utilized in the hours of planning before any work with a computer is undertaken, to avoid unnecessary weeks of programming.

Why Use Computer Science to Learn Computational Thinking Skills?

Computer programming is a testing process for computational thinking. Computer programming allows for the physical implementation of one's mental model of a solution. Computer programming generates a feedback loop that allows one to constantly adjust their evaluation processes, where each iteration of a process can provide a user with information that can be used to improve a course of action to be taken in the next cycle (Guzdial, 2019; Kynigos & Grizioti, 2018).

Programming a computer is not the only means of validating mental processes learned in computational thinking, but it does provide a convenient, consistent, and immediate response from an unnuanced device that will do exactly, and nothing more, what it is told (Guzdial, 2019; Krauss & Pottsman, 2016). Computers are capable of doing only what they are told to do. Computers have no residual knowledge; they do not remember anything from a previous attempt at solving a problem. They start fresh every time a student tries to implement a solution.

To illustrate this point, when performing outreach activities in local elementary schools, I sometimes mention that computers are quite dumb, and then ask students if they believe me. In a room filled with students most likely very experienced using the Xbox or PlayStation, or even an average smartphone, they are used to seeing all of the amazing games that can be played on modern computing devices. I mention that an electronic calculator is a computer, although somewhat limited in its capabilities. I suggest that they would not necessarily need a calculator to perform simple multiplications, but we will for "this example." I state that I want to multiply three times five, so I enter three, then hit the plus key, then I enter five, followed by the equals sign, and the calculator then displays eight for an answer. I then ask the class if the calculator displayed the correct answer. This generally brings up a lively conversation on whether the answer is correct. This provides me an opportunity to stress that: (a) yes, three plus five is eight, and I did receive the correct answer to the equation entered; (b) I entered the wrong equation into the calculator because I wanted to multiply two numbers, not add; and (c) there is no way for the calculator to recognize my mistake. Computers will only do what they are told. They do not "know" about my intentions. If I want correct output from a computer, I have to provide it with correct input(s).

Why Computational Thinking Rather Than Mathematics Education to Teach Problem Solving Skills?

Although there are similarities between computational thinking and mathematics education, computational thinking can extend mathematics in a unique way (Lee et al., 2011). Wing (2008) stated that computational thinking is "a kind of analytical thinking, and it shares with mathematical thinking for problem solving," but each computing area has its own methodologies, which do not necessarily require knowledge of the underlying mathematical systems. Another key difference, the ability to use computational tools to automate solutions is an essential component in computational thinking. Denning and Freeman (2009) stated, "It (CT) is distinctively different because of its central focus on information processes" (p. 30).

The National Council of Teachers of Mathematics (NCTM) offers five process standards: Problem Solving, Reasoning & Proof, Communication, Connections, and Representation. While these processes are similar to the goals of computational thinking, the NCTM also offers five content areas in which these standards are to be addressed: Number & Operations, Algebra, Geometry, Measurement, and Data Analysis & Probability (National Council of Teachers of

Mathematics, 2019). Where mathematical thinking “involves the application of math skills to solve math problems, such as equations and functions” (Shute, Sun, & Asbell-Clarke, 2017), computational thinking is a process for addressing real-world problems that require solutions which go beyond the use of standard mathematics.

Summary

In Chapter II, various definitions of computational thinking were discussed, as well as the rationale for a lack of consensus regarding a formal definition for computational thinking. To that end, a definition appropriate for the context of this study was presented. This definition included understanding a problem, clearly defining the problem at hand; abstraction, removing unnecessary details from a problem; decomposition, breaking down a complex problem into smaller, more manageable parts; pattern matching, looking for similarities, both within a sub-problem, or from previously solved sub-problems; and algorithmic thinking, developing a series of steps to solve the problem. In Chapter III, methods to be used in this study will be presented.

CHAPTER III
METHODOLOGY

Introduction

Outlined in this chapter are the theoretical framework, the methodology, and methods used to explore and explain barriers elementary school teachers may find toward implementation of computational thinking in their classrooms. Research tools and strategies have been selected to direct this study in addressing the following research questions:

1. What are educators' levels of understanding in regard to the definition of and opportunities to utilize computational thinking?
2. What are educators' levels of interest and comfort surrounding computational thinking?
3. What are educators' beliefs in regard to the importance of computational thinking in the classroom and their careers?
4. What are educators' beliefs regarding the use of computational thinking in specific K-5 content areas of English, mathematics, social studies, and science?

Survey Instrument

A survey instrument (Appendix A) was created by the researcher after a review of the literature. While the initial search for an instrument focused on evaluating an educators' computational thinking skills, it was determined that the focal point of the research should address educators' attitudes and beliefs regarding teaching of computational thinking, and not their personal skill set in performing computational thinking skills.

The instrument contained four parts. In Part I, respondents were asked if they were familiar with the concept of computational thinking. If a respondent indicated yes, they were asked to explain the concept of computational thinking. If the respondent indicated no, they are not familiar with the concept of computational thinking, they were asked what they thought it means. By asking for a respondents' perceptions on computational thinking before introducing any additional survey questions, the users' responses were gathered before being affected by any potential bias introduced by the remaining questions on the survey. This ordering was done to ensure a common ground, and to provide each respondent with a consistent basis when asked for their input.

In Part II of the survey, respondents were asked to respond to 24 Likert questions. The Likert scale used had a rating scale ranging from *strongly disagree* to *strongly agree*. Likert and Likert-type scales are commonly used in research because they make it easy to receive numerous responses in a relatively short amount of time (Johns, 2005). Likert scales allow respondents to go beyond a simple yes or no response, allowing a researcher to measure level or strength of an opinion in a response. In order to force a respondent into having an opinion, and to promote cognitive thought in the decision-making process, the neutral response sometimes present in Likert scales was removed, therefore eliciting a specific positive or negative response for each question (Johns, 2005).

Survey questions in Part II were based upon the Computing Attitude Questionnaire (CAQ). The CAQ was designed for use in a study by Yadav, Mayfield, Zhou, Hambrusch, and Korb (2014) entitled "Computational Thinking in Elementary and Secondary Teacher

Education." Permission was received from Dr. Aman Yadav to use his survey, and to modify as needed.

The CAQ contained twenty-one Likert-type questions organized into five categories: *definition*, containing four questions about participants' understanding of computational thinking; *comfort*, with six questions addressing participants' comfort levels regarding computers and computing; *interest*, with four questions about participants overall interest in computer science; *classroom*, with two questions on the future use of computational thinking in participants' future, and *career*, with five questions about how participants felt computational thinking may influence their careers. Since the initial survey was created for use with preservice teachers, a few questions were reworded or replaced to make them appropriate for use with working educators. Three additional questions were added, two to the *career* category and one to the *classroom* category to further explore educators' attitudes and beliefs derived from their classroom experiences. The statements, delineated by their categories, are listed in Table 1.

Part III asked each respondent their opinion (*don't know*, *disagree*, *agree*) about the use of computational thinking when teaching basic subject areas in an elementary school curriculum. For this study, basic areas were defined as English, mathematics, social studies, and science. In addition to their Likert-type quantitative response, respondents were asked to explain their answer for each subject area. These responses described where teachers felt they could introduce computational thinking into their classrooms, as well as adding insight into where professional development instructors could improve their message on how and where computational thinking can be used in classrooms. Finally, each respondent was asked to provide any positive or negative feedback they wished to offer. This open-ended question, with little provided direction, allowed respondents an opportunity to provide feedback they did not feel fit into prior questions.

Table 1

Survey Statements Grouped by Categories

Categories	Statement
Definition	• Computational thinking involves using computers to solve problems • Computational thinking involves thinking logically to solve problems • Computational thinking is understanding how computers work • Computational thinking involves abstracting general principles and applying them to other situations
Comfort	• I do not think it is possible to apply computing knowledge to solve other problems • I am not comfortable with learning computing concepts • I can achieve good grades (C or better) in computing courses • I can learn to understand computing concepts • I do not use computing skills in my daily life • I doubt that I have the skills to solve problems by using computer applications
Interest	• I think computer science is boring • The challenge of solving problems using computer science appeals to me • I think computer science is interesting • I will voluntarily take computing courses if I were given the opportunity
Classroom	• Computational thinking can be incorporated in the classroom by using computers in the lesson plan • Computational thinking can be incorporated in the classroom by allowing students to problem solve • Knowledge of computing will allow me to [be] more effective in the classroom
Career	• My career goals do not require that I learn computing skills • I expect that learning computing skills will help me to achieve my career goals • I hope that my future career will require the use of computing concepts • Having background knowledge and understanding of computer science is valuable in and of itself • Trying to introduce computing concepts will distract students from the task at hand • Using computing technologies can increase student interest in the material • I worry that my students will be more adept at using computing technologies than I am

Part IV of the survey instrument asked respondents basic information regarding their classrooms and their overall teacher experiences. Teachers were asked to provide information on the grade they were teaching, number of years of experience, both overall and in their grade level at the time they completed the survey, their highest earned degree, as well as to optionally indicate their gender.

Description of Population

Some elementary school teachers in North Dakota public schools were participants in this study. The survey was only offered to core content area educators, teachers in a specialty field such as physical education or foreign languages were not included in the study. Superintendents in the several school districts in North Dakota were contacted regarding the study. The largest school districts (Bismarck, Devils Lake, Dickinson, Fargo, Grand Forks, Jamestown, Mandan, Minot, Valley City, Watford City, West Fargo, Williston, and Wahpeton) in the state were selected for convenience, to maximize the number of potential participants per contact. A few smaller school districts (Grafton, Hillsboro, Larimore, and Northwood) close to the Grand Forks area were selected due to proximity or because of personal contacts the researcher had within school administrations. Several schools (Bismarck, Fargo, West Fargo) required the researcher to complete an online form requesting permission to conduct research. Once initial contact was made with either a superintendent or an authorized contact, an email was sent containing an abstract about the study, survey questions (in Word format for easier reading and the format provided by Qualtics, which allows for viewing the format and structure of the survey), and the Institutional Review Board (IRB) approved recruitment letter. For any school that did not respond to the initial request to conduct this study, a follow-up request was made one to two weeks after the initial email. Numerous schools chose not to participate or did not respond to phone calls or email requests (Bismarck, Devils Lake, Hillsboro, Mandan, Minot, Valley City, Watford City, and West Fargo). Reasons administrators gave for choosing not to participate, when offered, included time required of a teacher to complete the survey, a desire to limit the number of requests presented to teachers at that time, the demand on local resources to process the request, and a desire to limit research opportunities to only researchers from local school districts.

Once permission was granted from a school, including receiving a letter of permission from the school district allowing the researcher to conduct research with their educators, an email was sent to a school contact at each participating school for distribution to their teachers. Several schools sent out an email directly to their educators (Dickinson, Grafton, Larimore, Jamestown, Northwood, and Wahpeton). The Grand Forks school district sent out an email request to individual schools' principals for distribution of the survey. Given the anonymity of the process, there was no means of ensuring how many principals sent out the recruitment email to their teachers. The Fargo school district does not send out recruitment emails to their teachers; rather, they post an announcement on a website accessible by their teachers. Again, there was no way to determine how many educators from the Fargo school district read the recruitment offer. Based upon information provided by school districts, 1,111 teachers could have either received the recruitment email (751 teachers) or viewed the recruitment posting on a teacher website (360 teachers).

One hundred and sixteen participants responded to the survey (10% of the potential participants). Five of the initial participants chose not to take the survey, declining to accept the conditions presented in the Informed Consent landing page. Of the remaining 111 participants, 56 completely finished the survey (5% of the potential participants).

Methodology

Cross-sectional research is a methodology for conducting research using different groups of people who differ in one or more variables of interest, but share other characteristics. Cross-sectional survey design is “the most popular form of survey design used in education” (Creswell, 2012, p. 377). A cross-sectional survey allows a researcher to make inferences about a population without interfering or interacting with participants (Creswell & Creswell, 2017; Lavrakas, 2008). Cross-sectional surveys provide snapshots of variables at a specific point in time. One advantage of cross-sectional data is its usefulness in exploratory research as a preliminary step in creating hypotheses for future research (Sedgwick, 2014).

Cross-sectional research generally utilizes quantitative data. Quantitative data includes any data that can be measured numerically and on which statistical analyses can be performed. Quantitative data can be placed into categories, or ranked in order, or measured in units of measurement. Quantitative data can be collected through the use of instruments, such as surveys or questionnaires.

