Enacting Systemic Change: The Evolving Landscape of Computer Science Education in the State of Wisconsin

Heather Bort
Marquette University

ENACTING SYSTEMIC CHANGE: THE EVOLVING LANDSCAPE OF COMPUTER SCIENCE EDUCATION IN THE STATE OF WISCONSIN

by

Heather Bort

A Dissertation Submitted to the Faculty of the
Graduate School, Marquette University,
in Partial Fulfillment of the Requirements for the
Degree of Doctor of Philosophy

Milwaukee, WI 53201–1881

August, 2021

ABSTRACT

ENACTING SYSTEMIC CHANGE: THE EVOLVING LANDSCAPE
OF COMPUTER SCIENCE EDUCATION
IN THE STATE OF WISCONSIN

Heather Bort

Marquette University, 2021

Over the last decade, the Systems Lab at Marquette University has undertaken a grand challenge to positively impact learners and educators in the state of Wisconsin with computer science education innovation. We have moved through a progression in maturity around recognizing the need for and implementing a propagation plan to achieve this desired outcome. This work showcases several novel innovations with increasing overall effectiveness in creating a more diverse community of computer science educators and learners in the state. We demonstrate a pattern of increased engagement, sustainable change, and measurable impact, that has resulted in a more accessible and inclusive landscape for computer science education in the state of Wisconsin.

This dissertation will present five separate novel contributions to the Computer Science Education Field. 1.) The measurement of educators' ability to integrate Computational Thinking concepts into their lessons; 2.) The implementation of Computing content in humanities courses; 3.) The development of a low-cost replacement to the robotics module in the Exploring Computer Science curriculum; 4.) The establishment of a Scratch based programming competition to run in parallel with a traditional programming competition; 5.) Utilizing Disparate Impact Analysis to quantify impact to regional and economic groups in a statewide implementation of Exploring Computer Science.

Each of these initiatives will be discussed within an innovation-based propagation cycle framework. This framework, based on Diffusion of Innovation theory, expands on the scope of the typical propagation framework to accommodate the continuous development of Computer Science Education innovations and their injection into the innovation-decision process.

ACKNOWLEDGEMENTS

This work would not have been possible without the support and encouragement of my advisor, Dr. Dennis Brylow. Thank you Dr. Brylow for helping me overcome the many roadblocks encountered over this journey. I hope to one day be able to pay forward the patience and relentless motivation you have shown me.

Thank you to my family for supporting my ambition. My children, Daniel and Benjamin, will finally get to experience life with a mother who is not also a student! They have endured so much due to my constant split attention necessitated by my goal to earn this degree.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

Introduction

Computer Science Education has emerged as an area of National concern and there has been an immense push from researchers, educators, and private corporations to increase, broaden, and democratize participation in computing by all people and especially people from historically underrepresented backgrounds. The attention computer science education has been receiving has spurred a huge increase in programs aiming to expand access to computing through teacher education, policy recommendation, and the development of educational innovations across almost all Computing disciplines. While many programs have focused on the K-12 education space, few have been aimed at addressing state wide adoption. Many of those state wide initiatives have been built using cooperative efforts, such as the Expanding Computing Education Pathways (ECEP) Alliance, and focus on system wide change to expand access to Computer Science Education. Independent, innovation focused, state wide efforts are exceedingly rare in the literature, especially those whose efforts include the implementation of multiple Computer Science Education innovations instead of a limited thread focus on well established innovations meant for widespread adoption. For the few independent state wide initiatives like this in the K-12 Computer Science Education space, it is rare to find literature that undertakes to assist others in the iterative propagation planning, necessary for implementation, and scaling of those efforts that result in not only successful adoption but sustained momentum and growth of the variety of programs intended to support learners and educators. The existing literature focuses on a pathways approach with each next step leading forward towards the end goal. Our work aims to assist the independent, innovation (research and development) based, efforts that must work in a cycle rather than a path, where each next step leads back to the beginning while still supporting the achievement of the "end goal", given that the end goal will evolve over time.

Over the last decade, the Systems Lab at Marquette University has undertaken a

grand challenge to positively impact learners and educators in the state of Wisconsin with computer science education innovation. We have moved through a progression in maturity around recognizing the need for and implementing a propagation plan to achieve this desired outcome. This work showcases a number of novel innovations with increasing overall effectiveness in creating a more diverse community of computer science educators and learners in the state. We demonstrate a pattern of increased engagement, sustainable change, and measurable impact, that has resulted in a more accessible and inclusive landscape for computer science education in the state of Wisconsin.

This chapter will provide an introduction to the work by providing the necessary background and context, outlining the research problem, establishing the aims and objectives this research will focus on, and review the limitations of the research.

1.1 Background

The Preparing the Upper Midwest for Principles of Computer Science (PUMP-CS) project at Marquette University is currently engaged in it's third NSF grant. The initial grant had the ambitious goal of preparing at least 100 high school teachers for computer science classrooms by the 2016-17 school year. This goal was met in an effort to contribute to Wisconsins share of the 10,000 teachers requested for the NSF CS10K Project. At the time, Wisconsin was one of the few states with clearly defined licensing standards for K-12 computer science teachers, although we did suffer from a lack of programs that provided the necessary courses for a teacher to qualify for the license required by the WI Department of Public Instruction to teach those courses that have more than 25% computer programming content. With almost 500 public high schools in the state, there were fewer than 40 of those schools with students enrolled in an AP CS face-to-face course for the 2014-15 school year. As part of the effort to meet the PUMP-CS goal of preparing teachers for offering computer science courses, Marquette has hosted annual cohorts of high school teachers for Exploring Computer Science (ECS) professional development sessions. We have been leveraging the ECS program to develop the diverse teaching talent recruited for PUMP-CS, providing ECS training as part of an alternative pathway to secondary licensure for K-12 computer science.

Successful adoption of the ECS curriculum has been reported in places like Los Angeles [120], Ohio [101], or Chicago [58]. Even programs that are not based on ECS have reported success and provided insight to the factors affecting the high school computer science landscape [87]. Something that we must keep in mind with these success stories, and even with our own initiative, is that these efforts took nontrivial levels of funding and person hours, much of those dedicated to the question of "how" rather than "what". Going forward, we cannot be assured that the substantial amount of funding for computer science education initiatives will be available. We are challenged to find ways of optimizing the use of limited resources to ensure that initiatives like ECS and GAComputes! [82] are able to achieve scalability for replication in areas devoid of computer science education programs.

1.1.1 U.S. Computer Science Education

Computer science education has risen as a topic of importance in the United States, and while there are a vast amount of resources and attention being focused on STEM (science, technology, engineering, and mathematics) education, computer science as a field has suffered from not fitting well within any one area of STEM [132]. In fact, it has only been recently that computer science has been made visible as a separate discipline to learn and teach with the STEM+C initiative by the NSF [134]. Computer science is ubiquitous in our everyday lives, yet within the K-12 education system it has been marginalized[46]. Even as Computing is becoming visible in STEM education initiatives with STEM+C, it can still be perceived as merely a tacked on, superfluous discipline by the very nature of its initials appended position. Since computer science education has emerged as an area of concern there has been an immense push from researchers, educators, and private corporations to increase, broaden, and democratize participation in computing by all people and especially people from historically underrepresented backgrounds. In order to maintain this current momentum [132] argues that computer science needs to be seen as a core science and not perceived as a fringe or elective offering, or as purely application based literacy or coding.

The CSTA (Computer Science Teachers Association) 2013 report Bugs in the System: Computer Science Teacher Certification in the U.S. [46] points to education policy interactions at the federal, state, and local level as being responsible (perhaps

unintentionally) for perpetuating the perception of computer science as an unimportant, fringe subject. There have been policy recommendations following this work at a national level. We have identified three themes in the research surrounding policy recommendation and the state of computer science education [46, 132, 78, 77, 79]:

- Theme 1: **Computer Science must have status as a Core Subject** Students should be required to complete credits in computer science for high school graduation, or have computer science count for math or science credit towards graduation.
- Theme 2: **Equity in Opportunity** Underrepresented groups are less likely to have the opportunity to take computer science courses. Computer science should be offered at all high schools and schools should receive incentive for prioritizing computing courses.
- Theme 3: **Knowledgeable and Qualified Teachers** Licensing/certification programs should be more comprehensive and accessible for those wishing to teach computer science and states should require that computer science courses are taught by qualified teachers.

These themes can be seen referenced repeatedly in promotional media and advocacy material for computer science education. Code.org tracks these themes at the state level in the U.S., they publish maps and other documents for interested parties to use as resources in their local advocacy [44].

Many times these themes are used by initiatives to build their end goals. The Pump-CS program is no different, our original goals were focused on these themes but we have evolved over time to also include innovation as a theme to promote continued growth.

1.1.2 Wisconsin Computer Science Education

In the state of Wisconsin, the DPI (Department of Public Instruction) has begun to recognize computer science as a subject that is important and deserving of time in the K-12 curriculum. In December 2013, Wisconsin Act 63 provided local school boards with the ability to count a qualifying computer science course as a math credit towards high school graduation. [186]

> **Wisconsin State Statute 118.33 (1)(a)1. c. High school graduation standards**
>
> c. At least 3 credits of mathematics. The school board shall award a pupil up to one mathematics credit for successfully completing in the high school grades a course in computer sciences that the department has determined qualifies as computer sciences according to criteria established by the department

At the time of the initial PUMP-CS iteration, Wisconsin was one of only a few states with a computer science licensure/certification requirement for teachers. The Wisconsin Computer Science (405/1405) license was required to teach any course with at least 25% programming content [184]. This programming content percentage criteria has since been deprecated, however, at the time this was an important constraint on solution options for PUMP-CS. The state has worked to create certification pathways for teachers to obtain the computer science license and has published clear guidance for what can be considered a computer science course as opposed to a computer literacy or educational technology course [186, 184]. In 2017, the Wisconsin DPI adopted Computer Science Academic standards [187].

It would seem that Wisconsin is making progress toward providing all of its students with the opportunity to take computer science courses in high school. When we look at Wisconsin from the overall state level we see that policy has been put in place to ensure that we are meeting all three themes from section 1.1.1. However, when we take a look at the situation in individual schools across the state, we can see that not all schools offer computer science, and of the schools that do offer computer science courses, not all of them count those courses as a math credit towards graduation. Additionally, some of the schools that are offering courses that do not meet the criteria, set forth by the Wisconsin DPI, to count as a computer science course but rather a computer literacy course, are reporting those courses as computer science.

1.2 Research Problem

It has been well established that there is a need to expand access to Computer Science Education at the K-12 level, and multiple organizations are engaged in attempting

to meet that need. Almost all of the efforts that aim to expand access to or broaden participation in Computer Science at the K-12 level are focused on initial implementation and policy changes to promote sustained adoption, in essence, "carving out space for Computer Science in the schools". While our effort does indeed aim to promote policy changes and implement initial programming in schools where it is nonexistent as well as creating pathways to licensure for teachers, we recognize that at some point we will hit adoption critical mass. The current literature is lacking in material that supports innovation adoption at the K-12 level beyond the initial implementation of Computer Science as an available subject. Our work aims to plan for this by placing focus on an often overlooked role within these K-12 propagation frameworks, the innovator. The creation of innovative instructional technology, curricular innovations, and pedagogical innovations is growing significantly in the Computer Science Education field, developing a propagation model that provides for this activity is crucial for the continued growth of the K-12 programs our work supports.

1.3 Aims and Objectives

This research aims to promote the concept of iterative development in propagation planning for large scale initiatives in K-12 Computer Science Education. The focus will be on the growth of a K-12 Computer Science Education program in the State of Wisconsin and the importance of planned innovation within the propagation framework that supports that growth.

This dissertation will present five separate novel contributions to the Computer Science Education Field.

- The measurement of educators ability to integrate Computational Thinking concepts into their lessons
- The development of a low cost replacement to the robotics module in the Exploring Computer Science curriculum
- The implementation of Computing content in humanities courses
- The establishment of a Scratch based programming competition to run in parallel with a traditional programming competition

- Utilizing Disparate Impact Analysis to quantify impact to regional and economic groups in a statewide implementation of Exploring Computer Science

Each of these efforts will be placed within the larger scope of the PUMP-CS program, we will discuss the impact each has had to the statewide effort and explicitly define our proposed model for propagation. For each effort we analyze the impact that specific innovation attributes have had on the implementation and adoption of the innovation from the lens of a diffusion based framework. We then explore how these impacts have lead to the development of an enhanced framework to support continuing development of Computer Science Education innovation in the state of Wisconsin.

1.4 Significance

This work will contribute to researcher-practitioner partnerships model by describing a propagation framework specific to Innovation Development within K-12 Computer Science Education. This will help to address the need for continued growth in the development of new educational innovations to support adoption and expand access to quality Computer Science educational materials that are responsive to the specific needs of statewide programs.

1.5 Limitations

As with all research, there are several limitations to this work. Primarily, the participants in all of the interventions were self-selecting. A key component in the study of propagation of an innovation is the identification of both the time and the method by which "late majority" adopters or "laggards" are converted, since the participants sought out or chose to adopt the innovations instead of being directed or required to do so, we consider them to be members of the "innovator", "early adopter", or "early majority" categories of the adopter types classified by Rogers [155]. A longer term study focused on the sustained adoption across the state will be required to reflect those late conversions.

In addition, the geographical scope that this study was performed in, the State of Wisconsin, may be a limiting factor in the generalizability of this work. Follow on

efforts should be conducted to understand the outcomes of these innovations in other locations.

Another consideration to take into account is the availability of trusted, quality, data. All of the results in this work are compiled based on self reported data. Data collected at the time of the interventions from the participants was completely voluntarily given, not all individuals gave feedback or consent to use their data, and therefore may not represent an unbiased or comprehensive view of the intervention outcome. Individual student data was not available from K-12 classrooms for this study. All data representing student, classroom, school, or district demographics or function was determined based on information that had been reported by the schools to the Wisconsin Department of Instruction. This research was provided an anonymized version of the data for use. Therefore, demographics represent the school and are not necessarily representative of the individual classrooms. This data from the DPI was not presented in a normalized or well defined form, misinterpretation is always a risk under these circumstances. An important future avenue of research should focus on accurate data collection to provide results that are better than directional.

1.6 Structural Outline

In Chapter One, the context and background of this work has been introduced. We have identified the aims and objectives and discussed the limitations of the work presented in this dissertation.

In Chapter Two, the existing literature will be reviewed. Our focus will be on identifying State Level K-12 Computer Science Education initiatives and the Propagation Models in use for those efforts.

In Chapter Three, our work to measure the outcomes of a teacher professional development workshop will be presented.

In Chapter Four, we will explore the MUzECS module, a low cost replacement for the ECS robotics module.

In Chapter Five, we present a humanities based curriculum which incorporates computational thinking and computing concepts to meet the need for a context relevant quantitative literacy outcome.

In Chapter Six, our work to establish a Scratch based programming competition to run in parallel with a traditional programming competition is presented.

In Chapter Seven, we present the results of a disparate impact analysis, which quantifies the impact to economic and regional groups, from our state wide implementation of ECS.

In Chapter Eight, we describe our propagation cycle framework and discuss how each of the previously presented innovations contributes to that model.

In Chapter Nine, we summarize the contributions made with this work and explore the possibilities of future direction based on this work.

CHAPTER 2

Related Work

2.1 K-12 Computer Science Education

In a review of computer science education in the U.S., [132] indicates that while advocates for every academic subject may believe that the education system would be improved if only more time were dedicated to the study of their own subject and weakened by the reverse, there is a strong argument that can be made for putting more focus on computer science. They argue that there is a broad demand for computer science in IT professions, a broad demand for general knowledge of computer science, and pedagogical value in computer science. The motivation for computer science becoming a more important subject in education is commonly explained by how ubiquitous computing has become. It is argued that computer science skills are valuable in the job market and that the demand for these skills transcends the technology industry [132, 191]. In fact, some of the literature that promotes these motivators will go further by extending the argument of teaching valuable job skills into an argument that due to this we have a moral obligation to provide all students with these skills since they are required to live in a world where computing is so pervasive [46].

2.1.1 International

It is not only in the U.S. that the argument for computer science is being made. Internationally, computer science education has been an important topic for some time with rationales for including computer science as a subject in the K-12 curriculum being broken into many dimensions for further analysis. Many other countries are working on issues in computer science education; UK [30], Russia [107], Sweden [157], India [150], Italy [22], Korea [43], Germany [102, 111], France [12], Israel [74], Finland [112], and New Zealand [19]. Recently there has been awareness being brought to the lack of access to computing education in less affluent countries as well [6], work to build

an understanding of the state of Computer Science Education in Bangladesh, Nepal, Pakistan, and Sri Lanka was recently presented at ICER. In addition to International initiatives focused on formal K-12 (or the equivalent) education, one particularly extensive effort was conducted in Ireland [137] focused on outreach programs. [70] introduces us to the political dimension of computer science curriculum development. Pulling from [166], Fluck, et al., encourages us to accept that there is a political dimension in any curriculum as we investigate the development of such in Cyprus, United Kingdom, and Australia. Arguments for including computer science in schools are also broken into dimensions by [140]. With [70] labeling the major rationales as economic, social, and cultural; [140]breaks it further into economic, organizational, community, educational, learning, and learner. While this research will not directly explore computer science education at the international level, we will pull from the international arguments for including computer science in the curriculum to discover what factors may be considered important when the decision to offer computer science courses is made. For example, the economic dimension from [70] and [140] can lead us to look at factors related to the availability of computing job opportunities in a particular area and how that might have an effect on the offering of computer science courses.

Since the development of propagation plans for widespread Computer Science Education Initiatives is limited, it is important to consider the factors that have been documented in this international research. Broadening our literature review to an international scope will help to build towards generalizability in this work.

2.1.2 National

In the U.S., there have been several large-scale nationwide efforts to understand the current state of and factors that contribute to the availability of computer science education. The most well known publication to come out of these efforts is probably the CSTA report on computer science education policy Bugs in the System [46]. This report shines a spotlight on issues like the varying computer science teacher certification/licensure requirements in the U.S., the fact that it is often unclear, even within a state, what those requirements even are [153, 108]. It was found that there is not a clear picture of what computer science even is [46]. They found that even

though computer use is increasing in schools, knowledge about computing is not. With the ultimate, depressing conclusion that:

> ...it is difficult to draw broad conclusions about the certification of Computer Science teachers in the country beyond the fact that it is not working. Each state has its own process, its own definition of Computer Science, and its own ideas about where it fits in a young persons educational program (if at all). [46]

Another major source of information about the national state of computer science education has come from Google. Google has commissioned Gallup to help with their research efforts and is currently conducting a multiyear effort to better understand what influences students to choose computer science as a field of study, what barriers exist that prevent making computer science available to all students, and how stakeholders in K-12 education perceive the value of computer science to be. Their study found that while students, parents, and K-12 teachers and administrators seem to value computer science education, school and district administrators are not able to perceive a high level of demand for such in their communities [77]. Similar to the CSTA findings, the Google studies have found that a key barrier to offering computer science in more schools is the lack of qualified teachers. They also find that black students are less likely than white students to have access to computer science courses at their schools and that

> underrepresented groups face structural barriers in access and exposure to computer science (CS) that create disparities in opportunities to learn. [79]

2.1.3 State Level

The most well known example of computer science education research at the statewide level comes from Georgia. The Georgia Computes! initiative has worked to broaden participation in computing and to engage more members of underrepresented groups. Their program has included summer camps, after-school and weekend programs for 4th - 12th grades, professional development for secondary teachers, and professional development for post-secondary instructors and faculty [87, 82, 31]. In addition, The

Institute for Computing Education (ICE) was created in 2004 as a partnership between the Georgia Department of Education and the College of Computing at Georgia Tech with the goal to increase the number and quality of computer science teachers, increase the number, quality, and diversity of computer science students, and increase the number of students taking the CS-Advanced Placement (AP) course [85, 84]. The Georgia Computes! initiative administered a survey to students in Georgia colleges asking about demographics, middle and high school experiences, major, college and university experience, interest in computer science, factors influencing their engagement and how much support they felt that they had in studying computing, as well as their career aspirations. They used the results of the survey to gain an increased understanding on what interventions along the pathways might be having an effect, and where they might not be [86].

In Massachusetts, the CAITE (commonwealth alliance for IT education) effort measured progress on computing education goals (e.g., diversity, retention, and transfer from two-year to four-year programs) for the whole state with a focus on higher education, including community colleges [2]. Maryland and South Carolina both used surveys to get a deeper understanding of the computer science education landscape [57, 38]. One of the only studies to use data not collected from a survey was done in Nebraska. The Nebraska study used data reported to the state about courses offered and headcounts of students in those courses. They found that the majority of students taking computer science courses were in urban areas [62]. Utah now has the Exploring Computer Science course offered in high schools statewide [101].

We must also include acknowledgement that there are some programs that, while not functioning at a state wide level, include the same challenges and opportunities that the state wide programs do. The best example of this is the CS4ALL program which is focusing on the large NYC school district [66].

Interestingly enough, there is even information about computer science education to come out of state level research that is not focused on computing. An Arkansas study on teacher attrition rates found that teachers with certification for math, science, or computer were twice as likely as other teachers to realize opportunities for employment elsewhere [37].

Research that focuses on high school computer science teachers has been an interesting topic even before most of the current initiatives. High School computer science teachers in Kansas were the topic of a study done in 1996. Even then it was stated that computer science was an important part of the K-12 curriculum and that a trend was beginning for high school computer science teachers to lack necessary formal training [145].

2.1.4 Multi-State Collaboration

Recently, the two major state wide initiatives above, Georgia Computes! and CAITE, have come together to provide a foundation for effecting systemic change and improving the quality of Computer Science Education at the state level. The Expanding Computing Education Pathways (ECEP) program, uses the foundation developed in GA and MA to assist their cohort of 22 states in improving and broadening participation in computing education.

ECEP uses their 5-stage model of change, based on the Collective Impact model of social change at scale, to provide state wide programs a common guide for facilitating systemic change in computing education [69].

2.2 Propagation

There is not a shortage of innovation in the Computer Science Education research community, participation in the various education focused conferences within the Association of Computing Machinery (ACM) is growing and with it, the body of knowledge for K-12 Computer Science Education. However; it is often difficult to move from a proposed, or even implemented, innovation to a wide spread adoption of these improvements within Computer Science Education settings. The idea that dissemination is sufficient for promoting the adoption of innovation is common and is, perhaps, even promoted by policies such as the National Science Foundation Grant proposal process. It is clear in the guide that a plan for *dissemination* of results is required but there is not a requirement for a propagation strategy [135]. In fact, in STEM (Science Technology Engineering and Mathematics) there has been recent work to show that in order to successfully propagate a new educational method, the

publication of evidence showing their effectiveness is not sufficient [17, 73, 93]. To highlight the difference between propagation and dissemination a common understanding of how terms are used within the propagation literature would certainly be helpful to the reader at this point. For a more through review of the propagation literature in STEM and CS please refer to [168].

2.2.1 The Language of Propagation

- **Adoption**: use of an innovation beyond an initial trail
- **Propagation**: increasing the pool of adopters to meet some determined contextual threshold; scaling
- **Change Agent**: an active advocate or promoter for the usage of an innovation
- **Champion**: a specific type of **Change Agent**, typically characterized as having successfully adopted the innovation they are advocating for others to adopt
- **Dissemination**: spreading awareness and knowledge about the innovation
- **Diffusion**: the process by which an innovation spreads through a population

2.2.2 Theoretical Foundations of Innovation Propagation

Everett Rogers published his theory of the *Diffusion of Innovations* in 1962 to fill the gap between what is known and what is put into use [155]. He describes the Diffusion as

> ...the process by which an innovation is communicated through certain channels over time among the members of a social system.

This theory considers both *how* decisions to adopt innovations are made and *who* makes those decisions.

The Innovation-Decision Process

The *how* decisions are made is modeled by a five stage process shown in figure 2.1 that does not require consecutiveness or completeness. This process is and information-seeking and information-processing activity with the overall goal of reducing uncertainty about the advantages and disadvantages of an innovation.

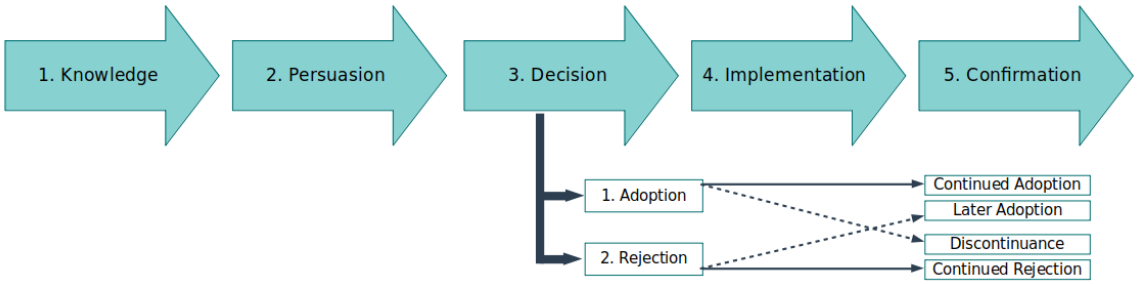


Figure 2.1: Rogers' five stage model of the innovation-decision process adapted from [155]

1. **Knowledge**: at this stage a potential adopter becomes aware the the innovation exists, this is a cognitive (or knowing) stage. This stage has its own "information seeking process" to follow as an individual moves from *awareness knowledge* (what is the innovation) to *how-to knowledge* (how does it work) and finally *principles knowledge* (why does it work)

2. **Persuasion**: here the individual forms a favorable or unfavorable attitude about the innovation, this is an affective (or feeling) stage. This stage is all about how an individual perceives the innovation.

3. **Decision**: a choice is made to adopt or reject the innovation. This stage often includes a trial of the innovation to remove an amount of remaining uncertainty about the innovation to balance with the relative advantage the innovation presents.

4. **Implementation**: the innovation is put into use. This is the first stage that

requires *behavior change* as all previous stages are mental exercises. The implementation stage can vary in length but its end point is defined by the routinization of the innovation.

5. **Confirmation**: the individual seeks to reinforce their decision. This stage lasts for an indefinite length of time where an individual endeavors to avoid or reduce (if it occurs) a state of dissonance brought about by the development of a problem or the acquisition of additional knowledge.

The implications of this decision process are important to understand. Awareness of what stage a potential adopter is in will assist with tailoring actions to influence their decision. Since most knowledge material is aimed at creating *awareness* knowledge [155], it is critical to keep in mind the ways in which *how-to* and *principles* knowledge should be presented to maintain a consistent knowledge narrative. When considering the **decision** stage, it is necessary to understand how the innovation can be "packaged" in a trial format to facilitate faster and easier adoption [155].

As stated earlier, this decision process is not necessarily dependent on an individual moving consecutively through the stages and not all stages are needed to be included in the decision making journey. Of additional note is the concept that at any stage the decision to adopt or reject the innovation can be made, even after a previous decision.

Adopter Categories

Rogers categorizes adopters into categories based on their level of innovativness and the time required for them to adopt an innovation. The categories are typically presented as in figure 2.2.

- **Innovators**: Venturesome and eager to try new and potentially untested innovations. This category has the greatest ability to deal with uncertainty and can actually be the developer of the innovation in cases.

- **Early Adopters**: Respectable, this category often serves as role model to other members of their social system, they aim to reduce uncertainty in the adoption of an innovation and have the greatest influence on others in establishing a perception of the innovation.

