DISSERTATION

COMPUTATIONAL THINKING: AN INVESTIGATION OF THE EXISTING SCHOLARSHIP AND RESEARCH

Submitted by

Andrea Elizabeth Weinberg

School of Education

In partial fulfillment of the requirements

For the Degree of Doctor of Philosophy

Colorado State University

Fort Collins, Colorado

Spring 2013

Doctoral Committee:

Advisor: R. Brian Cobb

Leonard Albright
Paul Kehle
Jerry Vaske

ABSTRACT

COMPUTATIONAL THINKING: AN INVESTIGATION OF THE EXISTING SCHOLARSHIP AND RESEARCH

Despite the prevalence of computing and technology in our everyday lives and in almost every discipline and profession, student interest and enrollment in computer science courses is declining. In response, computer science education in K-12 schools and universities is undergoing a transformation. Computational thinking has been proposed as a universal way of thinking with benefits for everyone, not just computer scientists. The focus on computational thinking moves beyond computer literacy, or the familiarity with software, to a way of thinking that benefits everyone. Many see computational thinking as a way to introduce students to computer science concepts and ways of thinking and to motivate student interest in computer science.

The first part of this dissertation describes a study in which the researcher systematically examined the literature and scholarship on computational thinking since 2006. The aim was to explore nature and extent of the entire body of literature and to examine the theory and research evidence on computational thinking. Findings reveal that there has been a steady increase in the popularity of the concept of computational thinking, but it is not yet developed to the point where it can be studied in a meaningful way. An examination of the research evidence on computational thinking found inadequacies in the conceptual characteristics and the reporting of studies. Weaknesses were identified in the theoretical conceptualization of interventions, definitions of key concepts, intervention descriptions, research designs, and the presentation of findings. Recommendations for bolstering the research evidence around this burgeoning concept are presented, including collaboration between computer scientists and educational researchers to apply social science research methods to conduct robust studies of computational thinking interventions.

The second part of this dissertation describes how computational thinking is currently incorporated into K-12 educational settings. The bulk of the literature on computational thinking describes ways in which programs promote this way of thinking in students. The K-12 programs that encourage computational thinking are classified, described, and discussed in a way that is intended to be meaningful for K-12 educators and educational researchers. Potential barriers and factors that might enable educators to use each category of interventions are discussed.

TABLE OF CONTENTS

CHAPTER 1: INTRODUCTION

It is nearly impossible to overemphasize the role of computing and technology in our everyday lives; computers, computational devices, and technology are so pervasive that that civic, economic, and personal participation are predicated on technological skills and knowledge. Reliance on technology goes beyond the daily use of digital electronics and technological applications in the ‘hard sciences’. Technology is essential in fields are diverse as agriculture, business, journalism, and social sciences. The influence of computing and technology applications extends beyond the borders and boundaries of the industrialized nations; they are used to help understand and address social problems across the world. In order for nations, including the U.S., to remain economically competitive in the increasingly global environment, a highly educated workforce skilled in computer science and technology is essential.

Academic and career achievement in an increasing number of disciplines is dependent on the ability to apply technology, yet many students are ill-equipped to meet this challenge. Many have noted the misalignment between computer science education and the ever-increasing digital world in which we live. A recent report published by the Association for Computing Machinery points out that K-12 computer science education paradoxically shrinking as the functions, influences, and significance of computer science in society are expanding (Wilson, Sudol, Stephenson, & Stehlik, 2010). In the five years prior to this report’s release, the number of computer science courses taught in secondary courses decreased by almost 20%, and high schools offering Advanced Placement Computer Science courses decreased by 35%. The field of computer science education is currently not keeping pace with the expanding technological environment.

A number of human capital issues currently exist in computer science. Foremost among these is the need for more computer science graduates. There is a lack of diversity among computer science graduates as evidenced by the underrepresentation of minority and female students in graduate computer science programs (Computing Research Association, 2010). A second issue is the scarcity of opportunities for U.S. students develop computational skills and explore how computational competencies may propel them toward careers of interest (Computer Science Teachers Association [CSTA], 2005, 2009). Course offerings are limited, teachers often are not adequately trained, rigorous high school computer science courses are rare, and introductory courses are often unappealing and unattractive (Repenning, Webb, & Ioannidou, 2010). Innovation is needed to develop curriculum for use in a wide range of computer science courses, along with professional development opportunities to prepare teachers to meet the needs of students (Wilson & Harsha, 2009).

One tangible direction for change can be found in the computational thinking movement. With the seminal article in 2006, Wing attempted to liken computational thinking to the basic skills of reading, writing, and arithmetic. It is believed that computational thinking would enable individuals to more effectively navigate today's society where technology is unavoidable. First, a focus on computational thinking in K-12 education would encourage equitable access to technological skills, devices, and other resources because it would enhance personal empowerment as individuals are taught how to apply computational thinking to their daily lives. Second, incorporating computational thinking into K-12 education would raise student interest in information technology, computer science, and other technologically oriented professions. Third, an increase would help maintain and enhance the competitiveness of the U.S. from an economic standpoint by better preparing students to enter the internationally competitive work force (Committee for the Workshops on Computational Thinking, 2010).

Examples of computational thinking are all around us. Sorting is a common example of computational thinking. Both sorting a list and using a sorted list involve computational thinking. By sorting items, we are able to locate items quickly and efficiently. In addition, duplicate items are easy to locate because they end up side-by-side, and extreme cases or potential data entry errors are easily identified because they are at the beginning or end of the list. Lists can be sorted a number of ways (e.g., alphabetically, numerically). A variety of methods (algorithms) can be used to sort items, and each of these methods requires computational thinking. A description of three of these methods follows. The *selection sort* method involves the following process: the item with the smallest (or largest) value is located and put it in the first position, then the next-smallest (or largest) item is found and put it in the second position, and so on until the entire list is sorted. Throughout this process, the list is divided into two parts: the part of the list that has been sorted and the part that has not yet been put in order. A second method is the *quicksort,* in which the list is divided into two smaller sub-lists, then these sub-lists are recursively sorted. For example, if a stack of nametags is to be alphabetized, one might use the last names on the nametags to 1) create two stacks: A-N and M-Z, 2) separate the A-N stack into A-G and H-N, 3) separate the A-G stack into A-C and D-G, 4) put the A-C stack in alphabetical order. This process would continue for all sub-lists until all the nametags were in order. Another method is the *bubble sort*, which involves moving through the list repeatedly, each time comparing two side-by-side objects at a time and swapping the pair when one is in the wrong order. When one moves through the entire list without making any swaps, the list is in order.

After a list has been sorted, multiple methods can be used to find an individual item. A linear approach involves beginning the search at the top and moving down the list until the item of interest is found. A binary search begins in the middle of the sorted list. If the item of interest is located above the middle value, then the middle value in the top half of the items is located. This value is compared to the item of interest, and the process repeats until the item of interest is found on the list.

This description of sorting and locating an item on a sorted list is intended to provide an example of how we might consider the world in computational terms. This exemplifies the description of computational thinking offered by the Value of Computational Thinking Across Grade Levels, which says

> [Computational thinking] begins with learning to see opportunities to compute something, and it develops to include such considerations as computational complexity, utility of approximate solutions, computational resource implications of different algorithms, selection of appropriate data structures and the ease of coding, maintaining, and using the resulting program. Computational thinking is applicable across disciplinary domains because it takes place at a level of abstraction where similarities and differences can be seen in terms of the computational strategies available. A person skilled in computational thinking is able to harness the power of computing to gain insights. At its best, computational thinking is multi-disciplinary and cross-disciplinary thinking with an emphasis on the benefits of computational strategies to augment human insights. Computational thinking is a way of looking at the world in terms of how information can be generated, related, analyzed, represented, and shared courses (Cozzens, Kehle, & Garfunkel, 2010).

The need to sorting a list and locate an item on the list is not a practice that is exclusive to any single discipline or category of disciplines. The need to conduct these activities extends to all disciplines and to tasks one might face in their daily personal lives. This example is one of many that could be used to demonstrate the need to better understand how computational thinking skills are learned and taught.

Purpose of the Dissertation

The purpose of this dissertation is to conduct a disciplined review of computational thinking literature, to examine the nature and extent of research evidence found within this literature, and to offer suggestions that could enable educators and researchers to collaborate and in efforts to both incorporate computational thinking into K-12 classrooms and study how students learn to think computationally. This current inquiry is driven by the need of researchers, scholars, and educators to have a unified understanding of the definition and a theoretical model or framework underpinning the concept of computational thinking to use as a starting point in developing and studying computational thinking interventions, the development quantitative measures of computational thinking to assess the success of these interventions, and to thoughtfully examine programs and interventions intended to promote computational thinking competencies.

Within the context of the recently NSF-funded (VCTAL) project, the National Science Foundation (NSF) expressed interest in providing support for the full development of an instrument to measure computational thinking. Both the VCTAL reviewers and the NSF Project Officer described the lack of an instrument or process to assess computational thinking, which makes this a fertile topic to consider. This exploratory research will set the stage for future efforts to enhance the rigor, strength, and visibility of both the theoretical model and the computational thinking assessment instrument.

Brief Overview of the Organization of the Dissertation

At the recommendation of my advisor and with the approval of my committee, this dissertation is presented as a set of two journal-ready manuscripts bracketed by an introduction and a conclusion. Manuscript ideas were presented to committee members at the Dissertation Proposal meeting, and modifications were made to the scope and direction as a result of this meeting. Outlines were prepared and approved by all committee members (Appendix A). A brief description of each chapter follows.

Chapter 2: Computational Thinking: An Investigation of the Existing Scholarship and Research

Chapter 2 is a manuscript written for the *Computer Science Education* journal. This journal's most recent call for papers and instructions for authors can be found in Appendix B. *Computer Science Education* is one of very few journals devoted to computer science teaching and learning. It publishes historical analysis and theoretical, analytical, or philosophical material. The study described in this manuscript study takes a systematic, disciplined approach as it first provides a broad look at the computational thinking literature, and then examines the nature and extent of research evidence found within this literature. The aims are to survey all of the scholarship on computational thinking from 2006 to June 2011, and to conduct a review to identify the nature and extent of the research evidence within this body of literature.

First, all the literature and scholarship on computational thinking since 2006 was examined and used to answer questions about the authors and the characteristics of each piece in order to better understand the literature base as a whole. To accomplish this, articles and publications on computational thinking were identified using search terms derived from the two most concrete and widely accepted descriptions of computational thinking available. A variety of bibliographic databases were searched, hand searches performed, as well as other electronic searches. A systematic screening process was used to determine eligibility for inclusion. If articles met the initial subject and date screening criteria, they were included in the literature map. Each of these articles was coded using a primary coding framework in which demographic information and data related to the primary purpose of the article was extracted.

Second, those computational thinking articles that describe research or evaluation studies were scrutinized further. A secondary coding framework was applied to this subset of the literature. Details of conceptual and methodological features were coded. No attempts are made to systematically appraise the studies, nor are findings synthesized across the studies. For these reasons, inclusion criteria for the scoping review can be quite broad; any article that includes data can be included, regardless of methods employed or study quality. Findings were described and conclusions were drawn in each section, and overall implications were discussed in the manuscript's conclusion.

Chapter 3: Computational Thinking: What Is It, How Do We Teach It, and How Do We Assess It?

The second manuscript was driven in large part by the findings from the first. It is applied in nature and will be submitted to SIGCSE for consideration to present the paper at the 2013 *SIGCSE Conference*. The call for participation and formatting instructions are included in Appendix C. This manuscript is intended to serve as an introduction to computational thinking for 'the rest of us'. In this case, 'the rest of us' includes elementary teachers, secondary teachers, school administrators, and educational researchers. This paper is intended to reach the audience who needs to be involved in the next step, which includes improving access to computer science and computational thinking in K-12 schools. This manuscript includes a discussion of the concept of computational thinking and its origins, an introduction to various programs and initiatives that aim to encourage computational thinking in classrooms, and a final section targeting educational researchers that discusses how computational thinking is and/or could be studied.

Chapter 4: Conclusion

Chapter 4 includes a brief synthesis and discussion of the key findings related to computational thinking. Following a discussion of this dissertation's limitations and the lessons learned, the researcher offers recommendations for the field of computer science education. Finally, potential next steps for the researcher and others interested in the computer science education field are presented.

REFERENCES

Committee for the Workshops on Computational Thinking. (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington, D.C.: National Academies Press.

Computer Science Teachers Association (CSTA). (2005) The new educational imperative: Improving high school computer science education. Retrieved November 15, 2010 from http://csta.acm.org/communications/sub/DocsPresentationFiles/White Paper07_06.pdf

Computer Science Teachers Association (CSTA). (2009). High school computer science survey. Retrieved November 15, 2010 from Research/sub/CSTAResearch.html

Computing Research Association (CRA). (2010). CRA Taulbee Survey. Retrieved November 11, 2010 from http://cra.org/resources/taulbee/

Repenning, A., Webb, D., Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 265-269). New York, NY. doi: 10.1145/1734263.1734357

Wilson, C., Harsha, P. (2009). IT policy: The long road to computer science education reform. *Communications of the ACM*, 52(9), 33-35. doi: 10.1145/1562164.1562178

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*. (Research Report). Retrieved from the Association for Computing Machinery website: http://www.acm.org/runningonempty/fullreport.pdf

CHAPTER 2: COMPUTATIONAL THINKING: AN INVESTIGATION OF THE EIXSTING SCHOLARSHIP AND RESEARCH

Introduction

Computers and technology are virtually everywhere. They influence nearly every aspect of our personal, professional, and civic lives including how we communicate, navigate through our physical environment, disseminate and acquire knowledge, and how we collect, store, and analyze information. Academic and career achievement in an increasing number of disciplines is dependent on the ability to use technology, yet most students are ill-equipped to face this challenge. Opportunities for U.S. students to develop computational skills and learn basic technology concepts are scarce. Course offerings are limited and introductory computer science courses at the high school and university levels are often unappealing and unattractive (Computer Science Teachers Association [CSTA], 2009; Computing Research Association, 2010; Repenning, Webb, & Ioannidou, 2010; Stephenson, Gal-Ezer, Haberman, & Verno, 2005). Rigorous, engaging high school computer science courses are rare and K-12 courses often consist of little more than teaching students how to operate computer applications or software (Wilson, Sudol, Stephenson, & Stehlik, 2010).

To empower learners to effectively face modern challenges, the US education system must reconsider how computer science is introduced and taught. Undergraduate courses in an array of disciplines have begun to be overhauled to introduce computer science concepts in compelling ways (e.g., Anewalt, 2008; Heines, Greher, Ruthmann, & Reilly, 2011; Martin et al., 2009). Changes at the K-12 levels have been much more subtle, but there are signs that an increased focus on computer science has begun to catch hold. While computer science is neither a core subject nor a requirement for graduation in any state, nine states now allow computer science courses to count toward graduation requirements in mathematics or science.

Efforts to advance computer science education at the federal, state, and corporate levels advocate a focus on computational thinking. The concept of computational thinking was popularized by Janette Wing in 2006, and the argument that computational thinking is a fundamental and universal skill has been embraced by the computer science community. The National Science Foundation's Computing Education for the 21st Century (CE21) and CISE Pathways to Revitalized Undergraduate Computing Education (CPATH) funding competitions encourage projects that develop computational thinking competencies. The College Board is currently testing a new Advanced Placement course, Computer Science: Principles, around a set of six Computational Thinking Practices. This course will advance students' understanding of computational thinking in order to advance efforts to develop a more competitive 21st century workforce (The College Board, 2011). Most recently, computational thinking was used as a common thread to link the three levels of K-12 learning standards developed by the Computer Science Teachers Association (CSTA, 2011a).

There has been much discourse in recent years about the precise definition of computational thinking (Bundy, 2007; Denning, 2007; Garcia, Lewis, Dougherty, & Jadud, 2010; Henderson, 2009; Committee for the Workshops on Computational Thinking, 2010; Wing, 2006), and the definition continues to be emergent. The National Academies convened workshops in 2009 and 2010 to explore the scope, nature, and pedagogical aspects of computational thinking. While no consensus on these aspects of computational thinking was reached, workshop participants agreed on the existence of computational thinking as a mode of thought with a distinctive character (Committee for the Workshops on Computational Thinking, 2010), and subsequent reports of these workshops widely disseminated the arguments and perspectives of prominent comptuer science education experts.