To collect more details, and to give respondents more freedom in their responses, several open-ended questions were used in this study. By including qualitative data gathered from open-ended questions, researchers can address their research questions from a broader perspective. Gathering data using multiple styles can allow a researcher to achieve a better understanding of a problem than if using only quantitative data and provides a researcher opportunities for additional depth and breadth of knowledge.

Coding

Coding is the process of breaking down data into a consistent, usable format prepared for data analysis. Elliot (2018) stated that coding is “essentially indexing or mapping data, to provide an overview of disparate data that allows the researcher to make sense of them in relation to their research questions" (p. 2850). Coding allows a researcher to make sense of the data in relation to research questions. Creswell (2016) noted that "coding is the process of analyzing qualitative text data by taking them apart to see what they yield before putting the data back together in a meaningful way" (p. 156). Coding allows a researcher to filter out of the text what is on-topic and remove the off-topic or irrelevant text, condensing data down to relevant, succinct points.

Given the exploratory nature of this study, there was no preexisting framework to use when analyzing the data. To the best of the researcher's knowledge, an a priori set of codes did not exist. And if such a set of codes did exist, Creswell (2013) noted a priori codes "serve to limit that analysis to the 'preconfigured' codes rather than opening up the code to reflect the view of the participants" (p. 185). Therefore, a grounded theory approach was used to code participants' responses. This method of coding ensures that codes are derived directly from the data, and not from pre-existing theories or beliefs. Open coding was used to identify emerging codes from participants' statements. All open-ended responses were coded using the same process throughout the entire study.

Hypothesis

It is anticipated that there will be a limited amount of interest in introducing a computational thinking component into curricula as well as participating educators reporting little value in introducing a computational thinking component into their personal curriculum. Educators face a variety of potential causes; any additions to their curricula, especially an addition not required by state standards, are likely to be seen as of little importance. There may also be concerns regarding the time and effort necessary when preparing to teach new content. Lack of support, both technical and from school administrations, may also be of concern.

It is expected that there will be a strong positive correlation between teachers with high levels of computer confidence and educators' interest in teaching computational thinking and the value teachers place on computational thinking. Likewise, it is also expected that there will be a strong positive correlation between how confident a teacher feels while in their classroom and both interest and value participants place on introducing new content, in this case, computational thinking.

Summary

In this chapter, the methods of the proposed study have been provided. Next, in Chapter IV, the results will be presented, and in Chapter V, results will be discussed.

CHAPTER IV

RESULTS

This study examined teachers' attitudes and beliefs regarding implementation of computational thinking into their classrooms. Specifically, this study examined participating educators' understandings of the definition of computational thinking, their level of interest and comfort with computational thinking, how important they perceive computational thinking to be, and where or how they feel they can use computational thinking in their classrooms.

Data were collected utilizing a researcher-designed survey based upon the Computing Attitude Questionnaire survey (Yadav et al., 2014). Quantitative data solicited from closed-ended questions where responses were limited to Yes/No responses, Likert scale responses (*strongly disagree*, *disagree*, *agree*, *strongly agree*), and Likert-type scale responses (*don't know*, *disagree*, *agree*) were analyzed using measures of central tendency and descriptive statistics. Qualitative data solicited from open-ended questions were analyzed using thematic analysis. This chapter presents an analysis of the data collected from participating educators.

Research Questions

The overall theme of this research was to investigate teachers' overall perceptions of computational thinking in order to gain insight into their knowledge of what computational thinking is and their overall impressions about introducing computational thinking into their classrooms. Four specific research questions were explored in this study:

1. What are educators' levels of understanding in regard to the definition of and opportunities to utilize computational thinking?
2. What are educators' levels of interest and comfort surrounding computational thinking?
3. What are educators' beliefs with regard to the importance of computational thinking in the classroom and their careers?
4. What are educators' beliefs regarding the use of computational thinking in specific K-5 content areas of English, mathematics, social studies, and science?

The study also examined participants' demographic data including: gender; grade they were teaching at the time of this study; number of years of teaching experience, both overall and in the grade they were teaching at the time of this study; and their highest earned degree.

Participants

One hundred and sixteen teachers initially responded to the survey request. Of those teachers, fifty-six completely finished the survey. The participants were all elementary school core content area educators in a North Dakota public school. Teachers in a specialty area, such as physical education, art, or foreign language were not included in the study.

Survey Instrument

The survey instrument contained four parts. Part I asked respondents if they were familiar with the concept of computational thinking, and to explain what they thought it means. Part II asked respondents 24 Likert questions, separated into five categories: definition, comfort, interest, classroom, and career. Several questions in Part II were reverse coded. In the presentations contained in this chapter, reverse coded questions are indicated by a trailing *RC*. Reverse coded questions are used as a means of addressing acquiescence response bias.

Acquiescence bias refers to the tendency of a respondent to agree or provide a positive response to all questions. The negatively worded questions were reverse scored prior to evaluation of the remaining data. Part III asked respondents their opinions about the use of computational thinking in areas of English, mathematics, social studies, and science. Part III also included an open-ended question, allowing the respondent the opportunity to provide any feedback. Part IV asked respondents several demographic questions, including amount of teaching experience, current grade level, and level of educational.

Descriptive Statistics

Descriptive statistics were calculated for the survey. Information requested from participating teachers included their gender, the grade they were teaching at the time of the study, number of years of experience they had teaching the grade they were teaching when they filled out their survey, number of total years of teaching experience, and their highest earned degree.

The majority of the participants, 88% (*n* = 49), reported they were female, with the remaining 13% (*n* = 7) reporting as male. This is consistent with the overall percentage of female primary school teachers in the United States at 87% (UNESCO Institute for Statistics [UIS], n.d.). Over half the participants (56%, *n* = 30) reported having earned a Bachelor's degree, with the remaining participants (44%, *n* = 24) reporting having earned a Master's degree. None of the participants reported earning a doctoral degree.

The data found in Table 2 suggests that the majority of participants taught at higher levels of elementary school with 72% (*n* = 28) reporting to be teaching the intermediate grades (third grade through fifth grade). An additional eight educators reported teaching second grade (21%). Only three educators (8%) reported teaching kindergarten or first grade.

Table 2

Grade Level Being Taught At Time of Study

Grade	Count	%
K	0	0
1	3	8
2	8	21
3	11	28
4	7	18
5	10	26

Data found in Table 3 suggests there was a wide variability in the experience of participants. Teachers with 1-5 years of experience and teachers with 21 or more years of experience were the groups which made up the highest percentage participants, both at 25% (*n* = 14). Teachers with 11-15 years of experience were the group accounting for the smallest percentage of participants at 15% (*n* = 8).

Table 3

Years of Teaching Experience, Including Grade Taught at Time of Study

Years	Years Teaching Grade at Time of Study		Years of Total Teaching Experience	
	Count	%	Count	%
1-5	14	25	27	54
6-10	10	18	13	26
11-15	8	15	6	12
16-20	9	16	2	4
21+	14	25	2	4

Research Question 1

To address Research Question 1 (*What are educators' levels of understanding in regard to the definition of and opportunities to utilize computational thinking?*), teachers were initially presented with a yes/no question asking, "Are you familiar with the concept of computational thinking?" (Table 4).

Table 4

Number of Participants Familiar With the Concept of Computational Thinking

Yes	15
No	41
Total	56

Each teacher was then asked a follow-up question, either "How would you explain the concept of computational thinking?" if they responded in the affirmative to being familiar with the concept of computational thinking or "What do you think computational thinking means?" if they responded in the negative.

Open-ended responses were coded after the survey was closed. Responses were grouped by those who answered Yes and those who answered No to the *Are you familiar with the concept of computational thinking?* question. Then significant statements were identified followed by coding analysis. Nonsensical or irrelevant responses were not coded. To improve the dependability of the coding, the entire coding process was independently conducted twice, from start to finish. Results from both attempts were then reconciled into a single document.

Of the 15 *Yes* responses, 14 responded to "How would you explain the concept of computational thinking?" with one participant not responding. Responses were coded and categorized for further analysis. Some responses included multiple codes, so total number of codes will not equal number of responses. The following categories in Table 5 were derived from teachers' comments.

Table 5

Categories Found in Comments Following a Yes Response to: Are You Familiar With the Concept of Computational Thinking?

Category	*n*	%
Problem solving	13	93
No reference to a computer	8	57
Use of a computer	3	21
Includes human or computer	3	21

N = 14

Problem solving – Problem solving is the process of using an approach to define a problem, identifying possible solutions, and then selecting, implementing, and testing solutions. Examples from participants' responses included:

1. "The systematic and process of thinking from beginning to the end of a set of tasks."
2. "Logical thinking process on sequential order and logic decisions impacting direction."

No reference to a computer – All of these responses included meaningful comments, yet made no reference to a computer or computing device. All responses in this category also included a reference to problem solving. Examples from participants' responses include:

1. "Not 100% confident, but my understanding would be that computational thinking is arranging steps to come to a solution (solve a problem) or creating various steps for a solution."

2. “A process of problem solving, coming up with solutions.”

3. “The systematic and process of thinking from beginning to the end of a set of tasks.”

Use of a computer – All of these responses made a specific reference to the use of a computer to complete tasks or implement solutions without any specific reference to programming or coding. Examples include:

1. “It is the thought process of solving problems or programming a computer to solve problems.”

2. “Solve (or create) a problem that a computer can be programmed to do.”

Includes human or computer – These responses specifically referred to computational thinking as a process which could be carried out by a computer OR a human. Examples include:

1. “Having a problem and thinking of solutions, like a computer or human would carry out.”

2. “Seeing a problem and identifying solutions that can be effectively carried out by human or machine.”

Of the 41 *No* responses, 38 responded to the open-ended question of “What do you think computational thinking means?” Responses were coded and categorized for further analysis. Some responses included multiple codes, so total number of codes did not equal number of responses. Categories resulting from analysis of data are listed in Table 6 and described in the text following Table 6.

Table 6

Categories Found in Comments Following a No Response to: Are You Familiar With the Concept of Computational Thinking?

Category	*n*	%
Problem solving	17	45
Use of a computer	15	39
Math / mathematical thought	11	29
No idea	3	8
Programming	3	8

N = 38

Problem solving – Problem solving is the process of using an approach to define a problem, identifying possible solutions, and then selecting, implementing, and testing solutions. Examples from participants' responses include:

1. "Problem solving visual and sequential problems with different possible solutions."
2. "Thinking in a way that solves a problem or analyzes data."

Use of a computer – All of these responses made a specific reference to the use of a computer to complete tasks or implement solutions without any specific reference to programming or coding. Examples include:

1. "Computer knowledge and using it for planning projects and assignments."
2. "Solving problems with computers in mind."

Math / mathematical thought – These responses made reference to mathematics or performing calculations.

1. "I believe it would have something to do with quantifying something, or calculations."
2. "Using mathematics or technology in some way."

Programming – These responses made reference to programming or coding. Examples include:

1. "The ability to create language for computer programming?"
2. "I think it is a problem-solving skill that helps develop computer programs."

No idea – These responses, as opposed to not answering, specifically stated that they did not know the definition of computational thinking.

To further explore teachers' understandings of the definition and meaning of computational thinking, each participant was then presented with five survey questions using a four point Likert scale. Responses included *Strongly Disagree* (SD), *Disagree* (D), *Agree* (A), and *Strongly Agree* (SA). Please see Table 7.

Responses for all five questions fell into two groupings, those which nearly unanimously agreed with the statement (chose either *Agree* or *Strongly Agree*) and those in which roughly two-thirds of the participants agreed (*Agree* or *Strongly Agree*) with the statement, with the majority of the remaining responses selecting *Disagree.* All five questions had high percentages of respondents choosing *Agree* (52% to 73%). Questions 1 and 3 both showed respondents choosing *Disagree* scoring in the 30-percentile range (Q1 – 34%, Q3 – 39%). Questions 2, 4, and 5 all showed significant responses of *Strongly Agree* (Q2 – 30%, Q4 – 21%, Q5 – 36%), with none of the questions receiving more two *Disagree* responses. None of the questions received more than one response of *Strongly Disagree.*

Questions with a trailing RC were negatively worded. Reverse coding is a common validation technique used to rephrase a positive item in a negative manner. All reverse coded questions were reverse scored prior to evaluating the data.