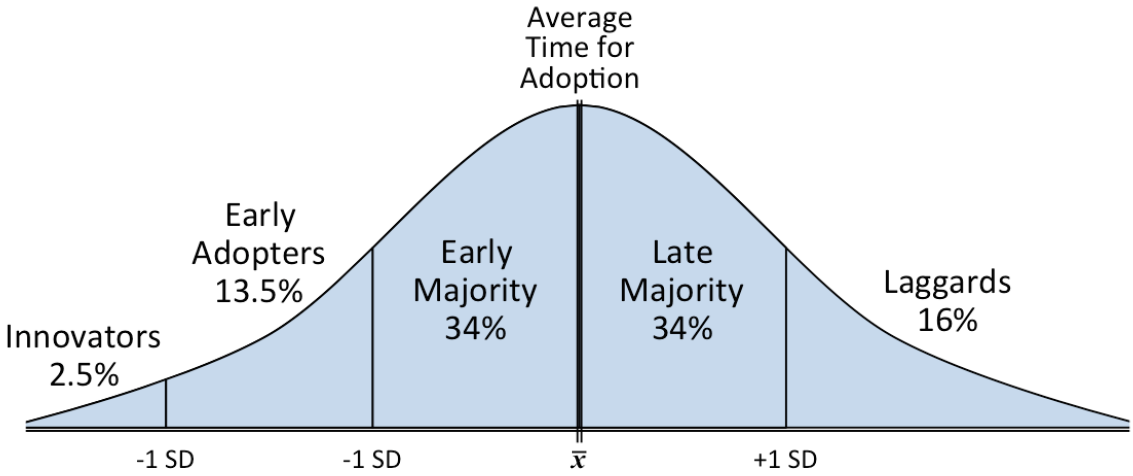


Figure 2.2: Rogers' Adopter categorization on the basis of innovativeness adapted from [155]

- **Early Majority**: Deliberate, adoption occurs ahead of the average member of the social system and deliberation may take some time. While they do not lead innovation adoption, they are willing adopter and can form the link between the early and late categories.

- **Late Majority**: Skeptical, social system pressure is a necessary component of the adoption of innovation for this category. This pressure can come from peers, economics, or resources, and nearly all of the uncertainty about the innovation must be removed before adoption can occur.

- **Laggards**: Traditional, the thought process here is focused on what has worked in the past. This category is generally isolated within social networks and their lack of resources contributes to the idea that they cannot afford to adopt an innovation until the uncertainty has been completely removed.

While Rogers cautions that these categories are generalizations,they can be leveraged to provide a summary level framework to consider the diffusion process from [155].

Bringing together the two concepts, the **Innovation-Decision Process** with the **Adopter Categories**, helps us to understand more fully that adoption is a multidimensional process. Not only can individuals (and organizations) be at different stages of the decision process but there will also be a variety of individuals from

multiple categories within any of the decision stages at any point in time. Propagation plans must consider this complication and be prepared with the right material and activities at the right time, and for the right adopter.

Innovation Attributes

In addition to the individuals level of innovativeness and their position within the decision process, there are attributes of the innovation itself that contribute to the adoption rate and success [155]. Rogers defines five attributes of innovation that have been identified as having a strong influence on this.

- **Relative Advantage**: The degree to witch an innovation is perceived to improve upon a previous (or currently implemented) innovation. Will this innovation provide a better result/outcome? Will this prevent an undesirable event? Since relative advantage has more to do with perceived advantage than actual advantage, incentives can be used to increase the perception, however, once the incentives run out the adopters may discontinue use of the innovation where the incentives were providing the necessary feedback within the confirmation stage. Relative Advantage is positively related to the rate of adoption

- **Compatibility**: The perceived compatibility of an innovation defines how well that innovation aligns with an individuals values, experience, and needs. Compatibility will inform the degree of behavior change necessary to adopt the innovation. Compatibility is positively related to the rate of adoption

- **Complexity**: The perceived complexity of an innovation can refer to either the difficulty an individual has with understanding the innovation (knowledge stage), the difficulty an individual has with use of the innovation (implementation stage), or both. This attribute is contextual to the individual adopter and should be carefully considered when building knowledge and implementation materials. Complexity is negatively related to the rate of adoption

- **Trialability**: Explicitly establishing the ability to trial an innovation will impact the perceived trialability of the innovation. At the decision stage a trial is desired and is a key attribute in reducing uncertainty about an innovation. Trialability is postively related to the rate of adoption

- **Observability**: The degree to which others can observe the results of the innovation can further reduce uncertainty and it can serve as a proxy for a trial when trialability is low. Observability is positively related to the rate of adoption

Criticism of the Diffusion of Innovations Model

Since Rogers original publication in 1962, there has been widespread criticism of the Diffusion of Innovation Theory. These criticisms largely fall into the following categories:

- **Pro-Innovation Bias**: Within diffusion research there is an implication that an innovation should be adopted by *ALL* members of the social system. Also, that the diffusion should be achieved as rapidly as possible and with little to no rejection. The cause of this issue can be attributed to the fact that many diffusion efforts are funded by the innovation change agents who have an implicit bias toward the "success" of the adoption effort. [156, 63]

- **Individual-Blame Bias**: There is a tendency to hold the individual responsible for their rejection or inability to successfully implement an innovation rather than to attribute that failure to the system within which the individual operates. Within diffusion of innovations, this can be observed when the source of the innovation is not held accountable for the inability to develop innovations that are appropriate across the adopter categories and instead apply the categorization of late adopters and laggards as being "change resistant". [92, 39]

- **Recall Problem**: There is a dependence on recall data from respondents regarding the timing and method of adoption. Through out diffusion research this information is typically gathered by point in time surveys that require respondents to condense what should be a timeline series of behaviors into a snapshot generalization. This issue can lead to inaccurate data and the inability to determine causality. [126, 48]

- **Issue of Equality**: Distribution of socioeconomic benefits of innovation may not be distributed equitably within a social system. There has been observed a

tendency for an increase in socioeconomic inequalities as a result of innovation diffusion. As described above, the pro-innovation and individual-blame biases allow for the source of the innovation to focus their efforts on the populations that are most able to adopt and most likely to show success in that adoption. This often leads to an under served population that would most benefit from the innovation and a widening of the gap between those able to implement and those that are not. [154, 24]

2.2.3 Propagation Models in Computer Science Education

There are two efforts in the K-12 Computer Science Education space that provide a detailed propagation model, the SCRIPT model from CSforAll [56] and the Collective Impact model from ECEP [69]. These models do not define a propagation practice for the adoption of specific educational innovation, rather they describe frameworks meant to promote adoption of computing or broadening participation in computing in general at a system level.

SCRIPT

CSforAll promotes a theory of change for use at the Local Education Agency (LEA) level. The purpose of this is to support LEAs in creating sustainable teams focused on planning for implementation of Computer Science Education programs within their schools. The framework they have developed to support that change theory, Strategic CSforALL Resource & Implementation Planning Tool (SCRIPT), provides LEAs with a series of workshops to develop long and short term goals, setting a vision, and assists in establishing teams that have representation at the teacher and administration or leadership levels. This theory of change suggests that short term outcomes, from the goals each district team identifies, will lead to long term impacts including more equitable outcomes for underrepresented students, and broader buy in for Computer Science Education efforts. While CSforAll has shown many positive results and have met their goals with the SCRIPT model in fostering team-led change at the district and LEA level [56], their model focuses on creating initial buy in and longer term support for those initial implementations. There does not appear to be a focus on fostering innovation within the SCRIPT model.

Collective Impact

ECEP has a focus on broadening participation in computing, this is a complex social problem further complicated by the decentralized nature of U.S. public education. They leverage the Collective Impact model of social change at scale as the foundation for their 5-stage model of change [69]. Like the SCRIPT model, ECEPs model leverages team building and goal setting, however, the 5-Stage model is described as being both top-down and bottom-up. Top-down refers to the common framework and language shared across the member states, and bottom-up suggests the ability for individual states to tailor the framework to meet their unique needs. One of the key components to the ECEP framework is their focus on the importance of data and a common measurement system to assess progress and direct areas of improvement. ECEP is using the collective impact framework to coordinate the change required to broaden participation in computing across 22 states engaged in their cohort. This framework does consider the different stages that any state may be in at any given time and presents itself as being well posed to promote sustained change within those involved states, however, as with the SCRIPT model, the teams are mainly comprised of K-12 administrative leadership and designated change agents. Only a small number of ECEP state leadership teams are positioned to foster the type of innovation where research and development are required.

CHAPTER 3

CS4Impact: Measuring Computational Thinking Concepts Present in CS4HS Participant Lesson Plans[1]

3.1 Introduction

Many current K-12 outreach efforts attempt to increase the number of students interested in majoring in computer science and related fields, with goals such as creating awareness of computer science, developing an interest in CS as a discipline, and communicating CS concepts in a way that is accessible to an audience broad in both interests and experience. These initiatives include workshops for K-12 teachers such as the Tapestry Workshop at Tennessee Technological University [162], the Computer Science for High School (CS4HS) workshops started at Carnegie Mellon and subsequently funded by Google [144, 23], and a variety of workshops sponsored either by single universities or through national funding agencies such as the National Science Foundation [65, 116, 130, 91]. Other efforts work to incorporate computer science and computational thinking into existing curricula [189, 148], develop new curricula to ensure student exposure to computer science and computational thinking principals [59], and use various media to inform teachers and students of the value of computer science [131, 21].

Most of these professional development efforts use pre- and post-test methodology to assess the transfer of factual knowledge about computer science and/or computational thinking to their target audience. While teaching the teacher is a laudable goal in itself ,there has been little effort in prior and related work to directly assess the impact of these professional development designs on actual teacher practice.

One recent study [129] attempted to measure how much of an impact programs like these actually have on teachers. This study concluded that the teachers had an improved understanding of computational thinking after the workshop they attended

[1] **This chapter is based on the peer reviewed publication [25] and has been minimally edited.**

based off of an increased percentage of correct answers to a multiple choice survey asking for a definition of the term and why they thought it might be important. Another measure that they used to determine the significance of the workshop was to ask the teachers if and how much of the material presented they thought could immediately be incorporated into their classrooms [129].

While this analysis directly measures some factual knowledge about computational thinking, it only indirectly addresses teachers' ability to incorporate this knowledge into teaching practice. Our anecdotal experience from many such previous workshops has been that teachers are typically very enthusiastic about the material immediately after a well designed session; however, follow-up surveys taken months later usually show that only a small number of teachers are able to transform the content of a 2- to 4-day CS4HS-style workshop into viable classroom practice.

In this paper, we aim to directly measure a CS4HS workshop's participating teachers' ability to synthesize the CSTA computational thinking core concepts [9] into actionable lesson plans for courses in their current teaching repertoire.

3.2 The Workshop

Over the past summer we received a grant from Google's Education Group to develop a CS4HS workshop for high school and middle school teachers to promote Computer Science and Computational Thinking in their classrooms. We undertook several goals in presenting this workshop, initially building on goals set at previous summer workshops [3, 91] to help high school and middle school teachers incorporate computer science and computational thinking into their classrooms. One of the goals was to show the direct correlation between the subjects taught by the teachers and computational thinking and to show the teachers how to use computational examples to teach current standards [3, 91]. Behind this aim is the prospect of cultivating a new crop of computer science teachers to renew the depleted resources available in our state. A second goal for this workshop was to take advantage of having assembled a number of our states computer science teachers in one space for network building, discussion of computer science educations standards, and to launch a chapter of the CSTA (Computer Science Teachers Association) for our state. Finally, we specifically targeted an audience of teachers with a background in teaching computer science in

order to identify schools and lend support to those schools interested in piloting the Exploring Computer Science [65] and AP Computer Science Principles [47] courses for the 2013 school year [34].

The participants were split into two tracks, a "basic" track A for teachers with little or no computer science background, and an "advanced" track B for those with experience teaching computer science. Both tracks had time allotted to learn and practice hands-on programming skills with either SCRATCH or Alice. The summary schedule is shown in Figure 3.1.

The basic track concentrated on computational thinking, some aspects of the Exploring Computer Science [65] curriculum, and ideas for working more computing-centric modules and motivation into existing high school STEM courses while still addressing relevant state standards.

The advanced track included sessions on the new AP Computer Science Principles exam being developed, with particular focus on topics that the AP CSP pilot sites have shown to be more troublesome in translating between the university context and the high school context. Some sessions met with everyone, and teachers were permitted to cross tracks if they were more interested in the opposing session.

Workshop structure included a variety of lecture-style presentations, hands-on lab sessions, mixed sessions, and panel discussions. We had two 2-hour sessions for lesson plan preparation late in the day on Monday and Tuesday, and then participants shared highlights of their lessons plans with the full group on the final day.

Because we drew participants almost exclusively from our state, we included sessions on state-specific curriculum and licensure issues, and worked hard on getting critical mass together for a pending CSTA chapter application.

Participants were given the opportunity to fill out an evaluation at the end of each day to inform us of which parts of the workshop they found most valuable, and in what areas we could improve.

3.2.1 Focus on Computational Thinking

A major focus common to both tracks in this workshop was to impart a working definition of computational thinking, how it is an entity separate from computer

Day 1	
7:30	Breakfast/Registration
9:00	Welcome and Introductions
9:45	Session 1: [A] Exploring CS and Computational Thinking [B] AP CS Principles
11:00	Session 2: Algorithms
12:15	Lunch
1:00	Session 3: SCRATCH
3:15	Session 4: [A] Boolean Building Blocks [B] Creativity
5:00	Dinner
6:00	TechSpots: Augmented Reality, POV Copter, Embedded Xinu
6:00	Lesson Planning / Open Laboratory
8:00	Daily Evaluation
Day 2	
7:30	Breakfast
8:30	Session 6: [A] HPC and Sciences [B] Big Data
9:45	Session 7: State and Curriculum Issues
11:00	Session 8: [A] CT and the Sciences [B] Impact and the Internet
12:15	Lunch
1:00	Session 9: [A] Alice [B] SCRATCH
3:15	Session 10: Problem/Project-Based Learning and Computational Thinking
5:00	Dinner
6:00	TechSpots: Mobile Computing, Finch, Lego MindStorms
6:00	Lesson Planning / Open Laboratory
8:00	Daily Evaluation
Day 3	
7:30	Breakfast
8:30	Presenting Lesson Plans
11:00	Google Keynote Speaker: *Go* Language
12:15	Lunch
1:00	Session 15: Careers Panel
2:00	Closing and Final Evaluations

Figure 3.1: Workshop Schedule

science, and how to integrate the computational thinking skill set and vocabulary in a classroom setting.

CSTA's K-12 Computer Science Standards manual makes the distinction between computer science and computational thinking, defining *computer science* as:

> "...the study of computer and algorithmic processes, including their principles, their hardware and software designs, their application, and their impact on society"

Whereas *computational thinking* is defined as:

> "...a problem-solving methodology that can interweave computer science with all disciplines, providing a distinctive means of analyzing and developing solutions to problems that can be solved computationally" [9]

In an article explaining her thoughts on computational thinking, Jeannette Wing states that:

> "...it represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use" [181]

Computational thinking is a method of problem solving, using abstraction and analytical thinking to arrive at a *best* answer. It is not about using a computer to solve a problem, it is more about thinking in an algorithmic way [182].

While applications of computational thinking in computer science may seem rather obvious and even redundant to those of us steeped in the field, when we try to define computational thinking outside of computer science it tends to be a somewhat esoteric concept; this can lead to some common misconceptions regarding the subject.

Firstly, computational thinking does not imply computer programming; a computer programmer can use computational thinking but so can any other person. Computational thinking is a way of conceptualizing a problem that

> "requires thinking at multiple levels of abstraction" [181]

Next, Computational thinking should not be defined as thinking like a computer. It is a fundamentally human way of thinking, drawing on concepts from mathematical and engineering thinking. Also, computational thinking does not promote cookie cutter

approaches to problems; it provides a method for thought organization and process optimization for solving complex problems. The computer is a tool, and it can be used to help us visualize abstract concepts [181, 182]. As there is not a concrete, finite definition of computational thinking, the CSTA and International Society for Technology in Education (ISTE) have entered into a project to develop one. They have compiled a list of core computational thinking concepts and capabilities and given examples of how to incorporate them in a variety of disciplines. The concepts are data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, parallelization, and simulation [14].

All of our workshop participants were given this information on computational thinking and spent time during sessions engaged in group and individual projects designed to give them experience using the CSTA and ISTE developed capabilities to highlight the core concepts in problem solving activities. The participants were also provided with information regarding the decline in computer science graduates, particularly the sharp drop in the participation of women in computer science, and the increased demand for qualified individuals in the field.

3.2.2 Under-representation

In an economy where perople are struggling to find jobs, the field of computer science is expected to have an increase in employment of 22 percent over the next decade and yield and average income 74.07 percent higher than the national average yearly income over all occupations [36]. There has been a concerning trend of decreased enrollment in computer science programs, particularly in the case of women. 2.61 percent of all bachelor's degrees awarded in 1990 were computer science degrees and in 2009 that percentage dropped to 2.38. In 1990, 30.24 percent of the bachelor's degrees awarded in computer science went to women; in 2009, it was only 17.91 percent. In fact, while the percentage of bachelor's degrees in all majors earned by women has increased from 53.31 percent in 1990 to 57.29 percent in 2009, the percentage of those women who majored in computer sciences has decreased from 0.79 percent in 1990 to 0.43 percent in 2009 [133].

3.2.3 Feedback

Overall, the participants gave favorable feedback. They were asked to complete an anonymous survey; some of the questions gave an opportunity to leave essay style feedback and other used a 5-point Likert scale (1=not at all, 5=extremely). The mean values of the Likert type questions are as follows:

1. Overall, I found this workshop to be worthwhile. 4.94
2. I felt a sense of community among participants of this workshop. 4.71
3. If you are a teacher, how likely is it that you will incorporate knowledge you've learned into your curriculum? 4.93

When asked if they would recommend this workshop to other colleagues, all respondents answered in the affirmative.

From this we conclude that our goal for helping teachers develop lesson plans incorporating the material from the workshop into their current teaching field was successful. As noted in an earlier section, it appears that most workshops of this type that ask teachers to rate the likelihood of incorporating workshop material into their classroom get reasonably positive responses. However, a separate follow-up survey will be necessary to determine if the teachers' self-assessment on this point was correct - that they did, indeed, incorporate material from the workshop into their teaching practice during the following school year.

Our secondary goal of network building and establishing support for a new CSTA chapter was accomplished as evidenced by the response to the question 2 and that all participants that were not currently on the CSTA mailing list did sign up to begin receiving materials and information from the CSTA.

3.3 Evaluation of Lesson Plans

Most prior work on examining the impact of professional development interventions for K-12 CS teachers stops with the indirect measures detailed in the previous section. Teachers are surveyed to see if they have learned anything, asked how much they liked the workshop, and asked to guess how much of it they'll use in the future.

Proper longitudinal follow-up studies of teacher participants are much rarer, and again only involve indirect measures. Follow-up with their students is even more unusual; the authors of this paper are not familiar with any such work in the literature specific to K-12 computer science. This is not a surprising state of affairs; longer-term studies are expensive and require a dedicated cadre of participating teachers - typically in a funding category beyond most K-12 outreach grants. Studies involving K-12 students directly present significantly higher barriers in terms of proper institutional review board approvals, and again require more resources than are available from the majority of funding agencies focused on K-12 CS outreach.

While we were not able to directly measure the impact of our workshop on participants' students in the subsequent school year, or to directly observe changes in teacher practice in their home classrooms, the structure of our workshop allows us to directly measure the next best thing - lesson plans crafted by participants during the course of the workshop, ready to incorporate computational thinking concepts into course they already teach.

We have developed an evaluation rubric (Figure 3.2) to rate the teachers effective use of the computational thinking core concepts based on a structure similar to the inquiry level rubric [35]. The lesson plans could receive a possible rating from zero to eighteen, each of the nine core concepts were assigned a level based off of the teachers use of that concept in their lesson plan. Points were awarded for each of the core concepts; zero points for not incorporating the concept, one point for incorporating the concept, and two points for incorporating the concept in a way that encourages high-level inquiry in their students. The teachers could then earn up to two additional points for connecting their lesson to another field, for a total possible score of twenty points.

3.3.1 Results

Videotaped recordings of 28 participant lesson plan presentations and the written deliverables shared with the group were categorized and coded according to the rubric described in the previous section.

The average score was 7.31, with no lesson plan scoring higher than 14. One of the most successful presented lessons required the students to learn about DNA molecules

Core Concept	0	1	2
Data Collection	Not incorporated	Provides the data that student will use	Students are required to collect their own data
Data Analysis	Not incorporated	An interpretation of the data is given to the student	Students will analyze the data
Data Representation	Not incorporated	The student is given a specific method to use	Students are able to choose their own method
Problem Decomposition	Not incorporated	An outline or similar structure is provided to the student	Students are required to break the problem down on their own
Abstraction	Not incorporated	Provides an expected outcome	Student arrives at an outcome
Algorithms and Procedures	Not incorporated	The basic steps for an algorithmic solution are provided	Students develop an algorithm or procedure
Automation	Not incorporated	Students are provided with a program or some other technology that automates their process	Students are able to automate their process
Parallelization	Not incorporated	Students are instructed to work in parallel	Students will decide how to distribute their workload
Simulation	Not incorporated	Students are shown a simulation	Students will produce their own simulation
Connection to Other Fields	Not incorporated	The connection is given to the student	Students are required to make a connection to another field

Figure 3.2: Lesson Plan Evaluation Rubric

Core Concept	0	1	2
Data Collection	7	6	3
Data Analysis	9	4	3
Data Representation	8	6	2
Problem Decomposition	5	10	1
Abstraction	5	9	2
Algorithms and Procedures	4	9	3
Automation	3	12	1
Parallelization	12	2	2
Simulation	0	13	3
Connection to Other Fields	10	6	0

Figure 3.3: Lesson Plan Evaluation Results

and then create a model of the molecule through a process using the computational thinking concepts. Other well-planned lessons involved creating truth tables to determine the care for patients in an emergency room and a project requiring students to present a report on an historical figure in computer science by programming an animated interview. The lesson plans that scored the least number of points did not involve any activity for the students; the teachers did not plan an activity, instead they gave an example of a program that they had written or found online. Figure 3.3 tabulates the number of lesson plans scored in each category.

Our evaluation of the lesson plans shows that the participants were largely unable to effectively integrate the computation thinking core concepts into lessons that would hold any meaningful significance with the students in their classrooms.

The large number of lesson plans scoring 0 levels on the rubric are to be expected. Our workshop specifically targeted a cohort of non-CS STEM teachers, most of whom were attending their first professional development experience in computing. Demonstrating proficiency in developing computational thinking core concepts in domain-specific lesson plans cannot be expected of everyone after only two days of instruction. Among the experienced CS teachers, some are firmly entrenched in a pedagogical style that still emphasized conveying facts and programming language syntax, not in focusing on skill building. Nevertheless, after only two days of instruction, a large number of participants were able to produce lesson plans with level 1 or level 2 components, sometimes in multiple core areas.

In the absence of other similar data for comparison, it is not yet possible to objectively quantify how good these results are. However, that was not the point of this iteration of data collection. With a baseline and a rubric, we can now design more specific interventions to increase the number of high-scoring lesson plans at next summer's workshop - to measure if we can improve teachers' abilities to construct lesson plans that should challenge students to a deeper level of inquiry in computational thinking tasks.

3.3.2 Other Analyses

Based upon a careful analysis of free-response survey results, it is not clear that most workshop participants were able to make the distinction between computational thinking and computer science. Of the 231 comments made through the daily evaluations and workshop survey, only 6 mentioned computational thinking; and while all of the lesson plans presented used some element of computer science, only 4 of them mentioned computational thinking explicitly or used the associated vocabulary.

There were 2 comments concerning the gender issues that were discussed in one of the sessions:

- "Gender divide: It was the males that were able to talk the tech talk about network configurations and hardware and electronic specs."
- "Let's talk about specific ways to improve middle school CS and females in CS - not just that we need to do it."

The first comment highlights our failure to present women as having an equal role in the field of computer science. While our sessions decry the stereotypes that inhibit women from entering into computer science, the only female presence we were able to recruit for our careers panel had the epitome of a typecast soft science non-technical job. Like many smaller computer science programs, we have graduated only a handful of female majors in the recent past, and our network of female alumnae working in the field and in our metropolitan area is nearly an empty set.

The second comment is concerning, as it implies that the concept of integrating computational thinking and computing concepts into existing non-computer science

courses content to increase awareness and interest in computer science was not realized. We feel that this may be due to an under-generalization of the computational thinking concepts and a failure to present this problem solving method in a way that fully expounds upon its multi-disciplinary reach.

It seems that the normal operating mode is that these workshops specifically target CS and STEM field teachers and therefore the goal of raising awareness and interest in computer science can only permeate about 30 percent of the student population [133].

In response to this trend, Towson University has recently added a series of general education courses focusing on computational thinking; they have developed outcome objectives for these courses. For their Computation Thinking in the Humanities course, students learn to design and use mapping and modeling tools, to put representational dynamics in perspective by applying the basic principals of graphs, networks, and systems, and also they will learn about the challenges associated with applying technology to the arts and humanities. Students develop algorithms and build models to define steps toward weight management in the Computational Thinking: Developing Life Skills for Weight Management course. They can use graph theory to explore social networks as well as learn to identify opportunities to use concurrency or parallel processing in everyday problem solving in the course for Everyday Computational Thinking. These along with the several other courses in this series exposes students to computational thinking as it relates to processes that they are already engaged in [59].

While the effort at Towson University is sure to have a substantial impact on the undergraduate students in their program and is a commendable endeavor that we hope to see spread to other universities, we hold that the first line for recruitment of future computer scientists and computational thinkers alike is the K-12 classroom. We feel that engaging students throughout their elementary and secondary education will promote an increased interest in computer science.

In an effort to correct the deficiency present in our workshop, we are working in collaboration with the Humanities department at a local women's college to develop lesson plans that adhere to the Department of Public Instruction common core standards [183] with the computational thinking core concepts as a problem-solving

framework. Our goal is to pilot this initiative in select humanities courses offered to education majors in the Spring 2013 semester and then open our summer workshop to high school teachers in the humanities in an effort to reach a broader, more diverse audience of educators and consequently a greater percentage of the student population that might not have had the opportunity for exposure previously.

3.4 Summary and Conclusions

Enrollment trends in university CS programs - while showing recent improvements - continue to lag disturbingly behind projected needs for graduates, and to show disheartening, persistent under-representation issues. These trends are driving a growing number of K-12 outreach efforts in the form of CS4HS-style summer workshops and other professional development interventions for K-12 teachers. The vast majority of these efforts, while well-intentioned and arguably effective in many dimensions, (teacher morale, more effective university and industrial contacts, updated tool and concept knowledge,) have done little to directly measure their results on participants' classroom teaching practice.

This paper leverages the structure of our workshop - which includes significant time dedicated to preparing and sharing discipline-specific lesson plans to incorporate core CSTA standards - to directly measure our participants' ability to construct lesson plans that encourage higher-level student inquiry with computational thinking concepts.

Our mixed results show that teacher enthusiasm about the workshop remains predictably high. (They're getting free professional development from experts, and STEM teachers of all sorts generally love to play with new instructional technology.) However, the quality of lesson plans developed leaves ample room for improvement in next year's workshop.

3.4.1 Future Work

While it is beneficial to us as researchers to be able to evaluate the impact that our workshop has on participants, we suspect that participants themselves may also benefit directly. It is one thing to check a box indicating that you have learned something

about a topic and quite another to be able to show how you will use what you have learned in your classroom; this is the reasoning behind our asking the participants to present a lesson plan in which they show how they have incorporated the ideas that they learned about computer science and computation thinking into their curriculum. We will go a step further at our next workshop by asking participants to complete an evaluation rubric for each of their peers, as well as a self-assessment of their work. We will provide them with our feedback and the anonymous feedback of their peers, as well as resources and support necessary to improve the assimilation of these concepts into their classrooms.

A brief, non-intrusive follow-up survey, next spring will allow us to correlate our initial rating of lesson plans with subsequent teacher reporting of classroom usage.

3.4.2 Acknowledgments

The authors are grateful to Google for funding our CS4HS workshop, and remain indebted to the many high school and university colleagues who contributed to planning and execution over the summer.

CHAPTER 4

MUzECS: Embedded BLocks for Exploring Computer Science [1]

4.1 Introduction

Despite its critical and growing importance, Computer Science is taught in only a small minority of United States high schools.[44] The MUzECS project has designed the hardware, software, and curriculum for a new, low-cost module to replace the expensive existing final module for Exploring Computer Science [65], an introductory high school curriculum. ECS is aimed at increasing the accessibility of Computer Science education for women and minorities in order to improve the diversity and student opportunity within computing courses.