Two definitions for computational thinking stand out at this point. The first is offered by the CSTA, who defines computational thinking as a problem solving process that includes (a) formulating problems in a way that enables us to use a computer and other tools to help solve them, (b) logically organizing and analyzing data, (c) representing data through abstractions, such as models and simulations, (d) automating solutions through algorithmic thinking, (e) identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources, and (f) generalizing and transferring this problem-solving process to a wide variety of problems (2011b). This definition underwent a review process that included a survey of over 700 experts including computer science teachers, researchers, and practitioners. The vast majority of respondents (*n*=697, 82%) indicated their agreement or strong agreement when asked if CSTA's definition captured the fundamental elements of computational thinking, and a further 9% indicated that the definition was sufficient to use to build consensus in the computer science education community. A second prominent definition is offered by Google's Exploring Computational Thinking initiative, the first large-scale program to provide an operational definition, disseminate resources, and promote discussion among K-12 educators about computational thinking. Google describes a process that includes four computational thinking techniques: decomposition, pattern recognition, pattern generalization and abstraction, and algorithm design (Google, 2011). Both of these definitions depict computational thinking as a series of skills and techniques that prepares a problem for computation, and conceptual parallels can be drawn between components of each definition, as seen in Table 1.

Table 1.
Computational Thinking Definitions

The Computer Science Teachers Association's Operational Definition of Computational Thinking	Google's Exploring Computational Thinking Techniques
• Formulating problems in a way that enables us to use a computer and other tools to help solve them	Decomposition
• Logically organizing and analyzing data	Pattern Recognition
• Representing data through abstractions, such as models and simulations	Pattern Generalization and Abstraction
• Automating solutions through algorithmic thinking • Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources	Algorithm Design
• Generalizing and transferring this problem-solving process to a wide variety of problems	

The CSTA's operational definition of computational thinking includes initial step that describes formulating problems so technology can be used to help solve them is similar to Google's concept of *decomposition*, which involves taking a large, complex problem and breaking it down into smaller and easier/more manageable ones. CSTA next describes the logical organization and analysis of data, which has parallels to Google's *pattern recognition*, or the ability to find similarities or differences that help "make predictions or lead to shortcuts" (Google, 2011). Representing data through abstractions is similar to *pattern generalization and abstraction,* or the ability to remove the details of a problem in order to make a solution that works to solve similar problems. This involves filtering out unnecessary details and designing a solution that can be used to solve similar problems. The CSTA's next two steps of using algorithmic thinking to automate solutions and the analysis of possibilities in order to determine an efficient and effective solution are similar to Google's *algorithm design*, which is described as involves the development of a step-by-step strategy or set of instructions for completing or solving a similar problem. The CSTA definition extends one step beyond this to include the idea of generalizing and transferring this process to diverse problems.

While both represent a processes, each suggests that the individual concepts can be approached and introduced as discrete concepts or skills. Computational thinking is not a stand-alone subject, but rather inherently interdisciplinary in nature because the set of skills and techniques are used help address problems in any discipline.

This recent emphasis on computational thinking has resulted in a surge of literature devoted to the discussion or study of its concepts. So far, there is no evidence of attempts to aggregate, scrutinize, or analyze this body of literature. This study focuses on computational thinking in education. The purpose is to represent the nature of empirical and theoretical literature at a point in time and identify recommendations for future research directions in computational thinking, and promote dialogue about computational thinking. It is intended to be used to computer science educators, policymakers, and researchers as they make decisions that could influence practice, policy, and future research directions.

This review employs a systematic, disciplined approach as it first provides a broad look at the computational thinking literature, and then comprehensively examines the nature and extent of research evidence found within this literature. The broad overview of the computational thinking scholarship offers a preliminary assessment of the scope and size of the literature base. It looks broadly at *all* literature produced on computational thinking and its domains since the term computational thinking was introduced by Janette Wing in 2006. In addition, the comprehensive study described in this paper examines the empirical studies discovered and creates a descriptive “map” of research conducted on the topic. The aim is to examine the studies that have been conducted around interventions to help researchers and educators understand computational thinking and its domains. No synthesis of findings or formal assessments of quality are included, but instead this review provides an overview of the evaluation and research that have been conducted to date.

This type of review was conducted for a variety of reasons. The desire to employ a systematic and transparent process to conduct this review was paramount. Prior to conducting this study, the author had extensive knowledge of the extant literature on computational thinking, so was aware that a rigorous meta-analysis was not an appropriate approach. The research amassed on computational thinking topics does not consist of a sufficient number of heterogeneous research studies that meet the highly selective inclusion guidelines for the statistical analyses conducted in a meta-analysis (Swanson & Deshler, 203). A scoping review was chosen because of the broad map of evidence and literature it create as it explores the extent of the literature in a domain and helps to identify the potential scope of a future systematic review to synthesize the findings (Armstrong, Hall, Doyle, & Walters, 2011).

Method

The aim is to isolate the literature on computational thinking in education and provide a meaningful and comprehensive description of the literature around this concept. The following questions are addressed in this review:

(1) What are the demographic characteristics of the entire set of literature?

(2) What kind of taxonomy might characterize this entire set of literature?

(3) What is the focus of the studies that have been conducted?

(4) How do study authors define computational thinking, and how do these align with the CSTA and Google definitions?

(5) What kinds of interventions are being described and tested?

(6) What outcomes are examined in studies and how are they measured?

A systematic process was used to identify articles and papers on computational thinking. The search parameters and procedures were established a priori, and a thorough, objective, and reproducible search was conducted. The guidelines established by the Cochrane Collaboration and its Cochrane Information Retrieval Methods Group were used as a guide (Hammerstrom, Wade, & Jorgensen, 2010).

Search Strategy and Sources

Search terms were established after an extensive review of the literature and consultations with a content expert, an academic librarian, and a methodologist experienced in systematic reviews. The review presented in this paper relied on a both the CSTA (2011b) and Google Exploring Computational Thinking definitions to arrive at search terms. Search terms extracted from each of these are considered a "domain" of computational thinking for this study. Because of the interdisciplinary nature of computational thinking the search also included the combination of computer science and other disciplinary areas. Search terms used include the following: *computational thinking, thinking, interdisciplinary, multidisciplinary, mathematics, science, biology, physics, reading, writing, journalism, music, art, problem decomposition, pattern recognition, abstraction, algorithm, data representation, simulation, recursive.* The phrase *computer science education* was included to filter unrelated articles.

These search terms were entered along with the data-range parameters (2006 – June 2011) into the following relevant electronic bibliographic databases: ACM Digital Library, ERIC (EBSCO and ProQuest), Web of Science, PsychInfo, ITicse, and Digital Dissertations. This resulted in 6,906 titles and abstracts. The concept was popularized in 2006, and the search was conducted in June 2011, which explains both end of the date-range parameters. Two institutional repositories were searched, but neither resulted in any additional abstracts: Directory of Open Access Repositories, and the Register of Open Access Repositories. There is no reliable way to obtain information about studies that have been conducted but never published (Hammerstrom, et al., 2010), but because of the importance of locating unpublished works and studies extensive non-database searches were conducted. Hand searches were conducted in two journals (*Computer Science Education* and *Journal on Educational Resources in Computing)* and in the Koli Calling proceedings. Papers presented at and published in the Koli Calling proceedings are not included in any electronic database, so the entire contents of the 2006-2011 proceedings for this computing education conference were examined. Hand searches resulted in 49 additional titles and abstracts to include, all from the Koli Calling proceedings. In an attempt to identify ongoing studies, the National Science Foundation list of ongoing projects was searched. While numerous projects that might eventually produce literature relevant to this study were identified, there were no additional papers available at the time of this review. Finally, a Google search for unpublished works resulted in 16 additional pieces of literature considered for inclusion.

These search procedures combined resulted in a total of 6,971 article titles to import into EndNote (Figure 1). Before the inclusion and exclusion criteria could be applied, duplicates were removed. An automatic search for duplicates was hindered by the fact that many identical papers were published in conference proceedings and in the governing body's journals; these were not automatically identified as duplicates because of the variation in the publication venue. After automatic and hand searches for duplicates, a total of 3,465 citations and abstracts were imported into the Evidence for Policy and Practice Information and Co-ordinating Centre's (EPPI-Centre) EPPI-Reviewer 4 (Thomas, Brunton, & Grazosi, 2010).

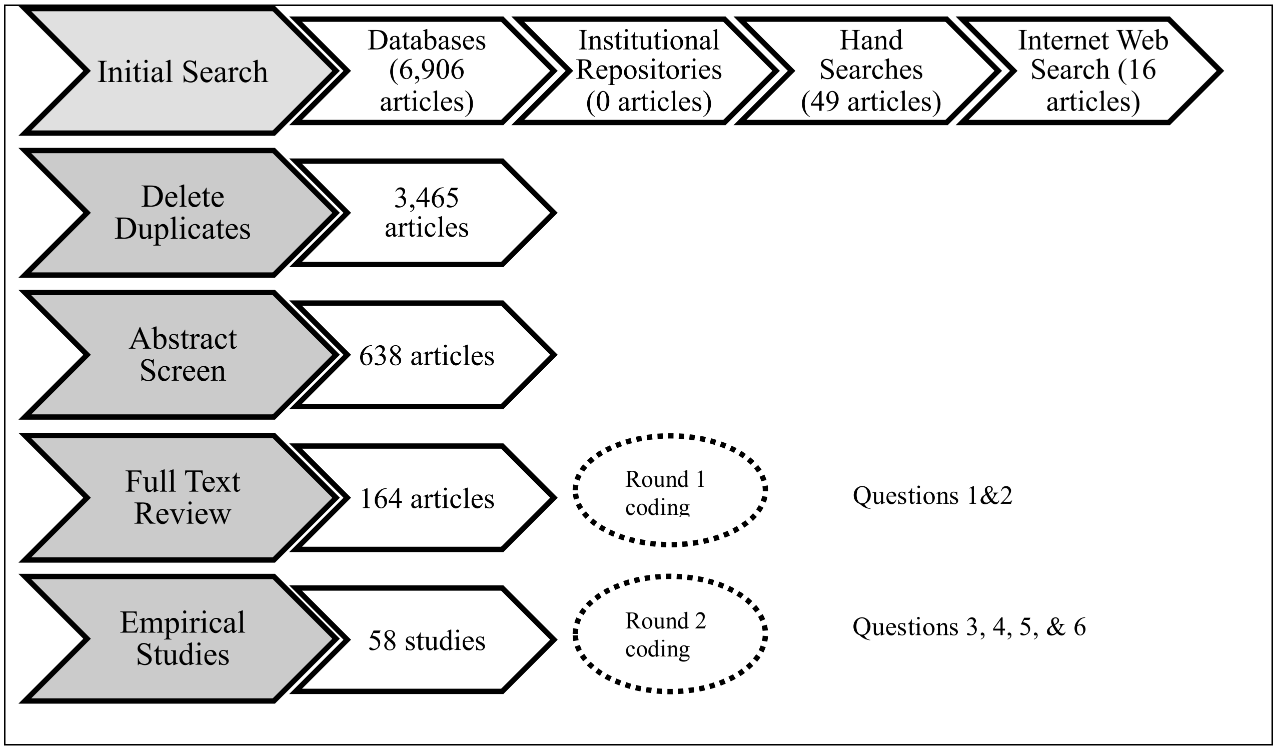


Figure 1: Search, Review, and Coding Proceure

Inclusion/Exclusion Criteria and Process

All abstracts went through an initial screening process where inclusion and exclusion criteria were applied. Any article that was outside of the date range was excluded, as were articles not related to computer science. If the phrase "computational thinking" was not included in the abstract, it was still eligible for inclusion if it was explicit about its intent to address one of the computational thinking domains identified in this study's search terms. There was no attempt to interpret the intent or otherwise deduce what outcomes could come of articles that described interventions; if the connection to computational thinking or one of the domains was not made explicit, the article was excluded. For example, the incorporation of real-world applications or programming environments like Scratch, Greenfoot, or Alice did not lead to automatic inclusion. While interventions that include these programming environments could be used to promote computational thinking skills, the author must have clearly made a connection to computational thinking or one of the domains specified in the definitions used for this study. Papers that introduce panel discussions, talks, video presentations, workshops, tutorials, posters, and other similar conference events were excluded. While there is valuable information in these sessions, the written explanations are brief and arguments too incomplete to be included in a meaningful way.

It was not always possible to ascertain from the abstract if the article should be included. In these cases, the full text article was obtained and reviewed to determine if the piece met the screening criteria for inclusion. At the conclusion of the abstract screening, 638 papers remained and moved on to the full text review stage before substantive coding began. An additional 474 papers were screened out with the full text reviews, which left 164 full text papers that advanced to the substantive coding phases in which relevant characteristics of the articles were coded.

Substantive Coding

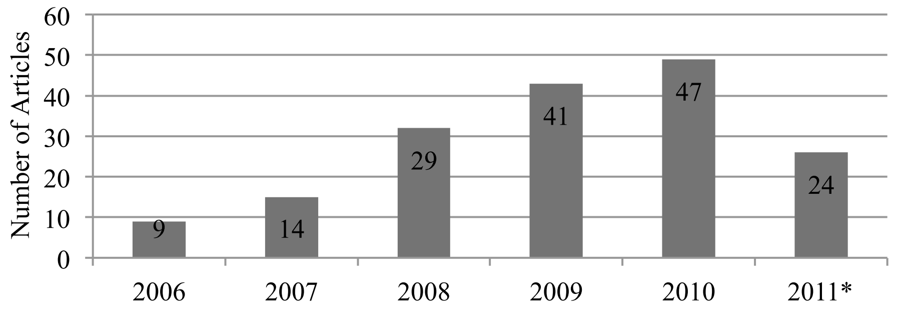
Substantive coding occurred in two rounds. The first round extracted demographic information on all articles. The second substantive round of coding examined the subset of the literature that included any paper or article that reported on data. The second round of coding addressed this study's research questions.

Findings

The findings from the first round of coding of the entire set of 164 articles on the topic of are used to answer research questions 1 and 2. A subset of these is examined more closely with a second round of coding. The literature included in this subset includes all reports of research related to computational thinking. Each of these studies was coded for: study design, population, type of intervention, outcomes, measures, and definitions of computational thinking used by studies. Since no attempts were made to statistically analyze the findings, all studies were included regardless of their methods, design, or quality.

Research Question 1: What are the demographic characteristics of the entire set of literature?

Year of Publication. The scholarship on computational thinking increased steadily from 2006 until June 2011 (Figure 2). This trend reflects its rise in popularity of the concept, and it is reasonable to assume this trend will continue given the recent focus on computational thinking at the national level (e.g., CSTA, NSF, Google).



**Note:* only papers published Jan-June 2011 are included

Figure 2. Computational thinking literature by year of publication (2006 – June 2011)

Author Characteristics. Two author characteristics were examined: the institutional affiliation of the primary author and area of expertise of all authors. An examination of the affiliation for the first author found that 38 (22%) of the 164 computational thinking articles were written by a primary author outside of North America. Regions include the Middle East (*n*=8), South America (*n*=1), Asia (*n*=10), Oceania (*n*= 2), and Europe (*n*=17). Randolph (2008) found that 55% of the studies in computer science were written by authors outside of North America. The term computational thinking was introduced in the United States, so it is not unexpected to find that the proportion of computational thinking scholarship produced abroad is lower than the proportion of internationally produced papers on computer science education (Randolph, 2008). It appears the emphasis on computational thinking or its domains is in the United States is not mirrored by other regions of the world.

The area of expertise of authors was primarily computer science. Only 46 (28%) articles included one or more authors with expertise in an area outside of computer science. These other areas include education, fine arts, other science (physics, biology, etc.), and engineering. Of those, 30 articles (18%) had one or more authors with expertise in education. Four articles had authors with expertise in evaluation or educational research. Since this review is focused on education, it is notable that so few authors have a background or expertise in education.

Target Populations. As seen in Table 2, there has been a considerable amount of literature aimed at the larger computer science and computer science education community (*n*=31, 19%), but the bulk of the literature has focused on students within a specific age range. To date, the scholarship on computational thinking has been distributed fairly evenly between the K-12 (*n*=69, 42%) and undergraduate (76, 46%) levels. A considerable amount of attention has been paid to undergraduate education, and this is likely in direct proportion to the number of undergraduate courses offered compared to the course offerings in K-12 settings. This proportion will shift given recent finding initiatives like National Science Foundation's Computing Education for the 21st Century (CE21), the CS10K project with its aim to have 10,000 teachers in 10,000 high schools teaching a new computer science curriculum by 2015 (Astrachan, Cuny, Stephenson, & Wilson, 2011), the attention to computational thinking in large-scale K-12 computer science advocacy efforts by organizations like CSTA and Google, and the emphasis on computational thinking in the newly developed AP CS Principles course. While the computer science community has been advocating for the expansion of computer science in K-12, it is only recently that policies and initiatives such as these have focused their attention on computer science and computational thinking at the K-12 levels. For example, it is reasonable to assume that there will be a substantial increase in the proportion of literature pertaining to K-12 in upcoming years.