Table 7

Teachers' Perceptions on the Definition of Computational Thinking

Survey Statement	SD	D	A	SA
1. Computational thinking involves using computers to solve problems (*N* = 56)	0 (0%)	19 (34%)	32 (57%)	5 (9%)
2. Computational thinking involves thinking logically to solve problems (*N* = 56)	0 (0%)	1 (2%)	38 (68%)	17 (30%)
3. Computational thinking is understanding how computers work (*N* = 56)	1 (2%)	22 (39%)	29 (52%)	4 (7%)
4. Computational thinking involves abstracting general principles and applying them to other situations (*N* = 56)	0 (0%)	2 (4%)	41 (73%)	12 (21%)
5. I do not think it is possible to apply computing knowledge to solve other problems (*N* = 56) . . . . RC*	1 (2%)	2 (4%)	33 (59%)	20 (36%)

Note. SD = *Strongly Disagree*, D = *Disagree*, A = *Agree*, and SA = *Strongly Agree*.
* RC means reverse coded.

Research Question 2

To address Research Question 2 (*What are educators' levels of interest and comfort surrounding computational thinking?*), teachers were presented with nine questions using a four point Likert scale with the responses of *Strongly Disagree*, *Disagree*, *Agree*, and *Strongly Agree*. The first five questions were used to evaluate comfort levels of teacher participants regarding computing concepts and their beliefs regarding basic computing skills (Table 8).

Table 8

Comfort Levels and Beliefs of Participants Regarding Computing Concepts

Survey Statement	SD	D	A	SA
6. I am not comfortable with learning computing concepts (*N* = 56) . . . . RC	4 (7%)	5 (9%)	34 (61%)	13 (23%)
7. I can achieve good grades (C or better) in computing courses (*N* = 56)	0 (0%)	3 (5%)	43 (77%)	10 (18%)
8. I can learn to understand computing concepts (*N* = 56)	0 (0%)	3 (5%)	41 (73%)	12 (21%)
9. I do not use computing skills in my daily life (*N* = 56) . . . . RC	0 (0%)	1 (2%)	35 (63%)	20 (36%)
10. I doubt that I have the skills to solve problems by using computer applications (*N* = 56) . . . . RC	0 (0%)	4 (7%)	36 (64%)	16 (29%)

Note. SD = *Strongly Disagree*, D = *Disagree*, A = *Agree*, and SA = *Strongly Agree*.
* RC means reverse coded.

While computing concepts do not equate to computer science or computational thinking, use of a computer is the most common and practical means of introducing computational thinking skills. Given the interconnected relationship between computational thinking and computing, it seemed prudent to ask educators about their beliefs regarding computing concepts. Therefore, the last four questions were used to evaluate teachers' interest levels in computing concepts.

All five questions designed to evaluate a teacher's comfort with computers and computing concepts demonstrated similar tendencies. All questions had roughly two-thirds of participants agreeing with the statement (*n* ranged from 34 (61%) to 41 (73%)). Questions 7, 8, 9 and 10 all showed an overall level of agreement (*Agree* or *Strongly Agree*) from 93% (*n* = 52) to 98% (*n* = 55). Only Question 6 showed an overall level of agreement less than 90% at 84% (*n* = 47). Levels of disagreement ranged from 1 (Q9, 2%) to 5 (Q6, 9%). Question 6 was also the only questions that received any *Strongly Disagree* responses, with four (7%).

The four questions designed to evaluate a teacher's interest in computer science also demonstrated similar tendencies (Table 9).

Table 9

Interest Levels of Participants in Computer Science

Survey Statement	SD	D	A	SA
11. I think computer science is boring (*N* = 56) . . . . RC	1 (2%)	15 (27%)	31 (55%)	9 (16%)
12. The challenge of solving problems using computer science appeals to me (*N* = 56)	3 (5%)	19 (34%)	31 (55%)	3 (5%)
13. I think computer science is interesting (*N* = 56)	1 (2%)	16 (29%)	34 (61%)	5 (9%)
14. I will voluntarily take computing courses if I were given the opportunity (*N* = 56)	3 (5%)	15 (27%)	32 (57%)	5 (9%)

Note. SD = *Strongly Disagree*, D = *Disagree*, A = *Agree*, and SA = *Strongly Agree*.
* RC means reverse coded.

All questions demonstrated comparable frequency levels across all four responses. The overall level of agreement (*Agree* or *Strongly Agree*) ranged from 71% (Q11, *n* = 40) to 61% (Q12, *n* = 34). Both *Agree* and *Disagree* responses showed similar consistencies in their range of values. Frequency counts for *Agree* ranged from 34 (Q13, 61%) to 31 (Q11 and Q12, 55%). Frequency counts for *Disagree* ranged from 19 (Q12, 34%) to 15 (Q11 and Q14, 27%). Frequency counts for *Strongly Agree* were minimal for Questions 12 (*n* = 3, 5%), 13 (*n* = 5, 9%), and 14 (*n* = 5, 9%), with Question 11 showing an increased level of agreement (*n* = 9, 16%). Frequency of responses for *Strongly Disagree* were lowest of all and ranged from 1 (Q11 and Q13, 2%) to 3 (Q12 and Q14, 5%).

Research Question 3

To address Research Question 3 (*What are educators' beliefs in regard to the importance of computational thinking in the classroom and their careers?*), teachers were presented with 10 questions using a four point Likert scale with possible responses of *Strongly Disagree*, *Disagree*, *Agree*, and *Strongly Agree*. The first six questions were used to evaluate participating teachers' beliefs regarding use of computational thinking in their classroom (Table 10). The last four questions were used to evaluate how important teachers felt learning computational thinking would be to their career (Table 11).

Table 10

Teachers' Beliefs on Using Computational Thinking in Classrooms

Survey Statement	SD	D	A	SA
15. Computational thinking can be incorporated in the classroom by using computers in the lesson plan (*N* = 56)	0 (0%)	3 (5%)	45 (80%)	7 (13%)
16. Computational thinking can be incorporated in the classroom by allowing students to problem solve (*N* = 56)	0 (0%)	1 (2%)	36 (64%)	19 (34%)
17. Knowledge of computing will allow me to [be] more effective in the classroom (*N* = 56)	0 (0%)	6 (11%)	38 (68%)	12 (21%)
22. Trying to introduce computing concepts will distract students from the task at hand (*N* = 56) . . . . RC	1 (2%)	4 (7%)	37 (66%)	14 (25%)
23. Using computing technologies can increase student interest in the material (*N* = 56)	1 (2%)	3 (5%)	37 (66%)	15 (27%)
24. I worry that my students will be more adept at using computing technologies than I am (*N* = 56) . . RC	3 (5%)	14 (25%)	32 (57%)	7 (13%)

Note. SD = *Strongly Disagree*, D = *Disagree*, A = *Agree*, and SA = *Strongly Agree*.
* RC means reverse coded.

Five of the six questions in Table 10 (Questions 16, 17, 22, 23, and 24) were designed to evaluate a teacher's beliefs regarding computers and computational thinking in their classroom.

All had an overall level of agreement (*Agree* or *Strongly Agree*) at or above 70%, ranging from 70% (*n* = 39) for Question 24 to 98% (*n* = 55) for Question 16. The consistency extended further for Questions 17, 22, and 23, where frequency counts for *Agree* ranged from 37-38, and frequency counts for *Disagree* were in the range of 3-6. While Question 15 displayed similar overall agreement levels to the other five questions, the level of *Agree* was much higher (*n* = 45, 80%), with a lower count in *Strongly Agree* (*n* = 7, 13%). Question 24 demonstrated a frequency count of *Agree* similar to the other questions (*n* = 32, 57%).

Four questions were used to evaluate participating teachers' beliefs regarding computers and computational thinking on their career (see Table 11). All four questions showed similar

Table 11

How Important Teachers Feel Learning Computational Thinking is to Their Career

Survey Statement	SD	D	A	SA
18. My career goals do not require that I learn computing skills (*N* = 56) . . . . RC	2 (4%)	5 (9%)	38 (68%)	11 (20%)
19. I expect that learning computing skills will help me to achieve my career goals (*N* = 56)	2 (4%)	8 (14%)	37 (66%)	8 (14%)
20. I hope that my future career will require the use of computing concepts (*N* = 56)	3 (5%)	13 (23%)	36 (64%)	4 (7%)
21. Having background knowledge and understanding of computer science is valuable in and of itself (*N* = 56)	0 (0%)	2 (4%)	38 (68%)	16 (29%)

Note. SD = *Strongly Disagree*, D = *Disagree*, A = *Agree*, and SA = *Strongly Agree*.
* RC means reverse coded.

frequency counts for *Agree*, with a range of 36 (Q20, 64%) to 38 (Q18 and Q21, 68%). The frequency counts for *Strongly Agree* varied from a high of 16 (29%) for Question 21 to a low of 4 (7%) for Question 20. The *Disagree* response showed a similar range, but with an inverse frequency count. Question 20 had the high count of 13 (23%), with Question 21 having the lowest count of 2 (4%). Questions 18 and 19 demonstrated similar tendencies, but with less disparity between *Strongly Agree* and *Disagree*. Only Questions 18, 19, and 20 reported any *Strongly Disagree* responses, with counts of 2 (4%), 2 (4%), and 3 (5%), respectively. Questions with a trailing RC are negatively worded and were reverse scored prior to evaluation of data.

Research Question 4

To address Research Question 4 (*What are educators' beliefs regarding the use of computational thinking in specific K-5 content areas of English, mathematics, social studies, and science?*), teachers were presented with a question for each of the four subjects asking if they believe computational thinking has a place in that subject. The available responses were *Agree*, *Disagree*, and *Don't Know* using radio buttons, ensuring the participant selected a single option (Table 12). After a participant made a selection for each subject, they were then asked to "Please

Table 12

Do You Believe Computational Thinking Has a Place in the Following Content Areas?

	Agree	Disagree	Don't Know
English	42 (75%)	4 (7%)	10 (18%)
Math	55 (98%)	0 (0%)	1 (2%)
Social Studies	40 (71%)	2 (4%)	14 (25%)
Science	51 (91%)	0 (0%)	5 (9%)

explain your response," using an open-ended response field. The use of an open-ended field allowed a participant to answer in their own words, eliciting feedback that would not be possible from a closed-ended question. This allowed for responses which were relevant, but unanticipated or for responses which were not necessarily relevant to the question associated with the response, but still important and noteworthy. Open-ended questions allowed for any and all feedback a participant wished to provide.

Categories identified from all four open-ended questions were fairly consistent across subjects, with frequency counts varying from subject to subject. There were a few identified categories which existed in only one subject, but those were few, and made specific reference to the subject area and not to computational thinking in general.

To avoid redundancy, a list of all identified categories, with a definition, are provided. For each subject, a table is provided listing the frequency counts for each category identified within that subject, along with a few selected comments from the teachers responding. Comments are meant to be representative of comments received from all teachers, and not an inclusive list of all comments received from teachers.

Categories

Use of a computer – All of these responses made a specific reference to the use of a computer to complete tasks or implement solutions without any specific reference to programming or coding.

Computational thinking skills – These responses included references to skills associated with computational thinking. These skills included, but were not limited to: problem solving, critical analysis, logic, sequencing of events, and cause and effect.

Vague computational thinking reference – All of the statements included in this code makes specific reference to computational thinking, without making a reference to any computational thinking skill or direct statement on how computational thinking can be used.

Interested but not sure how – This statement refers to educators explicitly stating they were interested in the notion of computational thinking, but literally had no idea on how to use computational thinking in their classrooms.

Not sure how – Educators did not know how to use computational thinking in their classrooms.

Programming – These statements specifically included a reference to computer programming, programming, or coding.

Literacy skills – All responses included a reference to reading and/or writing skills.

Computational Thinking and English

In response to believing that computational thinking has a place in English, 42 of the 56 who responded stated they agreed (75%), four disagreed (7%), and 10 did not respond to the question (18%). A follow up question asked a teacher to please explain their response. Of the 42 who agreed, 20 (36%) did not provide an explanation. Of the 22 (39%) who did provide a response, a total of 28 codes were derived from the responses. Some responses included multiple codes, so the total number of codes did not equal the number of responses. Codes were grouped into categories and identified categories and their frequency of occurrence are listed in Table 13.

Table 13

Categories Associated With an Agree *Response to Computational Thinking in English*

Category	Count
Use of a computer	9
Computational thinking skills	7
Vague computational thinking reference	6
Literacy skills	4
Programming	2

The category “Use of a computer” was derived from teacher comments that included:

- “Primarily for research and publishing purposes.”
- “For an example, doing a research project and need to cite sources.”
- “There are many selective programs that can be utilized to help build on language and grammar skills.”

The category “Computational thinking skills” was derived from teacher comments that included:

- “Sequencing is a very important concept in comprehension. This is also very important for computer programming. Editing is important to both subjects, too.”
- “You can apply analysis concepts to the materials that are being read.”
- “Using logic and laying out arguments is present in both.”