The curriculum has proven effective in its goal, and have improved upon it by both lowering cost and creating a dialect of Ardublock [7] upon, thus making the ECS curriculum a better and more widely available learning tool. This dialect is what our entire module relies upon to be the interface students have with learning basic programming through our module.

4.1.1 Project Scope

In order to offer a cheaper alternative for low-income schools to teach Computer Science courses, we have created a new module based on an entirely new platform. Our design utilizes the Arduino Leonardo, a streamlined set of simple peripherals, a rewritten class curriculum, and a custom block-based programming language.

Students write code in our dialect of Ardublock: an open-source hobbyist language already available for enthusiast use and modification which sends its blocks as strings of code to the Arduino IDE to deploy it directly to the Arduino board. The board receives input and provides output using the set of peripherals included in the Arduino shield, a board containing peripherals that can be plugged directly into the top of

[1] **This chapter is based on the peer reviewed publication [11] and has been minimally edited.**

the Arduino, we provide. The small embedded system created by this combination of parts will closely follow the content of the previous modules during which students learned to program using Scratch[152], a block-based programming language akin to Ardublock.

The curriculum we deliver has well-defined lesson plans, assignments, and projects which meet specific student learning needs. This curriculum has been piloted at half-a-dozen schools totaling seventy individual hardware units being tested and used with our software. The student data from this pilot study last spring are currently being analyzed to see if our curriculum meets the same objectives as the current module. The data that has been processed points towards a successful integration into the curriculum.

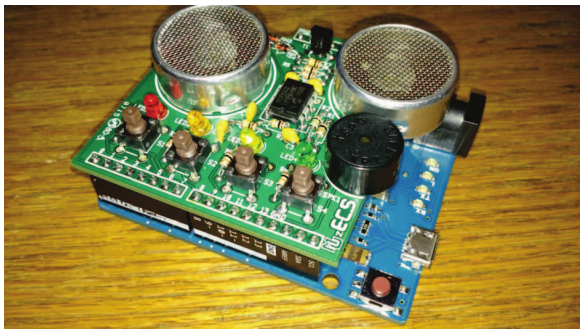


Figure 4.1: The shield and its peripherals attach to the Arduino GPIO headers

The curriculum is backed by a custom circuit board shield, as seen in figure 4.1, which mounts directly onto the General Purpose Input/Output (GPIO) headers of the Arduino Leonardo in order to power its peripherals, which the students will control via our final deliverable, the custom block-based language of MUzECS. This language, created by extension of Ardublock is to be used by students via a simple interface familiar to them from their Scratch training. Code will be deployed from this interface using a single button that prompts the Arduino software to compile to the board, ultimately executing the students programs.

We support the natural progression to an adept understanding to text-based languages from block-based by supplying control structures as blocks and making all other blocks compatible with these. The control structures in use are if-block, if-else,

if-else-if, repeat-until, repeat-while, and for-loop. A classroom could use these to create a variety of applications. For example, a block program where when an object is close it turns LED one on and when it is far away turns off and while button one is pressed a note plays, as seen in figure 4.2.

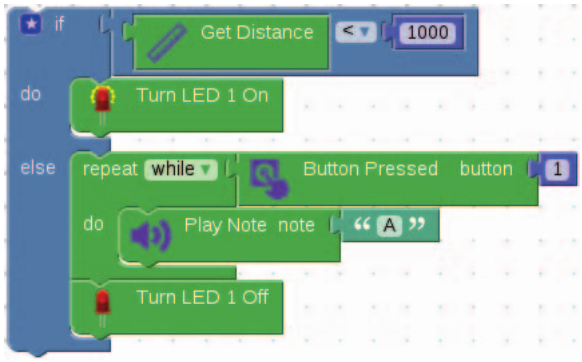


Figure 4.2: Example use of control structures similar to high level programming

4.2 Previous Research

There are many widely used visual block based programming languages and dialects in computer science education. These visual programming languages serve the needs of high school teachers with varying degrees of success. We built MUzECS to fill some of the needs of a high school computer science curriculum that some visual block based programming languages dont completely satisfy.

For instance, Scratch, is a visual programming language that makes it easy to create your own interactive stories, animations, games, music, and art. In addition, Scratch is very tinkerable, social, and diverse in the projects one can make with it. This is ideal for high school and middle school students who are just beginning to learn how to program. Scratch is easy to use and makes block programming interesting for students. Unfortunately, we have found that Scratch doesn't originally work with any open hardware and doesn't facilitate the progression to text based programming languages.

Furthermore, ECS found a block-based robot set, which supplied a visual block

based programming environment and interactive hardware, that was currently being deployed in many schools computer science curricula called Lego Mindstorms NXT[110]. There is a programmable robot set, which comes with several types of sensors, motor, and programmable brick, named Mindstorms NXT(NXT) made by the Lego Company.

MUzECS is designed to cost a fraction of Lego Mindstorms, while still supporting equivalent curriculum goals. Our MUzECS set is sixty dollars compared to six-hundred dollars for the LEGO Mindstorms NXT 2.0 by the Lego Company on Lego.com[2]. According to Barry Fagins, who is the developer of Ada for Mindstorms and has used Mindstorms in courses[123] there is no proof that Lego Mindstorms improves learning or enhances retention in computer science education.

In making our visual programming dialect MUzECS usable on all operating systems that can run the Chrome browser, we built it off of Googles open source visual block based programming environment called Blockly [76].

4.2.1 MUzECS Ensemble

Natural progression is further facilitated by requiring the user to click the 'Arduino' tab showing the Arduino code before they can upload to be compiled. This ensures the user sees the Arduino code after they use the blocks and can make connections between the conversion. This connection will hopefully ease the transition to a text-based language.

Within MUzECS there is also an 'Advanced (Pins)' section where more complicated blocks are available. We have chosen to implement this to give the user more freedom and thus creativity. The labeling 'Advanced' signifies a user with knowledge of the pin numbers on the Arduino which correspond to different hardware peripherals on the shield can use these blocks. This is also where users who would like to use the Arduino pins labeled 'advanced pins' on the shield can come to use their own peripherals with the Arduino. If a user would like to use MUzECS without the shield designed for it, this section allows for direct pin reference on the Arduino and can be used to set up other peripherals on the Arduino.

[2]htt;://shop.lego.com/en-US/LEGO-MINDSTORMS-EV3-31313?p=31313&track=checkprice

4.3 Blocks

Throughout the creation of the software, hardware, and curriculum in the MUzECS project the hardest obstacle to overcome in the whole project was creating the design for what blocks are available for student use and how they are presented in the MUzECS dialect of Ardublock. Although it is built to be usable by both beginners and those familiar with programming, the project is geared towards helping teach a ninth grade audience who are learning to program. The design decisions for how MUzECS blocks were all made with this goal in mind.

The block interface allows MUzECS to show users multiple versions of simple ideas with differing complexity while allowing more difficult programming to be abstracted behind simpler blocks. The entirety of the design attempts to make programming not only easy for a ninth grade beginner, but to provide them stepping stones towards the comprehension of more complicated ideas in programming and ease the transition from our block-based programming dialect to more standard text-based languages.

4.3.1 Overlapping Block Objective

In many high level languages there are many ways to accomplish a single objective which is why we implemented a similar ability in MUzECS. Some sets of blocks in the MUzECS dialect accomplish this in order to provide the tools for students to learn how the blocks behave in written code.

For example, the blocks 'play note time' and the combination of 'play note' or 'play note frequency' with 'no tone', and 'delay', as shown in Figure 4.3, all hold the same function: playing a musical note. 'Play note' time is the easiest to use for a beginner and holds the most abstraction of the three choices as it automatically takes a standard musical note and converts it to that note's frequency then sets a 'delay' until the note should be terminated at which point it sets a 'no tone'.

As students become more familiar with how the 'play note time' block works and the code behind it they are able to move forward and understand the building blocks of 'play note time' as they can access all of its components in block form. 'Play note' is an intermediary step that not only allows simple use of notes without having to look up frequencies for each note, but can be used to understand the underlying code

Figure 4.3: Example setup of how to play musical notes with the different methods provided by MUzECS

to 'play note time' without comprehension of what a frequency is and how that can be used to make musical notes.

The above layering of block operation offers user friendliness and intricate control in how a note is played. It allows a user to create a song from strict notes or for sounds via frequency to depend on a changing variable or to play a desired pitch between standard notes.

4.3.2 Block Abstraction

Using blocks to abstract more complex code can be used in ways other than layering their use with other blocks. This method of abstraction is helpful to hide code that is much too complex for a beginner to comprehend.

The 'get distance' block is an example of this abstraction. It may seem to be a simple data return as all it gives is the distance of an object, but it has complicated programming behind it. '[G]et distance' returns refined data that uses the time it takes an ultrasonic ping to bounce off of an object and return to the sensor. That data is then filtered and smoothed prior to being sent to the user's computer.

The block-based nature of the MUzECS dialect of Ardublock is perfect to put this code behind a simple block in the interface. We have also put it into its own function within the text-based code. This means that a beginning programmer can just as easily look at the text- based code created by MUzECS and see their program with a simple function call to getDistance(). A user may then look at both the text-based and block-based code and see a direct one-to-one correspondence between their blocks and the Arduino code as shown in figure 4.4.

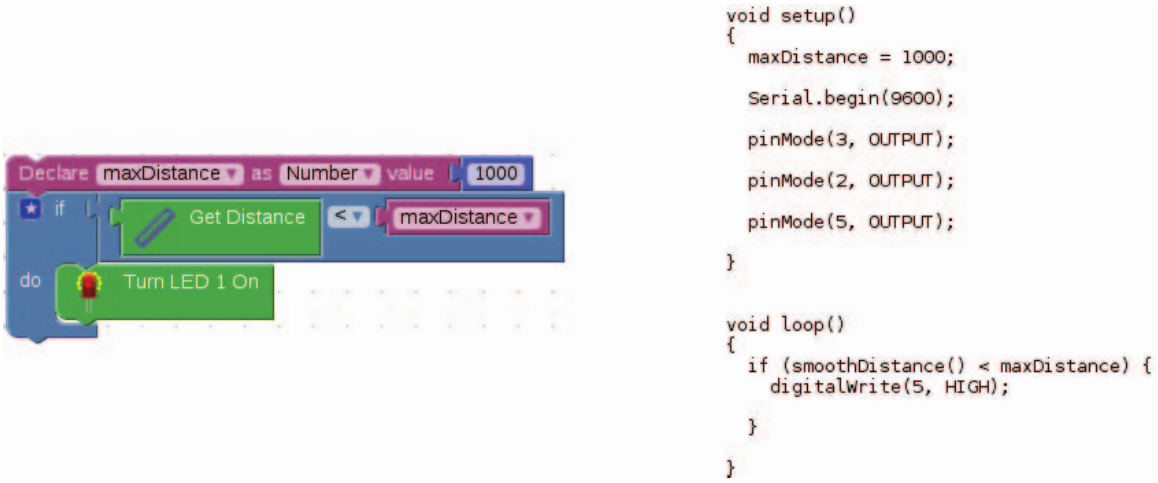


Figure 4.4: Example of the code mapping in both text-based and block-based of a sample program that uses a variable and lights an LED if an object is close to the shield

4.3.3 Visual Aids

We have found it is important to visualize the one-to-one relationship between code and output. It saves time and aids in a friendly user interface. To add this visualization, many blocks in MUzECS have associated images with them. An example of such is 'turn LED # on' with a bright colored LED image, 'turn LED # off', with a dark LED image, as seen in Figure 4.5. Similarly, 'play note' has a speaker and sound waves, and 'set up keyboard' with a keyboard image.

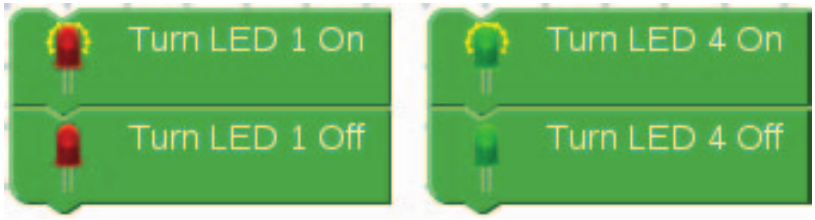


Figure 4.5: Visualization queue to quickly assist beginners in understanding what their code will do

These encourage the user to visualize what their code will do prior to compilation. When a user is able to imagine what their code will do without compiling it and seeing what the output is first, they are able to make fewer mistakes and save time. This is

also necessary for our users natural progression to adept programmers.

4.3.4 Block Expansion

We bolster the transition to text-based programming languages from block-based ones through use of setting up resources for input and output devices. The keyboard runs via the three blocks: 'setup keyboard', 'update keyboard', and 'key pressed' all of which in conjunction allow a student to read input from a keyboard just as they would a button on the shield. However, unlike a button the method to poll for a keyboard key is much more complicated than polling for a button press.

This means that while 'button pressed' is a single block the keyboard uses three. This complication is due to the button being directly on the shield, making it fairly easy to interact with while the keyboard needs to be found on the users local system and interacted with via a queue. In order to use the keyboard the student must use the blocks in the following order: 'setup keyboard', 'update keyboard', then 'key pressed', as shown in figure 4.6. While all three of the blocks could be combined into one, keeping them separate allows the students using MUzECS to learn the basic differences between communicating with a button on the shield. The 'setup keyboard' block defines the variables being used by the other two blocks while 'update keyboard' sets up communication with the local computer to talk to the keyboard and checks the buffer for recently pressed keys and flags them. Finally, 'key pressed' checks for the flag of a specific key and returns true or false.

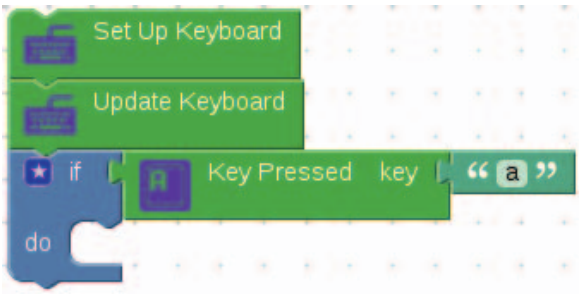


Figure 4.6: Setup of I/O to assist in transition to text-based languages

This separation allows a student to comprehend what steps are going into the

process to communicate with the keyboard and its complexity compared to communicating with a button on the shield even if they cannot grasp the specific code elements of each block. This introduces the student to the idea of communicating with different I/O devices on the computer and helps take a step towards a better understanding how the Arduino works and takes the student closer to being able to use and interact with the 'Advanced (Pins)' or 'Advanced (Code)' sections of MUzECS.

4.3.5 Error Handling

Similar to standard IDEs for text based coding which attempt to point out as many possible errors prior to compilation, MUzECS uses a system of alerts. For example, the 'button pressed' block takes any integer as input for a button, but only an integer containing only a valid button number will get passed on to be compiled. When an invalid button number is passed and the code is attempted to be sent to the Arduino IDE for compilation, MUzECS alerts the student prior to the code being sent and the blocks are converted to code. The alert message in this case consists of "Invalid button used in block. Hint: Valid button numbers are 1-4."

Additionally, 'play note' works in a similar fashion. This allows students to learn the basics of handling poor argument passing and other coding practices to be avoided prior to learning about the compilation process and how to interact with a more complex IDE in order to find the bugs in their code. Similar alerts are thrown whenever a user attempts to compile code that has errors in it such as attempting to use an undefined variable (Arduino uses its own flavor of C) or improperly setting up the overarching loop structure embedded systems like Arduinos run on.

4.4 Future Work

We will be collecting block usage statistics from high school students and teachers, who are already using MUzECS, through the use of surveys. We are eager to see if block usage statistics from MUzECS indicate a natural progression towards text based programming languages and stimulates a penchant for programming among our users. The kind of conclusions we hope to draw from this data collection is whether our users liked using MUzECS; whether they found using MUzECS difficult in any

way; what bugs they encountered with MUzECS; and what features they want added to MUzECS. Our analysis will include a coding of student final projects with both a MUzECS experimental group, and a Mindstorms control group to assess whether students are achieving similar or improved outcomes with MUzECS.

Our next step is to develop more additions to the shield in order to foster more creativity among our users and expand curriculum. We have plans to add more sensors in addition to the distance sensor on the shield, such as a temperature sensor, a light sensor, a pressure sensor, and an accelerometer.

After speaking with a significant part of our user base, we have noticed that some schools are shifting to using Chromebooks in their computer science curriculum. This is understandable as Chromebooks are cheap compared to standard laptop prices, and fit most of high school computer sciences needs. Unfortunately, Chromebooks cant install Arduino software. Due to this, we have developed a Chrome browser based web-page and web app to eliminate limited platforms. Anything that can run a Chrome browser can use our software and hardware in tandem. This is planned to be rolled-out to schools in Wisconsin this coming school year for feedback.

4.5 Conclusion

As a final point, our main goal was to design a low-cost alternative to the sixth module of the Exploring Computer Science curriculum that we piloted in schools during the Spring 2015 semester. While future work does exist and analysis of survey data will no doubt provide useful insight into the strengths and shortcomings of our designs, our team is confident that we have successfully produced hardware, software, curriculum, and resources which meet our customers needs and wants and fulfills our main objective.

As has been mentioned several times throughout this document, our projects ability to meet the subjective customer needs put forward during the initial design phase of this project are to be assessed in the future after the collection of teacher feedback and survey data post-piloting of our module. Once these data are collected, there will be extensive work to be done in improving our designs to meet the feedback. In addition, further testing work will need to be performed as new features are implemented in order to guarantee the continued reliability of the MUzECS platform. Our

team has already made plans with Dr. Brylow to continue these efforts into the fall and beyond to ensure the longevity of the project.

4.5.1 Acknowledgment

Some of the students on the MUzECS team were supported by the National Science Foundation, grants CNS-1339392, and ACI 1461264. We would like to thank our pilot teachers and students. We owe special thanks to curriculum consultants Robert Juranitch and Gail Chapman. The pilot curriculum testers both teachers and students alike, software and hardware included the work of previous student researchers Alex Calfo, Farzeen Harunani, Jonathan Puccetti, and John Casey. The MUzECS project was built on top of the existing Ardublock system which in turn was built on the Blockly architecture.

CHAPTER 5

Introducing Computing Concepts to Non-Majors: A Case Study in Gothic Novels [1]

5.1 Introduction

Computing based technology has never been more pervasive, accessible, or consumable. Computer science has altered the way we live, work, and learn. While the technology continues to advance rapidly, advancements in computer science education have been comparatively slow, creating a situation rife with significant social inequality. Foundational computer science concepts and the skills to create and innovate rather than solely consume computing technology are increasingly becoming privileged knowledge, students are attempting to enter the workforce and institutes of higher education without an understanding of the most influential discipline in science, academia, and the economy today [180].

Many believe that computer science should be elevated to a core subject in our schools. By relegating this relevant and engaging discipline to a superfluous filler status, we do a disservice to the students who will be the innovators of tomorrow [180, 71]. In order for computer science to earn a place amongst the other core academic subjects in our schools, we must show the subject to be independent and academically rigorous while at the same time proving the value of integrating its concepts across the curriculum.

This paper presents and evaluates an approach to integrating computer science and quantitative literacy concepts into an undergraduate English Literature course. The results of this approach are a set of cross-curricular activities that:

1. Deepen understanding of both computer science and English literature, without unnecessarily diluting the content of either, and

[1] **This chapter is based on the peer reviewed publication [26] and has been minimally edited.**

2. Inform students of the myriad opportunities available to them with interdisciplinary scholarship in computer science.

The concepts incorporated are used as a tool a tool that facilitates a deeper level of self-reflection and analysis, which translates to an ability to better understand and present material in the subject to which the tool is applied.

5.2 Prior Work

There is a great interest in increasing access to computer science education for all students, especially those from underrepresented populations. One of the first steps in an attempt to broaden participation should be to look at the factors that are contributing to the unbalance in representation. A major barrier to increased enrollment in computer science has been identified as student misperception of the discipline. This misperception takes the form of a general lack of understanding and familiarity with the subject [81, 190, 128, 121, 40, 95]. Add to that the pervasive cultural stereotypes that depict computer science as overwhelmingly for white males with innate ability and we have illustrated a large obstacle; not only to increasing participation, but also to effecting positive attitudes toward computer science. One of the goals of our project is to tackle this obstacle, so we directly explore the misperception and stereotypes, with the aim of altering our students understanding of computer science as an interesting and relatable subject.

Once the barrier has been identified, there are many ways in which we can work to overcome it. Some initiatives have focused on changing the way students view computer science outside of the classroom, using a widely popular approach to expose both children [64] and adults [42] to programming by gaming activities. While we applaud the success that these initiatives, and the many like them, have had; we have focused our development on students that would not be interested in the gaming aspect of computing.

Others have had success by redesigning their introductory computer science courses to be more attractive to a diverse student body [136, 4, 13]. One of the better known examples of this method, [4] uses an approach that focuses on three areas of improvement, updating the programming course to be attractive to a broader

student population, building community among students in the computer science program starting in the first year of study, and providing research experiences for their students. While our approach does focus on building community through social computing experiences and providing students with opportunities to engage in research; this project differs from these types of initiatives in that it is not a computer science course, and we are not attempting to increase the number of students graduating with a degree in computer science. The goal of our research is instead to introduce students to the ways in which computer science applications and concepts can enhance their work in other fields of study.

Another group of initiatives take advantage of the interdisciplinary nature of computer science and computational thinking by offering undergraduate computer science courses on the general education tract to appeal to non-majors [51, 178, 174, 60, 100]. The case for computing as a general education course as explained by [51], shows how having computer science courses outside of the computing major improves equity of access to learning computer science. Some of the programs use computational thinking concepts in disciplines other than computer science to improve students analytical and problem solving skills [174], expose students to concepts that they may have been previously unaware of in a variety of high interest freshman level courses [60], or in a single course designed to expose a broader pool of students in a transdisciplinary environment [178]. There are even courses that focus on improving a non-major students programming abilities as a way to meet other university requirements [100]. Much of the research in using an inter- or multi-disciplinary approach in computing and general STEM areas has have very positive results [41, 114] and is well recommended even at a K-12 level [165]. Our approach is interdisciplinary, we use computing and quantitative thinking concepts to expose students to computer science outside of a traditional computing course; however, we differ from similar work in that we are not focusing on students in STEM fields and we are not attempting to introduce computer science to students before they have clearly defined ideas about their major. We have based our work in upper level humanities courses so that students can have experiences working with computer science as it applies directly to their major.

5.3 Setting

Since the early 1970s, the second author's school has developed and implemented an ability-based curriculum, based on certain assumptions about student learning. These include: education goes beyond knowing to being able to do what one knows; and, educators are responsible for making learning more available by creating and publishing outcomes for student learning in the majors. In order to graduate, students must demonstrate development levels of eight college-wide abilities, as well as advanced outcomes in their disciplinary majors. The college-wide abilities are: communication (which includes quantitative literacy), analysis, problem-solving, social interaction, valuing in decision-making, effective citizenship, developing a global perspective, and aesthetic engagement [125].

In 1999, this institution received an NSF grant to explore and implement assessment of quantitative literacy across the curriculum, and at all levels of instruction. As a part of the project, one of the co-authors of this paper developed an assessment in a course entitled, The 19th Century British Novel, surrounding the monetary worth of characters in Great Expectations. Using a conversion formula from The Victorian Web [113], which translates 19th-century Victorian pounds into 20th-century American dollars, students calculated each character's wealth, compared the monetary possessions of two characters, and wrote an essay exploring what this exercise revealed about characters and character relationships in the novel. This example of quantitative literacy in the humanities differs from those described in the rest of this paper in that it was a single, free-standing assessment; while the students gleaned new knowledge and new strategies for engaging with Victorian novels, the assessment did not appear to lead to any longer-term changes in the ways the students approached literary analysis.

In the 2014 course offering, we integrated quantitative literacy and computing concepts throughout the course, in ways that supported the course outcomes, and built upon existing student knowledge and abilities in literary analysis. We were able to develop meaningful and useful activities by transposing current scholarly knowledge from its "original habitat" to school by working to rebuild the knowledge in a

teachable environment [28]. Information Processing Theory states that meaningfulness improves learning and retention by helping the learner in the process of encoding the information and storing it in long term memory [159].

Since the student population we were working with had no prior background in computing or quantitative literacy, we were careful to engineer the activities so that we stayed within the Zone of Proximal Development (ZPD). Vygotsky's Sociocultural theory explains that the ZPD is not only the level of potential development the learner can achieve through the guidance of a teacher, it is also the interaction between the learner and society where new forms of awareness can occur if the teacher provides the proper scaffolding [159]. As with the information processing theory, it is better to make new information by integrating it with known information, this is the reasoning behind bringing the new computer science and quantitative literacy learning into a context that the learner will already be familiar with.

5.4 Course Structure and Motivation

Mary Shelley believed readers of the Gothic novel- and she counted herself as one- derived pleasure from the genre because it spoke to the mysterious fears of our Nature. Almost one hundred years later, Virginia Woolf described the Gothic as the genre which fulfills the strange human craving for feeling afraid. At its birth in late eighteenth-century Britain, in part as a reaction to the Age of Reason, the Gothic novel reflected the crisis of identity of a nation moving into a new century. And today, in the 21st Century, the literary form still exerts a profound influence on Western popular culture.

This is a course that explores the evolution of the Gothic novel throughout the 19 th century. With readings consisting of a classic Gothic novella, an early spoof of Gothic novels, and Victorian fiction which incorporated Gothic elements. The course also examines ways that Gothic elements still appear in popular fiction, movies, and television.

Organizing questions of the course include:

- What is it about the Gothic that has made it so seductive to readers past and present?

- And what can we learn about our cultures and ourselves at this moment in human history through reading and exploring this genre?

5.4.1 Course Outcomes

Traditional Outcomes

The course outcomes for this senior-level class, incorporate the schools college-wide ability outcomes [128] of analysis and aesthetic engagement, and English major outcomes.

- To refine reading skills necessary for thorough and timely critical analyses of extended pieces of literary fiction. This includes the use of appropriate literary frameworks to analyze, evaluate, and place in context the novels we will read this semester,
- To analyze and evaluate professional literary criticism with an awareness of the main ideas, assumptions, literary concerns, and historical moment of the essay,
- To identify and describe relationships between the British novel and the culture out of which it grew,
- To formulate theories which explain changes in the form, content, and reputation of the novel, and
- To demonstrate engagement with literature through creative responses.

In addition to abilities that are integral to literary studies, such as analysis and aesthetic engagement, this course validates students in intermediate problem-solving. The student:

- Performs all phases or steps within a disciplinary problem solving process, including evaluation and real or simulated implementation,
- Independently analyzes, selects, uses, and evaluates various approaches to develop solutions, and
- Integrates quantitative abilities to effectively communicate information and respond to problems within a discipline-related context.

New Outcomes

Added to the course's abilities, for the purposes of this project, were school-wide intermediate quantitative literacy criteria, which are:

- Shows evidence of a reflective, deliberate choice to use quantitative information in a discipline related context,
- Considers use of and, as appropriate, effectively uses calculators, and spreadsheet, graphing, or discipline specific software to communicate quantitative information,
- Organizes, appropriately uses, and clearly communicates quantitative information,
- Shows a refined sense of effective ways to present quantitative information for a specific audience, and
- Evaluates her own use of quantitative information and argument and the implications of her choices.

5.4.2 Course Units

The course is broken up into 5 units, as shown in Figure 5.1, each focusing on a particular text and critical framework [15]. All of the texts are in the public domain, and available from sites such as Project Gutenberg [147].

5.5 Methods

For this study we used two different methods of collecting data. We had the students take a pre-course, free response survey. The survey asked about students' experience with computer science courses and computer programming, their opinions on computer science and mathematics as alternatives to their current major, and their perception of careers in computer science. This survey was taken again after completing the computing activities developed for the course to gain an understanding how the students perception of computer science and their confidence in computing and quantitative tasks was effected by the activities over the semester.