Table 2.
Populations targeted by computational thinking literature

	Articles *N* (%)
K-12 Students	28 (17%)
Elementary Students (K-5)	8 (5%)
Middle School Students (6-8)	13 (8%)
High School Students (9-12)	20 (12%)
Undergraduate Students	76 (46%)
Graduate Students	4 (2%)
Teachers or Instructors	10 (6%)
Computer Science Education Community	31 (19%)

Note: The total exceeds 164 because some studies targeted multiple populations

Research Question 2: What kind of taxonomy might characterize this entire set of literature?

A grounded, iterative process was used to develop a six-fold taxonomy to classify the articles and papers. The first category in the taxonomy, **Curriculum Description**, includes articles that explain a lesson, curriculum, activity, or course that is used to promote computational thinking or one of its domains. The purpose is to share ideas with other computer science educators to improve the computer science education environment and community. The distinguishing feature of **Program Descriptions** is that the intervention or idea described goes beyond the individual classroom level and is implemented on a larger scale – across a university, for example. **Evaluations** seek to make a judgment of the merit, worth, or value of a program or process (Scriven, 1991). Articles were placed in this category if the author indicated that the primary aim for the paper was to convey the results of a study focused on a specific program or intervention. Evaluation papers could also fit in either the Curriculum Description or Program Description categories if it were not for the emphasis on the evaluation methods and findings. These were not dual coded. The fourth category is **Research**. These are distinguished from evaluations by their focus on informing theory or contributing to the larger knowledge base rather than being focused on a single program or intervention. The next two categories are closely related. **Philosophy** papers are intended to create or promote debate about computational thinking in the broad computer science or computer science education communities. Works in this category might aim to share a perspective on an issue within the field or describe how CT applies to other disciplines. The sixth category in this taxonomy is Opinion, in which authors share their perspectives or outlook on a computational thinking topic. These are generally not peer-reviewed and the primary intent is for a single author to share their views on a topic.

The distribution of computational thinking literature across the six categories can be seen in Figure 3. The majority of the articles (*n*=78, 48%) described a curriculum, lesson, activity, or a course and 11 (7%) described larger programs. Approximately one fifth of the articles (*n*=37, 23%) were philosophical articles intended to promote discussion or debate around computational thinking. Research (*n*=19, 12%) and Evaluation (*n*=9, 5%) pieces reported on data that was collected as part of a study. The remaining 10 (6%) articles were written by individuals who were sharing their perspective on computational thinking.

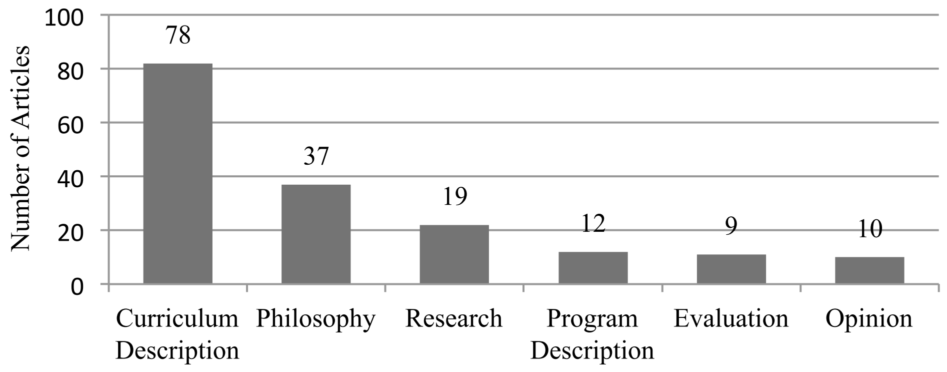


Figure 3. Distribution of computational thinking literature 2006 – June 2011.

To further characterize the literature, each was coded for the inclusion of study findings. In all, 58 papers included an explanation of a study, including the participants, methods, and a presentation of findings. All 28 articles in the Research and Evaluation categories included data, as did the 26 papers presenting curriculum descriptions included data, and four of the 12 program descriptions.

Research Question 3: What is the nature of the studies that have been conducted?

All reports of research or articles that included data on outcomes related to an intervention were examined, and methodological and conceptual details were extracted to answer research questions 3-6. An examination of study designs offers a cursory estimate of the methodological rigor of the research scholarship. The majority of studies (*n*=43, 74%) used some type of within-subjects or one-group design in which outcomes were examined for the treatment group only; no control groups were included (Table 3). The most common design employed was the one-group post-test only design (*n*=21, 36%). This is perhaps the weakest of all possible designs. Without a pretest it is difficult to ascertain if a change occurred, and the absence of a control group makes it impossible to know what might have occurred without the intervention (Shadish, Cook, & Campbell, 2002). Fifteen one-group studies (26%) included a pretest measure, which adds a small amount of strength to the design, but does not allow the researcher to make causal statements about the influence of the intervention.

Table 3.
Research and evaluation designs used to explore computational thinking

		All Studies *N* (%)
Within Subjects	Post only	21 (36%)
	Pre/Post	15 (26%)
	Repeated Measures	7 (12%)
Between Subjects	Post only	2 (3%)
	Pre/Post	2 (3%)
	Repeated Measures	1 (1%)
Correlational		4 (7%)
Causal Comparative		2 (3%)
Qualitative		10 (17%)
Did not include human participants		2 (3%)
TOTAL		66

Note: The total exceeds 58 because some studies used multiple designs

Quasi experimental studies that include a treatment and a control group (between-subjects designs) are necessary if the aim is to make causal statements about treatment effects. Only 3 (5%) of the studies included in this review are quasi-experimental studies. Of those, two included only a post-test measure, an improvement over a within-subjects design, but still a weak design. The incorporation of repeated measures bolsters both within subjects and between subjects designs. This was done in 8 (14%) of the studies. Correlational and causal comparative designs are also considered to be fairly weak research designs because the results remain open to many alternative explanations. Six described studies that employed two or more designs.

Research Question 4: How do study authors define computational thinking, and how do these align with the CSTA and Google definitions?

The conceptual feature explored were the computational thinking definitions used by authors in each of the studies. Each definition of computational thinking used and cited within the studies was examined. A large portion of the studies (*n*=25, 43%) did not include the term 'computational thinking', but focused one of or more of the computational thinking domains such as algorithmic thinking, problem solving, reduction, abstraction, recursive thinking, interdisciplinary application of computer science, or data reduction. Of the remaining 33 studies that do mention computational thinking, six (10%) use the term computational thinking but provide no definition, citation, or indication of what the phrase means either within the context of their study or to computer science education in general.

Several studies (*n*=12, 21%) simply cite Wing's 2006 paper without including an explanation or definition. While Wing popularized the term in this seminal piece, the description provided is not sufficient for use as an operational definition within a study. The page-long description was an appropriate and adequate introduction to the concept, but its use as a definition within a study of computational thinking is not sufficient. In that situation it is far too vague and non-specific to allow the researcher or the consumer of the research an understanding of what the phrase means. Quite simply, it is too broad a definition to be used to describe something that is the target of a single intervention.

Only fifteen (26%) studies include a definition of computational thinking. The detail and comprehensiveness of these seven definitions varied dramatically, but the definition included provided the reader with some understanding of what the phrase meant in the context of the study. These definitions included were compared to the definitions provided by Google and the CSTA. Of the 15 studies, nine offered definitions that were exceedingly vague compared to the processes described by Google and CSTA. These vague definitions are superficial descriptions of computational thinking as a "way of thinking", a "fundamental skill", or a "way of solving problems". They provide the reader little indication of what this mode of thought entails, and how the intervention described in the study might address it.

Four studies that define computational thinking go beyond these superficial descriptions and explain specific skills or concepts that comprise computational thinking. Each of the skills mentioned in these four studies overlap with those included in Google and the CSTA's definition (i.e. abstraction, decomposition, formulating problems). Two study authors address the need to operationalize computational thinking so it can be more meaningfully investigated. One of these authors included a laundry list of definitions provided by eight various authors, and explained that none of these was sufficient to operationalize the concept. The other author described a study in which the aim was to define computational thinking.

Research Question 5: Of the empirical studies, how many were intervention studies and how many were not? What kinds of interventions are being explored and tested?

The majority of the interventions studied were classroom-level interventions implemented in the traditional school settings (*n*=35, 60%). Further analysis of these curricula described for formal and informal settings found that the most common intervention in a school setting was an entire course redesign with the aim to infuse computational thinking across the entire semester or year (Table 4). Twelve of these redesigned courses were computer science courses, and six were courses focused on other disciplines such as science or mathematics. Other interventions included short computational thinking units or modules (*n*=3, 5%), activities or games (*n*=9, 8%), or a novel approach to teaching a computational thinking concept intended to be used in a traditional classroom setting.

Table 4.
Types of classroom-level curricula

	Articles *N* (%)
Course Focused on Computational Thinking	18 (51%)
Short Computational Thinking Units or Modules	3 (5%)
Activities or Games	9 (26%)
Approach to Teaching a Concept	3 (5%)

Computational thinking interventions used in out of school settings (*n*=8, 14%) were similar in nature, but were offered to students in informal settings such as summer camps or after school programs (e.g., Egan & Lederman, 2011). Another group targeted by computational thinking programs is teachers (*n*=6, 10%). Given the emphasis on incorporating computer science and computational thinking into non-computer science classrooms and the push to improve computer science teacher preparation, this is not surprising. Perhaps recent funding opportunities (e.g., CS10K) paired with the K-12 policy focus on computational thinking (i.e., AP, CSTE) will result in an increase in the proportion of interventions that fall into this category will increase dramatically in upcoming years. The 13 (22%) studies that explored how individuals think or learn or sought to understand differences in computational thinking skills among groups (e.g. between computer science professionals and novice computer science students) did not include an intervention. Some interventions aimed to influence individuals in multiple groups or settings (i.e., teachers, classrooms)

Research Question 6: What outcomes are examined in studies and how are they measured?

Of the studies that included an outcome or a dependent variable, many examined more than one. Some studies described numerous outcomes or measures, but did not report data or findings for each of the outcomes. In those cases, the only findings reported were significant ones. Some studies (correlational studies, for example) did not include a dependent variable. The outcomes examined fell into six major categories (Table 5). Examples of attitudinal outcomes include attitudes toward computing, attitudes toward other disciplines (i.e. mathematics and science), perceptions of computer scientists or computer science careers.

Table 5.
Outcomes examined in computational thinking studies

	Articles *N* (%)
Attitudes	23 (40%)
Skills/Knowledge	23 (40%)
Course Achievement	2 (3%)
Future Plans	5 (9%)
“did you like it”	21 (36%)
Course Enrollment	2 (3%)
None	13 (24%)

Note: The total exceeds 58 because some studies include multiple outcomes

Skills or knowledge outcomes were employed often. In some studies, the skills or knowledge were related directly to the intervention and the outcome examined in others studies was something more generally or not as directly related. A third common category of outcomes, seen predominately in evaluation studies, is the “did you like it” outcome. These differ from attitudinal outcomes because attitude outcomes require that one looks outside of the intervention and into one’s feelings or perceptions of a larger idea – like computer science or mathematics. “Did you like it” outcomes are exclusively interested in or target participant perceptions of the intervention being studied. Outcomes related to students’ future career or academic plans (e.g., intent to pursue a career or degree in CS, take future CS courses) were not commonly used, nor were those related to course achievement (final grade in the course) or subsequent course enrollment. Thirteen studies included no outcome variable.

Outcomes were assessed using a variety of measures across the various studies (Table 6). Questionnaires were by far the most commonly employed measure (*n*=34, 59%). These were used to collect data on many of the attitudinal outcomes and a number of studies used them to explore skills and knowledge gains (e.g. Kafura & Tatar, 2011). Self-assessments of knowledge gain are often flawed and are not as strong as other types of measures of knowledge and skills acquisition (Dunning, Heath, & Suls, 2004). More traditional measures of skills and knowledge including teacher or research made tests (*n*=14, 24%) or standardized or established tests that have undergone measurement of their reliability and/or validity (*n*=4, 7%) were used in some studies. Student work created as part of the course was used as a measure in 11 (19%), often paired with qualitative data analysis. Final course grades (*n*=3, 5%) and existing records (*n*=5, 9%) were used as measures of success, as were observations (*n*=11, 19%), interviews *(n=6*, 10%), and other measures including reflections, mentor ratings, focus groups, and computer-logged data.

Table 6.
Measures used in computational thinking studies

	Studies *N* (%)
Questionnaire	41 (61%)
Course Grades	3 (4%)
Teacher or Researcher Made Test	15 (22%)
Student Work	13 (19%)
Existing Records	4 (6%)
Standardized or Established Tests	5 (7%)
Interviews	7 (10%)
Observation	11 (16%)
Other	7 (10%)

Note: The total exceeds 58 because some studies used multiple measures

The methods used for data analysis were also examined. If multiple analysis methods were used within a study, each method was coded. The findings are displayed in Table 7. The majority of the studies used less robust descriptive (n=36, 62%) or qualitative (n=36, 62%) methods to analyze and represent the data. Only 8 (14%) of the studies used inferential statistics to report findings. Three used Pearson correlation, three used a t-test, one used ANOVA statistic, one a chi-square, and one reported z-test scores. One study used multiple analysis methods. Another reported to have several statistically significant findings, but did not give information on the statistical analysis method used. Only one of the eight studies that used inferential statistics included a discussion of the effect size.

Table 7.
Data Analysis Method Used

	Studies *N* (%)
Inferential	8 (14%)
Descriptive	36 (62%)
Qualitative	38 (69%)

Note: The total exceeds 58 because some studies used multiple data analysis methods

Limitations

There are several limitations to this study. The first arose in the development of a search protocol and gathering all the literature and studies on computational thinking was a challenge. The initial challenge was the lack of a widely agreed upon definition or structure of computational thinking. While there is agreement about the existence of CT as a mode of thought, there is no general understanding of the content or structure of computational thinking and great variation exists among the various definitions. Strides have been made because of events like the February 2009 workshop on the scope and nature of computational thinking hosted by the National Research Council (Committee for the Workshops on Computational Thinking, 2010), but general agreement remains elusive.

Another limitation arises from the fact that conference proceedings are the primary outlet for communicating about the computer science field (Moed & Visser, 2007). It is not a problem that the bulk of the studies and articles examined were published in conference proceedings. The emphasis on conferences is not surprising given the nature of computer science. In a field driven by innovation and a race to be the first to discover something new, only to have to immediately pursue the next new thing, time to publication is an important consideration. A National Research Council study (Snyder, 1994) found the median time from initial submission to publication in journals to be 31 months and in conference proceedings 7 months. The proportion of computer sciences literature published in this venue is markedly higher than the 8% of social science and 21% of science publications that are found in conference proceedings (Bourke & Butler, 1996). In other fields, conference proceedings are not held in as high esteem as academic journals, methodological quality is often lower in conference papers, and publication bias is seen when the proportion of non-significant to significant findings are compared for these two types of outlets. Journal manuscripts are more likely to include non-significant findings in other fields (Snyder, 1994). This is not the case for CS education publications where a study of the methodological quality of articles found no differences in the methodological quality of articles published in computer science education journals and those published in computer science education conference proceedings. Computer science conference papers have a "much, much" higher citation rate than social science conference papers (Heeks, 2010), which is further evidence of the approval of this dissemination method in the field. The limitation arises when information is shared at conferences in the form of panels, presentations, and talks. In this study, descriptions of almost 300 such sessions were excluded because the published descriptions did not include enough detail on the session’s content to allow for data to be extracted.

Two forms of publication bias, present in all reviews of this nature, are an additional limitation. Editorial publication bias is introduced when editors or reviewers reject manuscripts because of statistically non-significant results. Authorial bias refers to the failure to submit manuscripts that report on neutral or negative findings (Randolph & Bednarik, 2008). Since authorial bias is the most common form of publication bias (Lee, Boyd, Holroyd-Leduc, Bacchetti, & Bero, 2006; Olson et al., 2002), authorial bias is more of a threat to the validity of these findings.