The category “Vague computational thinking reference” was derived from teacher comments that included:

- “Though I am not exactly sure what computational thinking is, my sense is that it would be beneficial. I think it would involve helping students express themselves in a variety of ways.”
- “Computational thinking can have a place in any content area. It can be used differently.”
- “In elementary, computational thinking always has a place.”

The category “Literacy skills” was derived from teacher comments that included:

- “LA [language arts] is used in every aspect of life and if able to problem solve even slightly can be a benefit.”

- "I would say the writing process best exemplifies trial and error. The steps in the writing process is all about writing and making your writing better."

The category "Programming" was derived from teacher comments that included:

- "Sequencing is a very important concept in comprehension. This is also very important for computer programming. Editing is important to both subjects, too."
- "Writing code involves read skills and critical thinking."

Of the four teachers (7%) who disagreed with computational thinking having a role in English education, two (4%) provided an explanation and two (4%) did not. One statement indicated that a computer should not be used in English and the other indicated that they were "not sure how."

Ten teachers (18%) did not respond to the question for English: "Do you believe computational thinking has a place in the following content areas?" Four (7%) of those teachers who chose not to respond to the initial question still chose to provide an explanation requested in the follow up question. Even with the lack of a response, several teachers still demonstrated a degree of interest in computational thinking, while indicating a lack of understanding on how it could be used within this field (English). Categories identified by these four respondents and their frequency of occurrence are listed in Table 14.

Table 14

Categories Associated With No Response to Computational Thinking in English

Category	Count
Not sure how	3
Interested but not sure how	2
Use of a computer	1

The following comments were offered by the teachers in support of categories in Table 14:

- “I believe in [sic] can be but unsure.”
- “I have used computers in my classroom for years, this area is one that I haven't found an effective way of teaching by computer. The research is great, the writing and editing is great, however the basic grammar still needs to be taught more directly and is done better with paper.”
- “I know that when I'm stuck while writing a paper I use thesaurus and dictionary.com because it lists all the words I can use in the correct way.”
- “I would be interested in learning more about computational thinking as it relates to English.”

Computational Thinking and Mathematics

In response to believing that computational thinking has a place in mathematics, 55 of 56 who responded stated they agreed (98%), no one disagreed, and one did not answer the question (2%). The follow up question asked respondents to please explain their response. Of the 55 who agreed, 30 (54%) did not provide an explanation. Of the 25 participants who did provide a response, a total of 26 codes were derived from the responses. Some responses included multiple codes, so total number of codes did not equal total number of responses. Identified codes were grouped into categories and the frequency of occurrence of each category is listed in Table 15.

Table 15

Categories Associated With a Yes Response to Computational Thinking in Mathematics

Category	Count
Computational thinking skills	8
Vague computational thinking reference	8
Use of a computer	6
Programming	3
Interested but not sure how	2

The category "Computational thinking skills" was derived from teacher comments that included:

- "Breaking things down in small segments can be beneficial in seeing the mathematical portion."
- "Formulas, discreet [sic] math, logic used in both."
- "In Math we are constantly working on different ways to approach solving problems. There is a ton of exploration and trial and error."

The category "Vague computational thinking reference" was derived from teacher comments that included:

- "I believe computational thinking is essential to math especially with the new mathematical practices included in common core."
- "Computational thinking is in algebra."
- "Mathematics uses computational thinking already weather [sic] it is implied or not."

The category "Use of a computer" was derived from teacher comments that included:

- "There are many selective programs that can be utilized to help build on math skills."
- "Tutors are available online. Extra practice of basic skills. . ."
- "Graphing . . . creating and analyzing graphs; interactive manipulative to move students from concrete thinking to more abstract."

The category "Programming" was derived from teacher comments that included:

- "I can see using computer programming in math as an asset, if used in moderation and in supplement - not replacement."
- "Coding involves math skills."

The category "Interested but not sure how" was derived from teacher comments that included:

- "I would be interested in learning more about computational thinking as it relates to Math."
- "I am sure there are a lot of activities that you can do."

Computational Thinking and Social Studies

In response to believing that computational thinking has a place in social studies, 40 of the 56 of those who responded stated they agreed (71%), two disagreed (4%), and 14 (25%) did not answer the question. The follow up question asked teachers to please explain their response. Of the 40 who agreed, 22 (38%) did not provide an explanation or provided an explanation from which no meaningful code could be derived. Of the 18 (32%) who did provide a meaningful response, a total of 22 codes were derived from the responses. Some responses included multiple codes, so the total number of codes did not equal the total number of responses. Identified codes were grouped into categories and frequencies of occurrence of categories are listed in Table 16.

Table 16

Categories Associated With a Yes Response to Computational Thinking in Social Studies

Category	Count
Use of a computer	9
Vague computational thinking reference	4
Computational thinking skills	3
Interested but not sure how	2

The category "Use of a computer" was derived from teacher comments that included:

- "Research and sites with additional information on topics of study."
- "There are many selective programs that can be utilized to help build on history, geography, and other content areas related to social studies."
- "Tons of research material available that enhances what we talk about in social studies."

The category "Vague computational thinking reference" was derived from teacher comments that included:

- "Seeing how the past has used computational thinking (even if it wasn't called that) during war times and navigation in transportation advancements . . ."
- "This may allow students to understand how connected we are to the rest of the world."

The category "Computational thinking skills" was derived from teacher comments that included:

- "You can use analysis concepts to analyze history, geography impacts, etc."

- "Real world problems can be presented in such a way that the process of problem solving can be the forefront in Social Studies."
- "Understanding solutions (wars, great depression, etc.) why people decided to do what they did . . ."

The category "Interested but not sure how" was derived from teacher comments that included:

- "Again, I am sure there are a lot of activities that you can do."
- "I would be interested in learning more about computational thinking as it relates to Social Studies."

Computational Thinking and Science

In response to believing that computational thinking has a place in science, 51 (91%) of 56 who responded stated they agreed, no one disagreed, and five (9%) did not answer the question. The follow up question asked teachers to please explain their response. Of the 51 who agreed, 30 (48%) did not provide an explanation or provided an explanation from which no meaningful code could be derived. Of the 21 (38%) who did provide a meaningful response, a total of 22 codes were derived from responses. Some responses included multiple codes, so the total number of codes did not equal total number of responses. Identified codes were grouped into categories and frequency of occurrence of each category is listed in Table 17.

The category "Computational thinking skills" was derived from teacher comments that included:

- "Science experiments require critical thinking skills and the ability to think about things in multiple ways. Computer science used the same skills."
- "Again, science uses algorithms and equations. So yes."
- "Computational thinking is the scientific method."

Table 17

Categories Associated With a Yes Response to Computational Thinking in Science

Code	Count
Computational thinking skills	9
Use of a computer	9
Vague computational thinking reference	2
Interested but not sure how	2

The category “Use of a computer” was derived from teacher comments that included:

- “I strongly agree with science. There are many selective programs that can be utilized to help build on biology, genetics, and other science related content areas.”
- “Modeling, research, etc.”
- “There are lots of things that help extend and reinforce concepts. However, all of these areas should have the "hands on" model in them as well as computer models.”

The category “Vague computational thinking reference” was derived from teacher comments that included:

- “In elementary, computational thinking always has a place.”
- “Computational thinking is already used in many sciences already.”

The category “Interested but not sure how” was derived from teacher comments that included:

- “I would be interested in learning more about computational thinking as it relates to science.”
- “Again, I am sure there are a lot of activities that you can do.”

Free Response Question

Finally, participating educators were asked *to share anything (positive or negative) you would like to say about the use of computational thought in the K-5 classroom*. Open-ended responses were coded after the survey was closed. The same coding process was used throughout the entire study on qualitative data. Significant statements were identified, with nonsensical or irrelevant responses ignored. The next step was identification of codes from significant statements. The entire coding process was independently conducted twice, with results from both attempts reconciled into a single document. Once codes were identified, they were analyzed, allowing categories to emerge from the codes. Categories were used to group or organize those codes with similar ideas. As with the initial coding process, and given the exploratory nature of this study, categories were developed ad hoc from the codes. It was critical that categories were truly representative of the data, and not the result of pre-existing ideas or notions of what might, or could be present in the data.

Twenty-four unique codes were identified during the coding process, which are shown in Appendix B. Seventy total codes were found in significant statements. After analysis, five categories emerged from the codes: student experience, classroom experience, professional development/curriculum, student preparation, and technology concerns. From these categories, three themes were identified: students, teachers, and technology. Table 18 lists themes, associated categories, and number of codes per category.

Table 18

Themes, Categories, and Codes Within Each Category

Theme	Category	Unique Codes Within Each Category	Total Codes Within Each Category
Students	Student experience	8	36
	Classroom experience	2	3
Teachers	Professional development / curriculum	5	12
	Student preparation	2	8
Technology	Technology concerns	7	13

Theme - Students

The “students” theme contains codes that relate directly to students, whether in the context of their individual personal experiences or within a group setting in a classroom experience. Ten unique codes were assigned to the students theme, encompassing 39 total codes. These codes were organized into two categories: “student experience” and “classroom experience.” The codes refer to positive effects and outcomes teachers feel the introduction of computational thinking may have on students.

Student experience. The “student experience” category contains codes that relate directly to a student such as computational thinking being beneficial to a student likely promoting problem solving skills or a useful skill when looking to the future. There were eight unique codes taken from thirty-six identified student experience codes. Table 19 contains a list of unique codes from the student experience category.

Table 19

Unique Codes for Student Experience Category

Code	Count
Allows students to display strengths	1
Beneficial to students	15
Promote problem solving skills	5
Promote student confidence	1
Required skill for future	6
Students are willing to take risks	1
Students need to learn to use the computer	4
Teaches perseverance	3

Fifteen participants noted that computational thinking would be beneficial to their students. Comments indicated a general positive attitude toward introducing computational thinking into classrooms. Listed below are a few sample comments representative of the overall group of statements from teacher participants:

- “I think it would be very beneficial to get all students to engage in classroom learning.”
- “I think that it can be a valuable skill for students to learn.”
- “If it starts early enough, it will make learning down the road more understandable.”
- “Let's get this into our elementary classrooms ASAP!”
- “There are many benefits to utilizing computational thought in the classroom, even at the elementary level.”

Several educators suggested that computational thinking promotes the development of problem-solving skills. Several comments from teachers supporting this idea are listed below:

- “I like that we can take a problem apart and figure out the entire parts to be able to figure out the problem. I believe that it makes us think about away that we can attack a problem.”
- “I think that computational thinking is an essential skill to support and expand students problem solving skills.”
- “Allows them to be creative and reflective in their problem solving processes.”
- “I'm guessing that it's a way of breaking down a problem into smaller steps making it more manageable to think about and solve.”
- “It helps develop many skills that we are already working on.”

Numerous teachers offered specific comments on how computational thinking could be beneficial for their students. Responses included improved student confidence, increased willingness to take risks, improved perseverance, and increased student engagement. A few samples of teacher’s comments supporting these ideas are listed below:

- “I believe that it will make a student feel more confident in themselves because they are learning lifelong skills.”
- “Kids are so willing to take risks with technology!”
- “I use code.org and the students loved being able to control the programing.”
- “Computers can help students stay engaged.”
- “Not every problem is solved on the first, second, or third try.”

- “Computational thinking gives them an opportunity to have a chance to struggle through a problem and try out solutions until they find the best option (real world things).”
- “When we as teachers attempt to teach grit or perseverance I have yet to see a tool that does this better than coding.’

Finally, several educators also saw the introduction of computational thinking as a skill set that will be required for students as they move forward, both as a student, and in their future careers.

- “It is an essential skill for today's work force.”
- “Computer literacy is on the rise with the available technology currently and the unknown career paths in the future.”
- “I think that in order for students to be successful in their future careers, they will need to have many computational skills and flexible thinking.”
- “Working to prepare the students for the future.”
- “As our students progress through the grades, many skills are required to complete assignments."
- “It is the way of the future.”

Classroom experience. The “classroom experience” category contains codes that relate to the classroom experience of a student. While these codes obviously make reference to a student, the classroom experience takes into account a classroom environment and how a positive environment can promote student engagement and learning from other students. There were three total classroom experience codes that could be condensed into two unique codes. Table 20 contains a list of the codes from the classroom experience category.

Table 20

Unique Codes for Classroom Experience Category

Code	Count
Positive element in the classroom	1
Promotes student engagement	2

Several teachers' commented on how computational thinking promoted student engagement and had a positive effect in the classroom. Three example comments are listed below:

- "I use code.org and the students loved being able to control the programing."
- "Computers can help students stay engaged, and it's the way of the future."
- "It is a very positive element in the K-5 classroom."