Unit 1: History of the Novel and Introduction to Gothic Themes
Major Texts: *The Vampyre*, by John Polidori (1816), and *Northanger Abbey*, by Jane Austen (1817) Major Framework: Formalist
Unit 2: Madness and Marriage
Major Texts: *Jane Eyre*, by Charlotte Brontë (1847) Major Framework: Feminist
Unit 3: Victorian Childhood
Major Texts: *Great Expectations*, by Charles Dickens (1861) Major Framework: Historical
Unit 4: Fin de Siècle Worldviews
Major Texts: *A Picture of Dorian Gray*, by Oscar Wilde (1890) Major Framework: Psychoanalytical
Unit 5: The Gothic in the 21st Century
Major Texts: Student Choice Major Framework: Review of all

Figure 5.1: The Five Units of the Course

In addition to the survey, we looked at student work over the semester to gauge their understanding of the concepts being presented. The work done during class was collected and evaluated to ensure that students were correctly using the new methods presented to them. In addition to class work, students were asked to give written response to research articles employing the methods explained in class to current and relevant research in literary analysis.

5.5.1 Survey Demographics

The students in the course all identified themselves as female and English majors in the survey, none of the students indicated that they have taken any computer science or programming courses as undergraduates and many of them did not understand what was meant by the question: What programming languages do you use? Some of the representative answers to this question were:

- "What does that even mean?"
- "I am not sure I understand this question enough to answer this"
- "Apparently gibberish, because I'm afraid I'm not certain what this means."

5.5.2 Course Development

Due to the student demographic we needed to develop activities that would introduce computer science concepts without requiring much background knowledge but would also be involved enough to accurately represent usefulness of computer science to the field of Literary Analysis. We were also careful to leave the original course structure intact, our aim was not to force two courses together and only get half of each. Rather, we sought to find activities that enhanced the learning outcomes and course goals that were already in place, while providing an alternate method of analysis for the students to experience. Each activity was developed to build upon the previous activities in such a way that would provide students with an opportunity to develop their quantitative literacy skills over the course of the semester instead of have a single activity that was disconnected from the other parts of the course.

5.6 Word Frequency and Test Mining

In unit one the students focused on applying a Formalist framework of literary analysis to *The Vampyre* and *Northanger Abbey*. The first course activity reviews the formalist practice of close-reading applied to a passage from *The Vampyre*, where students analyze a text at the diction level, they look for patterns and vocabulary as well as symbol and metaphor among other literary elements. This is a very common activity in literature courses and lends itself well to computational tools. As an introductory computing activity in class, the students used the online word cloud tool, Wordle [75], to visualize word choice in the assigned passage. Following the prompt: *What does the Wordle tell you about the passage? What do you think about the use of word counts for literary analysis?*

Students engaged in a discussion of word count as literary analysis. They were given an article to read [170] which relayed findings of a recent research endeavor using text mining for literary analysis and were required to write a response to the article. These are responses that are representative of the class:

- I find it very interesting that diction and syntax can be analyzed with a technological tool and in turn, can give us quantitative data about literature according to different categories.
- It is extremely fascinating what these computational methods can discover.

The goal of this first exercise was to get the students thinking about how computer science could be used in literary analysis. We used this activity to introduce the students to scholarly research that joined their own discipline to computer science and quantitative literacy.

5.7 Network Analysis

In literary analysis, using an Historical (Marxist) framework [15], the reader is interested in how socioeconomic status shapes the experiences of the characters. In previous offerings of this course, the students create a character map to show relationships and status of characters from the novel *Great Expectations*. A character map

can look very much like a graph, with the characters as nodes and the relationships as edges.

Since *Great Expectations* has an extensive list of characters with complicated relationships, we chose to begin working on graphing with students at the beginning of the semester; the network analysis activity has been a part of every unit in the course. We began with *Northanger Abbey*, asking the students to make a character relationship map for the novel. This is a process that they are familiar with as it is a common assignment in literature courses.

The maps that the students produced in small groups were all very unique, there was no class definition for relationship and therefore each group interpreted it differently, the students used different shapes and colors to represent various types of character relationships. However, all of the maps were similar in that each character was a represented by a point and some form of line connecting the points represented the relationship between characters.

Using this similarity we introduced the concept of graph and showed how this was a construction they were already familiar with through their character relationship maps. After the students had practice constructing graphs from their character maps, we introduced the adjacency matrix and measures of centrality.

In other literature courses, the students have had extensive practice creating character relationship maps for character analysis; the question students use this map to answer is always *what do the relationships between the characters look like?* For this activity we wanted the students to see that by learning about graphs they were able to ask many other questions about the character relationships in the novel. The prompt they were given was: *Write a question about characters relationships in Northanger Abbey that you could explore using a graphing technique.* Responses that were representative of the class:

- Can the character relationships in *Northanger Abbey* be compared to those in *The Mysteries of Udolpho*?
- Do the relationships in *Northanger Abbey* strengthen over time in Bath, or lessen during the time in Northanger Abbey?

It is interesting to note that instead of looking at character relationships as a static

concept the students are now able to think in terms of time and how the characters develop. We can also compare the social structures of different novels once we have a standardized way of representing character relationships. Students were given a research article where graphing had been used to analyze social networks in literary pieces [139] so that they could see the in class activity in relation to its use in research.

For each novel throughout the course of the semester, students worked on building graphs of character relationships and considering which measures of centrality would be most helpful in answering questions they had about the novel. Each novel in the semester gave them progressively more complicated graphs to construct; when they got to *Great Expectations* they were working with 20 different characters and were getting frustrated having to organize the graphs by hand. At this point we introduced them to an open source graph visualization software, Graph Viz [75]. We decided to use Graph Viz instead of some of the more user-friendly programs available because we wanted to give the students the experience of playing with the code. The students had to abstract the data from their graphs to a structure specified for Graph Viz and then write a syntactically correct statement to represent that data. At this point, we observed many of the English majors experimenting on their own with compile-time and code annotation options to explore more advanced settings of the tool. Some examples of the students graphs can be seen in figure 5.2.

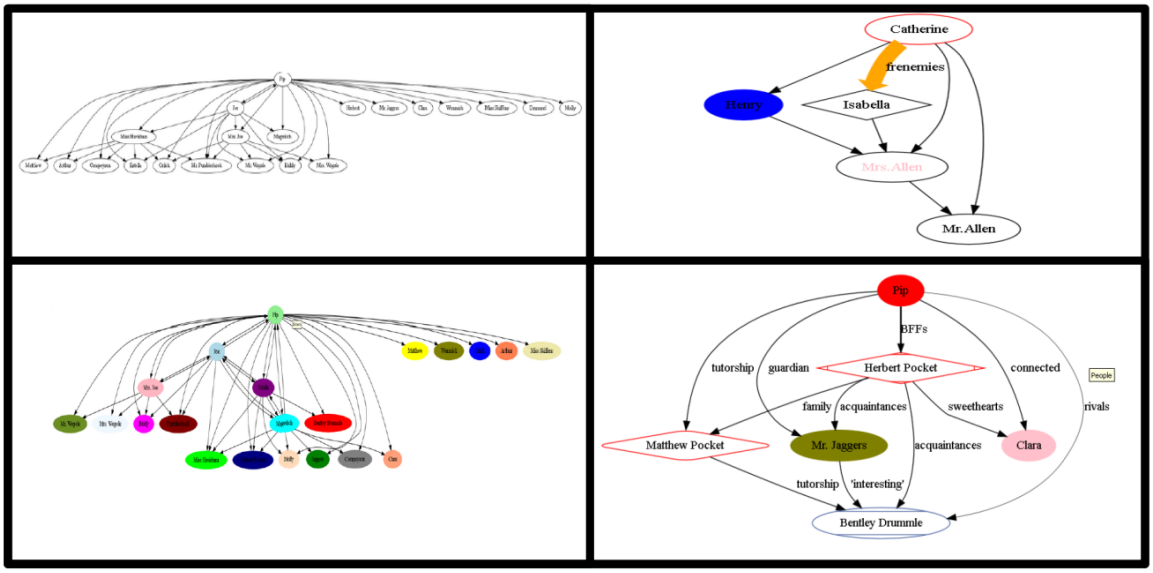


Figure 5.2: Graphs created by students using GraphViz

The social network analysis activity gave the students experience in constructing graphs, analyzing data, using the same data to answer different questions, and using technology to help with data representation.

One of the survey questions asked why they enjoyed their major; a majority of the students responded that they enjoyed English as a major because it allowed them to be creative. We also asked the students whether they considered computer science to be a creative field. As is clear in figure 5.3, many of the students did not think computer science would allow them to be creative. After the activities in this course their response to this question changed, showing a majority of students considering computer science to be a creative discipline.

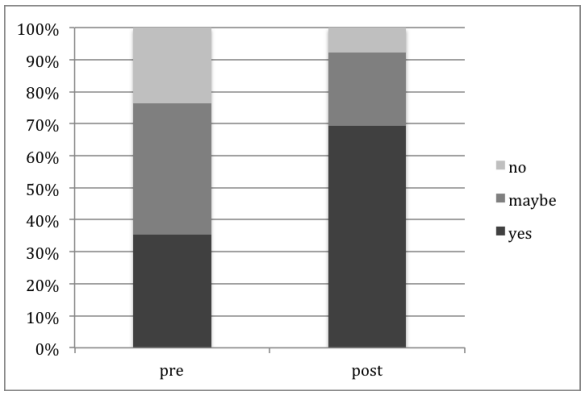


Figure 5.3: Pre- and Post-Survey Results: Does Computer Science Allow for Creativity?

5.8 Discussion

When we began to introduce computational strategies in this course, students' first experiences reinforced already-known and practiced approaches in literary analysis. For example, these upper-level students had mastered close-reading annotations, i.e., analyzing literature at the level of diction, looking for patterns, such as repetition, contrast, and development. When the students first employed on-line word-count tools with the 19th century novella, they noted the visual depiction of patterns in imagery that they had already identified through pen-and-paper annotations. They were able to reflect on some of the limitations of word counts, in that straight word counts

couldn't reveal multiple meanings of words, group words with similar meanings, or take into account words in their 19th-century literary contexts.

When students began work with graphing, they were exposed to a tool that improved upon their existing literary strategies. We frequently ask students to graph character relationships, as a way to analyze social class and connections in Victorian novels. Having a standard way to visualize character analysis data allowed students to quickly and accurately identify two new aspects of character and character relationships: 1) degree centrality, which character was most connected to other characters in the novel; and 2) the betweenness, how characters have access to other characters in the world of the novel. This technique led to some new insights beyond what we had been able to do with simple maps of characters. For example, students and instructor were surprised to find that John Thorpe was the most "central" character of *Northanger Abbey*, a discovery that led to an interesting discussion of influence. Thorpe is not the main character, nor the best-connected, nor the most appealing. However, our graphs revealed that he had a wide sphere of influence, mostly because of his ability to insinuate himself into the lives of other characters at every social class. As students began to compare graphs at different points of novels, and over time, they discovered that the plot of a Victorian novel became more democratic as the story progressed, and that Victorian novels overall became more democratic as the century progressed.

During the second half of the course, we observed students drawing upon these strategies without prompting; they were applying knowledge and skills from previous units to new texts. For example, when students read *The Vampyre*, they were asked to calculate the distances Aubrey travelled with his mysterious companion from European capital to European capital. Drawing upon knowledge of early 19th-century travel, they discovered that Aubrey's journey, while depicted in only a few pages of the novella, would have taken months, perhaps a year. Students reflected on the mileage in the context of Aubrey's character and relationship: such an arduous, prolonged journey might help to explain Aubrey's lack of awareness of his companion's identity. Later, when reading *Northanger Abbey*, several students noted what a trying journey it must have been for Catherine, a teenager, to travel alone from the Tilney's abbey to her own home. When students began reading *Great Expectations*,

they were aware of a difference in Dickens' treatment of cross-country travel. What was highly emphasized and even tracked in early Victorian novels was glossed over by the mid-century: Pip's move from the country to London is barely mentioned.

Finally, some literature students began imagining themselves studying computers and math in the future; these were the same students who began the semester rolling their eyes or expressing fear at anything related to math. One student approached the co-author about her desire to add a math minor to her degree program and another enrolled in a computer science course.

5.9 Conclusions

In this paper we presented our approach to integrating computer science and quantitative literacy concepts in an undergraduate English Literature course. The activities that we incorporated in the course gave students an opportunity to explore computer science applied to literary analysis. We believe that, by engaging with students in a discipline that they were already passionate about, introducing computing concepts applicable to their current interests was integral to the success of this project.

We were able to add computing activities that were of interest to the theme of the course without taking away from any of the learning outcomes present in previous iterations of the course. The original course was kept intact and additional learning outcomes were added that focused on computing as a means to engage students in quantitative skills directly applicable to their interests. Exposure to computing concepts in this course, as applied to literary analysis, gave students a new perspective on and deeper understanding of the nature of the nineteenth century Gothic novel. By focusing on developing activities that relied on the critical frameworks and in fact show the overlap and relationship between those frameworks, we reinforced the idea that computer science can work well to create interdisciplinary learning environments and engage students in new methods for continuing their work of interest. We have seen students' analytical and problem-solving abilities develop through these computational approaches. We have also observed a growing awareness of the importance and usefulness of analyzing literature as data, as well as data in literature.

CHAPTER 6

Multi-Track Programming Competitions with Scratch[1]

6.1 Introduction

Computer programming competitions for high school students are a time-tested method of outreach and engagement that have been widely used by academia and industry.

Universities organize and execute programming competitions for high school students for several reasons:

1. Competitions raise the visibility of a campus and its academic computing offerings among future students

2. Competitions offer an opportunity for universities to directly encourage promising future students

3. Bringing teams to campus offers a rare window of opportunity for networking and professional development for those teams computer science teachers

4. Many college students view helping run such a competition as worthy community service, and/or as preparation for the collegiate-level competition.

Conversely, high school computer science students and their teachers attend programming competitions because:

1. Students who take up an interest in computer science often would like a place to hone their skills and be recognized for their accomplishments

2. Students are eager to see college campuses and meet college students in a likely area of future study

[1]**This chapter is based on the peer reviewed publication [8] and has been minimally edited.**

3. Students often view such competitions as valuable practice for subsequent, high-stakes testing such as the Advanced Placement exam in Computer Science.

In most states in the U.S., more than 85% of high schools have no computer science teacher, and thus no computer science courses. High school programming contests held at college campuses in our region have been attractive only to a very narrow, elite band of high schools. Those schools are overwhelmingly suburban schools in orbit about metropolitan areas. Participants from those schools tend to be mostly white or Asian males, from middle-class backgrounds. Many are planning to major in computer science already.

Beginning in 2014, a National Science Foundation grant project started to roll out professional development for area teachers in schools with no computer science presence. Due to this project, 18 new school districts in our state began teaching ECS [65], and over 600 new high school students were taking computer science for the first time. We quickly realized that our traditional model of high school programming competitions was a poor match for the wave of new students entering computer science with broader introductory courses.

In response, we have adapted our traditionally-structured high school programming contest to make it more attractive to this new population of students. We added a second, parallel track to the traditional Java-based competition that uses the Scratch programming language to welcome students from courses such as ECS, or the new AP CS Principles [47] (AP CSP) course. Broadening the reach of this outreach beyond those schools with existing, well-established programs that culminate in Advanced Placement CS A is essential to engaging a more diverse set of participants. Our novel structure requires problems that can be quickly judged, but allow open-ended solutions rewarding creativity and artistic merit. Most significantly, two of the 11 schools participating this year were coached by brand-new ECS teachers, from districts that previously had no CS teacher or courses.

Although years of experience with traditional contest models have produced robust contest management software[2], a rich pool of existing and reusable contest tasks[3], and deep institutional knowledge about how to successfully engineer and execute an event,

[2]http://www.ecs.csus.edu/pc2/
[3]http://uva.onlinejudge.org/

there has been little published work on designing a Scratch-based competition with short-term, open-ended challenge tasks that must run in parallel with a traditional ICPC-style Java track.

This paper presents the first two years of our experience with the challenge of designing and fielding just such a competition.

6.2 Background

Programming competitions have been used to foster and encourage student interest in computing since at least the early 1970s [10]. In this section, we briefly summarize the large body of prior and related work.

6.2.1 Collegiate Programming Competitions

The Association for Computing Machinery (ACM) International Collegiate Programming Competition (ICPC) is among the oldest and most respected institutions for bringing computing students together in competition from around the world. The typical format has teams of three students working on eight or more complex algorithmic thinking tasks with a five-hour time limit and only one computer.[10]

Regional and local competitions at the collegiate level frequently follow the model of the ICPC, using the vast repository of past contest tasks as both practice and competition problems.

The International Olympiad in Informatics [105] is billed as, the most prestigious international algorithmic programming competition at the high-school level. National bodies, such as the USA Computing Olympiad [173], hold internal competitions to select a team of delegates to participate in the international competition, not unlike the Olympic Games for athletics.

For USACO, the internal rounds consist of online competitions, supplemented by many hours of free, online training.

Separately, the American Computer Science League [5] offers asynchronous, school-based competition with both a written exam component and a solo programming component. The asynchrony allows ACSL to scale to many schools, by allowing individual teachers to offer the competition during the school day at a time that suits

the local context in the high school building.

A wide variety of corporate-sponsored coding contests and web portals compete for the attention of high school students with a strong interest in computer science and related fields. A thorough survey of the landscape of programming competitions is provided in [72].

With many decades of experiences running programming competitions for high school students, more recent research has sought to classify and clarify the categories of competitions and tasks commonly found. Pohl [143] provides a taxonomy of computer science competitions, contests and challenges. Ragonis [149] classifies common questions types found in computer science challenge problems. Hakulinen [90] provides a survey of challenge task development. Foriek [72] gamely suggests new possible problem categories, and discusses an Internet Problem Solving Contest (IPSC) designed to escape some of the more common tropes found in ICPC- and IOI-style competitions.

Despite the longevity and popularity of the traditional model for such events, it is clear that there is room for improvement. Critics point out that time-bound contest problems evaluated solely through automated black box testing discourage many of the most important pedagogical goals of modern computer science [29]. The emphasis on speed in the ICPC-style contest crowds out documentation, refinement and design. The emphasis on platform-independent programming languages and development environments, as well as black-box testing, usually discourages the use of graphics and sound, sharply limiting the creative universe of both problem tasks and their solutions.

Finally, researchers have been commenting on the lack of diversity of participants attracted by the traditional contest model since at least the 1990s [68].

6.2.2 Alternative Competitions

While the ICPC-style competition is likely to persist in some form, many alternative competition styles have been proposed to address its known shortcomings, including its lack of participant diversity [68].

Alternatives such as the Kansas City Computer Science Fair [68] and the College of Charlestons alternative competition [29] emphasize quality-of-process by judging

criteria based on both technical and artistic merit, rather than just black box testing [29]. We reused these ideas in order to encourage creativity and reward artistic merit.

Others have combined CS Unplugged [172] activities with competition-style programming problems [177] to produce an effective hybrid. A wide variety of robotics-based competitions [141] allow participants to branch out beyond text-based input and output, as well as simple black-box testing.

Brebas [52] (Lithuanian for, Beaver), is an international, online competition series with divisions open to grades 5 through 12, and intended for those interested in computing regardless of skill level. Individual tasks are typically short, taking 3 minutes or so to solve. Typical Brebas tasks are not programming tasks, but use a computer to solve interactive and/or multiple-choice prompts. In contrast, our work retains the focus on algorithmic design and programming, using the Scratch [127] programming language.

6.2.3 Scratch Competitions

Competitions that use Scratch or SNAP [171] take advantage of these languages attractive, block-based graphical programming environment to make the programming task more welcoming to novices. Scratch prevents standard syntax errors, and offers a simple set of primitives to manipulate sprites (graphical elements with attached method behaviors) on a stage (background). Emphasis is on animation, sound, and reactive programming features such as mouse clicks and keyboard events.

The annual Harvard SCRatch International Programming Trial (SCRIPT)[4] began in 2012. Students ages 6-15 can participate in groups of 2-4, and submit a Scratch project under one of six categories: greetings, games, music and dance, stories, simulations, and educational software. Both Georgia Tech[5] and University of New Brunswick[6] have also offered a Scratch-based competition in several past years similar to SCRIPT described above.

Like many other alternative competition formats, all three of these alternative Scratch competitions differ from our work by focusing on online, long-term tasks,

[4]http://scratched.gse.harvard.edu/discussions/events/scratch-competition-symposium-script-2014

[5]http://home.cc.gatech.edu/TeaParty/346

[6]http://www.unb.ca/saintjohn/sase/dept/csas/competitions/jr-high/index.html

rather than an on-site track co-located with traditional ICPC-style contestants. However, all three efforts seek to accomplish similar goals to our work:

- Building students sense of belonging in computing
- Encouraging students to code
- Encouraging local teachers to teach Scratch

The use of Scratch in the Syrian Olympiad in Informatics [104] is the clearest antecedent to our current work. Like our contest tasks, the SOI participants were given sprites and a stage, and asked to add behavior. The overall structure of the SOI differs markedly from our typical high school outreach event, and was not designed to bring a growing community of Scratch-using high school students into a parallel competition track.

6.3 Year One Competition

For the Spring 2014 edition of our high school programming competition, we decided to add a new, parallel, Scratch-based track to our existing ICPC-style competition that allowed Java, Python and C++ solutions. We knew that the Scratch Division needed to remain open-ended to promote creativity while completing in the same timespan as the other competition.

At that time, we could find no readily available resources that suggested previously successful task prompts for such a competition. Existing models (see prior section) generally relied on much longer competition times, and also on longer judging times. There were no examples of on-site, open-ended, Scratch-based tasks suitable for grades 9-12, or corresponding judging rubrics.

6.3.1 Task Prompt

The final design envisioned a single, three-hour challenge task centered on a particular, classic fairy-tale motif. Students were asked to retell the story of *The Three Little Pigs*.

The task description included a summary of *The Three Little Pigs* and key requirements such as a minimum of three scenes, repetition of appropriate dialogue,

and a conclusion that emphasizes the moral of the story that hard work shall be rewarded when challenging times come.

Participants were admonished that their work should be original. They could use any of the sprites, backgrounds, audio and other narrative elements that are available through the Scratch.mit.edu site, but should not reuse or adapt existing Scratch projects.

6.3.2 Judging

The existing PC2 contest management software was determined to be unsuitable for running the Scratch Division. Because the entries would only be judged at the end, there was no need for features handling rapid submission, testing and feedback. Moreover, the complex and somewhat intimidating default interface of the software was deemed likely to detract from the overall welcoming message behind the Scratch Division in the first place.

A panel of faculty judges evaluated work submitted using four criteria including originality, creativity, elegance, and sophistication.

The small number of Scratch Division participants in the first year allowed judging to be completed successfully. However, entirely subjective judging went slowly, and three hours per task allowed participants to produce elaborate artifacts that were difficult for judges to fully explore in the time allotted.

6.4 Year Two Competition

In the second year, we made several important changes to the structure of the Scratch Division in order to more easily scale problem judging, while retaining the distinctive character of the competition.

Competitors were still permitted to compete in pairs, or individually. While high school students were the target audience, one coach requested and received permission to bring an advanced middle school team.

The competitors were given a packet of competition questions, including detailed instructions on how to access templates and submit their work. Additionally, the

packets were created to be colorful and playful (compared with the traditional, no-frills ICPC-style specifications) in order to encourage participants to maintain a creative and open mindset. (See Figure 6.1)

Team #: _____ Team ID: _____

Welcome to the 2015 edition of the Marquette University ACM computer programming competition. For today's competition, you and your team will have 3 hours to develop several Scratch solutions. Each question is worth **5 points**, except problem 5 which is worth **10 points**.

Instructions to **navigate to templates**:

- Navigate to the following URL: https://scratch.mit.edu/users/MUScratchDivision/projects/

Instructions on **using templates**:

- Must be signed into Scratch:
  - Click 'See Inside' then click 'Remix'
  - Edit the contents to complete the project

Instructions on **downloading** Scratch files

- Click File->Download To Your Computer and save the file with the name being the problem number and then the team name/number (ex. Problem2Team4.sb2)

Instructions on **uploading** completed problem solutions:

- Navigate to the following URL: http://www.mscs.mu.edu/~cohare/scratchcomp.php
- Type in the provided Team ID (top of this sheet) -> Click 'Choose File' and navigate to the previously downloaded file -> Click 'Open' -> Click 'Upload File'
  - If you receive any errors please contact a staff member

Instructions on **viewing feedback** on submitted solutions:

- Navigate to the following URL: http://www.mscs.mu.edu/~cohare/scratchcomp.php
- Type in the provided Team ID (top of this sheet) -> Click 'Get Feedback' and you will be brought to a .txt
  - Routinely check feedback after solution submission

Figure 6.1: Scratch Division Instruction Page

A mix of both closed- and open-ended questions were used, meaning quicker judging for the technical points during the competition, allowing more time for the judging of creative merit.

Under the question type taxonomy suggested in [149], this task set included:

- Type 1: Developing a Solution (tasks 1, 2, 3, 4)
- Type 2: Developing a Solution that Uses a Given Module (tasks 1, 2)
- Type 7: Completing a Given Solution (problem 1)

6.4.1 Task 1 - The Maze

For the first task, the designers sought to present the students with a straightforward, recognizable, and easy to implement question. Given a maze backdrop and a sprite, along with incomplete code blocks, the participants were expected to program the sprite to maneuver through the maze. They were expected to provide the test loop to determine when the sprite should stop moving (e.g. hitting a wall, crossing the finish line).

The designers intended for this problem to be solvable by all teams. Although not all teams received full marks, each team submitted a functioning copy of the sprite and maze. Not all of the teams were able to end the game successfully.

6.4.2 Task 2 - The Fly Swatter

The second question required students to make a fly swatter game. Participants were given a fly sprite, swatter sprite, and a kitchen background. The fly was pre-programmed to randomly move around the kitchen, and the students were to map the swatter sprite to the users mouse, "squashing" the fly sprite when clicked. The questions main objective was to see if students were able to make the game work well, which required the bug to wait a certain amount of time before moving again. If this was not present in the participants submission, it would be difficult to win at all.

All eight of the participating groups produced a submission for this problem, with only 6 receiving full points.

Despite the relatively structured nature of this task, participants still incorporated creativity into their solutions. Some teams replaced sprites, while others created elaborate win animations, such as in Figure 6.2, which incorporated a winged hippopotamus instead of the simple fly, and a popular Internet meme from that month[7].

[7] http://knowyourmeme.com/memes/he-man-sings

Figure 6.2: Flying Hippo Swatter and He-Man Sings

6.4.3 Task 3 - Rock-Paper-Scissors

The third task asked the participants to develop a working rock-paper-scissors game using the given sprites and their own animations. They were also required to find their own background to complement their game and animation, as well as promote the creativity of the question.

Creative solutions to the animation subtask included spinning the selection, keyboard movement for selection, and moving towards the opposing sprite.

6.4.4 Task 4 - Pong

The fourth task in the competition asked the participants to create a working two-dimensional one-player pong game (Figure 6.3) and for extra credit they could implement a custom artificial intelligence algorithm as a second player. This was intended to be the most challenging task in the Scratch Division.

As expected, this slightly more difficult problem did not elicit as much creative flourish as the earlier, easier problems.