Conclusions, Discussion, and Recommendations

The study described in this paper explored all of the literature on computational thinking between 2006 and June 2011. It described the demographic characteristics of this literature base and examined how well researchers exploring computational thinking align their methods, design, and analysis with commonly accepted definitions and standards for methodological rigor. Wing’s seminal paper was undoubtedly influential in the computer science education community and a nod to Wing’s introduction of the concept into the common language and modern thought of computer science educators is an appropriate reference. That said, the community continues to wrestle with a concrete definition for the phrase computational thinking. Until there is a universally accepted comprehensive understanding of what CT is , it is reasonable to expect that studies examining computational thinking explicitly state their definition of the phrase so readers can understand more about the intervention and its intent, or the focus of the research when there happens to be no intervention.

According to Hemmendinger (2010), most of the current definitions of computational thinking currently offered lack precision and fail to provide sufficient examples, often making the concept misunderstood or misconstrued. There has been much discourse in recent years about the precise definition of computational thinking (Bundy, 2007; Denning, 2007; Garcia, et al., 2010; Henderson, 2009; Committee for the Workshops on Computational Thinking, 2010; Wing, 2006). Wing (2006) introduced computational thinking as "reformulating a seemingly difficult problem into one we know how to solve, perhaps by reduction, embedding, transformation, or simulation" (p. 33). This definition is widely relied upon in projects that aim to promote computational thinking skills (i.e., Good et al., 2008; Hambrusch, Hoffmann, Korb, Haugan, & Hosking, 2009), but there has been little offered in the literature in the way of interpretation, further articulation, or constructive critique. The continued use of this broad, nonspecific definition and lack of elucidation could indicate that while there is acceptance of the idea that computer science is in need of a redesign, there is little understanding of the concept of computational thinking. It must be said that none of these articles claims to be focused on defining computational thinking. Given that computational thinking is such an ambiguous concept, it is problematic that these studies claim to be studying it, but provide no concrete definition. Only one study (Basawapatna, 2011) acknowledges the ambiguous nature of the concept and the need to provide a concrete description of how computational thinking is defined within the context of the study.

The intent of this review is to identify research trends and make recommendations to improve computational thinking and computer science education research practice. Reliable, generalizable methods were used to conduct this study, and two major conclusions were reached. First, computational thinking is far too underdeveloped a concept to be examined as a study's outcome in any meaningful way. Its use as a primary intervention target and outcome measure is problematic. Given the underdeveloped nature, one has to ask if it is inappropriate under any circumstances for a study to offer no definition or description of the intervention target, but particularly the case in studies reviewed that claim to target computational thinking, and yet fail to define it or describe how their intervention targets computational thinking or to offer any theoretical framework that explains how their chosen outcome measure is linked to computational thinking. There has been widespread agreement about the existence of the concept, but only recently have there been efforts to define it in a way that could lead to its operationalization and use as a general construct within studies.

Intervention research needs a strong basis of qualitative and quantitative research underpinning it. It is insufficient to provide only anecdotal evidence of claims, which is the type of evidence produced in the bulk of studies examined in this review. This type of data is useful for hypothesis generation, but can never confirm a hypothesis. The first step using a concept like computational thinking this is to conduct of qualitative research to articulate its theoretical components. The Computer Science Teachers Association's recently published definition can be used as a launching point for these efforts. The second step, the quantitative research, examines causal connections between the operational theoretical concepts. This includes the modification of existing measures or the development of new measures to assess each of the theoretical concepts that comprise the construct. These steps must be taken before robust studies addressing computational thinking can be conducted. Very little judgment or criticism about computational thinking has occurred; so far it is largely taken at face value. Perhaps the computer science education community should place more emphasis exploring the concept and taking further steps to arrive at a consensus about a theoretical framework that can be used to describe it.

The second conclusion that arose from this study relates to the quality of initial research and evaluation studies examining the efficacy of interventions that purport to improve computational thinking. The studies have, as a group, far too weak characteristics to allow for decent estimates of causal judgment. While the aim was not to report on the quality of the studies, it is evident that the rigor is lacking. This lack of rigor is apparent in both the conduct of the studies and in the reporting of them. The standards for reporting on studies of educational intervention in this field differ from those that have been established in other educational disciplines. There seems to be a *laissez faire* culture within computer science education that accepts research and evaluation studies that lack conceptual and methodological rigor. While there is no single set of standards that all of educational research relies on, there are some clear commonalities among the frequently cited standards (e.g. AERA, 2008; Ragin, Nagel, & White, 2004; National Center for Dissemination of Disability Research, 2012; Shavelson & Towne, 2002).

Evidence of robust research designs in the computational thinking literature is non-existent. The commonly relied upon designs (single group posttest only or single group pretest/posttest) are vulnerable to multiple threats to internal validity. Quality studies employ systematic designs and are underpinned by explicit theoretical or conceptual frameworks to address significant questions that will contribute to the knowledge base. Furthermore, controls for counterfactuals and threats to internal validity that threaten causal connections are vital components of studies that aim to make causal claims.

The majority of the computational thinking research and evaluation studies relied exclusively or in large part on measures of student attitudes toward the intervention or self-report of learning. In order to meet standards for educational research, measurement instruments should have psychometric information provided about them, and reliability and validity statistics should be presented. The research methods and instruments to measure variables of interest must be fully conceptualized and appropriate to address the research question. Studies that rely on perceptions of the intervention or on self-report of learning do little to inform the larger field and are overly focused on the context in which the study is conducted.

The reporting standards that are evident in the field are insufficient. Authors should be encouraged to provide sufficient detail about the intervention, the procedures, the participants, and the theoretical underpinnings that drive the study. In the articles scrutinized for this paper, there was often very little substantive information on the intervention included. Sampling procedures were not thoroughly described, and results and findings were not included for all outcomes described. All aspects of studies should be presented, not just the portions of the study that reveal positive findings. The omission of results hints at a bias toward reporting only positive findings. Furthermore, written reports must provide sufficient information to reproduce or replicate the study, which includes a description of the intervention, the sample, the methods, and present the findings from all aspects of the study. Beyond these methodological features, written reports should describe the philosophical assumptions made, the study's limitations, and present the implications of the study on the rest of the field. The portion of computer science education research represented in this study do not adequately meet these commonly describe upon standards for the conduct research and evaluation studies, nor for reporting. A discussion of what the computer science education research and evaluation standards could/should be would strengthen the quality of research in the field by encouraging reflection on research's role in advancing computational thinking and computer science education. If the computer science community maintains different expectations regarding research quality, these should be articulated and discussed publicly and shared with the larger community.

The lack of attention to details such as measurement of outcomes and the definition of constructs suggest a lack of appreciation and understanding of the complexity of social science research. These are not trivial aspects of educational research. Lack of theoretical framework is further evidence that these researchers are unfamiliar with ambiguities that are inherent in educational research. This is not the first call for computer science education to make improvements to their research practice. Randolph (2007) made a series of recommendations, and improvements are not yet apparent. Continued reliance on weak designs, ill- or non-defined concepts or interventions, and vague reports of the findings will do nothing to advance computational thinking and computer science education to where it wants to be – in K-12 classrooms across the country. In order to make evidence-based decisions, a collection of quality research studies must exist. By following the established educational research standards as well as the recommendations set forth by researchers who have examined the body of literature in the field (i.e. Randolph, 2007, the current study), the computer science education community will slowly amass the evidence it needs to establish its place in the K-12 curriculum.

This review captures the literature from the introduction of computational thinking into the common language of computer science educators. Judging by the limited quality of the designs, very little rigorous examination of computational thinking and how it can best be introduced, encourage, and fostered in students. This review will serve as a baseline for future projects that examine the body of work and evidence on computational thinking. In addition, hopefully it will encourage the community of researchers interested in computational thinking to apply rigorous methods that will allow for generalization to other settings. Policymakers could encourage projects that employ rigorous methods to study the computational thinning efforts agencies, like the National Science Foundation, are encouraging. All this in the hope that it won't be long until it is possible to conduct a disciplined inquiry that synthesizes and critiques the empirical literature – much like Randolph has done with the CS educational research – to derive or come up with suggestions which the field can advance. Through this current review, it has become clear that there currently are not enough empirical studies to conduct a quantitative review of the literature.

REFERENCES

American Educational Research Association (AERA). (2011). Definition of scientifically based research. Retrieved January 26, 2011, 2011, from http://www.aera.net/Default.aspx?id=6790

Anewalt, K. (2008). Making CS0 fun: an active learning approach using toys, games and Alice. *Journal of Computing Sciences in Colleges, 23*(3), 98-105.

Astrachan, O., Cuny, J., Stephenson, C., & Wilson, C. (2011). *The CS10K Project: Mobilizing the Community to Transform High School Computing*. New York: Assoc Computing Machinery.

Bourke, P., & Butler, L. (1996). Publication types, citation rates and evaluation. *Scientometrics*, 37(3), 473-494.

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing*, 1(2), 67-69.

Committee for the Workshops on Computational Thinking. (2010). *Report of a workshop on the scope and nature of computational thinking.* Washington, D.C.: National Academies Press.

Computer Science Teachers Association (CSTA) (2005). The new educational imperative: Improving high school computer science education. Retrieved November 15, 2010 from http://www.nsf.gov/cgi-bin/good-bye?http://csta.acm.org/Communications/sub/DocsPresentationFiles/White_Paper07_06.pdf

Computer Science Teachers Association (CSTA) (2009). High school computer science survey Retrieved November 15, 2010, from http://www.nsf.gov/cgi-bin/good-bye?http://csta.acm.org/Research/sub/CSTAResearch.html

Computer Science Teachers Association (CSTA) (2011a). K-12 Computer Science Standards: Revised. New York.

Computer Science Teachers Association (CSTA) (2011b). Operational definition of computational thinking for K-12 education Retrieved November 11, 2011, from http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

Computing Research Association. (2010). CRA Taulbee Survey Retrieved November 11, 2010, from http://www.cra.org/resources/taulbee/

Denning, P. (2007). Computing is a natural science. *Communications of the ACM*, 50(7), 18. doi: 10.1145/1272516.1272529.

Dunning, D., Heath, C., & Suls, J. (2004). Flawed self-assessment: Implications for health, education, and the workplace. *Psychological Science in the Public Interest*, 5(3), 69-106. doi: 10.1111/j.1529-1006.2004.00018.x

Garcia, D. D., Lewis, C. M., Dougherty, J. P., & Jadud, M. C. (2010). If _________, you might be a computational thinker! *Proceedings of the 41t ACM technical symposium on computer science education.* (pp. 263-264). Milwaukee, WI. doi: 10.1145/1734263.1734355

Good, J., Romero, P., du Boulay, B., Reid, H., Howland, K., & Robertson, J. (2008). An embodied interface for teaching computational thinking. Paper presented at the International Conference on Intelligent User Interfaces (IUI 2008).

Google. (2011). Exploring Computational Thinking, from http://www.google.com/edu/computational-thinking/

Hambrusch, S., Hoffmann, C., Korb, J., Haugan, M., & Hosking, A. (2009). A multidisciplinary approach towards computational thinking for science majors. ACM SIGCSE Bulletin, 41(1), 183-187.

Hammerstrom, K., Wade, A., & Jorgensen, A. K. (2010). Searching for studies: A guide to information retrieval for Campbell Systematic Reviews. Campbell Systematic Reviews 2010: Supplement 1, 74. Retrieved from http://www.campbellcollaboration.org/news_/new_information_retrieval_guide.php doi:10.4073/csrs.2010.1

Heeks, R. (2010, April 28). ICT4D conference papers: Impact and publication priority [web log post]. Retrieved from http://ict4dblog.wordpress.com/tag/ict4d-conferences/

Heines, J., Greher, G., Ruthmann, S., & Reilly, B. (2011). Two Approaches to Interdisciplinary Computing+ Music Courses. *Computer, 44*(12), 25-32. doi: 10.1109/MC.2011.355

Hemmendinger, D. (2010). A plea for modesty. *ACM Inroads*, 1(2), 4-7. doi: 10.1145/1805724.1805725

Henderson, P. B. (2009). Ubiquitous Computational Thinking. *Computer*, 42(10), 100-102. doi: 10.1109/MC.2009.334

Martin, F., Greher, G., Heines, J., Jeffers, J., Kim, H. J., Kuhn, S., . . . Yanco, H. (2009). Joining computing and the arts at a mid-size university. *Journal of Computing Sciences in Colleges, 24*(6), 87-94.

Moed, H. F., & Visser, M. S. (2007). *Developing bibliometric indicators of research performance in computer science: an exploratory study*. (CWTS Report 2007-01). Retrieved from the Centre for Science and Technology Studies website: http://www. cwts.nl/pdf/NWO_Inf_Final_Report_V_210207. pdf.

National Center for Dissemination of Disability Research. (2005). *What are the standards for quality research?* (Technical Brief No. 9) Retrieved January 26, 2012 from http://www.ncddr.org/kt/products/focus/focus9/

Ragin, C. C., Nagel, J., & White, P. (2004). *Workshop on Scientific Foundations of Qualitative Research*. Washington, DC: National Science Foundation.

Randolph, J. J. (2007). *Computer science education research at the crossroads: A methodological review of computer science education research, 2000--2005.* (Doctoral dissertation) Retrieved from http://0-search.ebscohost.com.catalog.library.colostate.edu/login.aspx?direct=true&AuthType=cookie,ip,url,cpid&custid=s4640792&db=psyh&AN=2008-99011-125&site=ehost-live Available from EBSCOhost psyh database.

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. *Proceedings of the 41st ACM technical symposium on Computer science education* (pp. 265-269). New York. doi: 10.1145/1734263.1734357

Shadish, W. R., Cook, T. D., & Campbell, D. T. (2002). *Experimental and quasi-experimental designs for generalized causal inference.* Boston: Houghton Mifflin.

Shavelson, R. J., & Towne, L. (Eds.). (2002). *Scientific research in education*. Washington, DC: National Academies Press.

Snyder, L. (1994). *Academic careers for experimental computer scientists and engineers*. Washington, DC: National Academies Press.

Thomas, J., Brunton, J., & Grazosi, S. (Producer). (2010). *EPPI-Reviewer 4: Software for research synthesis*.

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*. (Research Report). Retrieved from the Association for Computing Machinery website: http://www.acm.org/runningonempty/fullreport.pdf

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

CHAPTER 3: COMPUTATIONAL THINKING: WHAT IS IT, HOW DO WE TEACH IT, AND HOW DO WE ASSESS IT?

Introduction

Recent reports paint a bleak picture of K-12 computer science education. States have few standards focused on the conceptual facets that underpin computer science (e.g., an understanding of algorithm), but instead emphasize lower level skill-based concepts (e.g., using technology in other learning activities) (Wilson, Sudon, Stephenson, & Stehlik, 2010). The Computer Science Teachers Association (CSTA) proposed a national model for computer science standards (Tucker, Deek, Jones, McCowan, Stephenson, & Verno, 2006), but there is great disparity among the states in the adoption of these standards. Only 14 states have adopted the standards to a significant degree. Furthermore, no states require students to complete a computer science course, and only nine allow these courses to count toward the mathematics or science credits required for graduation (Wilson et al., 2010). The number of pre-Advanced Placement (AP) and AP computer science courses offered in high schools has declined (Gal-Ezer & Stephenson, 2009), and tremendous inequality is seen in minority student participation in AP computer science tests (Goode, 2011). This lack of access to engaging and rigorous curriculum could be one cause of the lack of student interest in computer science.

While the policy debates around computer science education continue, the reality persists: computing and technology are, and will remain, an integral part of our society and the future. This makes the need for computer science education reform undeniable, and the discipline appears to be headed in a clear and fairly unified direction. This tangible direction for change can be found in the computational thinking movement. Beginning with a seminal article by Janette Wing in 2006, many have begun to liken computational thinking to the basic skills of reading, writing, and arithmetic.

A focus on computational thinking in K-12 education would enable individuals to more effectively navigate today's society as well as encourage equitable access to technological skills, devices, and other resources. As individuals are taught how to apply computational thinking to their daily lives, personal empowerment is enhanced. This emphasis would raise student interest in information technology, computer science, and other technologically oriented professions; thus, it would enhance the competitiveness of the U.S. from an economic standpoint by better preparing students to enter the internationally competitive work force (Committee for the Workshops on Computational Thinking, 2010).