Theme - Teachers

Codes found in the "teachers" theme are related directly to a teacher. Several codes make indirect reference to students, which seems logical given the symbiotic relationship between teachers and students. Twenty total codes that condensed into seven unique codes were identified for inclusion in the teachers theme. These codes were organized into two categories: professional development/curriculum concerns and student preparation concerns.

Professional development/curriculum. The "professional development/curriculum" category contains codes that relate to concerns teachers may have regarding the lack of a clear understanding of what computational thinking is, the need for professional development in the area of computational thinking, how computational thinking can fit into an existing curriculum, or apprehensions over finding time to add computational thinking to an existing curriculum. Twelve codes were condensed into five unique codes (Table 21).

Table 21

Unique Codes for Professional Development/Curriculum Category

Code	Count
Curriculum concerns	3
Not sure what CT is	4
Teachers cannot be replaced	1
Teachers need professional development	2
Time concerns	2

Several comments respondents gave are listed below:

- “I would need to see how it could be accomplished effectively and efficiently.”
- “It isn't that I don't welcome new ideas and learning for students, but in an elementary classroom we are already drowning with so much to cover and get through.”
- “Would this be incorporated with existing curriculum and standards?”
- “I am not quite sure what it is.”
- “I'm not really sure what computational thinking is.”
- “Not sure I know enough about it to make a statement.”
- “I do think teachers need instruction on how to implement the use of devices into content areas.”

- "If you are really passionate about this, back it up with resources. This does not just mean a day where teachers learn about it."
- "We already have a lot on our plates as educators, and I hesitate to add more to an already overwhelming curriculum."
- "My concern is where do we fit it in?"

Student preparation. The "student preparation" category contains codes that directly relate to a teachers' thoughts on whether students are ready for the introduction of computational thinking into their curricula, or if students of this age will be able to generalize the computational thinking skills into real life. Eight codes were condensed into two unique codes as shown in Table 22.

Table 22

Unique Codes for Student Preparation Category

Code	Count
Skills do not generalize for the students	1
Students may not be ready	7

Several examples of the teachers' comments are listed below:

- "These skills do NOT generalize to real life for students this age."
- "Computational thought seems very abstract thus higher level thinking."
- "I am not sure it has a place in the primary classroom."
- "I see many students struggle with problem solving skills which involve multiple steps."
- "Technology is everywhere so you should use it."

- "I have many sixth graders who think they know how to use a computer because of these programs but cannot send an email, use office, remember usernames or even know the difference between a left click and a right click on the mouse."
- "The lack of skills needed to use a keyboard/compose sentences while typing."
- "I think that computational thought is very abstract and may present as a challenge to all students."

Theme - Technology

The "technology" theme contained one category, the "technology concerns" category. This category contains codes that directly relate to teachers' concerns regarding a student's use of computing technology and computational thinking or technology in general. Comments included a student's need to access technology, students need to learn to productively use technology, that technology is not required to teach computational thinking, concerns about the care of technology, appreciation for the tech support personnel who work on technology, and concerns regarding access to said technology. There were eleven codes in the technology concerns category that were condensed into seven unique codes (Table 23).

Table 23

Unique Codes for Technology Concerns Category

Code	Count
Appreciative of tech support	1
Computational thinking does not require a computer	2
Concerns about the care or misuse of technology	1
Concerns about too much time with technology	1
Digital divide concerns	1
Students need access to technology	4
Students need to use computer productively	1

Teachers' comments from this theme and category included:

- "I also believe that computational thinking isn't always directly correlated with physical technology."
- "I know that a person doesn't necessarily need computers or technology for this type of situation."
- "Technology can be very hard to come by in schools, which can be a huge negative in the school setting."
- "I would like each classroom to have more technology available."
- "We need to understand computers so that we know how to use computers to our advantage."
- "We need access to quality computers and iPads."
- "I believe that if the introduction to computers in a way that isn't just browsing YouTube or the internet (in a non-educational way), would be helpful for setting up the foundation for use of technology in the classroom."
- "We are lucky to have tech partners to help guide us in our teaching of technology."
- "We would need to have an insurance policy that held students accountable for care/loss of devices."
- "Students who have too much exposure to screen time run the risk of developing a multitude of conditions."
- "My concern is lack of technology parents of lower incomes have in the home."

Summary

This chapter contains results of qualitative and quantitative analyses and connects results back to research questions. Fifty-six participants completed the survey responding to both closed- and open-ended questions. The survey was designed to identify barriers to implementation of computational thinking in K-5 classrooms. All participants were core content area K-5 educators in a North Dakota public elementary school. Next, in Chapter V, a discussion of the results as well as implications from this study will be provided.

CHAPTER V

DISCUSSION

The purpose of this exploratory study was to gain insight and understanding into K-5 educators' attitudes and beliefs regarding the implementation of computational thinking content in their classrooms. This chapter includes a discussion of the findings regarding participating teachers' understandings of computational thinking, their interests and comfort levels regarding the use of computational thinking, the potential effect computational thinking may have on their career, and their understanding of where computational thinking could be introduced into their curricula. This chapter concludes with a discussion of the limitations of this study, potential areas for future work, and a brief summary.

Meaning and Definition

The first research question was: *What are educators' levels of understanding in regard to the definition of and opportunities to utilize computational thinking?* The most basic definition of computational thinking is that a problem, and its solution, is addressed with the idea that could the problem can, but does not necessarily have to, be solved using a computer. Basic skills used in computational thinking would include: the ability to clearly define a problem (removing any ambiguities or uncertainties), breaking the problem down into a sequence of smaller, more manageable problems (decomposition); removing unnecessary details (abstraction); looking for similarities and recognizing where a solution already exists for part of the overall problem (pattern matching); and constructing, a series of steps, and possibly testing/debugging the steps, to solve the overall problem (algorithmic thinking).

This study found that, overall, participating educators did not have a good working understanding of computational thinking, its definition, meaning, or how it could be implemented in their classroom. Only 27% (*n* = 15) of respondents indicated they were familiar with the concept of computational thinking. The remaining 73% (*n* = 41) of respondents indicated they were not familiar with the concept. This lack of understanding regarding computational thinking is not surprising, given that computer science, and by extension computational thinking, is neither a core content area in K-5 curricula or a key component in any of the standard core content areas (English, mathematics, science, social studies). For the most part, computer science has not been a required subject in K-12, or even addressed by most states' educational policies. While computer science is experiencing an increase in demand across all grade levels, there has been little incentive for elementary schools to include computer science or computational thinking in their curricula. In 2013, only 14 states had educational policies which addressed computer science; as of 2018, 44 states had enacted one or more policies regarding computer science (Code.org Advocacy Coalition and the Computer Science Teachers Association, 2018).

Educators who responded with "No" when asked if they were familiar with the concept of computational thinking were asked what they thought computational thinking means. Even though these educators did not report being familiar with the concept, it was important to determine what level of understanding they might have had about computational thinking to establish a foundation for future work. Overall, responses were not out-of-line with the definition or use of computational thinking. Seventeen comments (45% of "No" responses and 30% of overall responses) indicated respondents believed computational thinking was problem solving. Problem solving, while technically not the definition of computational thinking, does refer to the process of solving a problem without making specific reference to the means of solving the problem or of a particular type of problem.

Most of the remaining comments indicated areas in which computational thinking could be applied, such as being used in mathematics (*n* = 11, 29% of "No" responses, 20% of overall responses). A significant number of comments (*n* = 15, 39% of "No" responses, 27% of overall responses) indicated that computational thinking meant the ability to use a computer, such as accessing the Internet to perform research or using a word processor or spreadsheet. Three comments (8% of "No" responses, 5% of overall responses) indicated that computational thinking meant programming a computer. These comments, other than problem solving, tend to describe specific ways in which computational thinking can be applied, but lacked a general understanding of the concept of computational thinking.

Educators who responded with "Yes" when asked if they were familiar with the concept of computational thinking were then asked to explain the concept of computational thinking. Overall, this group of educators had the best insights into the meaning of computational thinking. Of the educators who answered "Yes," 93%, 13 (23% of overall responses) believed computational thinking related to problem solving. Eight (57% of "Yes" responses, 14% of overall responses) specifically stated computational thinking did not require the use of a computer, with an additional three (21% of "Yes" responses, 5% of overall responses) indicating that computational thinking meant addressing problems such that they could be solved by a human or computer. Three comments (21% of "Yes" responses, 5% of overall responses) indicated that computational thinking meant using a computer in some manner.

Overall, comments associated with "Yes" responses to participants being familiar with computational thinking were very positive. This sector of respondents understood that a computer is either not required, or is optional, as part of the computational thinking process. These participants recognized that computational thinking is just a process used to find a solution to a problem without regard to how the solution will be implemented. While these comments were encouraging, the limited number of teachers who possessed this knowledge of computational thinking was problematic and needs to be addressed.

Interest and Comfort

The second research question was: *What are educators' levels of interest and comfort surrounding computational thinking?* Participating teachers were asked nine questions to gauge their interest and comfort level in working with computer science concepts, and computing in general (Questions 6-14 in Part II of Appendix A). While computer science does not equate to computational thinking, use of a computer is the most common means of developing and testing computational thinking skills. While an educator could be interested in the concept of computational thinking and have no interest in applying those skills without a computer, the reality is that computers will need to be used in some fashion to promote the teaching of computational thinking.

Four questions were used to gauge teachers' interest levels (Questions 11-14 in Part II of Appendix A) in computer science. Findings showed that participating teachers consistently indicated an interest in computer science, but not a strong interest. Thirty-seven or 67% of participants (n = 5 for *Strongly Agree*, n = 32 for *Agree*) stated they would voluntarily take a computer science course, and 39 or 70% (n = 5 for *Strongly Agree*, n = 34 for *Agree*) stated they find computer science interesting. The levels of interest participants had in wanting the challenge of solving problems using a computer dropped, but not significantly, to 34 or 61% (*n* = 3 for *Strongly Agree*, *n* = 31 for *Agree*).

Data demonstrated participants held consistent levels of interest, showing neither strong interest levels or strong lack of interest levels in computer science. The range of percentages for teachers selecting *Agree* in the questions relating to interest levels ranged from 55% (*n* = 31) to 61% (*n* = 34). The number of participants who strongly agreed with survey questions relating to interest levels ranged from three to nine, showing very limited interest in learning about computer science or computational thinking. The number of participants who strongly disagreed with the four survey questions associated with interest ranged from one (three times) to three (once), demonstrating that very few teachers have little to no interest in learning about computer science or computational thinking.

Five questions were used to determine participating teachers' comfort levels in working with computers and with computing concepts (Questions 6-10 of Part II in Appendix A). As stated earlier, while computational thinking is not the same as computer science, on the practical/application level, working with computers is the most common and efficient means of learning computational thinking.

Findings showed participating educators felt very comfortable using a computer and in working with computing concepts. For four questions involving ability to learn computing concepts (Question 8), ability to pass a computing course (Question 7), daily use of computing skills (Question 9), and ability to solve problems using a computer (Question 10), respondents reported levels of agreement (*Agree* or *Disagree*) in the range from 93% (*n* = 52) to 98% (*n* = 55), with none of those questions receiving a *Strongly Disagree* response. For the question asking about comfort learning computing concepts (Question 6), participants reported slightly lower results (84% responded with *Agree* or *Disagree*, *n* = 47), and included four *Strongly Disagree* responses.

These results demonstrate that overall participating educators felt comfortable and confident in their ability to work with a computer and in their ability to learn new computing concepts, which may be required for teaching computational thinking. The reported lower levels of comfort in learning computing concepts, while still high, indicated that more research is necessary to determine why an educator may not be comfortable in learning new concepts and what can be done to improve their comfort level in this area.

Classroom and Career

The third research question was: *What are educators' beliefs in regard to the importance of computational thinking in the classroom and their careers?* To address this question the teachers were asked 10 questions to evaluate how important they thought computational thinking would be to their career and how it would impact their classroom experience (Questions 15-24 in Part II of Appendix A).

Four questions asked teachers if they thought computational thinking, and computing skills in general, would have an impact on their career (Questions 18-21 in Part II of Appendix A). Findings showed that educators do believe computing knowledge and understanding is useful and will help them in their careers. Ninety-six percent (*n* = 54) of participants either *Agreed* (*n* = 38, 68%) or *Strongly Agreed* (*n* = 16, 29%) when asked if "Having background knowledge and understanding of computer science is valuable in and of itself." Eighty-one percent (*n* = 45) of participating teachers also responded favorably to the question, "I expect that learning computing skills will help me to achieve my career goals" (*Agree*, *n* = 37, 66%; *Strongly Agree*, *n* = 8, 14%). After reverse coding responses to the question, "My career goals do not require that I learn computing skills," similar positive responses were received (*n* = 49, 88% overall agreement; *Agree*, *n* = 38, 68%; *Strongly Agree*, *n* = 11, 20%).

