Infusing Computational Thinking into Middle Grade Science Classrooms: Lessons Learned

Veronica Cateté[1], Nicholas Lytle[1], Yihuan Dong[1], Danielle Boulden[1], Bita Akram[1], Jennifer Houchins[1], Tiffany Barnes[1], Eric Wiebe[1], James Lester[1], Bradford Mott[1], Kristy Boyer[2]
[1]North Carolina State University, Raleigh, North Carolina
[2]University of Florida, Gainesville, Florida
{vmcatete,nalytle,ydong2,dmboulde,bakram,jkhouci,tmbarnes,wiebe,lester,bwmott}@ncsu.edu,keboyer@ufl.edu

ABSTRACT

There is a growing need to present all students with an opportunity to learn computer science and computational thinking (CT) skills during their primary and secondary education. Traditionally, these opportunities are available outside of the core curriculum as stand-alone courses often taken by those with preparatory privilege. Researchers have identified the need to integrate CT into core classes to provide equitable access to these critical skills. We have worked in a research-practice partnership with two magnet middle schools focused on digital sciences to develop and implement computational thinking into life sciences classes. In this report, we present initial lessons learned while conducting our design-based implementation research on integrating computational thinking into middle school science classes. These case studies suggest that several factors including teacher engagement, teacher attitudes, student prior experience with CS/CT, and curriculum design can all impact student engagement in integrated science-CT lessons.

CCS CONCEPTS

• **Social and professional topics → K-12 education**; **Computational thinking**;

KEYWORDS

Professional Development; STEM+C; Computational Thinking

ACM Reference Format:
Veronica Cateté, Nicholas Lytle, Yihuan Dong, Danielle Boulden, Bita Akram, Jennifer Houchins, Tiffany Barnes, Eric Wiebe, James Lester, Bradford Mott, Kristy Boyer. 2018. Infusing Computational Thinking into Middle Grade Science Classrooms: Lessons Learned. In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education (WiPSCE '18), October 4–6, 2018, Potsdam, Germany*. ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3265757.3265778

1 INTRODUCTION

Global economic challenges and the changing context of high-value employment in the U.S. has meant strong policy pressure to prepare a workforce that is computationally literate. This has resulted in increased interest in how best to integrate computer science (CS) in K-12 education. Until recently, a primary approach has been to position CS as out-of-class experiences or as stand-alone units [26, 31]. However, there has been emerging recognition for the need to integrate foundational CS skills (e.g. designing an algorithm, generating abstractions, etc.) into core academic Science, Technology, Engineering, and Mathematics (STEM) classes[13]. These foundational elements have been termed computational thinking (CT) [20].

Through integrating CT into the regular K-12 school day, all students can gain access and the playing field can be leveled for those with less programming experience [11, 21]. In order to create and pilot a series of CT activities for the core classroom, we partnered with two local middle schools (student ages 11-13), one recently designated as a "Center for the Digital Sciences Magnet School." We worked in partnership with school leadership and teachers to integrate CS/CT concepts into math and science classes.

In this paper we discuss our design-based implementation research in collaboration with middle school Science teachers. Through multiple cycles of inspection, prototyping, and piloting our team developed and tested two sets of week-long CT activities for life science. After a brief introduction to the STEM+Computing field, we present our case study, followed by a set of lessons learned and recommendations for infusing CT lessons into core content areas.

2 BACKGROUND

Historically, computing education disproportionately engages students with access to the "preparatory privilege" of extra experience in STEM and computer science [23]. The equity issues of a standalone computing course are compounded by additional systemic issues within the K-12 school structure. Few teachers have available room in their schedules to teach an additional computing course, there are limited classroom resources available within the schools, and there are a limited number of computing-certified teachers available. To truly address under-representation and equal access, efforts must be made to integrate computational thinking (CT) and computer science into the required K-12 curriculum [4] where all students will have access, and the playing field can be leveled for those with less experience [28].

Recognizing that this integration will take a different form than standalone courses, the framing of CT as a set of practices has been codified [8, 29] into Decomposition, Pattern Recognition, Abstraction, and Algorithmic Design (discussed further in Table 1).