Support for computational thinking is evidenced by at least three national initiatives. First, the curriculum for the new AP Computer Science Principles course is constructed around six Computational Thinking Practices (Astrachan, Cuny, Stephenson, & Wilson, 2011). Second, the National Science Foundation's Computing Education for the 21st Century (CE21) funding program encourages projects that develop computational thinking competencies (National Science Foundation, 2011). Third, the CSTA uses computational thinking as a common thread to link the three levels (i.e., elementary, middle, and high school) of learning standards that it recommended for K-12 computer science (Tucker et al., 2006).

If the ideas and projects that have arisen from these recent discussions are to take root, the computer science education community must appeal to those outside of the field to join in the campaign to overhaul K-12 computer science. The first step to doing this is to introduce the concepts, the programs, and the initiatives that are currently powering the efforts. Many who teach computer science in high schools do not have degrees in computer science or information technology, and it can be reasonably assumed that there are very few, if any, computer scientists in the teaching profession at the elementary level. This paper is intended to serve as an introduction to computational thinking for 'the rest of us'. In this case, 'the rest of us' includes elementary teachers, secondary teachers, school administrators, and educational researchers. This paper is intended to reach the audience who needs to be involved in the next step, which includes improving access to computer science and computational thinking in K-12 schools.

The Origins of Computational Thinking

Computational thinking was first mentioned by Papert to suggest a way to efficiently apply novel approaches to problem solving (1996). Wing expanded on Papert's ideas and popularized the term 'computational thinking' in a brief but compelling article in 2006 where it is described as a "fundamental skill for everyone, not just for computer scientists" (Wing, 2006, p. 35). Wing likened the skill to reading, writing, and arithmetic and asserted that with computational thinking, humans have the capability to solve problems and design systems that would be otherwise impossible. This mirrors the sentiments of Pfeiffer that "computers are thinking aids of enormous potentialities. Merely having them around is enough to change the way we think" (Pfeiffer, 1962). Other well-known computer scientists have also suggested that computer science be a part of all students' education. For example, Perlis argued in 1962 that all students should learn to program as part of their undergraduate education. Perlis believed exposing students to programming would allow them a new computational perspective on topics such as calculus and economics (Perlis, 1962). While the sole emphasis on programming is not generally seen in today's conceptualizations of computation thinking, the importance of exploring how computation can enhance other disciplines remains essential.

Computational thinking has ties to procedural thinking, algorithmic thinking, recursive thinking, and critical thinking. Procedural literacy means understanding the limitations and the possibilities of specific computing tools. Algorithmic thinking, popularized in the 1950's and 1960's, is the process of solving problems by exactly defining instructions using step-by-step procedures. These procedures give identical results when carried out by any human or suitable machine (Dening, 2003). Recursive thinking is solving large problems by breaking them down into smaller problems that have identical forms. Finally, critical thinking is "the art of analyzing and evaluating thinking with a view to improving it" (Paul & Elder, 2001, p. 4). Computational thinking includes all of these concepts, and more.

Defining Computational Thinking

Perhaps the most influential contribution since the introduction of computational thinking is the operational definition recently released by the CSTA in collaboration with The International Society for Technology in Education (ISTE) and leaders in education and industry. This definition describes computational thinking as a problem-solving process that includes, but is not limited to, the following characteristics: "(1) formulating problems in a way that enables us to use a computer and other tools to help solve them, (2) logically organizing and analyzing data, (3)representing data through abstractions such as models and simulations, (4) automating solutions through algorithmic thinking (a series of ordered steps), (5) identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources, and (6) generalizing and transferring this problem solving process to a wide variety of problems" (Computer Science Teachers Association, 2011b). A formal process for obtaining reactions to this definition showed broad support for this definition by computer science educators and professionals.

What sets computational thinking apart is its applicability across all disciplines. It is a unifying principle with ties not only to computer science, but also to all disciplines from the sciences to the humanities (Lu & Fletcher, 2009; Wing, 2006). Computational thinking goes beyond computer literacy, which is the efficient use of technology, and also beyond fluency, which is focused on skills that enable to use technology. Computational thinking includes analytical skills that enhance how one approaches a problem and an appreciation for how computing can augment a person's abilities and enhance efficiency.

Although computational thinking was quick to catch on in the computer science education community, most current definitions of computational thinking are vague and lacking in detail and examples (Hemmendinger, 2010). An early definition offered by The Center for Computational Thinking at Carnegie Mellon describes computational thinking as 1) a way of solving problems and designing systems that draws on concepts fundamental to computer science; 2) means creating and making use of different levels of abstraction to understand and solve problems more effectively; 3) means thinking algorithmically and with the ability to apply mathematical concepts to develop more efficient, fair, and secure solutions; and 4) means understanding the consequences of scale, not only for reasons of efficiency, but also for economic and social reasons.

This, and similar definitions, were sufficient to promote and foster widespread agreement of its importance within the community, but did not satisfy the field's desire to have a detailed and practical definition of computational thinking. Recent efforts have focused on developing a more concrete definition. The National Academies convened a workshop with the sole aim to explore the scope and nature of computational thinking in early 2009 (Committee for the

Workshops on Computational Thinking, 2010). While no consensus was reached, significant progress was made in clarifying the various perspectives on computational thinking.

Many, like Wing, believe computational thinking to be a revolutionary concept, one as important to a solid educational foundation as are reading, writing, and arithmetic (Bundy, 2007) (Day, 2011). Others believe its potential and significance are overstated (Denning, 2009; Hemmendinger, 2010), and some have voiced concern that by joining forces with other disciplines computer science might be diluting either one or both of the participating disciplines (Cassel, 2011; Jacobs, 2009). Both the praise and the criticism for computational thinking could perhaps be tempered by reflecting on a historical quote by Pfeiffer in 1962: "Computers are too important to overrate or underrate. There is no real point in sensationalizing or exaggerating activities which are striking enough without embellishment. There is no point in belittling either." (Pfeiffer, 1962). For the time being, computational thinking appears to be a unifying force in the computer science education community, but it seems prudent to heed these warnings to remain realistic about its potential to influence K-12 education.

Computational Thinking in the K-12 Classroom

Wing concluded the seminal piece on computational thinking by summoning computer science teachers to "spread the joy, awe, and power of computer science, aiming to make computational thinking commonplace" (Wing, 2006, p. 35). Almost immediately the computer science education community and scholars began to explore how it might be integrated into the existing computer science curriculum and applied in other disciplines.

The effective use of technology in the classroom to engage students in computing has been a source of debate for decades. Papert commented that the phrase "technology and education" too often implies "inventing new gadgets to teach the same old stuff in a thinly disguised version of the same old way” (1996, p.138). In 2000 Garofalio and colleagues argued that simply using technology to teach the same topics that could be explained without technology “belies the usefulness of technology” (Garofalo, Drier, Harper, Timmerman, & Shockey, 2000), and this belief persists today (Bull, 2005).Computational thinking is not about learning computer skills or how to solve problems like computers, nor is it learning how to use a piece of technology. Instead, computational thinking develops all critical skills of humans to solve problems (Wing, 2006). On the surface, this approach to computer science could be at least partially immune to many of the criticisms of how technology has historically been applied in K-12 classrooms.

Scholars and educators who are focused on computational thinking have taken a number of approaches as they strive to create tools and curricula that are applicable to K-12 students. The following brief explanation of some of these approaches is intended to provide an overview for K-12 teachers to use as they contemplate how they might best incorporate computational thinking concepts into classrooms.

Technology-Free Computational Thinking

An assortment of activities and curricula that are not reliant on computers or other technology have recently been developed. The most significant challenge to teaching computer science in K-12 settings is rapidly changing technology (Gal-Ezer & Stephenson, 2009), and the incorporation of activities that promote computational thinking without technology are a way to address this challenge. There are materials and ideas in this category that are appropriate for the full range of K-12 students.

Computer Science Unplugged is a compilation of free activities to introduce students of all ages to the central concepts in computer science without the use of a computer (Bell, Alexander, Freeman, & Grimley, 2009; Bell, Fellows, & Witten, 2002). One of the primary aims is to demonstrate to students that computer science is more than programming. A curriculum kit of Computer Science Unplugged materials are offered through The National Center for Women and Information Technology (NCWIT). This kit, *Computer Science-in-a-box: Unplug Your Curriculum* (National Center for Women & Information Technology, 2011), introduces essential elements of computer science to 9-14 year old students. Lessons that introduce how computers work while simultaneously teaching math and science concepts. *The Value of Computational Thinking across Grade Levels* (*VCTAL*) project is developing an innovative mix of twelve instructional modules designed to encourage computational thinking in high school students for grades 9-12. The modules and lessons are activity-based and can be used in computer science, science, mathematics, history, or other courses (Cozzens, Kehle, & Garfunkel, 2010).

Since one of the barriers to computer science and computational thinking is technology, it is reasonable to think that the technology-free options would appeal to many classroom teachers. This approach may have additional appeal to those who are not technologically savvy themselves or who lack the proper training. It might also be attractive to those who do not have easy access to computers for all students.

Programming Games or Robots

Another approach to introduce K-12 students to concepts in computer science and computational thinking is the use of games and robots to introduce computer science and computational thinking concepts. The belief is that games and robots will motivate students to broaden participation in computer science courses and careers, for they simultaneously instill an understanding of the targeted concepts and promote student interest in computing. The iDreams project is an effort to engage middle school students in computer science with game design.

iDreams created the *Scalable Game Design* curriculum in which students create a Frogger-like game while learning increasingly sophisticated concepts (Ioannidou, Bennett, Repenning, Koh, & Basawapatna, 2011). *TangibleK Robotics* is a program that introduces computational thinking to young children (Bers, 2010). Students in preschool to second grade construct robotic artifacts and program them to respond to some stimuli. For example, a young child might build a car and program it to follow a light. *Modular Robotics*, a spin-off company from Carnegie Mellon University's Center for Computational Thinking, is creating kits used to create robots for use in science centers and community museums. The robots require no programming, but instead consist of magnetic cubes that snap together. The order and configuration in which the cubes are put together determines how the robot behaves (Modular Robotics, 2011).

The downside to using games and robots is the training and specialized technology required to use these tools. Most gaming or robotics-based programs and tools that encourage computational thinking are offered in out of school settings (Hardnett, 2008; Modular Robotics, 2011). A few projects, like iDreams, offer summer workshops to train teachers to use the curriculum and the game design module (Ioannidou, Bennett, Repenning, Koh, & Basawapatna, 2011).

Initial Learning Environments

Initial learning environments (ILE's) are a third approach to incorporating computational thinking in the K-12 classroom. These environments support novice programmers by allowing the direct manipulation of objects or through visualizations to support learning. While they are primarily used to learn basic programming concepts, some projects focus on the teaching and learning of computational thinking skills using ILE's. Sontag (2009) describes the *Critical Thinking with Alice* model which uses the Alice 3D graphics programming environment to teach critical thinking, a fundamental facet of computational thinking. The model is applicable for students in grades 5-8, and is tied not only to curriculum standards, but also to 21st century skills. An example unit is situated in a social studies context where students learn computational concepts as they create a 3D animated world in Alice. The animation is based on primary source information about land disputes between Native Americans and settlers during the California Gold Rush of the 1850's. An advantage to this model is the existence of tutorials for teachers and for students as they learn Alice together, so there is limited prerequisite knowledge required.

A group of computer scientists at Hamburg University (Rick, Ludwig, Meyer, Rehder, & Schirmer, 2010) applied Greenfoot, an ILE, to a business informatics context with middle school students. Students were introduced to computational thinking concepts as they used the visual simulation framework to develop their own "micro world" that consisted of an airport baggage handling system. In addition to these widely-known ILE's, some projects develop their own language. One such language is Toque, a cooking-based programming language where cooking scenarios are used as programming metaphors (Tarkan et al., 2010) to teach computational thinking skills.

One downside to ILE's and other approaches that require the use of technology is the inconsistency among schools in relation to the availability of technology. Hardware requirements for computers to run initial learning environments are small compared to the technology available in most school systems, but some schools may have older computers that might not have adequate capacity to operate the ILE's. A second potential concern is scheduling time in a shared computer lab. Not all teachers have ready access to a classroom of computers. A third issue can arise when programs need to be installed or downloaded onto school-owned computers. Most schools have policies to prohibit teachers or students from doing this, so teachers must request that a technology specialist or systems administrator within the district download or install the necessary software. This process might be prohibitive for some teachers.

Integrating Computational Thinking with Other Disciplines

Another approach to introducing computational thinking concepts to K-12 students is the integration of computational thinking within other disciplines. The CSTA believes all teachers are capable of incorporating computational thinking into their classrooms. To aid in this, they have created a series called *Computational Thinking Learning Experiences*, which are short lessons designed to give teachers concrete examples of how computational thinking can be introduced in various content areas and grade levels. These are intended to get teachers thinking about computational thinking and how it can be infused into the classroom. They are not an all-inclusive resource for integrating computational thinking.

Google engineers and classroom teachers collaborated to create curriculum models, resources, and a community to help teachers incorporate computational thinking into classrooms. *Exploring Computational Thinking (ECT)* makes classroom-ready lessons, (including teacher editions, student worksheets, and applicable programs) available to teachers online. Lessons are searchable by subject and grade level, and educators are invited to share their own materials and discuss computational thinking with others using a discussion forum.

Several examples of courses or programs introduce computational thinking in liberal arts contexts. A three-week *Musicomputation* course for 11-17 year old students teaches the mathematical principles that underlie computer science and applies these to musical composition (Meyers, Cole, Korth, & Pluta, 2009). Other programs that integrate music and computational thinking skills exist at the secondary (Peterson & Hickman, 2008) and undergraduate (Ruthmann, Heines, Greher, Laidler, & Saulters, 2010) levels. The Inter Interactive Multimedia program integrates computational thinking into a middle school classroom alongside expository writing and interactive journalism (Wolz, Stone, Pearson, Pulimood, & Swizer, 2011; Wolz, Stone, Pulimood, & Pearson, 2011).

Science and mathematics are perhaps the most natural disciplines in which to integrate computational thinking. Numerous undergraduate level courses incorporate computational thinking with biology (Khuri, 2008; Matthews, Adams, & Goos, 2010; Qun, 2009; Robbins, Senseman, & Pate, 2011) or general science (Hambrusch, Hoffman, Korb, & Haugan, 2009). One initiative to bring computational thinking into K-12 classrooms is *Computational Thinking for the Sciences*, a three-day summer workshop for high school teachers to learn how to incorporate computational thinking into their mathematics and science classrooms (Ahamed, et al., 2010). Other programs offer computational thinking lessons that can be incorporated into classrooms (i.e., *VCTAL*, *ECT),* but so far there is no evidence of a completely integrated science and computational thinking course at the K-12 level. Given the increasing prevalence of this type of course at the undergraduate level, it is reasonable to assume a class of this sort of course could be on the horizon for high schools.

A significant advantage to this approach is that the context used to illustrate computer science and computational thinking concepts is directly related to the other discipline that is contributing to the course (e.g. music, journalism). This application of computer science to another discipline allows students to see links between the different disciplines, and hopefully encourages them to examine the connections between computer science and other disciplines. A limitation for K-12 teachers is, again, lack of training and resources.

Computational Thinking for the Researcher

The secondary audience of this paper is the educational researcher interested in computational thinking interventions or how students learn to think computationally. In order to support ongoing improvement, it is important to learn what good teaching practice is and how to best facilitate the student construction of knowledge. There has been discussion about the pedagogical approaches to teaching and integrating computational thinking in K-12 settings; rich descriptions of the interventions and approaches are often provided. An examination of these K-12 programs shows evidence of the ongoing debate around the role of programming and technology within the introduction and promotion of computational thinking skills. While some believe computational thinking should be taught through programming, technology, and software applications with a focus on core computing concepts, others disagree (Committee for the Workshops on Computational Thinking, 2010). Lu and Fletcher (2009), for example, believe exposure to programming can be detrimental to efforts to broaden participation in computer science. They believe that "programming is to Computer Science what proof construction is to mathematics, and what literary analysis to English" (Lu & Fletcher, 2009, p. 260). Others do not think computer hardware or software is necessary. These and similar discussions are motivating for those interested in developing a research agenda focused on computational thinking.

Those acquainted with conducting research within other STEM education disciplines are familiar with the complex array of factors that contribute to student learning. Narrow assessments of content knowledge are over-aligned with the intervention, and indirect measures such as student engagement, attitude, or motivation fail to adequately document any changes in the skills knowledge acquired by students. There is a need to conceive and develop alternative ways to assess the success of STEM education interventions, and for computer science to adapt these techniques to shape its own research methods (Fincher, Tenenberg, & Robins, 2011).