However, teachers did not report the same level of interest when asked "I hope that my future career will require the use of computing concepts." This question had an overall agreement level of 71% (*Agree*, *n* = 36; *Disagree*, *n* = 4), which shows a high level of agreement. The question also received 13 *Disagree* responses (23%) and three *Strongly Disagree* responses (5%), indicating a significant portion of the educators surveyed had no interest in using computing skills in the workspace.

Responses indicated that participating teachers were generally in agreement that their career would require the use of computing concepts, yet did not express the same level of agreement in wanting to use computing concepts in their future career path. Future research should help provide insight into the discrepancy between the perceived benefit from using computing concepts and the lower level of reported interest in said skills.

The remaining six questions were used to determine any effect teachers thought computational thinking may have on classroom environments (Questions 15-17 and 22-24 in Part II of Appendix A). Five of the questions indicated teachers feel strongly that computational thinking can have a positive effect on their classrooms; all participants showed a level of agreement (*Agree* or *Strongly Agree*) in the range of 89% (*n* = 50) and 93% (*n* = 52). These questions asked teacher participants if they thought computational thinking can be taught using computers in their lesson plans or by allowing students to problem solve, if having knowledge of computing will allow them to be more effective in the classroom, if they thought introducing computing concepts will distract students from the task at hand, or if they thought using computing technologies can increase student interest in material. Data indicated teachers significantly valued the use of computing concepts in their classrooms, both engaging their students in the work as well as allowing a teacher to be more effective.

Data showed that teachers strongly agree that computational thinking can be taught through problem solving. However, results from the question "Computational thinking can be incorporated in the classroom by using computers in the lesson plan" also indicated that the majority of the teachers equate using a computer and computational thinking. This shows a strong level of agreement with the misguided concept that computational thinking can be taught just by using computers in lesson plans. As mentioned earlier, it is understandable that there is a level of confusion regarding the difference between computer literacy and computational thinking. Education, through continued research leading to professional development, should help teachers understand the distinction between teaching students to be competent using a computer and the skill set associated with computational thinking.

Teachers did report a level of concern when asked if they "worry that my students will be more adept at using computing technologies that I am." Thirty percent (n = 17) reported that they *Agree* or *Strongly Agree* with that statement, indicating that work must be done to either improve a teachers' agency in their classroom, whether in their knowledge of computational thinking and computing concepts, or in their confidence in teaching computational thinking concepts to their students.

Computational Thinking in Content Areas

The fourth research question was: *What are educators' beliefs regarding the use of computational thinking in specific K-5 content areas of English, mathematics, social studies, and science?* There was substantial agreement that computational thinking can be used in the specified subject areas across all subjects. Levels of agreement ranged from a high of 98% in mathematics to a low of 71% in social studies (Table 24).

Table 24

Computational Thinking Has a Place in Specified Content Areas

	Agree	Disagree	Don't Know
English	42 (75%)	4 (7%)	10 (18%)
Math	55 (98%)	0 (0%)	1 (2%)
Social Studies	40 (71%)	2 (4%)	14 (25%)
Science	51 (91%)	0 (0%)	5 (9%)

It is not unexpected that mathematics and science received respondents' highest levels of agreement. There seems to be a logical link between computational thinking and the subjects of mathematics and science (Cho, Pauca, Johnson, & James, 2014). There also is the Science, Technology, Engineering, and Mathematics (STEM) connection, given that computer science has been formally recognized as a component of STEM (STEM Education Act of 2015). When using a computer, even the most trivial of applications generally involves mathematical calculations, coordinates systems, Boolean logic, or the use of mathematical functions. Discrete mathematics, a branch of mathematics, is at the very foundation of computing. Computational thinking has also been included as a core scientific principle in the Next Generation Science Standards (2019). Weintrop et al. (2016) stated, "In the last 20 years, nearly every field related to science and mathematics has seen the growth of a computational counterpart" (p. 128).

Near unanimous levels of educators selecting *Agree* when asked "if computational thinking has a place in" math (n = 55) or science (n = 51) shows participants felt computational thinking should be implemented in these fields. When asked to explain their response, the most common reply was that computational thinking skills can directly be used in mathematics and science to improve learning of students in those fields. The second most common response was that mathematics and science can use computers in their coursework solidifying the notion that computational thinking equates to using computers. While it is positive that educators believe computational thinking can be used, and taught, in these subject areas, the reoccurring impression that computational thinking equates to the use of a computer must be addressed.

Subjects such as English and social studies, commonly viewed as liberal arts or soft science fields, may not have the obvious connection to computational thinking found in other subjects. However, it has been suggested that modifying the K-12 curricula to include computational thinking can be beneficial to students across all disciplines (Lu & Fletcher, 2009; Settle et al., 2012; Yadav, Krist, Good, & Caeli, 2018). To that end, the use of computational thinking has been implemented in Grades K-12 in literally every subject, including English and science (Google, n.d.; Lockwood & Mooney, 2017; Nesiba, Pontelli, & Staley, 2015). The ability to utilize computational thinking in all subjects, while present, is not as obvious in some fields as others, and is something which needs to be addressed with educators to improve their understanding of what is available and how it can be introduced into their curricula.

While teacher respondents indicated that computational thinking can be used in English (n = 42) and social studies (n = 40), with very few *Disagree* responses (English, n = 4; Social Studies, n = 2), it appeared that participating educators as a group did not have a clear vision of how computational thinking could be used in these subjects. When asked to explain their responses, common responses were through the use of a computer, or that they were interested in learning more but did not know how to implement computational thinking in the indicated field. There were very few responses on how computational thinking could be used in either of these fields (English or social studies).

Results from the agree/disagree question on computational thinking having a place in specified content areas indicated that most educators responding felt computational thinking could be used in English and social studies yet struggled to find definite ways in which this could be accomplished. Education, through professional development or other means, will be necessary to help clarify how computational thinking can be used within a subject for those teachers who did not feel it has a place in either English or social studies or how it could be implemented within a subject area to improve student learning.

Demographics

The ratio of those reporting as female was 88% (*n* = 49), with the final 13% (*n* = 7) reporting as male. This was comparable to national statistics reported from 2017, with female educators accounting for 87% of elementary public school teachers in the United States (UIS, n.d.).

Grade levels taught by participating educators were skewed toward the upper grades. Of the 39 educators who reported the grade level they were teaching at the time of this study, 72% (*n* = 28) were teaching Grades 3-5. Eight educators (21%) reported teaching second grade, with the remaining three educators (8%) teaching in kindergarten or first grade. Given that students' reading, cognitive, and creative thinking skills should advance with age and grade level, it is not unreasonable to align teacher interest in computational thinking content with a progression through grade levels. While data cannot be used to infer that educators in Grades 3-5 are more likely to be interested in computational thinking, or at the least, take a survey about computational thinking, this finding suggests that there may be a correlation between grade level and teacher interest. Any future research should note the grade level of participants to determine if this finding is an anomaly, or if there is a need to explore why teachers in primary grades are more reluctant to utilize computational thinking.

Years of teaching experience was bi-modal, with lowest amount of experience (1-5 years; *n* = 14, 25%) and the most experience (21+ years; *n* = 14, 25%) having the most participants. This is an inverse trend from national statistics reported for 2015-16, where those with 3-20 years of experience accounted for 68% of elementary school educators (National Center for Education Statistics, 2017). As with data on grade level participants were teaching at the time of this study, while no inferences could be made from the data in this study, future research should include years of teaching experience in demographics data. If this result is consistent in future studies, additional research can address why teachers with the least experience (1-5 years) and teachers with the most experience (21+ years) are more likely to participate in a computational thinking survey.

Limitations

This study has several limitations. First, the survey was limited due to a limited response rate. There were several factors contributing to the lack of teacher responses. The initial contact with schools, where permission was requested to distribute the survey, was made in April, near the end of a semester. Several schools denied the request for a variety of reasons: not wanting to potentially burden their faculty during an already busy time of the year, or due to survey fatigue from the number of requests that had already been made on educators up to that point. Several schools refused to assist with graduate research as a basic rule, with one school only allowing student research studies for educators from their school district. All schools who declined due to time issues were contacted again in September, resulting in one additional school sending the survey request to their educators.

Second, the researcher had no direct ability to send out survey requests. While the researcher has no reason to doubt email requests were sent to potential participants, there was also no means to verify or guarantee that requests were sent to teachers. The largest school district would not send out an email request to teachers, rather, posted a link to the survey on a website accessible by their teachers. There was no means of determining how many educators from that school district took the survey, or even viewed the request on the school district's website.

Finally, teachers chose for themselves whether or not to take the survey. Lack of initial interest or understanding regarding computational thinking could have a negative influence on the likelihood of an educator choosing to take the survey. Participation bias suggests that those educators already familiar with computational thinking, or those who have introduced computer science in some fashion, such as the Hour of Code, would be more likely to have taken the survey than those educators without any experience with computational thinking or computer science in general (Fan & Yan, 2010; Groves, Presser, & Dipko, 2004).

Future Research

Future research will need to identify how best to introduce computational thinking into standard subject areas. Given that computational thinking is best presented in conjunction with a subject area, and due to a lack of time educators have to add new content to their existing K-5 curricula, research needs to continue to find best practices for including computational thinking into the entire K-5 curricula. This research should provide improved avenues for integration of computational thinking into a classroom.

Merely finding examples or problem sets where computational thinking can be included in standard subject areas is not enough. Future research will need to find pedagogical strategies which ensure that educators do not only learn the technical aspect of computational thinking, but learn how to effectively integrate computational thinking into specific subjects. Mishra and Koehler developed the Technological Pedagogical Content Knowledge (TPACK) model with the goal of providing a conceptual model to aid in technology-enhanced learning (Rienties, Brouwer, & Lygo-Baker, 2013). The TPACK provides a useful framework for studying teacher knowledge in relation to computational thinking (Mouza, Yang, Pan, Ozden, & Pollock, 2017). The TPACK identifies different types of knowledge teachers need to acquire in order to create a productive technology-enhanced learning environment. Two components of the TPACK specific to technology include Technological Pedagogical Knowledge (TPK), which addresses teaching using technology, and Technological Content Knowledge (TCK), which addresses how technology can be effectively used in a subject area. Using the TPACK framework as a guide, identifying best practices for both TPK and TCK in standard subject areas is (will be?) critical, ideally leading to improved teacher comfort, confidence, and self-efficacy.

Follow-up communications with teachers should help provide insight into benefits educators receive from using computational thinking and what might be done to improve interest in and the utility of computational thinking. These communications, initiated after teachers have used computational thinking in their classrooms, can provide insight into teacher comfort levels, difficulties, and potential suggestions for improving the professional development process.

Conclusions

All findings in this study indicate a need for education and professional development opportunities for teachers. Given the relatively recent introduction of computational thinking into K-5 curricula, it is not surprising educators lack an understanding of computational thinking or the insight to effectively use computational thinking in their classrooms. Creating opportunities for teachers to see the potential of computational thinking and to learn how to successfully introduce it into their classrooms will be key to the inclusion of computational thinking content into K-5 curricula.

Professional development sessions will have to improve the basic level of teacher understanding of computational thinking, as well as demonstrate the importance of computational thinking, and how it can benefit students. Showing teachers the utility which can be derived from including computational thinking into curriculum skills should help increase teacher interest, justifying the time and effort needed to learn new skills.

Work must also be done to address a message to administrations, both at the school level as well as the district level. Professional development opportunities will require buy-in not just from educators, but also from administrators, in terms of teacher release time, as well as academic and technical support. Benefits to administrators and their schools for supporting educators in this endeavor need to be clearly defined, with supporting documentation, providing administrators insight into why it is important to commit resources to allowing teachers to fully engage in learning to use computational thinking in their classrooms. It is also paramount that administrators understand benefits students derive from acquiring computational thinking skills; such as how computational thinking can improve problem solving skills. And as students learn to use computers as problem-solving tools, computational thinking skills can help students move from being consumers of content to having the ability to create and implement their own solutions to various problems they will encounter in their academic careers, and beyond.