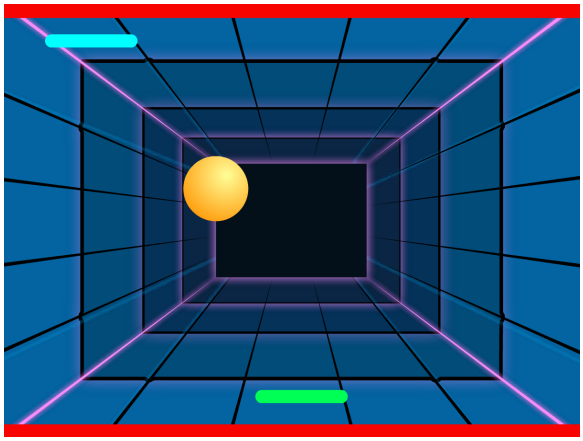


Figure 6.3: Task 4 - Pong

6.4.5 Task 5 - Sunken Ship Narrative

The fifth task was designed to be the most open-ended, allowing participants to demonstrate both their Scratch-fluency and creativity (Figure 6.4). Similar to our first year competitions sole task, the fifth task provided a story prompt and directed participants to finish the story. The plot included a tale of a sunken ship; however, it was up to the participants to provide their own developments to the story, conclude it with a somewhat logical ending, and include the required elements (e.g., underwater animation, loops, at least two sound clips). The participants were also asked to include as many additional objects as they saw fit. We made sure to emphasize that creativity was the ultimate goal for this question, and informed them that it was a very important qualifier for grading.

All eight groups submitted answers to this problem, with greatly varying levels of whimsy. One group made an incredibly creative story, but had difficulty implementing every requirement of the prompt, such as the code requirements.

In an echo of results from the previous year, the most creative code made the requirements difficult for judges to identify. On the far end of the spectrum, some groups made sure to hit every program requirement for the task, but did not invest

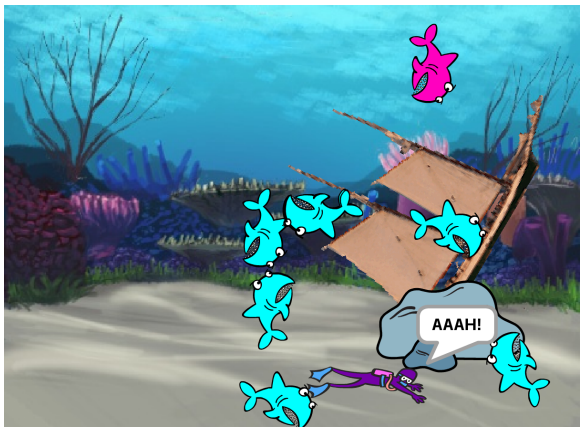


Figure 6.4: Task 5 given water, rock, and ship sprites

time into adding creative touches to their submission. In the happy middle, some teams met not only the basic requirements, but retained a logical program structure.

One group developed a game in which user could maneuver a character to collect all the coins within a sunken ship while avoiding the sharks on the screen. This group even included a winning and losing screen. (See Figure 6.5)

Although this question was the most difficult to grade, it provided the largest variety of solution types. This not only allowed each individual group to set themselves apart and present their personalities, but allowed us to take a look into how these groups performed in the wild. We were thus able to observe how the participants built their own blocks of code, while allowing them to personalize it and have fun along the way.

Anecdotal feedback from the teams confirmed that this question was both the judges and the participants favorite.

6.4.6 Submission and Judging

We deployed a simple, PHP-driven web submission system for the Scratch Division. This was deemed to be a simpler, lighter-weight and more theme-appropriate solution than the standard PC2 contest management software. (PC2 was still used for the Java Division.)

The judging rubric included a five point system the participants needed to fulfill in order to get full points. While grading submissions, the judges filled out corresponding

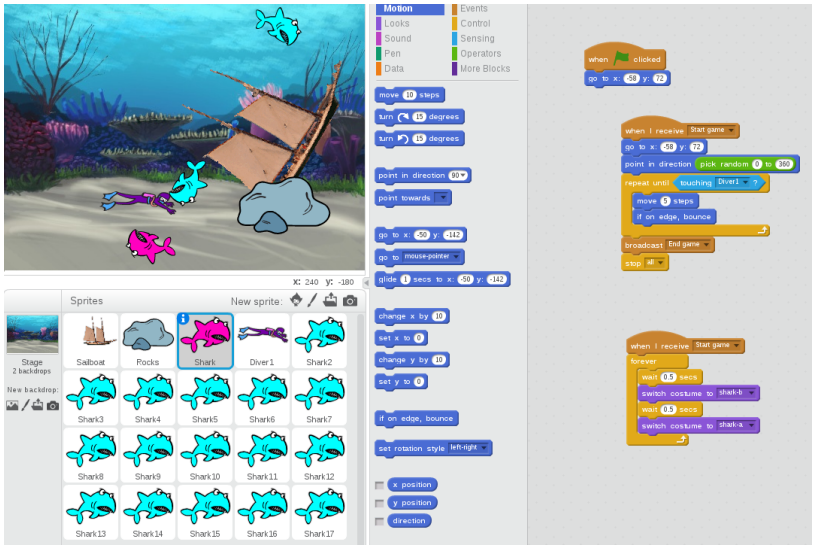


Figure 6.5: Task 5 Shark Sprite skin choices and Code

feedback.txt files that marked whether their submission was correct or incorrect. If incorrect, then feedback was given in order for the team to attempt corrections. The teams could view their feedback using the same aforementioned web system with unique key.

Tasks one to four were worth five points each with an extra three points of extra credit possible for task four. All partial credit was awarded three points as long as the submission had at least three of the problems criteria working for their submission. Open-ended task five was worth ten points; seven points were awarded if they reached the minimum requirements.

6.5 Results and Discussion

In total, 36 teams participated in our year two high school programming competition 28 teams in the traditional, ICPC-style Java Division, and eight in the Scratch Division. More than 120 students from 11 high schools across our state competed from 9am-Noon, taking up all five of the computer science departments instructional laboratories, as well as many other spaces to host registration, the coaches meeting, judging, contest headquarters, etc.

Most significantly, two of the 11 participating schools were coached by brand-new ECS teachers. Those schools had previously had no CS teacher or courses, and had

sent no previous teams to our events, but this year were able to join in on the Scratch Division.

Thanks to generous support from industrial sponsors and alumni, we were able to award prizes to the top 25% of teams in both the Java and Scratch Divisions, ranging from flatpanels to Arduino starter kits, to wireless peripherals.

Feedback from coaches and participants were overwhelmingly positive. In evaluating the success of the Scratch Division, one of our most experienced ECS teachers noted that most of the tasks resembled the in-class assessments she gives her students at the end of units. We take this as confirmation that we are on the right track. The number of ECS and CSP teachers in our area is growing sharply, due in part to outreach and professional development efforts led by our institution; a fall survey of teachers in our network suggests that we can expect both divisions to grow by another 25-50% in spring of 2016.

In practice, the design of the second year competition scaled well to triple the number of Scratch Division students from the previous year.

What we did not expect was the rather creative turn the participants took with the objective-based questions. For instance, the fly swatter problem (task 2) was rather straight- forward; however, a team redesigned it to track how many times the fly was clicked and when it reached one-hundred-and-eleven whacks a spinning animation of a picture from the internet was displayed along with sound. The originality made it more difficult to grade some submissions, but it is the opinion of the authors that this was a worthy trade-off, and one that we look forward to making again next year.

Finally, we were surprised to learn on competition day that the contestants in the Scratch Division varied more widely in age than expected. Despite ECS being intended as a 9th-grade course, our participants from ECS schools ranged from middle school students to high school seniors.

The total number of Scratch Division participants was still less than 20 this year, so it is not yet feasible to make statistically valid claims about the demographic shift we are seeing in the population participating in the Scratch Division. Anecdotally, the Scratch Division has had a better gender balance than the Java Division in both years.

6.5.1 Future Work

The originality and creativity that some students implemented indicates a need for more open ended questions. For several prompts, we outlined many of the steps and provided a template of sprites and some code fragments that contestants should use. This may have restricted some of the creativity that students could display. In the future, it may serve us better to give a general paragraph of what we require, and expect groups to take different routes to completing the prompt.

Moving forward, we expect to grow the competition next year, as more schools are affected by efforts to increase the number of new CS teachers in the state. We are considering creating a third division, based on previous work on CS Fairs, that would directly model one of the non-programming performance tasks expected for AP CSP.

As the number of participants grows, we hope to complete a statistically valid analysis of the differences between the populations in the two tracks. Our hypothesis is that the Scratch Division attracts a higher percentage of women than the current Java Division, and draws teams from schools with a larger population of underrepresented students.

6.6 Conclusion

We have presented the structure and rationale for an alternative, Scratch-based track that we have added to our existing ICPC-style high school programming competition. The design of this track incorporates both relatively constrained tasks that can be quickly judged, and more open-ended tasks that can be rewarded for creativity and artistic merit. Our overriding goal was to include in our annual outreach event the growing number of Scratch programmers coming out of broader introductory CS courses such as ECS and AP CSP. Students participate in the same, on-campus event, with a different set of tasks, proctors and judges, and participate in the same awards ceremony.

The competition ran well, and attracted markedly more participants than the previous years. The presence of a Scratch Division allowed new schools to participate who had previously never sent a team to such an event. Furthermore, while numbers are still small, it seems likely that the demographic balance of the Scratch Division

will at least initially beat the traditional Java Division competition.

6.6.1 Acknowledgments

The authors would like to thank the high school teachers of the CSTA Wisconsin-Dairyland chapter for their guidance and support in creating our event, and for bringing their teams of students. We are grateful to the Marquette University student chapters of ACM and Upsilon Pi Epsilon for the logistical and volunteer support that made our events possible. Aspects of the competition were supported by generous donations from Marquette computer science alumni as well as local employers, such as Rockwell Automation and Direct Supply.

Some of the authors were supported in part by NSF CE21 grant #CNS-1339392, the results of which are driving the need for high school programming contests that appeal to a broader audience in our region.

CHAPTER 7

The Impact of Exploring Computer Science in Wisconsin [1]

7.1 Introduction

Much of the literature discussing computer science (CS) education points to disparity in economic and demographic situations as a root cause for the lack of access to computer science for nontraditional students [2, 38, 46, 62, 77, 78, 86, 108, 132, 153, 191]. This research implies (and sometimes explicitly states) that if students were given the opportunity to learn CS, the gap in gender and minority participation would be lessened. Since economic and demographic factors are inherently spatial in nature, this paper will focus on using elements from the geography of educational opportunity framework.

The Geography of Opportunity framework is often referred to in research focused on housing and residential mobility. It is meant to refer to the ways that geography can influence an individuals opportunity, i.e., that an individuals options are limited by the social and economic conditions surrounding them [158]. The idea behind this framework can be succinctly summarized with a quote that Squires and Kubrin attribute to former Albuquerque mayor David Rusk, Bad neighborhoods defeat good programs [164].

This concept can be extended to the geography of educational opportunity because there are constraints placed on educational opportunity by the educational infrastructure of the community [96]. In Hillmans research on the geography of opportunity as it applies to college choice, he explains how geography can affect educational opportunity by drawing a comparison to the concepts of food deserts. Hillman uses the geography of opportunity framework to explore the importance of place and how geography shapes educational equity and opportunity [97]. Additionally, the geography

[1] **This chapter is based on the peer reviewed publication [27] and has been minimally edited.**

of opportunity is examined in [119] as it relates to outcomes of No Child Left Behind testing requirements and segregation in schools. Tate, et al. [167] reminds that certain interventions in STEM education have been geospatially-minded in the past, especially in urban contexts; Green [80] notes that Access to opportunities in the United States (U.S.) is inequitable across geographic spaces. Finally, Soja [163] advocates for using a spatial perspective to help build an understanding of the inequalities in communities so that action can be taken; spatial thinking through a geographical perspective can help to facilitate change. Where a student attends school has a direct impact on their opportunity to access CS education. Their geographic place can constrain or enable them more than other factors such as an understanding of what a computer scientist does, or seeing people in CS jobs that represent their gender, race, or ethnicity. Use of this framework can add the context of where to a discussion that is largely focused on the when and who. Instead of asking how much did CS enrollment increase from one year to the next, the question becomes where did CS enrollment increase. This shift moves the analysis to a different level of granularity. When not concentrating on merely the net gain or loss over an entire state, the focus can be on what the changes were for each school.

The primary questions in this study are:

1. Does publicly available data give enough information to track CS course enrollment over an entire state without the need for costly and time-consuming surveys?

2. Is it reasonable to represent high school CS course availability at the state level or is a more granular representation needed?

3. Has the introduction of the ECS program had a disparate impact on any economic groups in Wisconsin?

7.2 Geography in CS Education Research

Given the difficulty in collecting detailed data over a large area for hundreds of schools, very few prior studies have even a small focus on the underlying geography of their study area. The only recent CS-centric study explicitly mentioning geography is a report from South Carolina showing a statewide lack of geographical diversity for where

CS coursework is offered [38]. Based on survey responses from 158 K-12 educators, they concluded that Title 1 schools (where $> 40\%$ of the students qualify for free or reduced lunch) are less likely to offer computing coursework.

A second research study [83] indirectly focused on geography by using data from the Advanced Placement (AP) CS A exam in a regression analysis to explore the demographics of test takers across the U.S. In each state the relationships were explored between wealth and exam-taking, and the number of exam-takers from underrepresented groups, with the goal of explaining variances between states.

Both studies consider how economic status could relate to a students opportunity to take computing courses in K-12. While one uses statewide survey data for indication of CS availability with school level economic information, the other uses national AP data and state level economic factors. The current study differs from previous work in the following ways:

(1) While still interested in where CS coursework is offered in the state as in [38], this study uses public data collected from the state Department of Public Instruction (DPI) instead of attempting a statewide survey. Schools are required to self-report on many dimensions annually, and course level enrollment data is one of these dimensions. While there are always concerns for the validity of self-reported data, this source provides complete data for all the schools in our study area without putting any additional workload on the schools. In addition to data being more complete and reliable than what a large-scale survey would provide, this data is publicly available in many states, making it quite attractive for this type of analysis.

(2)The study in [83] investigates the relationship between wealth and the number of students taking the AP CS A exam for U.S. states. This paper follows the lead of [83] in examining the role that wealth can play in computing education; however, using a framework of geography of educational opportunity means considering the role of geographical place within the study, this leads the authors to disagree with [83] on three points: (a) **Analysis method** - A regression analysis assumes that the data being analyzed is random. This paper considers how CS course enrollment and availability is not randomly distributed within a state and therefore violates that assumption. (b) **Level of aggregation** - Aggregation to the state level for median income as the wealth variable and exam takers assumes that the nonaggregate data is

distributed within the state homogeneously. This paper argues that this type of data exhibits heterogeneity and therefore should be studied at a finer level of granularity. (c) **Choice of explanatory variable** - Using median income as a measure of wealth in [83] (even if the study had been at a more reasonable level of aggregation) assumes that the school inherits its wealth attribute from the surrounding community. This paper shows that while it is not a perfect proxy as an indication of wealth, an individual schools reported measure of economic disadvantage can more accurately describe the economic circumstances of the students in attendance. Since the unit of study is the school, this measure is within the proper context and not merely a convenient choice. Moreover, in a context such as Wisconsin, in which widespread school voucher programs allow many students to attend a school in a dissimilar economic area, median income for the surrounding community is less likely to align with the economic composition of a given schools student body.

7.3 The Role of Place

When considering why geographical place would have a role in the analysis of CS education in public high schools, one should keep in mind that the institution of interest in this work is the school. In the U.S., many aspects of educational policy are determined not at the national level, but at the state or school level. Schools play a major, central role in everyday social geographies in general. Collins and Coleman bring attention to the fact that they are one of the few institutions that can be found in almost every urban and suburban neighborhood, and with which almost every individual has meaningful, sustained contact at one or more points in their lives. [45] This paper focuses on Wisconsin public schools with a high grade of 12. The study is limited to regular schools, and does not include data related to charter, virtual, or private schools. Also discluded are data related to informal CS education, such as after school clubs or summer camps. The choice to limit the data in this way follows [1], which states that the best chance to broaden participation in computing is through formal education pathways; going through the formal education pathway is the only route that can ensure we are providing all students access to CS education.

This study uses student enrollment counts for each school at the individual course level. The number of students in a school who were enrolled in a CS course can be

extracted using a combination of NCES (National Center for Education Statistics) course codes, local course codes, and course descriptions. Economic data used in the study are aggregated at the school level. This initial study uses the school reported measure of economic disadvantage as a percentage of total students enrolled who are categorized as being economically disadvantaged.

All of the data that are used in this study are publicly available. We collected 6 years worth of data from the Wisconsin DPI related to course enrollments, school enrollments and demographics, educator license status and employment, as well as shapefiles for school attendance boundary zones. Within the course enrollment data, we compared course NCES codes to the local course codes and the local course descriptions to identify courses that could count as CS for mathematics graduation credit, defined by Wisconsin Act 63 [185]. There are 433 schools, making up 378 districts in this data. The six years collected from each school included academic years 2010-11 through 2015-16.

Wisconsin DPI defines economic disadvantage for students who are members of households that are eligible for free or reduced-price meals under the National School Lunch Program (NSLP). To be eligible, a familys income must be less than or equal to 185% of Federal Poverty Guidelines. Students must be identified by Direct Certification every year, and this information is reported by every school even if the school does not participate in the NSLP [186].

When considering American political geography in relation to K-12 CS education, it is important to note that autonomous governmental units, such as local school districts within metropolitan areas, are a defining feature [106, 176]. In fact, this type of geopolitical fragmentation is common in major U.S. metropolitan areas, especially in older industrialized regions of the Northeast and Midwest [99]. Information should not be aggregated at the state level to gauge the health or impact of CS education as in [83], because doing so assumes a fixed effect over the study area, potentially causing false correlation or confounded analysis. How these programs are implemented is greatly influenced by the local context of the individual units; [99] and [67] have found that even the level of fragmentation (number of school districts) within an area can influence efforts to implement reforms. The degree of challenge involved in implementing a reform is increased in areas where there are a greater number of units.

At the national level, entities like Code.org provide information about places where progress is being made in CS education. They show where CS can be counted towards high school graduation requirements for math or science. From this we can see that not all states in the U.S. have fallen in line with the recent CS For All initiatives from the Obama administration, and not all states allow CS courses to count towards high school graduation requirements.

Code.org also shows information at the state level, where they post data about employment in CS, numbers of CS graduates and what they call their Policy Environment (rubric), which point out whether a state has dedicated funding for professional development, if high schools in the state are required to offer CS courses, and the status of state K-12 CS standards.

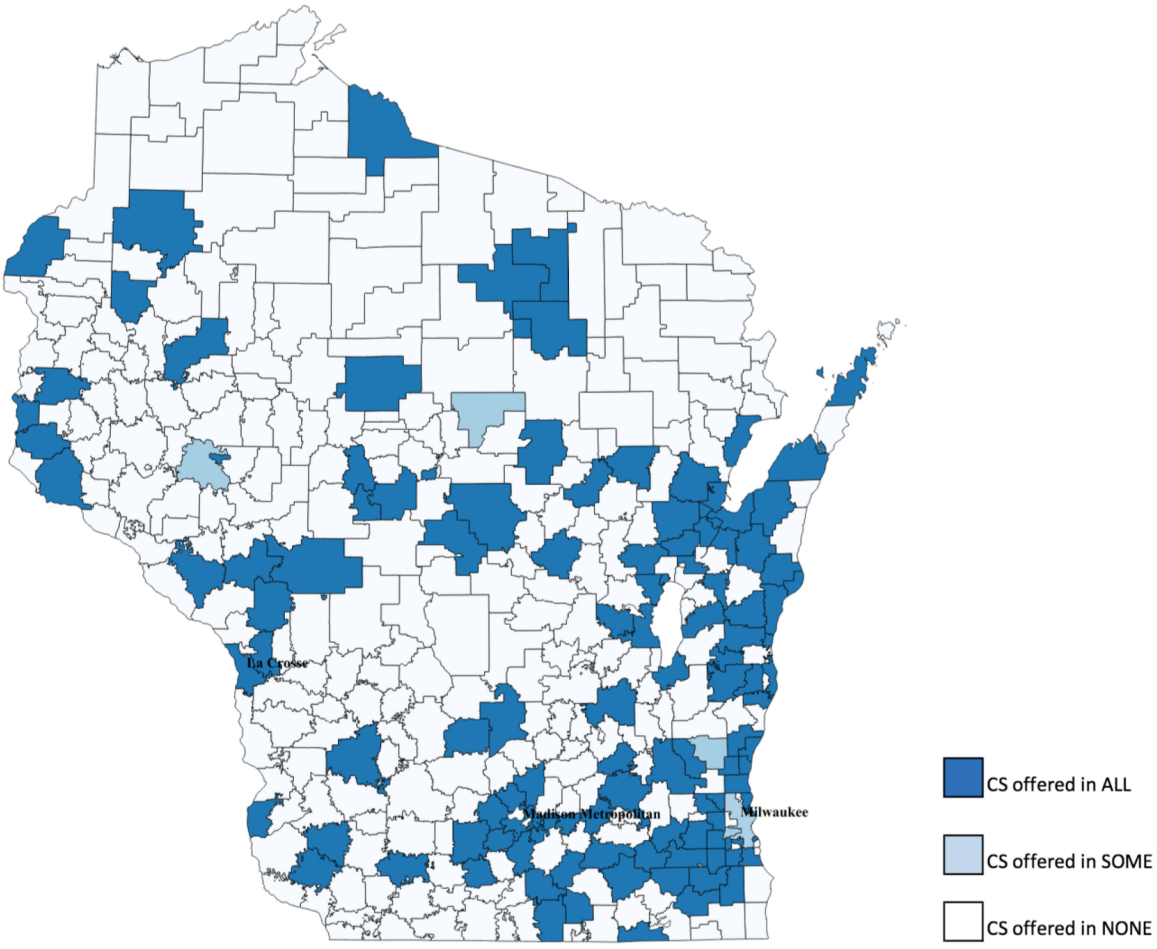


Figure 7.1: Districts where CS courses are offered in WI, '14-15

Figure 7.1 shows that considering where CS courses are being offered at the school district level gives a better understanding of where our state suffers from a lack of opportunity. We can then drill down even further to look at the attendance boundary zones for individual schools in a district, as in Figure 7.2, to see that even if the district level data shows that CS is present, the school level data can identify gaps in access to student groups.

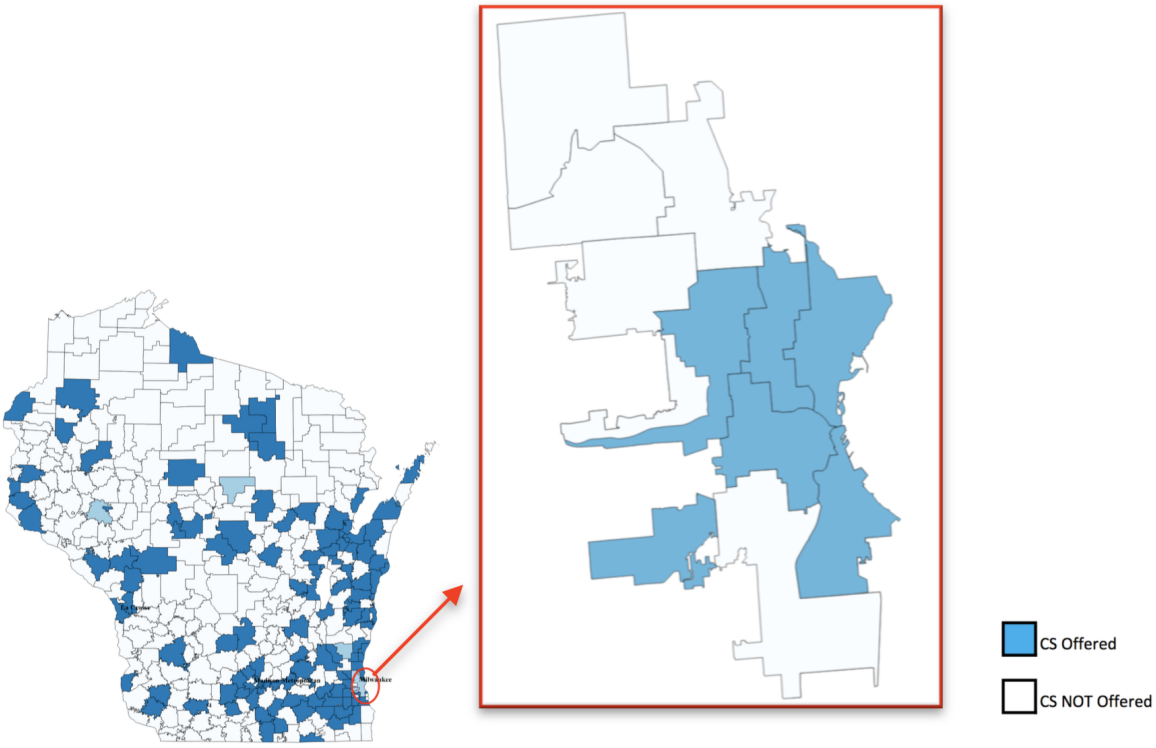


Figure 7.2: High school attendance boundaries where CS courses are offered in Milwaukee Public Schools, 2014-15

While it would be ideal to disaggregate this data all together and compile a dataset of family income for each individual student in addition to their course enrollment information, this is not a realistic goal for a statewide study like this one. Therefore, we have decided to aggregate our data at the school level, which is the most granular level reasonably available. Within the context of our study, school level aggregation is consistent within the framework, and provides minimal information loss.

7.4 Methodology and Analysis

This section describes the mapping of geographic opportunity in Wisconsin, the method used to identify randomness in our data, and the resultant need for finer-grained analysis than state aggregate.

Hogrebe and Tate discuss how building a visual representation in line with Soja relates to the geography of opportunity [98]. They focus on cognitive science literature stating that graphic representations support learning, visual modeling supports retention, and visual representation can help readers more easily grasp complex arrangements. They also note that maps have informed civic debates about regional development issues [98]. Maps bring a visual dimension to data through geospatial perspective and facilitate discussion among stakeholders. By locating data in the context of place, issues become more familiar and understandable to those who may not have experience in data analysis.[98] We can associate variables that are not spatial with a location so that they can be placed in the context of where they occur. In a discussion about why we should show data in map form, [49] refer to John Pickles [142] and how he rethinks mapping, stating that maps are active and that they actively construct knowledge, They exercise power and they can be a powerful means of promoting social change.[49]

The visual representation of where CS courses are offered in the state of Wisconsin, such as Figures 5.3 and 7.1, can help to identify areas of the state where intervention efforts are most needed. It also gives the opportunity to make the case against using a state level aggregation of measures for CS enrollment (or test takers as in [83]). It can be tested whether the data is randomly distributed in relation to space; since with a random distribution there is no value in knowing the spatial locations of the units, it will not provide leverage in predicting opportunity. We have used a join count analysis [53] as a diagnostic tool to test for a random distribution. If we assume the data are binary, presence or absence of CS in the school, we might observe a random distribution of the categories, a clustered distribution of the categories, or an evenly dispersed distribution of the categories. We may wonder what the chances are that a particular pattern of distribution in our data could occur at random. Of our 378 school districts, we can build a spatial weights matrix to describe the neighbor relationships

between the schools by defining neighbor with first-order queen contiguity as a base case. This means that we examine each of the polygons defining a schools geographic attendance boundary and identify where a polygon shares an edge with any other polygon, resulting in a contiguity matrix W . In the contiguity matrix W , each school polygon at i will be considered against another school polygon at j; each ij space will be assigned a 1 if the polygons share an edge, and a 0 otherwise. The diagonal of the matrix is 0 since a school cannot be defined as its own neighbor, and W is symmetric as we are defining the neighbor relationship to be bidirectional. Other potential definitions for W will be considered at a later time in order to build a reasonable and theoretically based weights matrix.