There are numerous impediments to researching computational thinking at this point, but none are insurmountable or permanent. The first barrier to researching computational thinking is the lack of a common definition. A recent review of the empirical literature on computational thinking found that less than a quarter of studies included a definition of this term (Weinberg, 2012). The remainder simply cited Wing's original definition or provided no explanation or operational definition of what the phrase meant in the context in their study at all. There is a clear need to have both a validated definition, as well as a theoretical model to underpin the concept of computational thinking. These must exist to serve as a starting point in the development measures of computational thinking to assess the success of computational thinking interventions. Only then can researchers begin to thoughtfully examine programs and interventions intended to promote computational thinking competencies. According to Hemmendinger (2010), the definitions of computational thinking which are currently offered lack precision and fail to provide sufficient examples. The fact that a six-year-old concept remains a bit abstract is not surprising. Great strides have been made in fleshing out a definition, but it is important for the researcher studying the concept to be specific about what the phrase means in the context of each study.

A second, related, barrier is that computational thinking is a latent variable. It cannot be directly observed but instead must be inferred from other variables that are observed or directly measured. In order for a measure of computational thinking to be developed, a solid theoretical definition of computational thinking must exist (Netemeyer, Bearden, & Sharma, 2003). The thorough articulation of a variable's definition is perhaps the most difficult step in the process of preparing a measure of that variable or concept (Haynes, Nelson, & Blaine, 1999). The work that has been done to explore computational thinking is promising in this regard.

A third barrier is the lack of rigorous empirical research conducted to explore computer science education in general (Randolph, 2008), and even less has been done to explore how students learn to think computationally (Weinberg, 2012). Most studies of computational thinking so far have used qualitative or descriptive methods. The use of qualitative or descriptive methods is not a condemnation of a study, but paired with outcomes that too often are focused on reactions to specific programmatic components (e.g., lessons, activities), limits the usefulness of these studies to the larger community. In order to move closer to determining how computational thinking can be assessed and used to advance these fields of study, research should examine outcomes that could be generalizable to a variety contexts and programs. Instead of asking "did you like it," researchers should instead explore how to measure computational thinking skills and competencies.

At least one group has made significant progress toward addressing these barriers. The *Scalable Game Design Project* (Basawapatna, Koh, Repenning, Webb, & Marshall, 2011) studied computational thinking patterns in students as they learned by designing games. The computational thinking concepts introduced were explicitly defined. To assess student learning, students were asked to transfer the computational thinking concepts learned in one context (game design) to another context (mathematical modeling). This group has developed an online assessment tool to measure the degree of transfer of understanding of the computational thinking concepts (patterns) to real-world situations (Marshall, 2011). Results of a large-scale analysis of data related to this assessment tool are forthcoming. This approach appears promising, and researchers interested in computational thinking will be watching this project closely.

Conclusion

Computational thinking has taken a strong hold in the computer science education community and beyond, but great strides must be made before it can be institutionalized across the entire K-12 system. One significant hurdle is teacher training. Opportunities for teachers to receive professional development or training in computational thinking are fairly limited. Most training programs are offered regionally or on a small scale. For example, *Exploring Computer Science* offers a week-long professional development program for teachers, along with a coaching program to provide ongoing support (Exploring Computer Science, 2011). Carnegie Mellon partners with several industry partners to offer *Explorations in Computer Science for High School Educators* (CS4HS) workshops each summer. This program provides materials and training for teachers to use to emphasize computational thinking. It was expanded to include workshops at other universities as well (Blum, Cortina, Lazowska, & Wise, 2010)

Currently, there are no a teacher certification programs available in the United States (Margolis, 2008), and there is only one computer science teaching methods course (Yadav, Zhou, Mayfield, Hambrusch, & Korb, 2011), and this course had only one student during the 2010-11 school year. There is broad recognition that K-12 teachers need training to effectively introduce computational thinking concepts in the classroom using any of the approaches described in this paper. The National Science Foundation has initiated the CS10K project, an effort to have 10,000 qualified high school teachers in 10,000 high schools teaching a new curriculum by 2015 (Astrachan, Barnes, & Garcia, 2011). The NSF's Computing Education for the 21st Century (CE21) program has expressed its interest in supporting programs that align with the CS10K initiative.

The lack of emphasis placed on computer science at the policy level is another impediment to its incorporation into the K-12 curriculum. Until computer science courses are required for graduation, or at least allowed to count as a mathematics or science credit, the computer science standards mean very little and the emphasis placed on computer science will continue to be weak. Numerous advocacy groups and organizations have surfaced in recent years to address this obstacle. The Association for Computing Machinery (ACM), in collaboration with various corporate sponsors, has spearheaded a variety of initiatives to advance computer science education. It formed the Education Policy Committee in 2007 to engage policymakers in conversations about computer science education. The Computer Science Teachers Association, introduced in 2005, promotes and supports computer science in K-12 schools. Finally, the National Center for Women in Information Technology (NCWIT) aims to increase the participation of girls and women in computing by "(1) building a learning community, (2) creating and sharing research-backed resources, data, and research, and (3) providing a unified, amplified voice."

A clear and comprehensive definition of computational thinking will enable those outside of computer science to begin to come 'on board' and not only support, but also participate in efforts to increase students' exposure to computer science and computational thinking. It is easy to support a theoretical idea like computational thinking, but much more difficult to participate in the efforts without a clear understanding of what the movement is about. The computer science education community has made great strides toward the widespread adoption of a concrete, articulate, and understandable definition, so it is time to accelerate efforts to share that with the larger community of K-12 educators and educational researchers.

Computer science educators need to recruit others. In order to do this, they must make their goals understandable to those to whom they are appealing. The CSTA has done some of this with their publications. There are many resources available for teachers, and as of yet these have not yet been shared outside of the computer science education community other than by the schools that are involved in program development.

By introducing computational thinking and computer science into the K-12 curriculum, students will acquire skills that allow them full participation in our increasingly technological society and to broaden the variety of careers and academic programs that are available to them.

REFERENCES

Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A. & Early, J. P. (2010). Computational thinking for the sciences: A three day workshop for high school science teachers. *SIGCSE 10: Proceedings of the 41st ACM Technical Symposium on Computer Science Education,* (pp. 42-46). Milwaukee, Wisconsin. doi:10.1145/1734263.1734277

Computer Science Teachers Association (CSTA) (2011b). Operational definition of computational thinking for K-12 education. Retrieved November 11, 2011, from http://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

Astrachan, O., Barnes, T. and Garcia, D. D. (2011).CS principles: Piloting a new course at national scale. *SIGCSE '11 Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, (*pp. 397-398). Dallas, TX. doi: 10.1145/1953163.1953281

Astrachan, O., Cuny, J., Stephenson, C. and Wilson, C. (2011).The CS10K project: Mobilizing the community to transform high school computing. *SIGCSE '11 Proceedings of the 42nd ACM Technical Symposium on Computer Science Education,* (pp. 85-86). Dallas, TX. doi: 10.1145/1953163.1953193

Basawapatna, A., Koh, K. H., Repenning, A., Webb, D. C. & Marshall, K. S. (2011). Recognizing computational thinking patterns. *SIGCSE '11 Proceedings of the 42nd ACM Technical Symposium on Computer Science Education,* (pp. 245-250). Dallas, TX. doi: 10.1145/1953163.1953241

Bell, T., Alexander, J., Freeman, I. & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *New Zealand Journal of Applied Computing and Information Technology*, 13(1), 20-29.

Bell, T., Fellows, M. & Witten, I. (2002). *Computer Science Unplugged* [website]. http://csunplugged.org.

Bers, M. U. (2010). The TangibleK robotics program: Applied computational thinking for young children. *Early Childhood Research and Practice, (2)*2. Retrieved from http://ecrp.uiuc.edu/v12n2/bers.html

Blum, L., Cortina, T. J., Lazowska, E. & Wise, J. (2008). The expansion of CS4HS: An outreach program for high school teachers. *SIGCSE '08 Proceedings of the 39th ACM Technical Symposium on Computer Science Education,* (pp. 377-378). Portland, OR. doi: 10.1145/1352135.1352263

Bull, G. (2005). Children, Computers, and Powerful Ideas. *Contemporary Issues in Technology and Teacher Education*, *5*(3/4), 349-352.

Bundy, A. (2007). Computational thinking is pervasive. *Journal of Scientific and Practical Computing*, 1(2), 67-69.

Cassel, L. N. (2011). Interdisciplinary computing is the answer: Now, what was the question? *ACM Inroads*, 2 (1), 4-6.

Committee for the Workshops on Computational Thinking. (2010). *Report of a workshop on the scope and nature of computational thinking*. Washington, D.C.: National Academies Press.

Cozzens, M., Kehle, P., & Garfunkel, S. (2010). *The Value of Computational Thinking Across Grade Levels (VCTAL)*. National Science Foundation: 09-602. Discovery Research K-12., Rugters University.

Day, C. (2011). Computational thinking is becoming one of the three R's. *Computing in Science and Engineering*, *13*(1), 88-88.

Denning, P. J. (2003). *Encyclopedia of Computer science*. John Wiley and Sons Ltd., Chichester, UK.

Denning, P. J. (2009). The profession of IT: Beyond computational thinking. *Communications of the ACM, 52(*6), 28-30. doi: 10.1145/1516046.1516054

Exploring Computer Science (2011). *Exploring Computer Scinece: A K-12/University partnership.* [website]. http://www.exploringcs.org/.

Fincher, S., Tenenberg, J., & Robins, A. (2011). Research design: necessary bricolage. *ICER '11: Proceedings of the Seventh International Workshop on Computing Education Research.* Providence, RI. doi: 10.1145/2016911.2016919

Gal-Ezer, J., &Stephenson, C. (2009). The current state of computer science in U.S. high schools: A report from two national surveys. *Journal for Computing Teachers*, Spring 2009. Retrieved November 14, 2011 from http://csta.acm.org/Research/sub/Projects/ResearchFiles/StateofCSEDHighSchool.pdf

Garofalo, J., Drier, H., Harper, S., Timmerman, M. A., & Shockey, T. (2000). Promoting appropriate uses of technology in mathematics teacher preparation. *Contemporary Issues in Technology and Teacher Education*, 1 (1), 66-88.

Goode, J. (2011). *Comparing the state rates of APCS participation & enrollment by race*. (CSTA Report). Retreived from Computer Science Teachers of America website: http://csta.acm.org/Research/sub/CSTAResearch.html

Hambrusch, S., Hoffman, C., Korb, J. T. and Haugan, M. (2009). *Teaching computational thinking to science teachers*.

Hardnett, C. R. (2008). Gaming for middle school students: building virtual worlds. *GDCSE '08: Proceedings of the third internatinal conference on game development in Computer Science Education,* (pp. 21-25). doi: 10.1145/1463673.1463678

Haynes, S., Nelson, K. and Blaine, D. (1999). Psychometric issues in assessment research. In P. C. Kendall, J. N. Butcher, & G. N. Holmbeck (Eds.), *Handbook of research methods in clinical psychology* (pp. 125-154).

Hemmendinger, D. (2010). A plea for modesty. *ACM Inroads*, *1*(2), 4-7. doi: 10.1145/1805724.1805725

Ioannidou, A., Bennett, V., Repenning, A., Koh, K. H. & Basawapatna, A. (2011). Computational Thinking Patterns. [website] http://scalablegamedesign.cs.colorado.edu/wiki/Category:Computational_Thinking_Patterns

Jacobs, J. A. (2009, November 22). Interdisciplinary hype. *The Chronicle of Higher Education*. Retrieved from http://chronicle.com/article/Interdisciplinary-Hype/49191/

Khuri, S. (2008). A bioinformatics track in computer science. *SIGCSE 08: Proceedings of the 39th SIGCSE Technical Symposium on Computer Science Education.*, (pp. 508-512). doi: 10.1145/1352135.1352305

Lu, J. J., Fletcher, G. H. L. (2009). Thinking about computational thinking. *SIGCSE 09: Proceedings of the 40th SIGCSE Technical Symposium on Computer Science Education,* (pp. 260-264). Chatanooga, TN. doi: 10.1145/1508865.1508959

Margolis, J. (2008). *Stuck in the shallow end: Education, race, and computing*. The MIT Press: Cambridge, MA.

Marshall, K. S. (2011). Was that CT? Assessing Computational Thinking Patterns through Video-Based Prompts. *Proceedings of the Annual Meeting of the American Educational Research Association.* New Orleans, LA.

Matthews, K. E., Adams, P. & Goos, M. (2010). Using the principles of BIO2010 to develop an introductory, interdisciplinary course for biology students. *CBE-Life Scicences Education*, 9(3), 290-297. doi: 0.1187/cbe.10-03-0034

Meyers, A. L., Cole, M. C., Korth, E. & Pluta, S. (2009). Musicomputation: Teaching computer science to teenage musicians. *Proceedings of the Seventh ACM Conference on Creativity and Cognition*, (pp. 29-38). Berkeley, CA. doi: 10.1145/1640233.1640241

Modular Robotics (2011). *Modular robotics: Cubelets*. Retrieved November 3, 2011 from http://www.modrobotics.com/

National Center for Women & Information Technology (2011). *Computer science-in-a-box: Unplug your curriculum*. Retreived October 11, 2012 from http://www.ncwit.org/sites/default/files/resources/computerscience-in-a-box.pdf

National Science Foundation (2011). *Computing Education for the 21st Century (CE21) program solicitation (NSF 10-619)*. Washington, DC.

Netemeyer, R., Bearden, W. and Sharma, S. (2003). *Scaling procedures: Issues and applications*. Los Angeles: Sage Publications.

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

Paul, R. & Elder, L. 2001.*The Miniature Guide to Critical Thinking-Concepts and Tools*. Retreived September 2, 2011 from http://www.criticalthinking.org/files/Concepts_Tools.pdf

Perlis, A. J. (1962). *The Computer in the University*. Cambridge, MA: MIT Press.

Peterson, J. & Hickman, C. (2008). Algorithmic music composition as an introduction to computing. *Journal of Computing in Small Colleges, 24*(1). 212-218.

Pfeiffer, J. (1962). *The thinking machine*. Philadephia, PA: Lippincott.

Qin, H. (2009). Teaching computational thinking through bioinformatics to biology students. *Proceedings of the 40th ACM Technical Symposium on Computer Science Education,* (pp. 188-191). Chatanooga, TN. doi: 10.1145/1508865.1508932

Randolph, J. J. (2008). A methodological review of the program evaluations in K-12 computer science education. *Informatics in Education*, 7(2), 237-258.

Rick, D., Ludwig, J., Meyer, S., Rehder, C. & Schirmer, I. (2010). Introduction to business informatics with Greenfoot using the example of airport baggage handling. In *Proceedings of the Proceedings of the 10th Koli Calling International Conference on Computing Education Research*, (pp. 68-69). Berlin, Germany. doi: 0.1145/1930464.1930474

Robbins, K. A., Senseman, D. M. & Pate, P. E. (2011). Teaching biologists to compute using data visualization. In *Proceedings of the Proceedings of the 42nd ACM technical symposium on Computer science education*, (pp. 335-340). Dallas, TX. doi: 10.1145/1953163.1953265

Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P., & Saulters, C. I. (2010). Teaching computational thinking through musical live coding in scratch. *SIGCSE 10: Proceedings of the 41st ACM Technical Symposium on Computer Science Education,* (pp. 42-46). Milwaukee, Wisconsin. doi: 10.1145/1734263.1734384

Sontag, M. (2009). Critical thinking with Alice: a curriculum design model for middle school teachers. *ALICE '09: Proceedings of the 2009 Alice Symposium, (*Article No. 2). Durham, NC. Doi: 10.1145/1878513.1878515.

Tarkan, S., Sazawal, V., Druin, A., Golub, E., Bonsignore, E. M., Walsh, G., & Atrash, Z. (2010). Toque: designing a cooking-based programming language for and with children. *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (pp. 2417-2426). Atlanta, Georgia. doi: 10.1145/1753326.1753692

Tucker, A., Deek, F. P., Jones, J., McCowan, D., Stephenson, C. & Verno, A. (2006). *A model curriculum for K-12 comptuer science: Final report of the ACM K-12 task force curriculum committee*. Retreived from : http://www.acm.org/education/education/curric_vols/k12final1022.pdf.

Weinberg, A. E. (2012). *Computational Thinking: An Investigation of the Existing Scholarship and Research*. Unpublished manuscript.