The CT framework can be used as a guide for CT integration into STEM disciplinary courses [4, 5, 16, 22, 27]. CT and the use of computational tools has been shown to enable deeper learning of STEM content areas for students [10, 18, 32]. These foundational

WiPSCE '18, October 4–6, 2018, Potsdam, Germany

ACM ISBN 978-1-4503-6588-8/18/10…$15.00
https://doi.org/10.1145/3265757.3265778

Table 1: Elements of Computational Thinking [8]

Element of CT	Expanded Explanation
Decomposition	Breaking down data, processes, or problems into smaller, manageable parts
Pattern Recognition	Observing patterns, trends, and regularities in data
Abstraction	Identifying the general principles that generate these patterns
Algorithmic Design	Developing the step by step instructions for solving this and similar problems

computing and CT abilities have also increased student retention and learning performance in computing courses and outreach programs [6, 14, 19]. Middle grades has been identified as a critical age range to study the potential for developing CT. Some attempts have designed curricula using block-based programming languages [12, 17], as visual programming environments have been shown to improve student performance and affect [24]. However, as not every classroom has access to computers, other research has focused on developing unplugged CS [2]. Through designing STEM-oriented block-based programming and unplugged activities we can further integrate CT into the K-12 school day.

This new approach is critical as there is a growing need to educate students in computational methods and techniques to support the rapidly changing landscape of research across mathematics and scientific disciplines [30]. Today, biological and environmental data, for example, is multivariate, multidimensional, and multi-causal, and it exists at multiple scales in enormous volume (increasing at terabytes of data per day) [10]. These non-linear problems require computationally powered solutions envisioned by CT.

Infusing CT directly into a STEM course provides the teacher with improved mastery of their disciplinary concepts with new instructional approaches [29]. In courses where students pose and solve open-ended integrated STEM+C problems, teachers lose much of the control they traditionally have over the learning process and may become uncomfortable [10]. They need new skills to guide individual learners. Supporting students engaging in self-directed collaborative processes requires an ability to diagnose difficulties and give hints, rather than supplying solutions. Research shows that targeted professional development (PD) on facilitating CT and creative problem solving, can increase teachers confidence in ability to teach content and guide students [25].

Our work at this middle school is modeled as a research practice partnership [7]. We observe teachers in their natural classrooms, collaborate together to develop CT enriched curricula, provide scaffolded PD, and conduct field observations and data collection to improve the materials, and reiterate the cycle.

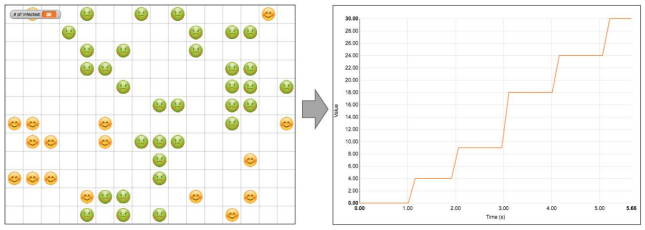
3 INTEGRATING CT AND SCIENCE

3.1 Curricula

In this case study we describe implementations of two CT-integrated life science curricula. The Epidemics curriculum focused on modeling the spread of epidemic diseases such as the flu and the Invasive Species curriculum focused on data manipulation and predicting the presence of invasive species in a given area. Each of these curricula was developed by a multidisciplinary team of computer science researchers, educational psychology researchers, and classroom teachers. Individual activities are described as “plugged" if students programmed or used a computer modeling tool, or “unplugged" if students were learning CT/STEM topics without a computer. Additionally, tutorials on block-based programming were given to students as part of the developed curricula. Both curricula were developed to be standard aligned with Essential Science Standards and the k12cs.org Computational Thinking Framework.

3.1.1 Epidemics. This curriculum features 2 unplugged days focused on modeling agent-host relations in the transmission of diseases and 4 plugged days where students developed a simulator using the block-based programming environment, Cellular [1]. This culminated in students using their simulator to answer research questions on what environmental factors influence the rate of infection across a population. An overview of the day’s topics, assignments, and aligned CT curricula are shown in Table 3.

Figure 1: Epidemics: A run of the simulation is shown (left) with graphical output showing the rate of infection (right)



3.1.2 Invasive Species. Students learned about invasive species in ecosystems integrated with learning concepts of data science. Students investigated feature data for each state in the United States (average annual temperature, average rainfall, etc.) to make models for predicting whether an invasive species, Kudzu, is likely to be present in that state. Students completed this through unplugged activities designed to learn about properties and relationships of data and how to calculate error; a data modeling tool, CODAP [9]; and a block-based programming tool, Netsblox [3]. The sequence of activities for this assignment is shown in Table 5.