Our queen contiguity matrix shows 2112 distinct neighbor relationships, with each school having a number of neighbors from 1 to 16. We can count the number of relationships that fall into each of three categories:

1. Neither school has CS (NN)

2. Both schools have CS (YY)

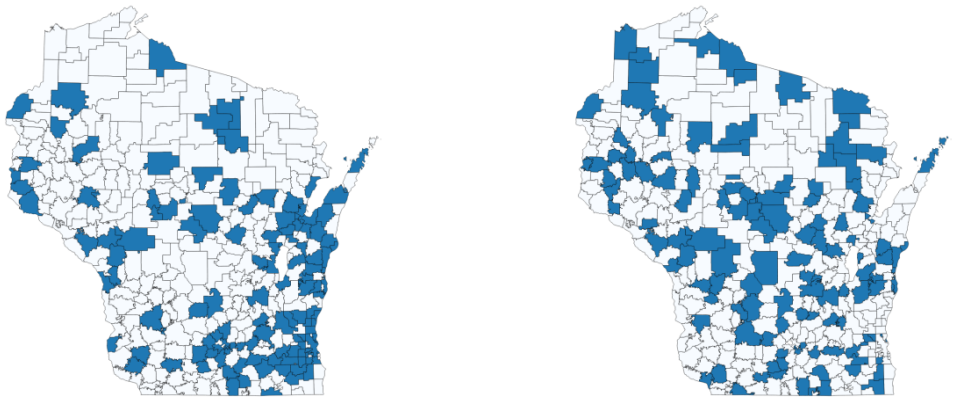
3. One of the schools has CS and the other does not (YN)

We then ran a random permutation on the attribute matrix 1000 times to produce a distribution of random placements of the dependent variable so that we could compare the true counts to the expected counts under a random distribution.

The join counts analysis results for the 2014-15 school year are shown in Figure 7.3, and clearly indicate that the distribution of course offerings is not random. Since this is the case, a regression to show the relationship between wealth and CS course enrollment is not appropriate for our data. It is important to note that patterns can be perceived even when they are not present. The benefit of using a visual representation along with a test for randomness is that we can be assured that our perception of clustering in the offering of CS courses is not merely apophenia.

7.5 ECS in Wisconsin

Wisconsin was the site of one of several NSF-funded initiatives that aimed to prepare annual cohorts of high school teachers for teaching the Exploring Computer Science



(a) Actual districts offering CS courses 2014-15

(b) Random Permutation of districts offering CS courses 2014-15

	True counts	Expected counts	Standard Deviation	z-score	p-value
NN	1012	952.102	25.96081	2.307247	0.021*
YN	772	933.426	30.78347	-5.24392	1.57E-07**
YY	328	226.472	17.63633	5.756752	8.57E-09**

Figure 7.3: Results of Join Count analysis of CS course offerings by high school district in Wisconsin (*-indicates significance at the .05 level, **-indicates significance at the .01 level)

(ECS) curriculum [65] in the years 2014-2017. While the program collected its own data from participating students and their teachers, to date there has been no attempt to assess the statewide impact of large scale deployment of ECS. The first cohort of ECS teachers were deployed to Wisconsin classrooms in the 2014-15 school year. In keeping with our interest in the findings of [83], where wealth is shown to have an indirect impact on the number of students taking the AP CS A exam, we looked to see if there was a relationship between school level economic disadvantage and CS course offerings. Instead of a linear relationship, we found that the implementation of ECS in a school disproportionately impacted schools with a higher percentage of economically disadvantaged students.

Table 7.1 shows enrollment percentage in CS and ECS courses for the 4 years before the first year of ECS in Wisconsin, and how those percentages changed with the first 2 years of implementation of ECS, broken into 4 categories of economic disadvantage. It can clearly be seen that the schools with a higher population of economically disadvantaged students have had an increase in enrollment that is significantly higher than the other economic groups.

Of particular interest is the difference between the schools in the upper and lower

ranges of economic disadvantage. In general, we see a drop in CS enrollment, particularly for ECS, between the 2014-15 school year and the 2015-16 school year for the 76-100% economically disadvantaged group. We also note that in the 0-25% group there is little variation in CS enrollment from one year to the next, and that ECS has had little impact on enrollment. This is where our focus on place will allow us to more clearly identify the mechanics of these general observations.

We observe that in the 76-100% group, at many schools prior to the implementation of ECS, CS courses were not available. Therefore, when ECS began to be offered, students were enrolled in the course at the 9th-12th grade levels. After the initial year, overall enrollment for the group would drop, because only the incoming 9th graders would not have had the previous opportunity to enroll.

As an example, North Division High School (76-100% economic disadvantage), a Milwaukee Public School, enrolled 84 students in ECS in 2014-15 with 27 of those being 12th graders, 27 11th, 19 10th and 11 9th. In fact, 84 students were their total enrollment in any CS courses, since they did not offer any course that was not ECS, and prior to 2014-15 did not offer any CS courses at all. In 2015-16 their enrollment in CS overall dropped to 52. However, the 20 students enrolled in ECS were from 9th and 10th grade, and the other 32 10th-12th grade students were enrolled in an Introduction to Programming course. We would expect that the 2016-17 data will show this in the 51-75% range since the 2015-16 ECS enrollment numbers are spread across the grades like the 76-100% group in 2014-15. Additionally, Whitefish Bay High School (0-25% economic disadvantage), had offered CS courses prior to the Wisconsin ECS initiative and, despite being one of the cohort schools, did not show enrollment in ECS courses. However, they did show consistent enrollment among all grades for 3 years in the CS Principles course.

7.6 Conclusion

The publicly available data that was used in this study provides ample information to track CS course enrollment in the state of Wisconsin. We are able to observe how enrollment changes within economic groups as well as within individual schools, and can make inferences about how these observations could be used to guide interventions. However, additional evidence collected through targeted interviews with individual

educators would enable us to consider more fully the role that teachers play in those changes. The next step in our work will consider the status of the teachers along with enrollment data, since our initial analysis reveals that there are more districts where educators holding a license for CS are employed at the high school or district-wide level than there are districts with schools actually offering CS courses (see Figure 7.4).

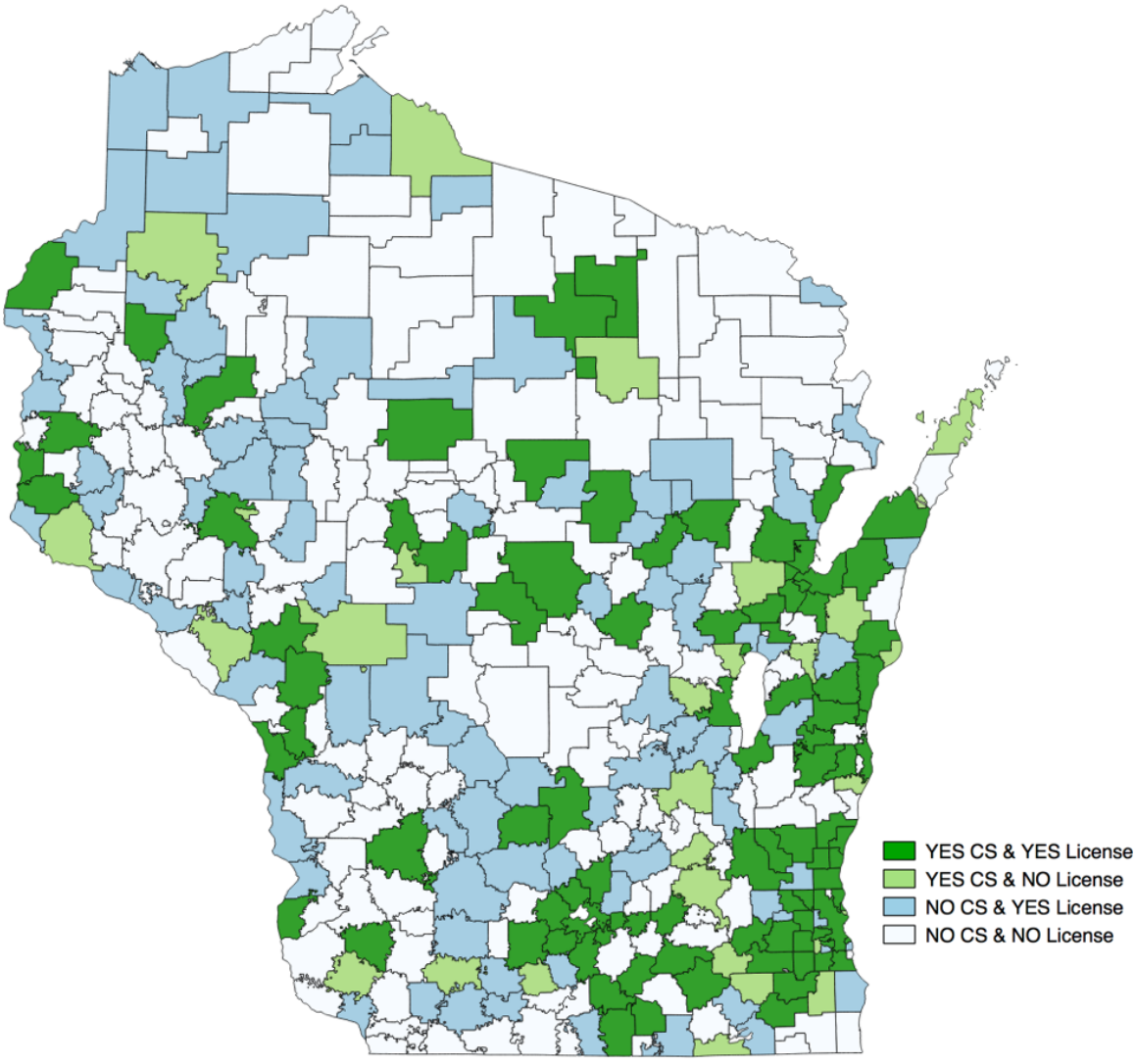


Figure 7.4: CS courses offered vs. CS licensed teachers, 2014-15

Since much of the narrative surrounding the subject of how to increase presence of CS courses in K-12 education points to a lack of qualified teachers, we feel that it is

worth investigating the ways in which Wisconsin seems to break from that narrative.

We can see that analysis at the individual school level gives us a much clearer view of what is going on in a state overall, and that we cannot aggregate our data to the state level. In this situation, the sum of the parts does not necessarily add up to the whole. Information like this can lead to more thoughtful policy and planning for widespread K-12 CS initiatives. We might suggest that a school interested in implementing ECS with a limited budget would need more funding available in the initial year, when more of the student population will have access for the first time to any CS course. In subsequent years, less funding would be required to sustain the course only for incoming 9th graders. Based on this analysis, we would most likely target groups with lower economic disadvantage with an AP CS initiative, rather than an ECS initiative.

The introduction of ECS in Wisconsin has had much more of an impact on schools with a higher level of economic disadvantage. We have seen that schools that are more economically disadvantaged were less likely to have offered any CS courses before the initiative, and that adding ECS led to a wider variety of CS courses for those schools after the initial intervention year. Schools which were less disadvantaged were more likely to have already been offering CS courses, and the addition of ECS did not significantly change their enrollment percentages.

7.6.1 Acknowledgments

This work was supported in part by US NSF grant CNS-1339392.

% Economic Disadvantage		School Year					
		2010-11	2011-12	2012-13	2013-14	2014-15	2015-16
0-25%	Schools	135	125	118	117	113	115
	Total Enrolled	96,785	92,150	84,048	79,681	79,650	83,501
	CS Enrolled (pct)	2045 (2.113%)	2150 (2.333%)	2255 (2.683%)	2427 (3.046%)	3213 (4.034%)	3341 (4.001%)
	ECS Enrolled (pct)	0 (0)	0 (0)	0 (0)	57 (0.072%)	172 (0.216%)	246 (0.295%)
	ECS pct of CS	0	0	0	2.349%	5.353%	7.363%
26-50%	Schools	225	234	236	239	245	242
	Total Enrolled	128,260	125,971	130,302	131,865	130,245	127,852
	CS Enrolled (pct)	1811 (1.412%)	1931 (1.533%)	1993 (1.530%)	2498 (1.894%)	2691 (2.066%)	3142 (2.458%)
	ECS Enrolled (pct)	0 (0)	0 (0)	0 (0)	8 (0.0006%)	88 (0.068%)	192 (0.15%)
	ECS pct of CS	0	0	0	0.320%	3.270%	6.111%
51-75%	Schools	65	62	66	66	59	65
	Total Enrolled	34,404	31,836	32,192	33,003	32,164	30,955
	CS Enrolled (pct)	477 (1.386%)	311 (0.977%)	316 (0.982%)	436 (1.321%)	277 (0.861%)	400 (1.292%)
	ECS Enrolled (pct)	0 (0)	0 (0)	0 (0)	0 (0%)	0 (0%)	208 (0.672%)
	ECS pct of CS	0	0	0	0	0	52%
76-100%	Schools	10	14	15	13	18	13
	Total Enrolled	7,355	12,129	11,608	10,507	12,691	10,177
	CS Enrolled (pct)	57 (0.775%)	140 (1.154%)	54 (0.465%)	49 (0.466%)	408 (3.215%)	244 (2.398%)
	ECS Enrolled (pct)	0 (0)	0 (0)	0 (0)	0 (0%)	255 (2.009%)	45 (0.442%)
	ECS pct of CS	0	0	0	0	62.5%	18.443%

Table 7.1: Enrollment % in CS and ECS courses aggregated by school % of economic disadvantage, 2010-2016

CHAPTER 8

Discussion

8.1 Propagation

One of the aims of this research is to promote the concept of iterative development in propagation planning. We will discuss what that means as it relates to current propagation frameworks for large scale initiatives in K-12 Computer Science Education and suggest a new model for propagation that includes planned innovation.

Both the SCRIPT and the ECEP models for adoption and scaling of K-12 Computer Science Education initiatives successfully impact their intended populations. They focus on 1. understanding the current state of the problem, 2. developing goals to positively impact that state, 3. promote accountability of the adopters to sustain the change. This model works well for introducing programs where there were none to begin with [55].

While we also invest in the focus areas above, our propagation model differs from this type of framework in some critical ways. Given that our initiative leadership is based at a University and functions within a research lab, a crucial component of our model leverages this research function to build novel innovations. Our research efforts endeavor to understand the current state of Computer Science Education in our state, through landscape reports and data collection just as is expected within the ECEP. We have focused efforts to measure impact of the initiative on the state [27] as well as generate new information about student learning outcomes [122]. However, we also engage in research activity to develop instructional technology, curricular innovations, and pedagogical innovations [26, 25, 11, 8, 103]. This branch of research focus allows for us to extend beyond the SCRIPT and ECEP frameworks in that we are able to iteratively inject innovation into our social system, moving our propagation framework from a pathway to a cycle.

8.1.1 An Analogy

The differences between these frameworks can be described with an analogy comparing each to a development lifecycle. The existing frameworks function like a waterfall development process (figure 8.1) and our framework functions as an agile development process (figure 8.2).

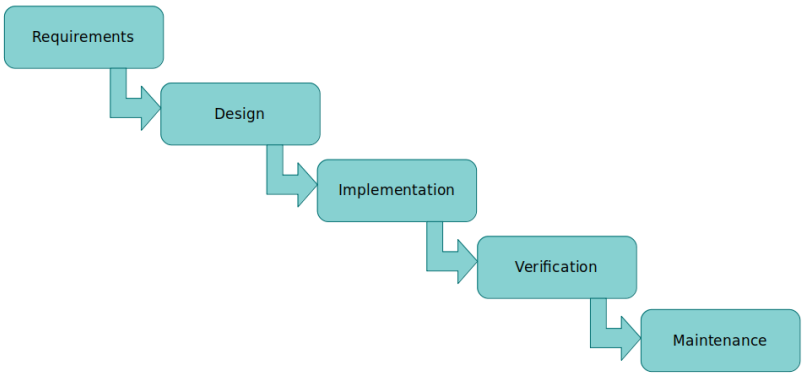


Figure 8.1: Basic Example of a Waterfall Approach

The waterfall process is easy to implement and produces an agreed upon product, this is like the existing frameworks in that goals are set and there is a framework of steps to get you from your starting point to those goals.

In this analogy, the agile cycle describes our framework better than the waterfall model. Our research efforts allow for us to continuously cycle through 1. the identification of a need in our system, 2. the development and testing of a solution to that need, and 3. implement the solution and measure the impact, which will undoubtedly result in identification of new needs and begin the cycle all over.

8.1.2 A Propagation Cycle

The ability to have a cycle based framework instead of a path based framework is important when considering the future. While broadening participation in computing is a challenging mission, it will not always mean that we should be focusing on ensuring the existence of computing in K-12 classrooms. If the current national, state, and multi-state initiatives are successful in meeting their objectives we will have a solid foundation to work from across the major themes identified in the literature:

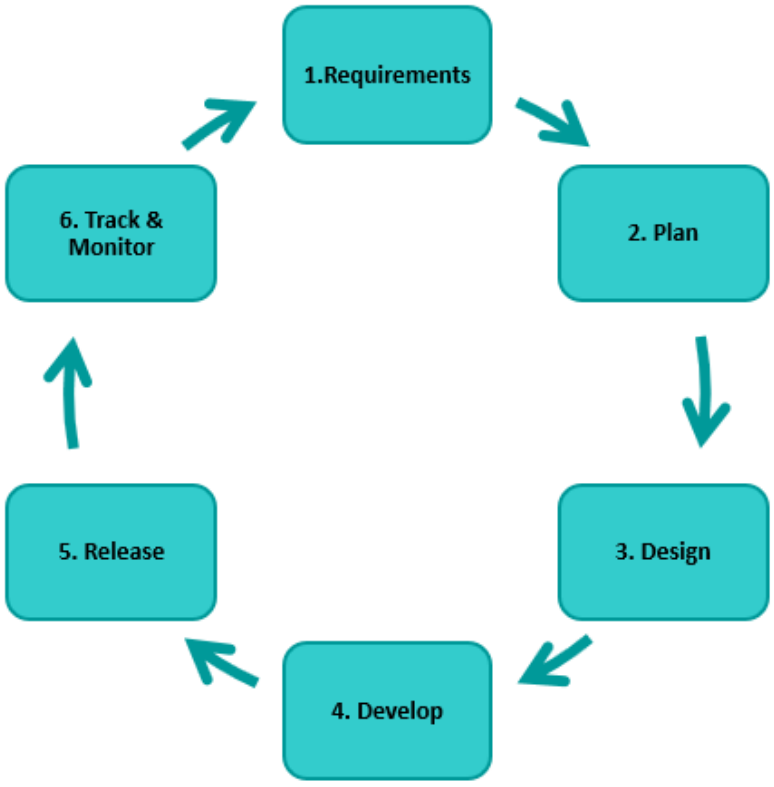


Figure 8.2: Basic Example of an Agile Approach

- Theme 1: **Computer Science must have status as a Core Subject** Students should be required to complete credits in computer science for high school graduation, or have computer science count for math or science credit towards graduation.

- Theme 2: **Equity in Opportunity** Underrepresented groups are less likely to have the opportunity to take computer science courses. Computer science should be offered at all high schools and schools should receive incentive for prioritizing computing courses.

- Theme 3: **Knowledgeable and Qualified Teachers** Licensing/certification programs should be more comprehensive and accessible for those wishing to teach computer science and states should require that computer science courses are taught by qualified teachers.

What are the implications of this shift? Today, we start from the assumption

that these themes need to be "solved". If we instead assume that the foundation of Computer Science Education consists of these themes as already solved:

- Theme 1: **Computer Science is a Core Subject** Students are required to complete credits in computer science for high school graduation
- Theme 2: **Opportunity is Equitable** Underrepresented groups have the opportunity to take computer science courses. Computer science is offered at all high schools and schools receive incentive for prioritizing computing courses.
- Theme 3: **Teachers are Knowledgeable and Qualified** Licensing/certification programs are comprehensive and accessible for those wishing to teach computer science and states require that computer science courses are taught by qualified teachers.

It will be difficult to maintain the momentum of the current movements if we are still focusing on propagation plans that are built to enable initial adoption of computing as an option in schools.

In fact, our analysis of the implementation of the Exploring Computer Science curriculum in Wisconsin showed declining enrollment after initial implementation in individual schools. This implies that while our implementation was successful, the schools needed more than ECS to create a self sustaining computing program. In general terms, an initial implementation of a introductory computing course is insufficient. This is where our new, fourth, theme, **Innovation**, can play a role. Our efforts, being cyclical, should aim for innovation in each cycle to produce the next feature needed to sustain and promote growth of K-12 Computer Science Education.

There are implications beyond the need for innovation, or the "what" based implications, there are also "who" based implications for this shift. It has been our experience that the students who belong to the classrooms where our initiatives have taken hold are uniquely positioned to grow into the role of an innovator when they move from the K-12 social system to post secondary social system. We have had a number of students come from K-12 Computer Science programs where our initiatives are implemented and become members of the Systems Lab, working to develop new innovations that go right back into those K-12 classrooms where they were once members. Thus making a person based cycle as well and an innovation based propagation

cycle.

8.1.3 A New Model for Propagation

Based on Rogers' Diffusion of Innovation, our propagation model is meant to serve a broader audience than the current frameworks, it is also intended to remain relevant further into the future.

Cycle and Scope

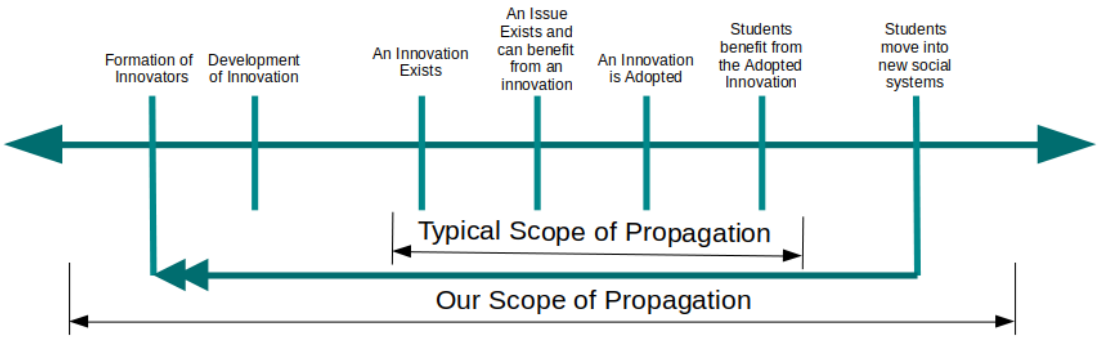


Figure 8.3: Expanded Scope of Our New Propagation Model

Figure 8.3 illustrates our expanded scope and person based cycle. Within Rogers' innovation-decision model, the process begins with an individual gaining knowledge that an innovation exists. This implies that we are to assume that the innovation does in fact already exist. This is the case with many of the efforts to make computing courses available in the K-12 education space. Most initiatives, Marquette's PUMP-CS included, mark their starting line with identification of a curriculum, like ECS, that they will be aiming to implement broadly. Our model will provide the flexibility to begin from that point, or from the point of needing to bring that innovation into existence. With the cycle concept in mind, there is not a prescribed starting point; and, as we will discuss further in the next section, a program can be operating at multiple points within the cycle at any one time.

The change in scope of our model does not mean that the innovation-decision

model is not in alignment. Figure 8.4 shows where the innovation-decision process exists within the scope of our cycle base framework. It is important to note that continued attention being given to the decision process defined by Diffusion of Innovation theory is critical for innovation adoption. More than ever, planning for propagation, as defined by [168], must be a key part of the innovation development process for this cycle framework to produce the intended outcomes.

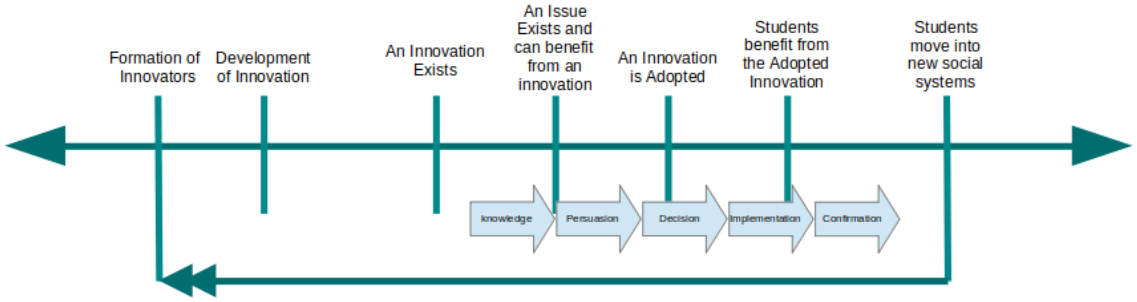


Figure 8.4: Alignment with Innovation-Decision Model

In addition to the scope expansion, the "person cycle" component is a new feature. Since this model is particularly meant for an audience of propagators functioning within a research context, the role of innovator takes on additional meaning from how they are defined within the adopter categories by Rogers. Within our framework, the innovator is a very special role, no longer are they solely playing the part of adopting an innovation that is assumed to exist, they are now playing the key role of innovation developer. While Rogers' classification does indicate that the innovator adopter is most likely open to being a beta tester or iterating through versions of the innovation with the developer, we suggest that within a research context in an academic setting that iterative development and testing can indeed be components of that innovator role. Not only does that adopter category expand, but we are also able to start putting in place the means by which a perpetuating cycle that feeds individuals into that role can exist. While we have already seen this process in work within the Systems Lab at Marquette, formalization by means of marketing materials or summer research opportunities as outreach may contribute to this being a reliable pipeline not only for students choosing to move into a Computer Science major at

the undergraduate level, but also for research, development, and innovation minded students to begin creating the types of educational opportunities they would have benefited from and implementing them within the K-12 space. Who better to feed back into the cycle and provide valuable perspective on what helped and what did not than those very students who were impacted? This also creates a new branch in the diffusion network, a new bridge between propagator and adopter, which we will explore in the next section.

Roles

Much of the focus in a propagation model is given to the role of the adopter. This should be expected since propagation cannot be successful if the adopter does not decide to adopt the innovation. In addition to the adopter, an innovation driven propagation framework should consider the various roles that change agents have within a system.

We will consider some of the important change agent rolls within the propagation framework.

- **Change Agent**: Anyone actively advocating for or promoting an innovation
- **Adopter**: Anyone who is part of the group of potential users of the innovation
- **Champion**: Anyone who has successfully adopted the innovation and evangelizes among their peers to encourage adoption of the innovation
- **Opinion Leader**: Has high social capital and influences others' opinions about the innovation
- **Developer**: The creator of the innovation (can also be though of as the propagator)

If we consider the adopter role as categorized by Rogers, within our framework it might be assumed that a focus on developing innovations equates to a focus on engaging the *Innovator* category of adopter. This is true, however, since the model we are promoting functions as a cycle, while Innovator adopters are being engaged for recent innovations, other adopter populations can focus on innovations that are at the appropriate stage in the cycle for their needs. This is illustrated in figure 8.5

(K)nowledge (P)ersuasion (D)ecision (Adopt, Reject, Trial) (I)mplementation (C)onfirmation	Innovator	Early Adopter	Early Majority	Late Majority	Laggards
Innovation A (Beta Testing)	D	P	K		
Innovation B (Tested, relatively new)	I	D	P	K	
Innovation C (Implemented by Early Adopter Groups with initial successful results)	C	I	D	P	K
Innovation D (Has been successfully implemented by much of the population and has proven results)	C	C	I	D	P
Innovation E (Has become the standard)	C	C	C	I	D

Figure 8.5: Multi-Innovation Initiatives can have categories of adopters engaged in different stages of the cycle at any given point in time

One concept, that is repeated across a variety of initiatives within the K-12 Computer Science space, is that in order to broaden the reach of your initiative and to ensure widespread adoption you must find your champions. Depending on the flavor of initiative, a champion could be a teacher who has already successfully adopted, or an advocate for the importance of your cause, even local industry partners who associate your initiative with providing a social good. These champions are supposed to help support your initiative by networking with potential adopters, promoting the effort within their social influence, and even help identify "new" things to include in your program. Reviewing the STEM and CS propagation literature suggests that involving diverse champions is an important factor in the success of your initiative [168].

Commonly, we look to the innovator category of adopters to serve the roll of champion. It makes sense, they are enthusiastic about the innovation, they have successfully implemented, they can provide information to their peer network. While these adopters can play a part in the champion roll, this is not always the most

effective method of diffusing knowledge through the system. Rogers suggests that relying on innovator category adopters to serve this purpose may result in diffusion only to other innovators [155]. This is due to the fact that while innovator adopter may be the earliest to implement an innovation, they are not necessarily *opinion leaders*.