In an ever-increasing digital world, it is important that today's students are prepared for whatever is next in life. In no way is the goal of introducing computational thinking into K-5 curricula an attempt to turn students into future computer science majors. However, it does seem reasonable to think that in the future, students will be expected to analyze data and create graphs, using spreadsheets and/or databases (all forms of programming) more often and earlier than ever. Students will be expected to work, and thrive, in a digital world. The introduction of computational thinking can help facilitate learning across all subjects, while also helping to prepare students for life beyond their formal education.

APPENDICES

Appendix A

Computing Attitude Survey

You are invited to participate in a research study titled *Identifying Barriers to the Implementation of Computational Thinking in the K-5 Classroom*. The study is being done by Tom Stokke as part of his dissertation from the University of North Dakota.

The purpose of this research study is to determine if K-5 teachers know what computational thinking is, are they interested in adding it into their curriculum, and if not, why not? The survey will take you approximately 15 minutes to complete. Your participation in the study is voluntary, and you may withdraw at any time.

We believe there are no known risks associated with this research study. The survey has been approved by the Institutional Review Board at the University of North Dakota (UND IRB 201805-309). All data will remain confidential.

Thank you for your time and consideration.

PART I

First, are you familiar with the concept of computational thinking?
_____ Yes
_____ No

If yes, how would you explain the concept of computational thinking?

____

____

____

If no, what do you think computational thinking means?

____

____

____

PART II

Next, I would like you to provide your level of agreement with the following statements

	Strongly Disagree	Disagree	Agree	Strongly Agree
1. Computational thinking involves using computers to solve problems				
2. Computational thinking involves thinking logically to solve problems				
3. Computational thinking is understanding how computers work				
4. Computational thinking involves abstracting general principles and applying them to other situations				
5. I do not think it is possible to apply computing knowledge to solve other problems				
6. I am not comfortable with learning computing concepts				
7. I can achieve good grades (C or better) in computing courses				
8. I can learn to understand computing concepts				
9. I do not use computing skills in my daily life				
10. I doubt that I have the skills to solve problems by using computer applications				
11. I think computer science is boring				
12. The challenge of solving problems using computer science appeals to me				
13. I think computer science is interesting				

14. I will voluntarily take computing courses if I were given the opportunity				
15. Computational thinking can be incorporated in the classroom by using computers in the lesson plan				
16. Computational thinking can be incorporated in the classroom by allowing students to problem solve				
17. Knowledge of computing will allow me to [be] more effective in the classroom				
18. My career goals do not require that I learn computing skills				
19. I expect that learning computing skills will help me to achieve my career goals				
20. I hope that my future career will require the use of computing concepts				
21. Having background knowledge and understanding of computer science is valuable in and of itself				
22. Trying to introduce computing concepts will distract students from the task at hand				
23. Using computing technologies can increase student interest in the material				
24. I worry that my students will be more adept at using computing technologies than I am				

PART III

Do you believe computational thinking has a place in the following content areas?

	Don't know	Disagree	Agree
English			
Please explain your response			
Mathematics			
Please explain your response			
Social Studies			
Please explain your response			
Science			
Please explain your response			

Next, please use the space below to share anything (positive or negative) you would like to say about the use of computational thought in the K-5 classroom?

PART IV

Finally, I would like to ask you a few demographic based questions.

Demographics	
What grade are you currently teaching?	Kindergarten (1) 1st (2) 2nd (3) 3rd (4) 4th (5) 5th (6)
Years of teaching experience, including the current year	Dropdown list 1-40
Years teaching your current grade, including the current year.	Dropdown list 1-40
Gender	Male (1) Female (2) Other (3) Prefer not to say (4)
Highest degree obtained. – dropdown list	Bachelors (1) Masters (2) Doctorate (3)

THANK YOU FOR TAKING THE TIME FOR FILLING OUT THE SURVEY. THE DATA COLLECTED WILL BE USED TO INFORM K-5 CURRICULUM DEVELOPMENT

Appendix B

Codes Identified From Open-Ended Responses

Codes	Count
Allows students to display strengths	1
Appreciative of tech support	1
Beneficial to students	15
Computational thinking does not require a computer	2
Concerns about the care or misuse of technology	1
Concerns about too much time with technology	1
Curriculum concerns	3
Digital divide concerns	1
Not sure what CT is	4
Positive element in the classroom	1
Promote problem solving skills	5
Promote student confidence	1
Promote student engagement	2
Required skill for future	6
Skills do not generalize for the students	1
Students are willing to take risks	1
Students may not be ready	7
Students need access to technology	4
Students need to learn to use the computer	4
Students need to use computer productively	1
Teachers cannot be replaced	1
Teachers need professional development	2
Teaches perseverance	3
Time concerns	2

REFERENCES

Code.org Advocacy Coalition and the Computer Science Teachers Association. (2018). *2018 State of computer science education: Policy and implementation*. Retrieved from https://code.org/files/2018_state_of_cs.pdf

Abelson, H. & Sussman, G. J. (1986, July). *1A: Overview and Introduction to Lisp* [Video lecture file]. Retrieved from https://ocw.mit.edu/courses/electrical-engineering-and-computer-science/6-001-structure-and-interpretation-of-computer-programs-spring-2005/video-lectures/1a-overview-and-introduction-to-lisp/

Ajzen, I. (1985). From intentions to actions: A theory of planned behavior. In J. Kuhl & J. Beckmann (Eds.), *Action control: From cognition to behavior* (pp. 11-39). New York: Springer-Verlag.

Ajzen, I. (1991). The theory of planned behavior. *Organizational Behavior and Human Decision Processes*, *50*, 179-211.

Ajzen, I. (2015). Consumer attitudes and behavior: The theory of planned behavior applied to food consumption decisions. *Italian Review of Agricultural Economics*, *70*(2), 121-138.

American Association for Employment in Education. (2016). *Educator supply and demand report 2015-16: Executive summary*. Retrieved from http://www.aaee.org/resources/Documents/2015-16_AAEE_Supply_Demand_Summary.pdf

Bandura, A. (1982). Self-efficacy mechanism in human agency. *American Psychologist*, *37*(2), 122−147.

Bogost, I. (2008). The rhetoric of video games. In K. Salen (Ed.), *The ecology of games: Connecting youth, games, and learning* (pp. 117-139). Cambridge, Mass: MIT Press.

Briggs, A., & Snyder, L. (2012, June). Computer science principles and the CS 10K initiative. *ACM Inroads*, *3*(2), 29-31.

Brinkerhoff, J. (2006). Effects of a long-duration, professional development academy on technology skills, computer self-efficacy, and technology integration beliefs and practices. *Journal of Research on Technology in Education*, *39*(1), 22-43.

Carnegie Mellon University. (1999-2017). *Our history*. Retrieved from http://www.alice.org/about/our-history/

Center for Computational Thinking. (n.d.). *What is computational thinking?* Retrieved from http://www.cs.cmu.edu/~CompThink/.

Chiazzese, G., Fulantelli, G., Pipitone, V., & Taibi, D. (2017, October). Promoting computational thinking and creativeness in primary school children. *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality*, *TEEM 2017*, Article No. 6. doi:10.1145/3144826.3145354

Chitu, A. (2007, November 5). *Google launches Android, an open mobile platform*. Retrieved from http://googlesystem.blogspot.com/2007/11/google-launches-android-open-mobile.html

Cho, S., Pauca, P., Johnson, D., & James, Y. (2014, March). Computational thinking for the rest of us: A liberal arts approach to engaging middle and high school teachers with computer science students. In M. N. Ochoa & M. Searson (Eds.), *Site 2014: Society for Information*

Technology & Teacher Education International Conference (25th international conference; pp. 79-86). Waynesville, NC: Association for the Advancement of Computing in Education (AACE).

College Board, The. (2019a). *About AP computer science principles*. Retrieved from https://advancesinap.collegeboard.org/stem/computer-science-principles

College Board, The. (2019b). *AP data – archived data 2012*. Retrieved from http://research.collegeboard.org/programs/ap/data/archived/2012

Google. (n.d.). *Computational thinking for educators: What is computational thinking?* Retrieved from https://computationalthinkingcourse.withgoogle.com/unit

Creswell, J. W. (2012). *Educational research: Planning, conducting, and evaluating quantitative and qualitative research* (4th ed.). Boston, MA: Pearson.

Creswell, J. W. (2013). *Qualitative inquiry & research design: Choosing among five approaches* (3rd ed.). Los Angeles, CA: SAGE.

Creswell, J. W. (2016). *30 essential skills for the qualitative researcher*. Los Angeles, CA: SAGE

Creswell, J. W., & Creswell, J. D. (2017). *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed.). Los Angeles, CA: SAGE.

Creswell, J. W., & Plano Clark, V. L. (2011). *Designing and conducting mixed methods research* (2nd ed.). Los Angeles: SAGE Publications.

Cuny, J., Snyder, L., & Wing, J. M. (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript, referenced in http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Denning, P. J. (1981, June). ACM president's letter: Eating our seed corn. *Communications of the ACM*, *24*(6), 341-343. doi:10.1145/358669.358672

Denning, P. J., & Freeman, P. A. (2009, December). Computing's paradigm. *Communications of the ACM*, *52*(12), 28-30.

diSessa, A. A. (2001). *Changing minds: Computers, learning, and literacy*. Cambridge, Mass: MIT Press.

Dubner, S. J. (Produced by Tam, S.). (2018, January 10). *How to be a modern Democrat — and win (Ep. 313)* [Freakonomics podcast]. Retrieved from http://freakonomics.com/podcast/modern-democrat-win/

Education Standards and Practices Board. (2017, September 29). *License Codes/K-12 Courses*. Retrieved from https://www.nd.gov/espb/sites/www/files/documents/License-Code-Manual-2017-2018%20-Final.pdf

Elliott, V. (2018). Thinking about the coding process in qualitative data analysis. *The Qualitative Report*, *23*(11), 2850-2861.

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 – Final report of the CSTA teacher certification task force*. New York: The Computer Science Teachers Association. Retrieved from https://www.researchgate.net/publication/304674511_Ensuring_Exemplary_Teaching_in_an_Essential_Discipline_Addressing_the_Crisis_in_Computer_Science_Teacher_Certification_-_Final_Report_of_the_CSTA_Teacher_Certification_Task_Force

Fan, W., & Yan, Z. (2010). Factors affecting response rates of the web survey: A systematic review. *Computers in Human Behavior*, *26*(2), 132-139.

Fisher, T. (2006). Educational transformation: Is it, like 'beauty', in the eye of the beholder, or will we know it when we see it? *Education and Information Technologies*, *11*(3-4), 293-303.

Fishman, B., & Davis, E. (2006). Teacher learning research and the learning sciences. In R. K. Sawyer (Ed.), *The Cambridge Handbook of the Learning Sciences* (pp. 535-550). New York: Cambridge University Press.

Franke, B., Century, J., Lach, M., Wilson, C., Guzdial, M., Chapman, G., & Astrachan, O. (2013, March). Expanding access to K-12 computer science education: Research on the landscape of computer science professional development. In R. McCauley, T. Camp, P. Tymann, J. D. Dougherty, & K. Nagel (Eds.), *SIGCSE'13 Proceedings of the 44th ACM technical symposium on computer science education* (pp. 541-542). New York, NY: ACM.

Friedman, T. L. (2005). *The world is flat: A brief history of the twenty-first century*. New York: Farrar, Straus and Giroux.

Furst, M., Isbell, C., & Guzdial, M. (2007, March). Threads™: How to restructure a computer science curriculum for a flat world. *ACM SIGCSE Bulletin*, *39*(1), 420-424. doi:10.1145/1227504.1227456

German, K. (2011, August 2). *A brief history of Android phones*. Retrieved from https://www.cnet.com/news/a-brief-history-of-android-phones/

Google for Education. (2015, July 8). *Exploring computational thinking: CT overview*. Retrieved from https://edu.google.com/resources/programs/exploring-computational-thinking/#!ct-overview

Groves, R. M., Presser, S., & Dipko, S. (2004, March). The role of topic interest in survey participation decisions. *Public Opinion Quarterly*, *68*(1), 2-31. doi:10.1093/poq/nfh002

Guzdial, M. (2008, August). Education: Paving the way for computational thinking. *Communications of the ACM*, *51*(8), 25-27.