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*. (Research Report). Retrieved from the Association for Computing Machinery website: http://www.acm.org/runningonempty/fullreport.pdf

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

Wolz, U., Stone, M., Pearson, K., Pulimood, S. M., & Switzer, M. (2011). Computational Thinking and Expository Writing in the Middle School. *ACM Transactions on Computing Education (TOCE)*, 11(2), Article No. 9. doi: 10.1145/1993069.1993073

Wolz, U., Stone, M., Pulimood, S. M. & Pearson, K. (2010). Computational thinking via interactive journalism in middle school. *SIGCSE 10: Proceedings of the 41st ACM Technical Symposium on Computer Science Education,* (pp. 239-243). Milwaukee, Wisconsin. doi: 10.1145/1734263.1734345

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S. and Korb, J. T. 2011. Introducing Computational Thinking in Education Courses. *IGCSE '11 Proceedings of the 42nd ACM Technical Symposium on Computer Science Education, (*pp. 465-470). Dallas, TX. doi: 10.1145/1953163.1953297

CHAPTER 4: CONCLUSION

With this dissertation project, I set out to survey the entire body of literature on computational thinking, identify the nature and extent of the research evidence on computational thinking, and introduce computational thinking to K-12 educators and educational researchers who might wish to incorporate the concept into their own classrooms or research. Within this final section, I will present several insights gained from this dissertation, reflect on the directions computer science education might move to advance the field, and present the next steps I will take in my pursuit to better understand and conduct research on computer science education and computational thinking.

Lessons Learned & Insights Gained

In addition to the knowledge gained from the systematic identification and examination of the literature on computational thinking, several lessons were learned and key insights gained. A few of those will be presented and discussed here.

Barriers to Systematic Review in Computer Science Education

Significant barriers exist for those who wish to conduct a systematic examination of the literature within the field of computer science education. Popular databases such as ERIC or EBSCO allow the user to conduct a search and save all of the identified articles as a batch, or list. This entire batch, including all fields associated with each article, can be imported into a bibliographic software or systematic review web application. The Association for Computing Machinery (ACM) maintains a digital library which serves as the primary database where computer science and computer science education literature can be found. This digital library presented numerous challenges during this review. First, the ACM database does not have the capacity to export search results into an external location in batches. The export of citations and abstracts to a bibliographic software (e.g. EndNote) or systematic review web application (e.g. EPPI Reviewer-4, RefWorks) is an essential step in conducting a systematic examination of a body of literature, particularly when the intent is to code or categorize it. When a search is conducted in the ACM Digital Library and the list of articles is produced, each citation must be individually opened and to the external location (i.e., EndNote or EPPI Reviewer-4). The digital library has a "Binder" feature that allows citations to be saved, but this offers no advantage over the direct export when the second major challenge is taken into consideration. The absence of abstracts in exported citations is a second challenge with the ACM Digital library. To have both citations and abstracts for review, a five step process had to be completed for each of the over 7,000 citations. Each was 1) opened individually, 2) imported into EndNote, 3) the abstract text was manually highlighted and cut, 4) the EndNote reference file was opened manually, and 5) the abstract text was pasted into the appropriate EndNote field. Numerous processes were attempted, and this was the least time-intensive.

A third limitation to a systematic review using the ACM Digital Library is that it does not allow behaviors that appear to be automated. During this project, the repetitive process necessary to obtain citation information for each article appeared automated; the process described above caught the ACM's attention and access to the digital library was repeatedly blocked. Each time this occurred, an email was sent to the ACM Digital Library staff with a request to reinstate access and searching privileges. At times it took multiple days for access to be reinstated. This time consuming import process, combined with the accessibility issues were a substantial hindrance and a barrier to this and subsequent reviews.

A fourth limitation to the ACM Digital Library is that many articles are included more than one time in different publication formats. For example, an article would appear identically in a conference proceeding and also in a journal. Inconsistencies in how the citation fields were included within each of the publication sources or slight variations in the title prevented these from being recognized in an automatic search for duplicate records. A manual search through the citations was required to identify and eliminate duplicates.

While these challenges are not insurmountable, they turn a process that should require a relatively small time commitment into an activity that takes days or even weeks rather than hours. Neither academic librarian at Colorado State University nor staff members at the ACM Digital Library was able to offer suggestions to improve the process. The academic librarians were surprised at the lack of sophistication or features available.

Because of these substantial limitations, it is not surprising there have been so few reviews of literature in Computer Science education. Only one dissertation (Randolph, 2007) and three articles (Randolph, Julnes, Sutinen, & Lehman, 2008; Randolph, Julnes, Bednarik, & Sutinen, 2007; Valentine, 2004) describe reviews of computer science education research and evaluation studies, and three of these describe different components of a study that examined a single set of articles.

Need for Collaboration

The need for collaboration between computer science experts and educational or social science researchers is the second major insight gained. The computer science education community as a whole does not seem to be aware that its research is not up to par with educational research in other disciplines. Computer science education articles and studies are written by computer scientists, who approach research from a vastly different perspective than a social scientist might. Computer science has its foundation in mathematics and logic, which is distinct from the social sciences in terms of the degree of certainty that is the result of research.

Logic and mathematics, and therefore computers science, claims are based on the simplest of all objects of investigation: abstract objects such as propositions and numbers (Dodig-Crnkovic, 2002). Research methods and data analysis are clear-cut and the resulting findings are rarely questioned and studies are not often replicated by other researchers. Admittedly, there are aspects of Computer Science that extend into the realm of natural sciences (e.g. artificial intelligence uses physics, biology, possibly even psychology), but these forays into the natural and social sciences do not prepare the computer scientists for social science research. The social sciences are much more complex than this, and require the research to rely on theoretical frameworks and provide links to existing research. The thorough articulation of a variable's definition is perhaps the most difficult step in the process of preparing a measure of that variable or concept (Haynes, Nelson, & Blaine, 1999).

These issues signify the need for collaboration with social scientists and educators. The claims made about the need for and efficacy of computer science interventions bring along with them a responsibility to understand the social science discipline and the acceptable research methods required to produce credible evidence. This begins with an understanding of what an intervention study is and all that must go into it, including design, measurement, and analysis considerations. If claims of efficacy are to be made, the psychometric properties of constructs (e.g. computational thinking) and subconstructs must be articulated. The measures must be theoretically aligned with the intervention and have undergone rigorous development and testing procedures to ensure their reliability and validity, and design techniques must be considered as the study is developed.

Challenges Not Unique to Computer Science

A third and final insight gained is that the challenges computer science education currently faces mirror what other disciplines have experienced. The problems the field is encountering mirror what engineering education experienced some years ago as it struggled to find its place in schools, and to learn to conduct educational research that was considered robust and credible. While this is not a topic that came out in either of the manuscripts, it is an area of investigation that I embarked upon when I began to ponder how computer science educators might come to realize the need to make changes to how research is being conducted in their area of interest.

Next Steps

I embarked on this effort with the realization that the establishment of a theoretical model of computational thinking is a precursor to widespread pedagogical reform. Researchers and curriculum developers focused on encouraging computational thinking must assess this concept to determine the effectiveness of the proposed interventions. Computational thinking is a latent variable, one that cannot be directly observed but instead must be inferred from other variables that are observed or directly measured. In order for a measure of computational thinking to be developed, a solid theoretical definition of computational thinking must exist (Netemeyer, Bearden, & Sharma, 2003). This dissertation was conducted, in part, to explore the theoretical models and measures used by researchers studying computational thinking so that I could continue to refine how computational thinking is being studied as an outcome variable in the current *The Value of Computational Thinking Across Grade Levels 9-12 (VCTAL)* study and in future studies where the concept is quantitatively measured or assessed.

The concept is not as advanced as I believed when I began this project. The recent introduction of an operational definition by the CSTA is a positive step, but should be seen as a launching point for the development of measures to assess computational thinking rather than an endpoint in the development of this concept. I would like to develop alternative ways to assess the success of STEM education interventions, including those that target computational thinking. Narrow assessments of content knowledge are over-aligned with the intervention, and indirect measures such as student engagement, attitude, or motivation fail to adequately document any changes in the skills or knowledge acquired by students. The assessment of computational thinking will target a generalizable skill that is fostered by some STEM education interventions.

Finally, instead of continuing to focus exclusively on the narrow area of computational thinking, I intend to take a step back and examine at the entire field of computer science education. I will closely examine the history and development of engineering education in K-12 settings and apply the lessons learned in this field to the context of Computer Science Education. I intend to first produce a manuscript that explores the possibilities of incorporating some of the ideas into a NSF Proposal submission. I believe that the field can be advanced by acquiring an understanding of how other disciplines created a niche for themselves in the K-12 curriculum. One concrete idea arose from researcher conducted by Borrego (2007) in which researchers described the conceptual difficulties encountered by engineers as they attempted to shift their perspective to that of an Engineering Educator. This study offered suggestions to help engineers overcome difficulties and conduct robust social science research. I intend to propose a similar study, and to propose a training session at a computer science education conference to share my results and share knowledge with computer scientists that will allow them to begin to apply social science research methods in their field. It is through these next steps that I hope to influence the computer science education field.

REFERENCES

Computer Science Teachers Association (CSTA). (2005) The new educational imperative: Improving high school computer science education. Retrieved November 15, 2010 from http://csta.acm.org/communications/sub/DocsPresentationFiles/White Paper07_06.pdf

Computing Research Association (CRA). (2010). CRA Taulbee Survey. Retrieved November 11, 2010 from http://cra.org/resources/taulbee/

Borrego, M. (2007) Conceptual difficulties experienced by trained engineers learning educational research methods. *Journal of Engineering Education, 96*(2), 91.

Borrego, M. (2007). Conceptual difficulties experienced by trained engineers learning educational research methods

Dodig-Crnkovic, G. (2002). *Scientific methods in computer science*. Proceedings of the Conference for the Promotion of Research in IT. Vasteras, Sweden.

Haynes, S. Nelson, K., & Blaine. (1999). Psychometric issues in assessment research. In P. C. Kendall, J. N. Butcher, & G. N. Holmbeck (Eds.), *Handbook of research methods in clinical psychology (2nd ed.).* (pp. 125-154). Hoboken, NJ: John Wiley & Sons.

Netermeyer, R. Bearden, W., Sharma, S. (2003). *Scaling procedures: Issues and applications.* Los Angeles: Sage Publications, Inc.

Randolph, J. J. (2007). *Computer science education research at the crossroads: A methodological review of computer science education research, 2000--2005.* (Doctoral dissertation) Retrieved from http://0-search.ebscohost.com.catalog.library.colostate.edu/login.aspx?direct=true&AuthType=cookie,ip,url,cpid&custid=s4640792&db=psyh&AN=2008-99011-125&site=ehost-live Available from EBSCOhost psyh database.

Randolph, J., J., G., Sutinen, E., & Lehman, S. (2008). A Methodological Review of Computer Science Education Research. *Journal of Information Technology Education*, *7*, 135-162.

Randolph, J. J., Julnes, G., Bednarik, R., & Sutinen, E. (2007). A Comparison of the Methodological Quality of Articles in Computer Science Education Journals and Conference Proceedings. *Computer Science Education, 17*(4), 263-274.

Repenning, A., Webb, D., & Ioannidou, A. (2010). Scalable game design and the development of a checklist for getting computational thinking into public schools. *Proceedings of the 41st ACM technical symposium on computer science education (pp. 265-269).* New York. doi: 10.1145/1734263.1734357

Valentine, D. W. (2004). *CS educational research: a meta-analysis of SIGCSE technical symposium proceedings*. Proceedings of the 35th SIGCSE technical symposium on comptuer science education (pp. 255-259). Norfolk, VA. doi: 10.1145/971300.971391

Wilson, C., & Harsha, P. (2009). IT policy The long road to Computer Science education reform. *Communications of the ACM, 52*(9), 33-35.

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*. (Research Report). Retrieved from the Association for Computing Machinery website: http://www.acm.org/runningonempty/fullreport.pdf

APPENDIX A: COMMITTEE APPROVED MANUSCRIPT OUTLINES

Manuscript #1: A Literature Map And Scoping Review Of Computational Thinking
Target Journal: *Computer Science Education*

I. **Introduction/background**

This study takes a systematic, disciplined approach as it first provides a broad look at the computational thinking literature, and then systematically examines the nature and extent of research evidence found within this literature.

II. **Aims**

A. To create a literature map of computational thinking from 2006-2011
B. To conduct a scoping review to identify the nature and extent of the research evidence on interventions intended to promote computational thinking.

III. **Method**

A. **Study Identification**
   1. ***Search Terms*** (*terms 1-4 derived from Google's Exploring CT site, terms 5-9 are from Computer Science Teachers' Association's definition of Computational Thinking. Further search terms will be identified using database thesauruses*) Some terms may be added post hoc based on increased familiarity with the literature
      a. problem decomposition
      b. pattern recognition
      c. pattern generalization to define abstractions or models
         i. algorithm design
      d. data analysis and visualization.
      e. data organization
      f. data representation
      g. simulations
      h. any integration of computer science with other disciplines

B. **Sources**
   1. ***Major Bibliographic Databases***
      a. ERIC
      b. PsychInfo
      c. ACM digital library

   2. ***Conference Proceedings***
      a. The Association for Computing Machinery's (ACM) SIGCSE Technical Symposium
      b. Innovation and Technology in Computer Science Education Conference (ITiCSE)

c. Koli Calling: Finnish/Baltic Sea Conference on Computer Science Education

3. ***Google Scholar***

4. ***Hand Search***
    a. Prominent CS education journals: *Computer Science Education, Journal of Research on Computing Education, The Journal of Information Technology Education*
    b. Reference lists and citation searches

5. ***Search for Grey Literature***
    a. technical reports, working papers, blogs

C. **Inclusion Criteria**
1. Computational Thinking Education or Computer Science Education
2. Time frame: 2006-2011
3. English language
4. Some criteria may be added post hoc based on increased familiarity with the literature

D. **Screening Process** *(A sub-set of the articles will be examined by another individual and IRR will be calculated)*
1. ***Step 1: Title***
2. ***Step 2: Abstract***
3. ***Step 3: Full Article***

E. **Coding Framework**

1.***Primary Coding Framework*** (literature map) – Extract the following from all
    a. **Purpose –** Categorized using a modification of Valentine's (2004) framework. This existing framework includes: Experimental, Marco Polo, Philosophy, Tools, Nifty, John Henry
    b. **Author** – name, affiliation, area of expertise
    c. **Year of Publication**
    d. **Computational Thinking definition**
    e. **Computational Thinking domains/topics**
    f. **Other data TBD** – possibly bibliometrics such as citation analysis

2. ***Secondary Coding Framework*** (scoping review)- Only reports of studies that involve human participants will be included in the secondary coding process.
    a. **Conceptual Features**
        i. Intervention description
        ii. Theoretical framework
    b. **Methodological Features**
        i. Research design

ii. Research question
iii. Participant description
iv. Participant sampling design
v. # of participants
vi. Duration of intervention
vii. Outcomes examined
viii. Measures employed

c. **Study Findings**

IV. **Findings**
  A. **Literature Map**
  B. **Scoping Review**

V. **Implications & discussion**

Manuscript #2: Computational Thinking Research: things I've learned so far

This paper will be driven in large part by the findings of Manuscript 1. It will be applied in nature and will be submitted to be presented at the 2013 *SIGCSE conference*

I. **Introduction to CS Education and CT**
  A. **Past**
  B. **Present**
  C. **(Presumed) future**

II. **The current state of CS and CT educational research**

III. **Challenges to CS Research and Evaluation**

IV. **Opportunities in CS an CT Research and Evaluation**

V. **Promising directions**
  A. **Outcomes**
  B. **Measures**

VI. **Next Steps**

VII. **Conclusion**

APPENDIX B: *COMPUTER SCIENCE EDUCATION* CALL FOR PAPERS & AUTHOR GUIDELINES

Computer Science Education aims to publish high-quality papers with a specific focus on teaching and learning within the computing discipline that are accessible and of interest to educators, researchers and practitioners alike.
Depending on their special interests, those working in the field may draw on subject areas as diverse as statistics, educational theory and the cognitive sciences in addition to technical computing knowledge.
Papers may present work at different scales, from classroom-based empirical studies through evaluative comparisons of pedagogic approaches across institutions or countries and of different types from the practical to the theoretical.
The Journal is not dedicated to any single research orientation. Studies based on qualitative data, such as case studies, historical analysis and theoretical, analytical or philosophical material, are equally highly regarded as studies based on quantitative data and experimental methods. It is expected that all papers should inform the reader of the methods and goals of the research; present and contextualise results, and draw clear conclusions.