Opinion leaders play the role of influencing other individuals opinions about innovations and they can have a significant impact on the rate of adoption. Defining characteristics of the opinion leader are that they they exhibit more social participation than their followers (they are accessible to a broader range of people), they are typically at a higher social status than their followers as well, and they are trusted but not necessarily experts. Opinion leaders do not have to be adopters but if they are, they are typically in the early adopter category instead of the innovator category.

Having diversity among your change agents can help to support the propagation cycle. Since the potential adopters of the innovation are likely to be K-12 Computer Science teachers, we should also consider change agents outside of this group. It is common to recruit change agents from school and district leadership (principals, guidance councilors, school board members).

Some other groups to consider should be:

- **Former Students**: As we described in the expanded scope of our propagation framework, former students of the teachers in the potential adopter group, especially students who have moved into an innovation developer role, can make amazing opinion leaders
- **Teachers in other fields**: These teachers, while they are not potential adopters, can expand the diffusion network for the innovation. They are often opinion leaders.
- **Industry Professionals**: These individuals can serve as trusted knowledge experts as well as opinion leaders.

8.2 Innovation in PUMP-CS

With the overarching goal of expanding access to K-12 Computer Science Education in the state of Wisconsin, PUMP-CS has made progress toward the themes identified

as necessary for a successful Computer Science Education program.

- Theme 1: **Computer Science must have status as a Core Subject** PUMP-CS leadership has worked with the Wisconsin Department of Public Instruction to establish the adoption of Computer Science Education standards as well as enabling Computer Science courses (that meet those standards) to count as a Science credit toward High School Graduation

- Theme 2: **Equity in Opportunity** PUMP-CS has focuses a major portion of their effort on high need schools, specifically in the Metro Milwaukee area. These schools have a high percentage of economic disadvantage (73% district average for the 2014-15 school year) as well as serving a significant minority population (90% district average for the 2014-15 school year). These populations are in contrast to the overall state averages of 36% economic disadvantage and 83% white student population for the same school year.

- Theme 3: **Knowledgeable and Qualified Teachers** PUMP-CS has been fostering multiple pathways to Computer Science teacher licensure and training, through professional development, alternative licensure programs, and university initial licensure (major and minor) options.

In addition to this, the theme of **Innovation** has played a role in the success of PUMP-CS. Along side the implementation of proven curriculum like Exploring Computer Science, Code.Org, and Computer Science Principles, multiple innovations have been developed to support the adoption and expansion of Computer Science Education in the state.

8.2.1 Propagation + Innovation

The five novel contributions described earlier are examples of how the Computer Science Systems Lab at Marquette has contributed to the PUMP-CS initiative. Each innovation builds on the overall goal to expand access to Computer Science Education while supporting the propagation framework to promote successful adoption and scaling.

Measuring Adoption

Leadership for the systems lab and PUMP-CS has been engaged in the mission of expanding access to Computer Science Education well before establishing a state wide program [3, 91]. These workshops for educators were focused on showing how computing was related to STEM subjects and how computing and computational thinking concepts could be leveraged in those classrooms. We can consider the innovation here being the use of computing and computational thinking concepts in K-12 STEM courses. These early workshops had the goal of promoting the innovation for adoption by the educators that attended.

Our dual track CS4HS workshop [25] was the first in this series to attempt to measure the degree to which adoption would take place. We strove to understand, by studying the result of the teachers incorporating computational thinking concepts into their lesson plans, how those concepts would surface in the classroom setting. This effort offered the realization that few of the teachers were able to incorporate the computational thinking concepts into a typical lesson plan for their class, and the recognition that this inability did not change the perception that the teachers had about the workshop content being important and valuable.

The **relative advantage** of incorporating Computational Thinking Concepts into their courses was perceived to be high (the teachers average response to whether they found value in the workshop was 4.945) and **compatibility** was perceived just as high as evidenced by the 4.93 average response to how likely they would be to incorporate what they had learned into their curriculum. Where we failed to effectively navigate the requirements for propagation were in the innovations' **complexity**, asking non-CS STEM teachers to incorporate computing material into their lessons with only two days of introduction to the concepts was much too complex a task to leave so open ended. We did not support those teachers with example lesson plans, limiting the **trialability** of the changes we wanted them to adopt. In addition, our intention to keep the implementation requirements open and not prescriptive as a way to allow for creativity and flexibility for teachers from such a broad range of backgrounds, we failed to provide an adequate level of **observability** to reduce the unknown conditions as required for successful adoption of an innovation.

This understanding was the catalyst for PUMP-CS to identify an established curriculum as our path forward. The decision was made to move forward with Exploring Computer Science [65] as our initial focus for widespread adoption in the state of Wisconsin.

Overcoming Barriers

The high degree of **relative advantage** that a proven innovation like ECS had over either the nonexistent or self developed curriculum that existed in the classrooms was apparent. In addition, the ECS model being inclusive of training by teacher leaders using the pedagogy expected within the curriculum, achieved the requirements of high **compatibility, trialability, and observability**, while being of low **complexity**. We quickly established our change agents across various roles and levels with champions assisting from successful implementations in Chicago and opinion leaders like Joe Kmoch and CSTA leadership. We also established the expectation that as adopters, we intended to grow new change agents from this cohort, both champions and opinion leaders.

Over the course of our initial implementation we found that many of the ECS adopters in our cohort were skipping the Robotics module of the curriculum. This module was not being left out due to it being irrelevant, or a lack of time or interest, the factor in determining if a teacher would include the module seemed to be based on cost. Within a typical propagation framework, the lack of **compatibility** perceived due to cost constraints could cause rejection at any stage of the innovation-decision process, this could even cause discontinuance of the adoption during implementation due to difficultly with funding or as uncertainty would rise during confirmation.

The introduction of the **innovation development** stage in our propagation framework allows for pathways to overcome this type of obstacle. The Systems Lab developed an embedded systems replacement module consisting of hardware, software, curriculum, and training [11] named MUzECS. Given that this was a true innovation and not a perceived innovation to our adopters (like ECS was), we needed to ensure that as much uncertainty as possible was mitigated in our solution. Our attempts to ensure the innovation was able to be easily adopted are as follows:

- **Relative Advantage**: The relative advantage of the MUzECS module over

the Robotics module is high for cost, a set for MUzECS cost $60 while the Lego Mindstorms required for ECS Robotics is $600, as well as flexibility, while our module is constructed within the context of "music" the Ardunio platform that we developed from has hundreds of functions and applicable components which would make it a simple process for a teacher or student to expand on the prescribed activities.

- **Compatibility**: A critical component of the development of the MUzECS module was to ensure that the curriculum mapped to the same standards and outcome as the ECS Robotics module, this enabled high compatibility with the existing ECS curriculum and therefore made it possible for teachers to easily substitute as needed.

- **Complexity**: There were three critical steps taken to reduce the overall complexity of the MUzECS innovation. First, we were able to maintain a block-based language for this module by utilizing Ardublock. The real and perceived complexity of incorporating a new programming language into your course is greatly reduced when the new language is just as visual and incorporates the same "snap in place" blocks as you are currently using in the other modules with Scratch. Next, we were able to create step by step video tutorials for each of the components and lessons in the new module to ensure that teachers were supported with references specific to the module. Finally, the lessons within the module were constructed in the same form as the existing ECS modules so that the hardware and software components were able to be used right away, without the need for the teacher to develop their own lessons.

- **Trialability**: The MUzECS solution was built with trialability in mind. The board is portable and has the ability to store your program, this enables the developer (propagator), or other champion, to literally put the trial in the palm of your hand. A basic routine is stored on the board and the adopter is able to test functionality and preview results without having to go through any training.

- **Observability**: Just as the video tutorials serve the purpose of reducing complexity, in the early stages of propagation where there is not an observable user

base, those same videos serve the roll of creating observability in the innovation. Not only do the videos give the ability to observe the innovation "at work", the videos were conceived to put the developers in a visible roll. This is important because those developers were students in the classrooms of teachers in the potential adopter group only a short time before hand. This serves the purpose of the developers being seen as near-peers to the students who would benefit from the innovation should the decision be made to adopt. In addition, those developers can act as change agents in this network given they have a level of social capital with the teachers in the potential adopter group.

Building Diverse Change Agent Networks

While much of the focus for building change agents is on the potential adopters and their immediate networks, diffusion theory suggests that the reach within a close network is limited [155]. In addition, one of the risks of adopters being in the late majority or laggard categories is that they are socially isolated and not a member of the diffusion network the earlier adopting categories are. There are two initiatives that the Systems Lab has implemented to help with these issues.

First, given that many of the potential adopters are physically isolated from the rest of the adopter group due to most schools only employing a single teacher for the Computer Science courses, we have established a virtual peer network. This community of practice is a self organizing group led by teachers who have successfully adopted one or many of the innovations through the PUMP-CS program. These groups meet regularly to combat the isolation a teacher new to the program may feel without having any similarly trained teachers in their own school. The community of practice also serves as a continued learning environment, giving "just in time" content knowledge and lesson feedback to teachers who are implementing for the first time, as well as providing the needed reinforcement of **relative advantage** to teachers in the confirmation phase of the innovation-decision process. This community of practice itself should be considered an innovation produced by the Systems Lab, as the original ECS product did not include such a device.

Our second initiative established to support a diverse change agent network was the development of a humanities curriculum inclusive of computing and computational

thinking concepts [26]. The benefits of this course are two fold, first there is direct benefit to the students in the course, and second there is benefit to the PUMP-CS program by way of these students becoming change agents and expanding our diffusion network.

The benefits to the students, as described in chapter 5, are many. This curriculum was built to replace an outdated and out of context quantitative literacy requirement. The students were able to engage in a broad range of computing activities, like network analysis, text analysis, and geographical mapping, to earn competence in and should prof of learning within the quantitative literacy content.

The benefits to our diffusion network are based on the Granovetter's theory of "the strength of weak-ties" [155]. This is more commonly known in network theory as *communication proximity*, and defines the degree of proximity based on the overlap in network ties between two points. In figure 8.6 A and B have a low communication proximity since their network ties do not overlap, while in figure 8.7 C and D have a high communication proximity since they have multiple overlapping network ties.

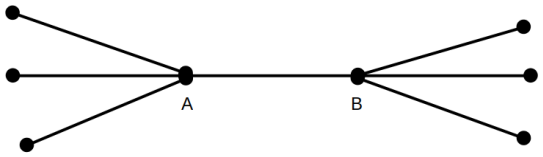


Figure 8.6: low proximity network

The implications for this are as follows, within a network with low communication proximity an innovation may diffuse from A to B reaching all of the ties that A has as well as all of the ties that B has. While a network with high communication proximity will have far less range of diffusion since the ties are shared between C and D already. What this tells us is that to ensure that our innovation diffuse to a broader set of potential adopters, we need to take advantage of networks with low communication proximity, in other words, create change agents from people outside of the K-12 Computer Science teacher networks and their direct support networks.

Our work to implement computing curriculum for college level humanities students creates those change agents. The institution where we implemented produced a high

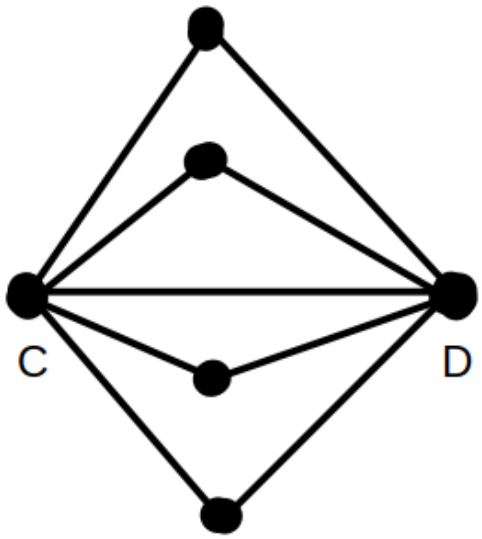


Figure 8.7: high proximity network

number of new K-12 teachers every year, none of those new teachers are Computer Science teachers, but many of them will work in schools what have an need for a Computer Science program or the need to expand the existing program. Creating an awareness within those pre-teachers of what computing education is and what value it provides to students provides us with an opportunity to create change agents that serve as "weak-ties". These new teachers can provide access to their networks, and those networks are likely not the same as the K-12 Computer Science teacher networks. They can act as the tie that bridges between two previously unconnected networks (as shown in figure 8.8, thereby expanding the influence of the PUMP-CS program to a new set of change agents and potential adopters.

A Community of Learners

In creating a diverse change agent network, we should not be content with creating champions from successful adopters or opinion leaders from non-STEM teachers. These are great strides forward in a successful propagation cycle, however, the majority of what these change agents are able to accomplish is spread of knowledge, we also require the catalyst for potential adopters to seek that knowledge out in the first place. We need a way to create need, or at least the perception of need.

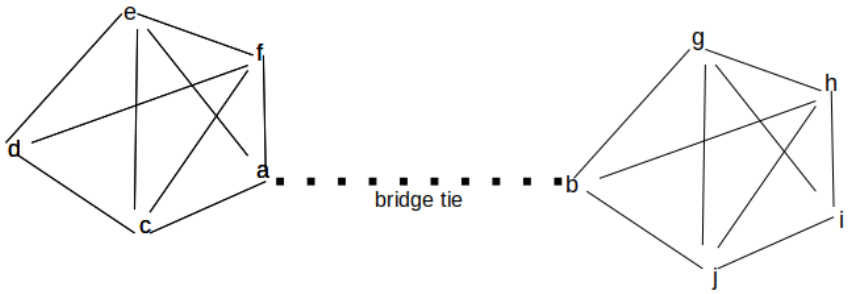


Figure 8.8: The bridge tie acts to diffuse awareness of the innovation to a previously unconnected network

The development of a Scratch based programming competition to run in parallel with a standard Java based competition allowed us to create an environment of engagement with the student beneficiaries of the ECS implementation. Bringing the Scratch participants into the same space as the "more advanced" Java students created a connection point between these groups and enabled the Scratch students to view themselves as belonging to this student community. Once they were members of that community there was every reason for them to expect that they would have access to courses beyond ECS. That expression of expectation is the generator of need for the teacher adopters to seek awareness knowledge of what comes after ECS.

An exciting effect of the Scratch competitions is that the schools that brought teams to the competition at the start of our efforts have started to bring teams to the Java competition now as well, some of the Java teams even have student members that had been on the Scratch teams in previous years. The implication of this observation is that expanding the scope of the propagation framework also requires an expansion in actor roles within that framework. Our propagation strategy should look at students of the potential innovation adopters as a large pool of opinion leaders, we should be creating targeted messaging to that student group that will generate the perception of need within a diffusion process from those students to the teachers as potential adopters of future innovations. While the Scratch competition provides all of the benefits outlined in chapter 6 [8], there is also the benefit of the competition as messaging to those students that there is a "next step" in this learning process and they should desire to be a part of it.

Continuous Measurement

In order for the propagation framework to function as a cycle instead of a path, there is a need to continuously measure and use that measurement to determine needs. An example of this measurement begets innovation process is given in chapter 7 [27]. In this example, adoption rates of the ECS program were measured based on student enrollment in those ECS courses within the state of Wisconsin. What we found was that in the initial year of adoption there were almost three times as many students enrolled at a school than in the following year. After some investigation into this result, it was determined that, in schools where no computing courses existed prior to the implementation of ECS, multiple grade levels were given the opportunity to take the ECS course as soon as it was available but the following school year only the students that had not already taken the course were eligible to enroll.

This makes sense since you would not want a student to take the same course a second time, however, it did highlight the need for a progression of courses to be available for K-12 instead of only the single level option. It is important to understand that if the measurement undertaken had been at a different level of granularity we would not have been able to glean this information. Answering the question "Has ECS been implemented in Wisconsin?" would have given us a binary "yes". Similarly, the question of "How many schools have implemented ECS in the state of Wisconsin?" would have left us with a measurement that would not have perpetuated the innovation cycle. When determining what to measure about the adoption of an innovation and how to measure that what, it is important to be thinking about the true value to the indented beneficiary of the innovation. In this case, our goal was to expand access to this computing course for students across the state. Therefore, we needed to measure if there was truly an increase in access at the student level. This line of inquiry lead to the metrics showing for each school what percentage of the student population was enrolled in the ECS course year over year, not just that ECS enrollment was present.

Answering the right questions and even knowing what those questions are, is not easy and should not be left to chance. As part of the propagation recommendations from [168], it is clear that meaningful measurement is a key component to the planning phase of your propagation plan and innovation development. The closest

analogy points back to the SDLC process presented earlier in this chapter. When a team is developing a new system or software, a critical component is the presence of acceptance and testing criteria within the stories and requirements that describe the product. Developing meaningful and usable acceptance and testing criteria allows for the team to understand what they are developing at a deeper level, they can use these criteria to build an understanding of how the user intends to interact with the product and what the expectations are for success and failure. This is the type of thinking that should go into identifying measures for innovations that will foster continuous improvement, assist in identification of gaps that need to be addressed, and create the need for additional innovation.

8.3 Addressing Diffusion Shortcomings

Given the criticism of the application of Diffusion of Innovation Theory, the documented shortcomings of this framework should be addressed.

- **Pro-Innovation Bias**: to avoid the implicit bias that might be exacerbated by an innovator lead propagation framework, our model relys heavily on user feedback, both from successful adopters and potential adopters who have rejected any of our innovations, to set the priority for new development and iterative improvement of innovations. In addition, adequate attention must be given to the appropriate pacing of adoption rates and overall success should be defined based on the overall system need and across the adopter categories.

- **Individual-Blame Bias**: careful attention should be given to the systems where the innovation may be applied instead of solely focusing on the individuals within those systems. Our work attempts to address this issue by ensuring measurement of innovation impact is occurring at the system level as in [27]. In addition, the expansion of the diffusion model to enable the researcher practitioner partnership, incorporates a three party model where the innovator party lies between the researcher and practitioner parties. The implication of this approach is that there is an independent research effort that is not directly accountable as the source of the innovation. This disassociation can provide for an unbiased assessment of the need for and impact of the innovation on the

system.

- **Recall Problem**: Continuous and meaningful measurement are critical components to a successful propagation cycle. Survey data should never be the sole method of data generation, instead, the focus should be on field observation, longitudinal panel studies, use of recorded data over time, and case studies of the innovation.

- **Issue of Equality**: In order to prevent an unintended increase in socioeconomic gaps as a result of our innovations, particular close attention should be paid to individual versus system blame and pro-innovation bias. The feedback mechanisms in place to provide for identification and prioritization of new development efforts must account for and apply additional effort to collecting and understanding the unique needs of potential adopters that may fall into the late adopter or laggard categories. In addition, research objectives should be constructed to better understand the impacts of our innovations to these under served populations and particularly the effect that our propagation effort have on the social system as a whole.

CHAPTER 9

Conclusion and Future Work

The aim of this work was to present a cycle based and innovation focused propagation framework for use within K-12 Computer Science Education initiatives and discuss how a series of innovations developed in the Systems Lab at Marquette University have supported this framework.

9.1 Contributions

We presented five separate novel contributions to the Computer Science Education Field.

9.1.1 Chapter 3

We present a framework to directly measure workshop participants ability to incorporate computational thinking concepts into lesson plans for use in their classrooms. This work in significant in that enabled our program to evaluate where we needed to make changes to achieve desired results in our continuing efforts to expand access to computing education.

This work [25] has been cited repeatedly as a model for measuring outcomes of teacher professional development in CS Education [50, 179, 160, 115, 61, 54, 89, 118, 124, 18, 188, 146, 138, 117, 20, 169, 88]. In addition, it is interesting to note that the year after [25] was published, Google's CS4HS program began to incorporate more rigorous outcome measurement into all of their grants, pushing the entire CS4HS community to assess their PD workshops beyond mere pre-/post-surveys.

My individual contribution to this effort was in the design of the measurement instrument, the development of the lesson plan building requirement for the workshop, and the collection and analysis of the video and survey data from the participants.

9.1.2 Chapter 4

We developed a low cost replacement for the robotics module in the Exploring Computer Science curriculum. Our product was based on an Ardunio Leonardo board and we contributed the development of hardware in a custom shield, software in custom Ardublock blocks, and a curriculum that mapped back to the learning outcomes from the original robotics module. This was able to be provided for teachers at one tenth the cost of the Lego product required for the robotics module and greatly reduced the barrier to adoption of the ECS curriculum in the state of Wisconsin.

Since the development of MUzECS, Marquette has produced close to 200 custom shields which have been provided at cost to at least 10 schools in the state. Innovation has continued on this product in the form of a cloud-based development environment [103] to address the need identified by adopters for availability of this module while using chromebooks. The ECS team has since released a less expensive Edison-based option and the Code.org CS Discoveries physical computing unit has replicated many of the key features originally developed by the MUzECS project.

My individual contribution to this effort was in the design of the course curriculum, the development of material to foster ease of adoption in the form of video tutorials, and the development of the initial product vision. The Systems Lab undergraduate student researchers were responsible for the design and manufacture of the custom shields and the software components of the solution.

9.1.3 Chapter 5

We developed a new course to incorporate computing and computational thinking concepts within an English Literature course. The activities that we incorporated in the course gave students an opportunity to explore computer science applied to literary analysis. Exposure to computing concepts in this course, as applied to literary analysis, gave students a new perspective on and deeper understanding of the nature of the nineteenth century Gothic novel.By focusing on developing activities that relied on the critical frameworks and in fact show the overlap and relationship between those frameworks, we reinforced the idea that computer science can work well to create interdisciplinary learning environments and engage students in new methods

for continuing their work of interest. We have seen students analytical and problem-solving abilities develop through these computational approaches.

The development of context based course content to meet the need for a quantitative literacy requirement has continued. There is collaboration across disciplines to accomplish this as needed. This work has served as a guide for a number of new programs aiming to introduce computing within a specific course context [94, 16, 192, 32, 33, 175, 151].

My individual contribution to this effort was in the design of the computing curriculum, the mapping to learning outcomes, teaching the course and advising the students, as well as all of the collection and analysis of the data from the students.

9.1.4 Chapter 6

We developed a Scratch based programming competition to run in parallel with a traditional programming competition. This new competition track has assisted in the broadening of participation by students that would otherwise not be eligible for this type of community and efficacy building experience.

Participation in this event has grown substantially since our initial implementation in 2014 with fewer than 10 students participating in this team challenge. By 2018, our Scratch track had participation by more than 100 students across 38 teams, nearly a third of those participants from Milwaukee Public School which supports a minority student population above 75% on average and suffers from a high degree of economic disadvantage.

The Scratch division continues to be a major component of the competition every year, and took place virtually during the pandemic spring of 2021.

My individual contribution to this effort was in the development of the original product concept. I identified the need for, developed the content, secured the resources, and implemented the original version of this effort. In addition, for the second year of competition, I developed the scoring matrix and assisted in creating the tasks.

9.1.5 Chapter 7

We applied a place based disparate impact analysis to understand the behavior of adoption for our ECS initiatives. This type of analysis, with a focus on place as a component and use of publicly available data with course enrollment data supplied by the Department of Public Instruction, had not been applied to K-12 Computer Science initiates at the time. This work has been leveraged as a model for recent work focused on understanding the changing landscape of CS education [161, 109].

Appropriate data collection and measurement are key concepts to successful widespread adoption of an innovation. Perhaps more significantly, since the publication of this work, ECEP has matured their framework to include the type of data collection we outline and they have defined a standard set of measurements for their cohort states which are incredibly similar to the measurements suggested in our work.

My individual contribution to this effort was in identifying the methods, collecting and processing the data, performing the analysis, interpreting the results, and creating the visualizations.

9.1.6 Chapter 8

We have defined a model for a cycle based propagation framework which expands on the typical scope of propagation models to include an innovation development stage. This is significant for the continued expansion of K-12 computing programs as well as refinement of the researcher-practitioner partnerships model suggested by NSF.

9.2 Future Work

Through this work, we have identified many limitations and gaps in the current state.

9.2.1 Information Governance

We suggest that a program for information governance be developed and implemented for state data collection and management. While the current process for schools to report a vast amount of information covering dozens of topic areas to the Department of Public Instruction, the processing of this data is time consuming and often leads to data loss and inaccuracy. We have found many cases where course and demographic

information has been misleading due to inaccurate and incomplete table entries. In addition, there does not appear to be a consistent definition process in place and no formal governance program is indicated in any of the DPI material related to the procurement and processing of this data.

9.2.2 Model Validation

Additional application of the propagation cycle framework is required to validate the generalizability of this model and its appropriateness as a framework for our researcher-practitioner partnership work. Our ability to test this framework both within the state of Wisconsin and as it can be applied to other state wide initiatives will require an extensive and long term research focus.

9.2.3 Consequences of the Propagation Model

Another line of inquiry should be considered to better understand the consequences of a diffusion based propagation model like the one we describe. Given the interconnectedness of social and economic status with the adopter categories utilized within the diffusion framework, the implications of the "Innovativeness-Needs paradox", as Rogers calls out [155], is particularly concerning. The paradoxical relationship between the innovation adoption and the need for the benefits provided by that innovation has not been studied at all within the context of K-12 computer science education. We must strive to better understand if we are truly meeting our goal to expand access given the behavior presented by this paradox whereby the individuals who are most in need of the benefits from an innovation are likely the last to adopt and that the consequence of this behavior may well be a widening of the socioeconomic gap between the higher and lower socioeconomic groups within our potential adopter groups for technological innovations.

REFERENCES

[1] Adrion, R., Fall, R., and Guzdial, M. Broadening access to computing education state by state. *Communications of the ACM 59* (01 2016), 32–34.

[2] Adrion, R., Fall, R., Matos, M., and Peterfreund, A. Integrating evaluation into program development: Benefits of baselining a nsf-bpc alliance. pp. 27–31.

[3] Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A., and Early, J. P. Computational thinking for the sciences: A three day workshop for high school science teachers.

[4] Alvarado, C., and Dodds, Z. Women in cs: An evaluation of three promising practices. pp. 57–61.

[5] American Computer Science League. American computer science league.

[6] Anwar, T., Jimenez, A., Bin Najeeb, A., Upadhyaya, B., and McGill, M. M. Exploring the enacted computing curriculum in k-12 schools in south asia: Bangladesh, nepal, pakistan, and sri lanka. In *Proceedings of the 2020 ACM Conference on International Computing Education Research* (New York, NY, USA, 2020), ICER '20, Association for Computing Machinery, p. 7990.

[7] Ardublock. Ardublock.

[8] Arnold, J., Bort, H., Naugle, R., O'Hare, C., and Brylow, D. Multi-track programming competitions with scratch. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education* (New York, NY, USA, 2016), SIGCSE '16, Association for Computing Machinery, p. 228233.

[9] Association for Computing Machinery. Curricula recommendations, csta k-12 computer science standards, 2011.

[10] Association for Computing Machinery (ACM). Icpc factsheet.

[11] Bajzek, M., Bort, H., Hunpatin, O., Mivshek, L., Much, T., O'Hare, C., and Brylow, D. Muzecs: Embedded blocks for exploring computer science. In *2015 IEEE Blocks and Beyond Workshop (Blocks and Beyond)* (2015), pp. 127–132.

[12] Baron, g.-l., Drot-Delange, B., Grandbastien, M., and Tort, F. Computer science education in french secondary schools. *ACM Transactions on Computing Education 14* (06 2014), 1–27.

[13] Barr, V. Create two, three, many courses: an experiment in contextualized introductory computer science. *Journal of Computing Sciences in Colleges 27* (06 2012), 19–25.

[14] Barr, V., and Stephenson, C. Bringing computational thinking to k-12. *ACM Inroads 2*, 1 (2011), 48 – 54.

[15] Barrowman, C., and et al. Critical theory as critical thinking: A book of classroom practice in literary criticism. *Alverno College* (2010).

[16] Bart, A. C., Tibau, J., Kafura, D., Shaffer, C. A., and Tilevich, E. Design and evaluation of a block-based environment with a data science context. *IEEE Transactions on Emerging Topics in Computing 8*, 1 (2020), 182–192.

[17] Beach, A., Henderson, C., and Finkelstein, N. Facilitating change in undergraduate stem education.