Guzdial, M. (2010, August 12). Go to the data: Two stories of (really) computational thinking [Web blog post]. Retrieved from https://computinged.wordpress.com/2010/08/12/go-to-the-data-two-stories-of-really-computational-thinking/

Guzdial, M. (2012, July 17). Yum, tasty seed corn: Where will we find the teachers? [Web blog post]. Retrieved from https://computinged.wordpress.com/2012/07/17/yum-tasty-seed-corn-where-will-we-find-the-teachers/

Guzdial, M. (2019, May 3). What's NOT computational thinking? Curly braces, x = x + 1, and else [Web blog post]. Retrieved from https://computinged.wordpress.com/2019/05/03/whats-not-computational-thinking-maybe-x-x-1/

Guzdial, M., & Soloway, E. (2003, June). Computer science is more important than calculus: The challenge of living up to our potential. *ACM SIGCSE Bulletin*, *35*(2), 5-8. doi:10.1145/782941.782943

Harvey, B. (n.d.). *Berkeley Logo (UCBLogo)*. Retrieved from https://people.eecs.berkeley.edu/~bh/logo.html

Hubwieser, P., Armoni, M., Giannakos, M. N., & Mittermeir, R. T. (2014, June). Perspectives and visions of computer science education in primary and secondary (k-12) schools. *ACM Transactions on Computing Education (TOCE)*, *14*(2), Article 7.

Computer Science Teachers Association and the International Society for Technology in Education. (2011). *Computational thinking teacher resources* (2nd ed.). Retrieved from https://id.iste.org/docs/ct-documents/ct-teacher-resources_2ed-pdf.pdf?sfvrsn=2

Johns, R. (2005). One size doesn't fit all: Selecting response scales for attitude items. *Journal of Elections, Public Opinion & Parties, 15(2)*, 237-264. doi:10.1080/13689880500178849

Kafai, Y. B., & Burke, Q. (2013, September). Computer programming goes back to school. *Phi Delta Kappan*, *95*(1), 61-65.

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

Kafura, D., Bart, A. C., & Chowdhury, B. (2015, June). Design and preliminary results from a computational thinking course. In *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 63-68). New York: ACM.

Kan, M. P. H., & Fabrigar, L. R. (2017). Theory of Planned Behavior. In V. Zeigler-Hill & T. K. Shackelford (Eds.), *Encyclopedia of personality and individual differences* [Advance online edition]. Retrieved from https://link.springer.com/content/pdf/10.1007%2F978-3-319-28099-8_1191-1.pdf

Krauss, J., & Prottsman, K. (2016). *Computational Thinking and Coding for Every Student: The Teacher's Getting-Started Guide*. Corwin Press.

Kynigos, C., & Grizioti, M. (2018). Programming approaches to computational thinking: Integrating turtle geometry, dynamic manipulation and 3D space. *Informatics in Education*, *17*(2), 321-340. Retrieved from https://files.eric.ed.gov/fulltext/EJ1195612.pdf

LaMorte, W. W. (2018, August 2). *Behavioral change models: The Theory of Planned Behavior*. Retrieved from http://sphweb.bumc.bu.edu/otlt/MPH-Modules/SB/BehavioralChangeTheories/BehavioralChangeTheories3.html

Lavrakas, P. J. (Ed.). (2008). *Encyclopedia of survey research methods* (Vol. 2). Thousand Oaks, CA: SAGE Publications.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . . & Werner, L. (2011, March). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32-37. doi:10.1145/1929887.1929902

Lifelong Kindergarten Group. (n.d.). *About Scratch*. Retrieved from https://scratch.mit.edu/about

Locke, L. F., Spirduso, W. W., & Silverman, S. J. (2014). *Proposals that work: A guide for planning dissertations and grant proposals* (6th ed.). Thousand Oaks, CA: SAGE Publications.

Lockwood, J., & Mooney, A. (2017, March). *Computational thinking in education: Where does it fit? A systematic literary review*. Retrieved from https://arxiv.org/ftp/arxiv/papers/1703/1703.07659.pdf

Lu, J. J., & Fletcher, G. H. L. (2009, March). Thinking about computational thinking. *ACM SIGCSE Bulletin*, *41*(1), 260-264. doi:10.1145/1539024.1508959

Maio, P. (2016, August 23). *New computer science course's challenge is finding qualified teachers to teach it*. Retrieved from https://edsource.org/2016/new-computer-science-courses-challenge-is-finding-qualified-teachers-to-teach-it

Marder, M. (2016, October 25). Is STEM education in permanent crisis. *Education Week*. Advance online publication. Retrieved from https://www.edweek.org/ew/articles/2016/10/26/is-stem-education-in-permanent-crisis.html

Martinez, L. S., & Lewis, N. (2016). The moderated influence of perceived behavioral control on intentions among the general U.S. population: Implications for public communication campaigns. *Journal of Health Communication*, *21*(9), 1006–1015. doi:10.1080/10810730.2016.1204378

Mateas, M. (2005). Procedural literacy: educating the new media practitioner. *On the Horizon*, *13*(2), 101-111.

McPeak, A. (2018, January 24). *A brief history of web browsers and how they work*. Retrieved from https://crossbrowsertesting.com/blog/test-automation/history-of-web-browsers/

Microsoft. (2019). *Kodu*. Retrieved from https://www.microsoft.com/en-us/research/project/kodu/

Mouza, C., Yang, H., Pan, Y.-C., Ozden, S. Y., & Pollock, L. (2017). Resetting educational technology coursework for pre-service teachers: A computational thinking approach to the development of technological pedagogical content knowledge (TPACK). *Australasian Journal of Educational Technology*, *33*(3), 61-76. doi:10.14742/ajet.3521

National Center for Education Statistics. (2017). *Digest of education statistics*. Retrieved May 20, 2019, from https://nces.ed.gov/programs/digest/d17/tables/dt17_209.20.asp?current=yes

National Council of Teachers of Mathematics. (2019). *Principles, standards, and expectations.* Retrieved from https://www.nctm.org/Standards-and-Positions/Principles-and-Standards/Principles,-Standards,-and-Expectations/

National Governors Association. (2007). *Building a science, technology, engineering and math agenda* [Innovation American initiative]. Washington, DC: Author. Retrieved from https://files.eric.ed.gov/fulltext/ED496324.pdf

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

Nesiba, N., Pontelli, E., & Staley, T. (2015, October). DISSECT: Exploring the relationship between computational thinking and English literature in K-12 curricula. *2015 IEEE Frontiers in Education Conference, El Paso, Texas, USA (FIE 2015)*, 249-256. Retrieved from http://fie-conference.org/sites/fie-conference.org/files/1570093959.pdf

Next Generation Science Standards. (n.d.). *Read the standards*. Retrieved March 15, 2019, from https://www.nextgenscience.org/search-standards

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

Papert, S. (1996). An exploration in the space of mathematics educations. *International Journal of Computers for Mathematical Learning*, *1*(1), 95-123.

Perlis, A. J. (1982, September). Special feature: Epigrams on programming. *ACM Sigplan Notices*, *17*(9), 7-13.

Ponticell, J. A. (2003, September). Enhancers and inhibitors of teacher risk taking: A case study. *Peabody Journal of Education*, *78*(3), 5-24.

Popper, B. (2017, May 17). *Google announces over 2 billion monthly active devices on Android*. Retrieved from https://www.theverge.com/2017/5/17/15654454/android-reaches-2-billion-monthly-active-users

President's Council of Advisors on Science and Technology (US). (2010, September). *Report to the president: Prepare and inspire: K-12 education in science, technology, engineering, and math (STEM) for America's Future* (Executive Report). Washington, DC: Executive Office of the President, President's Council of Advisors on Science and Technology.

Randolph, J. J. (2007). *Computer science education research at the crossroads: A methodological review of computer science education research: 2000-2005* (Doctoral dissertation, Utah State University). Retrieved from https://archive.org/details/randolph_dissertation

Resnick, M., Bruckman, A., & Martin, F. (1996, September/October). Pianos not stereos: Creating computational construction kits. *interactions*, *3*(5), 40-50. Retrieved from https://llk.media.mit.edu/papers/pianos.html

Rienties, B., Brouwer, N., & Lygo-Baker, S. (2013, January). The effects of online professional development on higher education teachers' beliefs and intentions towards learning facilitation and technology. *Teaching and Teacher Education*, *29*, 122-131. doi:10.1016/j.tate.2012.09.002

Sadaf, A., Newby, T. J., & Ertmer, P. A. (2012a). Exploring factors that predict preservice teachers' intentions to use Web 2.0 technologies using decomposed theory of planned behavior. *Journal of Research on Technology in Education*, *45*(2), 171-196.

Sadaf, A., Newby, T. J., & Ertmer, P. A. (2012b). Exploring pre-service teachers' beliefs about using Web 2.0 technologies in K-12 classroom. *Computers & Education*, *59*(3), 937-945.

Sanford, G. (1996-2019). *iPhone*. Retrieved from https://apple-history.com/iphone

Sedgwick, P. (2014, March 26). Cross sectional studies: Advantages and disadvantages. *BMJ: British Medical Journal*, *348*, g2276. doi:10.1136/bmj.g2276

Settle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., & Wildeman, B. (2012, July). Infusing computational thinking into the middle- and high-school curriculum. *Proceedings of the 17th ACM annual conference on Innovation and technology in computer science education, Haifa, Israel, ITiCSE'12*, 22-27. New York, NY: ACM. doi:10.1145/2325296.2325306

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review*, *22*, 142-158.

Smarkola, C. (2008, May). Efficacy of a planned behavior model: Beliefs that contribute to computer usage intentions of student teachers and experienced teachers. *Computers in Human Behavior*, *24*(3), 1196-1215. doi:10.1016/j.chb.2007.04.005

Snyder, L., Astrachan, O., Briggs, A., & Cuny, J. (n.d.). *Seven big ideas of computer science*. Retrieved from https://csprinciples.cs.washington.edu/sevenbigideas.html

Squeakland: Home of squeak etoys. (n.d.). *Etoys is* . . . Retrieved from http://www.squeakland.org/

STEM Education Act of 2015, Pub. L. No. 114-59, 129 Stat. 540 (2015).

Stephenson, C., & Barr, V. (2012). Defining computational thinking for k-12. In P. Phillips (Ed.), *Computer science K–8: Building a strong foundation* (pp. 4-5). New York, NY: Computer Science Teachers Association.

Sugar, W., Crawley, F., & Fine, B. (2004, October). Examining teachers' decisions to adopt new technology. *Educational Technology and Society*, *7*(4), 201-213. Retrieved from https://pdfs.semanticscholar.org/862d/0a4c870323a7688e375cf38ad9675caf0116.pdf

Tang, D. (2016, October 31). Learn to affect customer behavior using the Theory of Planned Behavior [Web blog post]. Retrieved from http://flevy.com/blog/learn-to-affect-customer-behavior-using-the-theory-of-planned-behavior/

Taylor, S., & Todd, P. A. (1995). Understanding information technology usage: A test of competing models. *Information Systems Research*, *6*(2), 144–176. Retrieved from http://home.business.utah.edu/actme/7410/TaylorTodd.pdf

Teo, T. (2009). Modeling technology acceptance in education: A study of preservice teachers. *Computers and Education*, *52*(1), 302−312.

Teo, T., Lee, C. B., & Chai, C. S. (2008). Understanding preservice teachers' computer attitudes: Applying and extending the Technology Acceptance Model. *Journal of Computer Assisted Learning*, *24*(2), 128−143.

UNESCO Institute for Statistics (UIS). (n.d.). *Education : Percentage of female teachers by teaching level of education*. Retrieved May 20, 2019, from http://data.uis.unesco.org/index.aspx?queryid=178

van Driel, J. H., Beijaard, D. D., & Verloop, N. (2001). Professional development and reform in science education: The role of teachers' practical knowledge. *Journal of Research in Science Teaching*, *38*(2), 137-158.

Vee, A. (2013). Understanding computer programming as a literacy. *Literacy in Composition Studies*, *1*(2), 42-64.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016, February). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, *25*(1), 127-147.

Wilson, C. (2015, March). Hour of code---a record year for computer science. *ACM Inroads*, *6*(1), 22-22.

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). *Running on empty: The failure to teach K-12 computer science in the digital age*. New York, NY: Association for Computing Machinery. Retrieved from https://runningonempty.acm.org/fullreport2.pdf

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

Wing, J. M. (2008, July 31). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, *366*(1881), 3717-3725.

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

Yadav, A., Krist, C., Good, J., & Caeli, E. N. (2018). Computational thinking in elementary classrooms: Measuring teacher understanding of computational ideas for teaching science. *Computer Science Education*, *28*(4), 371-400.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., & Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE)*, *14*(1), Article 5.

Yushau, B. (2006, December). Computer attitude, use, experience, software familiarity and perceived pedagogical usefulness: The case of mathematics professors. *Eurasia Journal of Mathematics, Science and Technology Education*, *2*(3), 1−7.