The Editors would like to invite you to submit your article to *Computer Science Education*

Articles for consideration should be written in English and e-mailed electronically in Word or PDF format to the Editors Sally Fincher and Laurie Murphy at **cse.editors@gmail.com**
Please ensure your article includes an abstract of 100-500 words. Papers should normally be around 7000 words in length, but longer or shorter articles may be considered. For further instructions please visit the journal homepage at **www.tandf.co.uk/journals/cse** and click on the **'instructions for authors'** tab.

Instructions for authors

Papers must be original. Please send your manuscripts in Word or PDF format tothe Editors by email .All articles from authors in the USA, Canada, and South America should be sent to: Laurie Murphy, Associate Professor, Department of Computer Science and Computer Engineering, Pacific Lutheran University: cse.editors@gmail.com
Manuscripts from all other areas should be sent to: Sally Fincher, Computing Laboratory, University of Kent at Canterbury, UK: cse.editors@gmail.com
Papers should normally be around 7000 words in length, but longer or shorter articles may be considered.

Manuscripts should be typed on one side of paper with double spacing and a wide margin to the left. All pages should be numbered. All submissions must be properly formatted for reviewing (see Publication Manual of the American Psychological Association, 5th edition, 2001, for instructions). Authors' names and institutions should be typed on a separate page. The full postal and email address of the author who will check proofs and receive correspondence and offprints should also be included.
Each paper should include an abstract of 100 to 150 words on a separate page.
Style guidelines
Any consistent spelling style may be used. Please follow the APA manual for punctuation.
LaTeX template (Please save the LaTeX template to your hard drive and open it for use by clicking on the icon in Windows Explorer)

For information about writing an article, preparing your manuscript and general Guidance for authors, please visit the Author Services section of our website.
If you have any questions about references or formatting your article, please contact authorqueries@tandf.co.uk (please mention the journal title in your email).

Word templates
Word templates are available for this journal. If you are not able to use the template via the links or if you have any other queries, please contact authortemplate@tandf.co.uk

Tables and captions to illustrations. Tables must be on separate pages and not included as part of the text. The captions to illustrations should be gathered together on a separate page. Tables and Figures should be numbered consecutively by Arabic numerals. The approximate position of tables and figures should be indicated in the manuscript. Captions should include keys to any symbols used.

Figures. Please supply one set of artwork in a finished form, suitable for reproduction. Figures will not normally be redrawn by the publisher.
As an author, you are required to secure permission if you want to reproduce any figure, table, or extract from the text of another source. This applies to direct reproduction as well as "derivative reproduction" (where you have created a new figure or table which derives substantially from a copyrighted source). For further information and FAQs, please see http://journalauthors.tandf.co.uk/preparation/permission.asp Citations of other work should be limited to those strictly necessary for the argument. Any quotations should be brief, and accompanied by precise references.

Proofs will be sent to authors if there is sufficient time to do so. They should be corrected and returned to the Publisher within three days. Major alterations to the text cannot be accepted.
Free article access. Corresponding authors will receive free online access to their article through the journal website and a complimentary copy of the issue containing their article. Reprints of articles published in this journal can be purchased through Rightslink® when proofs are received. If you have any queries, please contact our reprints department at reprints@tandf.co.uk
Copyright: It is a condition of publication that authors assign copyright or license the publication rights in their articles, including abstracts, to Taylor & Francis. This enables us to ensure full copyright protection and to disseminate the article, and of course the Journal, to the widest possible readership in print and electronic formats as appropriate. Authors retain many rights under the Taylor & Francis rights policies, which can be found at http://journalauthors.tandf.co.uk/preparation/copyright.asp. Authors are themselves responsible for obtaining permission to reproduce copyright material from other sources. Visit our Author Services website for further resources and guides to the complete publication process and beyond.

APPENDIX C: SIGCSE SUBMISSION REQUIREMENTS AND CALL FOR PARTICIPATION

Formatting Requirements for all Paper, Panel, and Special Session Submissions

The requirements listed in this section apply to all papers, panels, and special sessions:

- **Title:** The title should be centered, Arial or Helvetica, bold, 18 point, and Initial Letters Capitalized Like This.
- **Author information:** The author's name(s) should be centered using Arial or Helvetica 12 point. The affiliation and address should be Arial or Helvetica 10 point, and email should be Arial or Helvetica 12 point. Two or more authors may be listed side by side. If co-authors are at the same institution and share most information, you may use only one address. Please see the templates for examples.
    - **SPECIAL NOTE FOR PANEL SUBMISSIONS:** Indicate which of the panelists is the moderator by placing the word "Moderator" in parentheses after her/his name.
- **Paper size:** You should format your submission for 8.5 x11-inch paper.
- **Margins:** Top and bottom margins should be 1 inch, left and right margins should be 0.75 inch. This is for every page including the first.
- **Columns:** Text should be presented in two columns each 3.33 inches wide. There should be a 0.33 inch space between the columns. The two columns on the last page should be the same length approximately.
- **Section heads:** Section heads are flush left, Times Roman, bold, 12 point, ALL CAPITALS, and numbered starting at 1. There should be an additional 6 points of white space above the section head.
- **Subsection heads:** If your paper has subsections, they are flush left, Times Roman, bold, 12 point, and subnumbered (for example, 1.1). Initial letters of the subsection heading should be capitalized. There should be an additional 6 points of white space above the subsection head unless it immediately follows a section head. (Please see the templates for examples.)
- **Subsubsections:** If your paper has subsubsections, they are flush left, Times Roman, italics, 11 point, with initial letters capitalized, and subnumbered (for example, 1.1.2 or 1.2.3.4). There should be an additional 6 points of white space above the subsubsection heading, unless it immediately follows a subsection heading.
- **Text:** All text including abstract should be single spaced, full justification, Times Roman, and 9 point.
- **References:** Use the standard Communications of the ACM format for references. That is, references should be a numbered list at the end of the article, ordered alphabetically by first author, and referenced by numbers in square brackets, like this [1]. Use commas for multiple citations like this [3,4]. The reference section has a regular section head (i.e., numbered, ALL CAPITALS, Times Roman, bold, 12 point), and the references are 9 point Times Roman but with ragged right justification.

- **Copyright Space:** Leave 1.5 inches of blank space at the bottom of the left column of the first page for the copyright notice. Use a placeholder copyright notice with the number X-XXXXX-XX-X/XX/X for your submission. Please see the templates for examples.
- **Required Sections:** The following unnumbered sections are required at the beginning of the document in the following order:
- **Abstract:** The abstract should be a short description of the work described in the document. The title of the section ("ABSTRACT") should be formatted as a section head (i.e., flush left, Times Roman, bold, 12 point, ALL CAPITALS).
- **Categories and Subject Descriptors:** The ACM Computing Classification Scheme is available at acm.org/class/1998/ Most submissions are likely to use category K.3.2 Computer and Information Science Education. The title of this section ("Categories and Subject Descriptors") should be formatted as a subsection head (i.e., flush left, Times Roman, bold, 12 point, Initial Letters Capitalized).
- **General Terms:** This section is limited to the following 16 terms: Algorithms, Management, Measurement, Documentation, Performance, Design, Economics, Reliability, Experimentation, Security, Human Factors, Standardization, Languages, Theory, Legal Aspects, Verification. The title of this section ("General Terms") should be formatted as a subsection head.
- **Keywords:** This section is your choice of words you would like your publication to be indexed by. The title of this section ("Keywords") should be formatted as a subsection head.
- **Other Requirements:** Do NOT use page numbers or headers/footers. Use a blank line between paragraphs.

New ACM Reference guidelines.

Elements (in most cases):

1. Author(s)
2. Year of publication
3. Title of 'document' - use initial caps on keywords and end in period.
4. Name of Site in italics if given, and followed by period.
5. Date accessed - Use 'Retrieved' followed by date as Month, DD, YYYY followed by 'from'
6. Address - Given as '{http|ftp|telnet}://path' and underlined.

Note: a web address should never be given for a formally published document whose citation is complete or for which there is a DOI. Only give a web address for informal works or online-only works or resources that cannot otherwise be found by citation and/or DOI. Author Home page URLs or Institutional Repository URLs are not the way to cite formally published literature. If citing a formally published online-only publication, use the format for that genre and add elements 5 and 6 above.

Examples:

```
H. Thornburg. 2001. Introduction to Bayesian Statistics. Retrieved March 2,
2005 from http://ccrma.stanford.edu/~jos/bayes/bayes.html

Rafal Ablamowicz and Bertfried Fauser. 2007. CLIFFORD: a Maple 11 Package for
```

```
Clifford Algebra Computations, version 11. Retrieved February 28, 2008 from
http://math.tntech.edu/rafal/cliff11/index.html

Poker-Edge.Com. 2006. Stats and Analysis. Retrieved June 7, 2006 from
http://www.poker-edge.com/stats.php
```

Page Limits

All submission must adhere to the following page limits:

- **Paper:** 6 (Increased from 5 starting with SIGCSE 2011)
- **Panel:** 2
- **Special Session:** 2

Copyright/Permission forms

All authors of accepted papers will need to submit a signed copyright form with the FINAL document.

All authors of accepted panels or special sessions will need to submit a signed permission form with the FINAL document.

Information will be sent to authors after notification of acceptance by the program committee.

Templates and Samples

Templates for submissions can be found at the ACM SIG Proceedings website. LaTeX users should use option #2 (tighter alternate style) when formatting your document.

Questions?

Contact the Publications chair:

Brad Miller
Luther College
sigcse12-publications@cs.holycross.edu

Call for Participation

SIGCSE 2013: The Changing Face of Computing
The 44th ACM Technical Symposium on Computer Science Education

March 6-9, 2013, Denver, Colorado, USA
http://www.sigcse.org/sigcse2013/
SIGCSE 2013 continues our long tradition of bringing together colleagues from around the world to present papers, panels, posters, special sessions, and workshops, and to discuss computer science education in birds-of-a-feather sessions and informal settings. The SIGCSE Technical Symposium addresses problems common among educators working to develop, implement and/or evaluate computing programs, curricula, and courses. The symposium provides a forum for sharing new ideas for syllabi, laboratories, and other elements of teaching and pedagogy, at all levels of instruction.

Submissions in line with the conference theme, 'The Changing Face of Computing', are ideal. The theme focuses our attention on how computing is changing, and how we must change in education to address the changes in computing.

PAPERS

Papers describe an educational research project, classroom experience, teaching technique, curricular initiative, or pedagogical tool. Two versions of a submission are required: a full version having author names and affiliations and an anonymous version for use in reviewing. Papers will undergo a blind reviewing process and must not exceed six pages. Authors will have approximately 25 minutes for their presentations, including questions and answers.

PANELS

Panels present multiple perspectives on a specific topic. To allow each panelist sufficient time to present his or her perspective and still enable audience participation, a panel will normally have at most four panelists, including one moderator. Panel submissions should include a list of the panelists, their affiliations, and a description of the topic, with brief position statements from panelists. Proposals with more than four panelists must provide a statement connecting the extra panelist to the effectiveness of the panel and must convincingly show that each panelist will be able to speak, and the audience able to respond, within the session time. Panel abstracts must not exceed two pages. A panel session is approximately 75 minutes.

SPECIAL SESSIONS

Special sessions are your opportunity to customize and experiment with the SIGCSE conference format. Possible special sessions include a seminar on a new topic, a committee report, or a forum on curriculum issues. More generally, they must be 75 minutes in length, held in standard conference spaces, and justifiably distinct from the panel, paper, and poster tracks. Within those constraints, the form is yours to design. Special session abstracts must not exceed two pages.

WORKSHOPS

Workshops offer participants opportunities to learn new techniques and technologies designed to foster education, scholarship, and collaborations. A workshop proposal (including abstract) must not exceed two pages. Proposals must specify equipment needs (e.g., participant-supplied laptops, room configurations, and A/V equipment) and any limitation on the number of participants. Workshops are scheduled for a three-hour session and do not conflict with the technical sessions.

BIRDS OF A FEATHER SESSIONS

Birds of a Feather (BOF) sessions provide an environment for colleagues with similar interests to meet for informal discussions. A maximum one-page description (including abstract) is requested to describe the informal discussion topic. A/V equipment will not be provided for these sessions. Approximately 45 minutes are allocated to each BOF topic.

POSTERS

Posters describe computer science education materials or research, particularly works in progress. Proposals (including abstract) are limited to two pages. Poster demonstrations are scheduled to permit one-on-one discussion with conference attendees, typically during session breaks. Prepared handouts are encouraged in order to share your work.

STUDENT RESEARCH COMPETITION

Research from all areas of computer science is considered for awards in two categories of competition: graduate and undergraduate. All submissions must represent a student's individual research contribution and a student must be an ACM student member to qualify for awards and travel grants. Entry due date is September 30, 2

APPENDIX D: BOOLEAN LOGIC

The following logic was used to search the websites and databases described in Chapter 2.

- (computational thinking)

- (computer science education) AND (thinking)

- (computer science education) AND (interdisciplinary OR multidisciplinary)

- (computer science education) AND (mathematics OR science OR biology OR physics OR reading OR writing OR journalism OR music OR art)

- (computer science education) AND ((problem decomposition) OR (pattern recognition) OR (pattern generalization) OR (abstraction) OR (algorithm design) OR (data analysis and visualization) OR (data organization) OR (data representation) OR (simulation) OR (recursive thinking)) **Note: ACM Digital Library search was conducted with (computer science education) AND (1 search term at a time)*

APPENDIX E: CODING SHEET

Screening Criteria

1. Is the literature dated January 2006-June 30, 2011?
    a. Yes – Proceed with screen
    b. No – EXCLUDE
2. Is the literature focused on computational thinking or one of the CT domains specified in the search terms?
    a. Yes – Proceed
    b. No – EXCLUDE
3. Is the literature focused on education?
    c. Yes – Proceed
    d. No – EXCLUDE
4. Is the literature introduce a conference session, tutorial, poster, event
    e. Yes – EXCLUDE
    f. No – Proceed

Substantive Coding

Round 1:

1. Year of Publication
    a. 2006
    b. 2007
    c. 2008
    d. 2009
    e. 2010
    f. 2011

2. Institutional affiliation of the primary author?
    a. United States
    b. International
        i. Country:____(list all)______

3. What is the area of expertise of each author?
    a. Computer Science
    b. Education
    c. Other area(s) ____(list all)_____

4. What population is the article focused on?
    a. K-12
    b. Elementary (K-5)
    c. Middle (6-8)
    d. High School (9-12)
    e. Undergraduate
    f. Graduate

5. What type is the primary purpose of the article?
    a. Opinion
    b. Program Evaluation
    c. Description of a Curriculum, Lesson, or Course
    d. Research
    e. Philosophy
    f. Literature Review
    g. Program Description

6. Does the article include data?
    a. No – STOP coding
    b. Yes – Include in Substantive Coding Round 2

Round 2:

1. What research methods were used?
    a. Experimental/quasiexperimental
    b. Correlation
    c. Nonexperimental
    d. Survey
    e. Qualitative
    f. Causal comparative
    g. Did not include human subjects
    h. No intervention

2. What research design was used
    a. Post only (one group)
    b. Post only (treatment/control)
    c. Pre/Post (one group)
    d. Pre/Post (treatment/control)
    e. Repeated measures (one group)
    f. Repeated measures (treatment/control)
    g. Other
    h. Correlational
    i. Causal comparative
    j. Multi methods (*also indicate each method a-i*)

3. What type of intervention was explored?
    a. Student instruction (in class)
    b. Teacher instruction
    c. Out of school time
    d. Other
    e. None

4. What outcome(s) were examined?
    a. Attitudes (student)
    b. Attitudes (teacher)
    c. Skills/knowledge
    d. Course achievement
    e. Future plans

    f. Teaching practices
    g. “did you like it”
    h. None
    i. Other

5. Measures (select all that apply)
    a. Questionnaire
    b. Course grades
    c. Teacher/researcher made test
    d. Student work
    e. Existing records
    f. Standardized or established tests
    g. Interviews
    h. Observation
    i. Other ________________
    j. Multiple measures (*also indicate each measure a-i*)

6. Type of Analysis
    a. Inferential
    b. Descriptive
    c. Qualitative

7. Computational Thinking Definition
    a. Cites Wing – no definition included
    b. Defines CT
    c. Phrase CT not used
    d. CT phrase used, but no definition or citation provided