[18] Bean, N., Weese, J., Feldhausen, R., and Bell, R. S. Starting from scratch: Developing a pre-service teacher training program in computational thinking. In *2015 IEEE Frontiers in Education Conference (FIE)* (2015), pp. 1–8.

[19] Bell, T., Andreae, P., and Robins, A. A case study of the introduction of computer science in nz schools. *ACM Transactions on Computing Education 14* (06 2014), 1–31.

[20] Bell, T., Andreae, P., and Robins, A. A case study of the introduction of computer science in nz schools. *ACM Trans. Comput. Educ. 14*, 2 (June 2014).

[21] Bell, T., Witten, I., and Fellows, M. Cs unplugged.

[22] Bellettini, C., Lonati, V., Malchiodi, D., Monga, M., Torelli, M., and Zecca, L. Informatics education in italian secondary school. *ACM Transactions on Computing Education (TOCE) Special Issue on Computing Education in (K-12) Schools 14* (01 2014), 15.115.6.

[23] Blum, L., and Cortina, T. J. Cs4hs: an outreach program for high school cs teachers. *Proceedings of the 38th SIGCSE Technical Symposium: Computer Science Education* (2007), 19 – 23.

[24] Bordenave, J. D. Communication of agricultural innovations in latin america: The need for new models.

[25] Bort, H., and Brylow, D. Cs4impact: Measuring computational thinking concepts present in cs4hs participant lesson plans. In *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2013), SIGCSE '13, Association for Computing Machinery, p. 427432.

[26] Bort, H., Czarnik, M., and Brylow, D. Introducing computing concepts to non-majors: A case study in gothic novels. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2015), SIGCSE '15, Association for Computing Machinery, p. 132137.

[27] Bort, H., Guha, S., and Brylow, D. The impact of exploring computer science in wisconsin: Where disadvantage is an advantage. In *Proceedings of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (New York, NY, USA, 2018), ITiCSE 2018, Association for Computing Machinery, p. 5762.

[28] Bosch, M. Mathematics and science: The relationships and disconnects between research and education. *In Compendium for Math and Science* (2014).

[29] Bowring, J. A new paradigm for programming competitions. vol. 40, pp. 87–91.

[30] Brown, N., Sentance, S., Crick, T., and Humphreys, S. Restart: The resurgence of computer science in uk schools. *ACM Transactions on Computing Education 14* (07 2014).

[31] Bruckman, A., Biggers, M., Mcklin, T., Dimond, J., Disalvo, B., Hewner, M., Ni, L., and Yardi, S. "georgia computes!": Improving the computing education pipeline. vol. 41, pp. 86–90.

[32] Brunvand, E., and McCurdy, N. Making noise: Using sound-art to explore technological fluency. *ACM Inroads 8*, 2 (Mar. 2017), 6065.

[33] Brunvand, E., and McCurdy, N. Making noise: Using sound-art to explore technological fluency. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (New York, NY, USA, 2017), SIGCSE '17, Association for Computing Machinery, p. 8792.

[34] Brylow, D. Cs4hs @ mu.

[35] Buck Laura, B., Bretz Stacey, L., and Towns Marcy, H. Characterizing the level of inquiry in the undergraduate laboratory. *Journal of College Science Teaching 38*, 1 (2008), 52 – 58.

[36] Bureau of Labor Statistics, U.S. Department of Labor. Occupational outlook handbook, 2012-13 edition.

[37] Bureau of Legislative Research. Teacher attrition in arkansas: A supplement to teacher supply and demand in arkansas.

[38] Burke, Q., Schep, M., and Dalton, T. Cs for sc: A landscape report of k-12 computer science in south carolina. pp. 705–705.

[39] Caplan, N., and Nelson, S. D. On being useful: The nature and consequences of psychological research on social problems.

[40] Carter, L. Why students with an apparent aptitude for computer science don't choose to major in computer science. vol. 38, pp. 27–31.

[41] Carter, L. Interdisciplinary computing classes: Worth the effort. pp. 445–450.

[42] Charters, P., Lee, M., Ko, A., and Loksa, D. Challenging stereotypes and changing attitudes: The effect of a brief programming encounter on adults' attitudes toward programming. pp. 653–658.

[43] Choi, J., An, S., and Lee, Y. Computing education in koreacurrent issues and endeavors. *ACM Transactions on Computing Education 15* (04 2015), 1–22.

[44] Code dot Org. Code dot org.

[45] Collins, D., and Coleman, T. Social geographies of education: Looking within, and beyond, school boundaries. *Geography Compass 2* (01 2008), 281 – 299.

[46] Committee, C. C. Bugs in the system: Computer science teacher certification in the u.s. *Association for COmputing Machinery* (2013).

[47] Computer Science: Principles. Computer science: Principles.

[48] Coughenour, C. M. The problem of reliability of adoption data in survey research.

[49] Crampton, J., and Krygier, J. An introduction to critical cartography. *ACME: An International E-Journal for Critical Geographies 4* (07 2010).

[50] Curzon, P., Waite, J., Maton, K., and Donohue, J. Using semantic waves to analyse the effectiveness of unplugged computing activities. In *Proceedings of the 15th Workshop on Primary and Secondary Computing Education* (New York, NY, USA, 2020), WiPSCE '20, Association for Computing Machinery.

[51] Cutts, Q., Esper, S., and Simon, B. Computing as the 4th "r": A general education approach to computing education. pp. 133–138.

[52] Dagiene, V., and Futschek, G. Bebras, a contest to motivate students to study computer science and develop computational thinking. In *learning while we are connected, vol 3, book of abstracts* (2013), N. Reynolds, M. Webb, M. Syslo, and V. Dagiene, Eds., pp. 139–141. Vortrag: WCCE 2013, Torun; 2013-07-01 – 2013-07-07.

[53] Darmofal, D. *Spatial Analysis for the Social Sciences*. Analytical Methods for Social Research. Cambridge University Press, 2015.

[54] de Araujo, A. L. S. O., Andrade, W. L., and Serey Guerrero, D. D. A systematic mapping study on assessing computational thinking abilities. In *2016 IEEE Frontiers in Education Conference (FIE)* (2016), pp. 1–9.

[55] DeLyser, L. A., and Wright, L. A systems change approach to cs education: Creating rubrics for school system implementation. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education* (New York, NY, USA, 2019), ITiCSE '19, Association for Computing Machinery, p. 492498.

[56] DeLyser, L. A., Wright, L., Wortel-London, S., and Bora, A. Evaluating a systems approach to district cs education implementation. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2020), SIGCSE '20, Association for Computing Machinery, p. 11201126.

[57] desJardins, M., and Martin, S. Ce21–maryland: the state of computer science education in maryland high schools. pp. 711–716.

[58] Dettori, L., Greenberg, R. I., McGee, S., and Reed, D. The impact of the exploring computer science instructional model in chicago public schools. *Computing in Science & Engineering, Comput. Sci. Eng 18*, 2 (2016), 10 – 17.

[59] Dierbach, C., Hochheiser, H., Collins, S., Jerome, G., Ariza, C., Kelleher, T., Kleinsasser, W., Dehlinger, J., and Kaza, S. A model

for piloting pathways for computational thinking in a general education curriculum. *Proceedings of the 42nd ACM Technical Symposium: Computer Science Education* (2011), 257 – 262.

[60] Dierbach, C., Taylor, B., Zhou, H., and Zimand, I. Experiences with a cs0 course targeted for cs1 success. vol. 37, pp. 317–320.

[61] Dodero, J. M., Mota, J. M., and Ruiz-Rube, I. Bringing computational thinking to teachers' training: A workshop review. In *Proceedings of the 5th International Conference on Technological Ecosystems for Enhancing Multiculturality* (New York, NY, USA, 2017), TEEM 2017, Association for Computing Machinery.

[62] Dorn, B., Babb, D., Nizzi, D., and Epler, C. Computing on the silicon prairie: The state of cs in nebraska public schools. *SIGCSE 2015 - Proceedings of the 46th ACM Technical Symposium on Computer Science Education* (02 2015), 296–301.

[63] Downs, G. W., and Mohr, L. B. Conceptual issues in the study of innovations.

[64] Esper, S., Foster, S., and Griswold, W. On the nature of fires and how to spark them when you're not there. *SIGCSE 2013 - Proceedings of the 44th ACM Technical Symposium on Computer Science Education* (03 2013).

[65] Exploring Computer Science. Exploring computer science.

[66] Fancsali, C., Mark, J., and DeLyser, L. A. Nyc cs4all: An early look at teacher implementation in one districtwide initiative. In *2020 Research on Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT)* (2020), vol. 1, pp. 1–8.

[67] Finnigan, K., Holme, J., Orfield, M., Luce, T., Diem, S., Mattheis, A., and Hylton, N. Regional educational policy analysis. *Educational Policy 29* (06 2014).

[68] Fitzgerald, S., and Fritts, M. The computer science fair: An alternative to the computer programming contest. vol. 28, pp. 368–372.

[69] Fletcher, C. L., Dunton, S. T., Torbey, R., Goodhue, J., Biggers, M., Childs, J., DeLyser, L. A., Leftwich, A., and Richardson, D. *Leveraging Collective Impact to Promote Systemic Change in CS Education*. Association for Computing Machinery, New York, NY, USA, 2021, p. 994999.

[70] Fluck, A., Webb, M., Cox, M., Angeli, C., Malyn-Smith, J., Voogt, J., and Zagami, J. Arguing for computer science in the school curriculum. *Educational Technology & Society 19* (01 2016), 38–46.

[71] Force, C. S. T. Csta k-12 computer science standards.

[72] Foriek, M. Pushing the boundary of programming contests. *Olympiads in Informatics 7* (01 2013), 23–35.

[73] Froyd, J. E., Henderson, C., Friedrichsen, D., Khatri, R., and Stanford, C. From dissemination to propagation: A new paradigm for education developers.

[74] Gal-Ezer, J., and Stephenson, C. A tale of two countries: Successes and challenges in k-12 computer science education in israel and the united states. *ACM Transactions on Computing Education 14* (06 2014).

[75] Gansner, E., and North, S. An open graph visualization system and its applications to software engineering. *Software: Practice and Experience 30* (03 2003).

[76] Google. Google blockly.

[77] Google; GALLUP. Searching for computer science: Access and barriers in u.s. k-12 education.

[78] Google; GALLUP. Diversity gaps in computer science: Exploring the underrepresentation of girls, blacks and hispanics.

[79] Google; GALLUP. Trends in the state of computer science in u.s. k-12 schools.

[80] Green, T. Places of inequality, places of possibility: Mapping opportunity in geography across urban school-communities. *The Urban Review 47* (08 2015), 717–741.

[81] Grover, S., Pea, R., and Cooper, S. Remedying misperceptions of computer science among middle school students. pp. 343–348.

[82] Guzdial, M. Georgia computes!: an alliance to broaden participation across the state of georgia. *ACM Inroads 3* (12 2012), 86–89.

[83] Guzdial, M. Measuring demographics and performance in computer science education at a nationwide scale using ap cs data. pp. 217–222.

[84] Guzdial, M., and Biggers, M. A model for improving secondary cs education. vol. 37, pp. 332–336.

[85] Guzdial, M., and Biggers, M. Improving secondary cs education: progress and problems. vol. 39, pp. 298–301.

[86] Guzdial, M., Ericson, B., Mcklin, T., and Engelman, S. A statewide survey on computing education pathways and influences: Factors in broadening participation in computing. *ICER'12 - Proceedings of the 9th Annual International Conference on International Computing Education Research* (09 2012).

[87] Guzdial, M., Ericson, B., Mcklin, T., and Engelman, S. Georgia computes! an intervention in a us state, with formal and informal education in a policy context. *ACM Transactions on Computing Education 14*, 2 (2014).

[88] Glbahar, Y., Kert, S. B., and Kaleliolu, F. The self-efficacy perception scale for computational thinking skill: Validity and reliability study. *Turkish Journal of Computer and Mathematics Education (TURCOMAT) 10* (2019).

[89] Haden, P., Gasson, J., Wood, K., and Parsons, D. Can you learn to teach programming in two days? In *Proceedings of the Australasian Computer Science Week Multiconference* (New York, NY, USA, 2016), ACSW '16,

Association for Computing Machinery.

[90] Hakulinen, L. Survey on informatics competitions: Developing tasks. *Olympiads in Informatics 5* (01 2011), 12–25.

[91] Hart, M., Early, J. P., and Brylow, D. A novel approach to k-12 cs education. *Proceedings of the 39th SIGCSE Technical Symposium: Computer Science Education* (2008), 286 – 290.

[92] Havens, A. E. Diffusion of new seed varieties and its consequences: A colombian case.

[93] Henderson, C., Cole, R., Froyd, J., Friedrichson, D., Khatri, R., and Standford, C. Designing educational innovations for sustained adoption: A how-to guide for education developers who want to increase the impact of their work.

[94] Heo, G. Arduino-Compatible Modular Kit Design and Implementation for Programming Education. *Advances in Science, Technology and Engineering Systems Journal 5*, 5 (2020), 295–301.

[95] Hewner, M. Undergraduate conceptions of the field of computer science. pp. 107–114.

[96] Hillman, N. Geography of college opportunity: The case of education deserts. *American Educational Research Journal 53* (06 2016).

[97] Hillman, N., and Weichman, T. Education deserts: The continued significance of place in the twenty-first century. *Viewpoints: Voices from the Field* (2016).

[98] Hogrebe, M., and Tate, W. Geospatial perspective: Toward a visual political literacy project in education, health, and human services. *Review of Research in Education 36* (02 2012), 67–94.

[99] Holme, J., and Diem, S. Regional governance in education: A case study of the metro area learning community in omaha, nebraska. *Peabody Journal of Education 90* (01 2015), 156–177.

[100] Honig, W. Teaching and assessing programming fundamentals for non majors with visual programming. pp. 40–45.

[101] Hu, H. H., Heiner, C., and McCarthy, J. Deploying exploring computer science statewide. *Technical Symposium on Computer Science Education* (2016), 72 – 77.

[102] Hubwieser, P. Computer science education in secondary schools – the introduction of a new compulsory subject. *Trans. Comput. Educ. 12* (11 2012), 16:1–16:41.

[103] Hunpatin, O., OHare, C., Thomas, R., and Brylow, D. A browser-based ide for the muzecs platform. pp. 112–118.

[104] Idlbi, A. Taking kids into programming (contests) with scratch. 17–25.

[105] International Olympiad in Informatics. International olympiad in informatics.

[106] Judd, Dennis, R., and Swanstrom, T. *City politics: The political economy of urban America*. Longman Press, 2010.

[107] Khenner, E., and Semakin, I. School subject informatics (computer science) in russia. *ACM Transactions on Computing Education 14* (06 2014), 1–10.

[108] Khoury, G. Csta certification committee report.

[109] Killen, H., Weintrop, D., and Garvin, M. Ap computer science principles' impact on the landscape of high school computer science using maryland as a model. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2019), SIGCSE '19, Association for Computing Machinery, p. 10601066.

[110] Kim, S., and Jeon, J. Programming lego mindstorms nxt with visual programming. pp. 2468 – 2472.

[111] Knobelsdorf, M., Magenheim, J., Brinda, T., Engbring, D., Humbert, L., Pasternak, A., Schroeder, U., Thomas, M., and Vahrenhold, J. Computer science education in north-rhine westphalia, germanya case study. *ACM Transactions on Computing Education 15* (04 2015), 1–22.

[112] Kurhila, J., and Hellas, A. A purposeful mooc to alleviate insufficient cs education in finnish schools. *ACM Transactions on Computing Education 15* (04 2015), 1–18.

[113] Landow, G. Victorian web.

[114] Lawson, B., Szajda, D., and Barnett, L. Introducing computer science in an integrated science course.

[115] Lee, S., Kim, J., and Lee, W. Analysis of factors affecting achievement in maker programming education in the age of wireless communication. *Wireless Personal Communications 93* (2017).

[116] Liu, J., Lin, C.-H., Hasson, E. P., and Barnett, Z. D. Introducing computer science to k-12 through a summer computing workshop for teachers. *Proceedings of the 42nd ACM Technical Symposium: Computer Science Education* (2011), 389 – 394.

[117] Liu, J., Lin, C.-H., Wilson, J., Hemmenway, D., Hasson, E., Barnett, Z., and Xu, Y. Making games a "snap" with stencyl: A summer computing workshop for k-12 teachers. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2014), SIGCSE '14, Association for Computing Machinery, p. 169174.

[118] Liu, J., Wilson, J., Hemmenway, D., Xu, Y., and Lin, C.-H. Oh snap! a one-week summer computing workshop for k-12 teachers. In *2015 10th International Conference on Computer Science Education (ICCSE)* (2015), pp. 663–668.

[119] Logan, J., Minca, E., and Adar, S. The geography of inequality: Why separate means unequal in american public schools. *Sociology of education 85* (06 2012).

[120] Margolis, J., Goode, J., and Chapman, G. An equity lens for scaling: A critical juncture for exploring computer science. *ACM Inroads 6*, 3 (2015), 58 – 66.

[121] Martin, C. Draw a computer scientist. *SIGCSE Bulletin 36* (12 2004), 11–12.

[122] McGee, S., McGee-Tekula, R., Duck, J., McGee, C., Dettori, L., Greenberg, R. I., Snow, E., Rutstein, D., Reed, D., Wilkerson, B., Yanek, D., Rasmussen, A. M., and Brylow, D. Equal outcomes 4 all: A study of student learning in ecs. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2018), SIGCSE '18, Association for Computing Machinery, p. 5055.

[123] Mcnally, M., Goldweber, M., Fagin, B., and Klassner, F. Do lego mindstorms robots have a future in cs education? vol. 38, pp. 61–62.

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

[125] Mentkowski, M. Learning that lasts: Integrating learning, development, and performance in college and beyond.

[126] Menzel, H. Public and private conformity under different conditions of acceptance in the group.

[127] MIT Media Lab. Scratch.

[128] Mitchell, A., Purchase, H., and Hamer, J. Computing science: what do pupils think? vol. 41, p. 353.

[129] Morreale, P., Goski, C., Jimenez, L., and Stewart-Gardiner, C. Measuring the impact of computational thinking workshops on high school teachers. *Journal of Computing Sciences in Colleges 27* (06 2012), 151–157.

[130] Morreale, P., Joiner, D., and Chang, G. Connecting undergraduate programs to high school students: Teacher workshops on computational thinking and computer science. *Journal of Computing Sciences in Colleges 25* (01

2010), 191–197.

[131] Myketiak, C., Curzon, P., Black, J., Mcowan, P., and Meagher, L. cs4fn: A flexible model for computer science outreach. 297–302.

[132] Nager, A., and Atkinson, R. The case for improving u.s. computer science education. *SSRN Electronic Journal* (01 2016).

[133] National Science Foundation. Women, minorities, and persons with disabilities in science and engineering: 2011 (nsf 11-309), 01 2004.

[134] National Science Foundation. Stem + computing partnerships (stem+c).

[135] National Science Foundation. Proposal and award policies and procedures guide, 06 2020.

[136] Newhall, T., Meeden, L., Danner, A., Soni, A., Ruiz, F., and Wicentowski, R. A support program for introductory cs courses that improves student performance and retains students from underrepresented groups. pp. 433–438.

[137] Nolan, K., Faherty, R., Quille, K., Becker, B. A., and Bergin, S. Developing an inclusive k-12 outreach model. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (New York, NY, USA, 2020), ITiCSE '20, Association for Computing Machinery, p. 145151.

[138] Oliveira, O. L., Nicoletti, M. C., and del Val Cura, L. M. Quantitative correlation between ability to compute and student performance in a primary school. In *Proceedings of the 45th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2014), SIGCSE '14, Association for Computing Machinery, p. 505510.

[139] Park, G.-M., Kim, S.-H., Hwang, H.-R., and Cho, H.-G. Complex system analysis of social networks extracted from literary fictions. *International Journal of Machine Learning and Computing* (01 2013), 107–111.

[140] Passey, D. Computer science (cs) in the compulsory education curriculum: Implications for future research. *Education and Information Technologies 22* (03 2017).

[141] Petrovi, P. Ten years of creative robotics contests. *In Proceedings of Select Papers, ISSEP 2011: 5th International Conference on Informatics in Schools: Situation, Evolution and Perspectives* (10 2011), 201–212.

[142] Pickles, J. Geography gis and the surveillant society. *Papers & Proceedings of Applied Geography Conferences - State University of New York at Binghamton 14* (01 1991), 80–91.

[143] Pohl, W. Computer science contests for secondary school students: Approaches to classification. *Informatics in Education 5* (04 2006), 125–132.

[144] Pokorny, K., and White, N. Computational thinking outreach: reaching across the k-12 curriculum. *Journal of Computing Sciences in Colleges 27* (05 2012), 234–242.

[145] Pontier, D. A. Kansas high school computer science teachers' professional development, 1996.

[146] Prieto-Rodriguez, E., and Berretta, R. Digital technology teachers' perceptions of computer science: It is not all about programming. In *2014 IEEE Frontiers in Education Conference (FIE) Proceedings* (2014), pp. 1–5.

[147] Project Gutenberg. Project gutenberg archive.

[148] Qualls, J., and Sherrell, L. Why computational thinking should be integrated into the curriculum. *Journal of Computing Sciences in Colleges 25* (05 2010), 66–71.

[149] Ragonis, N. Type of questions - the case of computer science. *Olympiads in Informatics 6* (01 2012), 115–132.

[150] Raman, R., Venkatasubramanian, S., Achuthan, K., and Nedungadi, P. Computer science (cs) education in indian schools. *ACM Transactions on Computing Education 15* (05 2015), 1–36.

[151] Reades, J., and Rey, S. J. Geographical python teaching resources: geopyter. *Journal of Geographical Systems* (2021).

[152] Resnick, M., Maloney, J., Monroy-Hernndez, A., Rusk, N., Eastmond, E., Brennan, K., Millner, A., Rosenbaum, E., Silver, J., Silverman, B., and Kafai, Y. Scratch: Programming for all. *Commun. ACM 52* (11 2009), 60–67.

[153] Roberts, E., and Halopoff, G. Results of the 2005 csta national secondary computer science survey: 2005 survey analysis.

[154] Rogers, E. M. Communication and development: The passing of the dominant paradigm.

[155] Rogers, E. M. *Diffusion of Innovations*, 5 ed. Free Press, New York, NY, 2003.

[156] Rogers, E. M., and Shoemaker, F. F. *Communication of Innovations: A Cross-Cultural Approach*. Free Press, 1971.

[157] Rolandsson, L., and Skogh, I.-B. Programming in school: Look back to move forward. *ACM Transactions on Computing Education 14* (06 2014), 12:2.

[158] Rosenbaum, J., Reynolds, L., and Deluca, S. How do places matter? the geography of opportunity, self-efcacy and a look inside the black box of residential mobility. *Housing Studies 17* (01 2002), 71–82.

[159] Schunk, D. Learning theories: An educational perspective.

[160] Simmonds, J., Gutierrez, F. J., Casanova, C., Sotomayor, C., and Hitschfeld, N. A teacher workshop for introducing computational thinking in rural and vulnerable environments. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education* (New York, NY, USA, 2019), SIGCSE '19, Association for Computing Machinery, p. 11431149.

[161] Simmonds, J., Gutierrez, F. J., Meza, F., Torrent, C., and Villalobos, J. *Changing Teacher Perceptions about Computational Thinking in*

Grades 1-6, through a National Training Program. Association for Computing Machinery, New York, NY, USA, 2021, p. 260266.

[162] Siraj, A., Kosa, M., and Olmstead, S.-M. Weaving a tapestry: Creating a satellite workshop to support hs cs teachers in attracting and engaging students.

[163] Soja, E. W. *Seeking spatial justice.* Globalization and community series. University of Minnesota Press, 2010.

[164] Squires, G., and Kubrin, C. Privileged places: Race, uneven development and the geography of opportunity in urban america. *April 42* (08 2004).

[165] Stage, E., Asturias, H., Cheuk, T., Daro, P., and Hampton, S. Opportunities and challenges in next generation standards. *Science (New York, N.Y.) 340* (04 2013), 276–277.

[166] Stenhouse, L. An introduction to curriculum research and development.

[167] Tate, W., Jones, B., Thorne-Wallington, E., and Hogrebe, M. Science and the city thinking geospatially about opportunity to learn. *Urban Education 47* (03 2012), 399–433.

[168] Taylor, C., Spacco, J., Bunde, D. P., Butler, Z., Bort, H., Hovey, C. L., Maiorana, F., and Zeume, T. Propagating the adoption of cs educational innovations. In *Proceedings Companion of the 23rd Annual ACM Conference on Innovation and Technology in Computer Science Education* (New York, NY, USA, 2018), ITiCSE 2018 Companion, Association for Computing Machinery, p. 217235.

[169] Thompson, D., and Bell, T. Adoption of new computer science high school standards by new zealand teachers. In *Proceedings of the 8th Workshop in Primary and Secondary Computing Education* (New York, NY, USA, 2013), WiPSE '13, Association for Computing Machinery, p. 8790.

[170] Underwood, T., Black, M., Auvil, L., and Capitanu, B. Mapping mutable genres in structurally complex volumes. *Proceedings - 2013 IEEE International Conference on Big Data, Big Data 2013* (09 2013).

[171] University of California at Berkeley. Snap (build your own blocks).

[172] University of Canterbury. Cs unplugged: Computer science without a computer.

[173] USA Computing Olympiad. Usa computing olympiad.

[174] Van Dyne, M., and Braun, J. Effectiveness of a computational thinking (cs0) course on student analytical skills. pp. 133–138.

[175] Van Wart, S. J. Computer science meets social studies: Embedding cs in the study of locally grounded civic issues. In *Proceedings of the Eleventh Annual International Conference on International Computing Education Research* (New York, NY, USA, 2015), ICER '15, Association for Computing Machinery, p. 281282.

[176] Vogal, R., and Harrigan, J. *Political change in the metropolis*. Pearson Education, 2007.

[177] Voigt, J., Bell, T., and Aspvall, B. Competition-style programming problems for computer science unplugged activities. *A new learning paradigm: competition supported by technology* (2010), 207–234.

[178] Walden, J., Doyle, M., Garns, R., and Hart, Z. An informatics perspective on computational thinking. pp. 4–9.

[179] Wang, J., Brylow, D., and Perouli, D. Implementing cybersecurity into the wisconsin k-12 classroom. In *2019 IEEE 43rd Annual Computer Software and Applications Conference (COMPSAC)* (2019), vol. 2, pp. 312–317.

[180] Wilson, C., Sudol, L., Stephenson, C., and Stehlik, M. *Running on Empty: the Failure to Teach K–12 Computer Science in the Digital Age*. 10 2010.

[181] Wing, J. Computational thinking. *Commun. ACM 49* (03 2006), 33–35.

[182] Wing, J. Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences* (10 2008).

[183] Wisconsin Department of Public Instruction. Academic standards.

[184] Wisconsin Department of Public Instruction. Computer science guidance.

[185] Wisconsin Department of Public Instruction. Economically disadvantaged status/food services eligibility.

[186] Wisconsin Department of Public Instruction. K-12 computer science education in wisconsin.

[187] Wisconsin Department of Public Instruction. What is computer science in wisconsin?

[188] Wohl, B., Porter, B., and Clinch, S. Teaching computer science to 5-7 year-olds: An initial study with scratch, cubelets and unplugged computing. In *Proceedings of the Workshop in Primary and Secondary Computing Education* (New York, NY, USA, 2015), WiPSCE '15, Association for Computing Machinery, p. 5560.

[189] Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S., and Korb, J. Introducing computational thinking in education courses. 465–470.

[190] Yardi, S., and Bruckman, A. What is computing?: Bridging the gap between teenagers' perceptions and graduate students' experiences. *Third International Computing Education Research Workshop, ICER'07* (06 2021).

[191] Zinth, J. Computer science in high school graduation requirements. *Education Commission of the States* (2016).

[192] Zug, M., Hoffman, H., Kobayashi, F., President, M., and Dodds, Z. Cs for all academic identities. *J. Comput. Sci. Coll. 33*, 4 (Apr. 2018), 130137.