4 METHODS

4.1 Context

The case study took place within two local area magnet middle schools, which provide innovative and/or expanded educational opportunities while promoting diverse populations and reducing high concentrations of poverty. Demographics for these schools are provided in Table 2. For both schools, over 50% of the population are racially and ethnically underrepresented minorities in STEM and nearly half are on a Free or Reduced Payment Lunch (FRPL) program (a financial aid program for low SES American families).

Table 2: Demographics for case study locations.

Location	White	Asian	Black	Hispanic	FRPL
School 1	22.0%	2.6%	42.3%	28.1%	62.2%
School 2	39.5%	5.9%	22.5%	25.8%	47.1%

Table 3: Epidemics curricula outline, P: programming, U: unplugged

Day	Lab Description	Science Outcomes	CS Outcomes	CT Outcomes
1	U: Simulate epidemics in-person and create agent diagrams (healthy/sick)	Model real life factors in epidemics.	Understand Hosts and Agents share properties with modified values.	Abstraction
2	P: Animate healthy and infected agents including basic transmission	Define Infection and infection rate; Reinforce day 1.	Demonstrate understanding of agent properties. Use loops and conditionals to automate life cycles.	Decomposition, Algorithm
3	P: Extend Day 2 to track how populations get sick over time; generate graphs based on initial conditions	Understand disease spread and rate of transmission/infection.	Use variables to maintain count. Analyze trends in data to identify patterns. Demonstrate an understanding of how interaction properties can affect simulations.	Pattern Recognition
4	U: Use a transition diagram to model interactions and variable changes with 4 health states	Understand Morbidity/Mortality rates and their influence on spread.	Understand and use Finite State Machines to model complex logic flow. Model algorithmic thinking through transition modeling.	Abstraction, Algorithm
5	P: Use given simulation to explore effect of environmental factors on disease spread.	Understand how environmental factors affect disease spread; Learn experimental procedure for hypothesis testing	Data visualization; Using simulations to test hypothesis	Pattern Recognition, Decomposition

The breakdown of curriculum implementation is shown in Table 4; each class contained 20-25 students. All teachers led instruction on days where computers weren't needed. Each teacher at School 2 led instruction on computer days for 4 of their afternoon classes.

Table 4: Implementation profile of activities and teachers.

	Topic	Teachers	Classes	PD	Personality
School 1	Epidemics	3	6 (7th)	No	Disinterest
School 2	Epidemics	2	12 (8th)	Yes	Proactive
	Invasive	2	10 (7th)	Yes	Disinterest

4.2 Teachers

Teachers self-enrolled their classes for a programming study ran by a subset of the research team. After that study, teachers were offered several other computing activities for their students. Teachers were able to choose which activities they wanted to implement in their classroom; School 1 teachers took a vote based on which seemed most interesting, School 2 teachers choose those that aligned with the pre-existing science learning objectives for their grade level.

Researchers offered a short professional development beforehand to all classroom teachers for each of the plugged activities. PD occurred during two of the teachers' 1-hour planning periods. The PD covered activity content, teacher instruction guides, and scaffolded student instructions. The four School 2 teachers eagerly opted to partake in PD for the activities. The three School 1 teachers declined PD, citing limited time availability and uncertainty in if they would reimplement the activity the following semester.

4.3 Classroom Implementation

Each of the teachers' classes were presented the same curriculum activity on the same day. The teachers primarily taught all unplugged activities as they were strongly oriented in the science domain. For plugged activities, the research team provided each classroom with an instructional assistant to help lead the topic. Additionally, when a teacher indicated less confidence, a third support person attended the session to help answer student questions.

For all School 2 implementations of Epidemics (and one implementation of Invasive species), a faded scaffolding teaching approach was carried out. In order to help strengthen the teachers, a member of the research team with former middle school teaching experience, led the first class period in the activity while the science teacher observed. In the second class period, the teacher co-taught with our team member to get supported hands-on teaching experience for the CT activities. By the third class period, the teacher would lead the full activity while the researcher moved into an observation-only role. The teacher then taught the remaining three afternoon class periods on their own.

4.4 Data Collection

Data was collected in a variety of forms. During the run of the activities, a member of the research team acted as an observer recording classroom behavior. These observers also conducted cognitive interviews with select students based on observed proficiency level and performance on the activity. These interviews typically consisted of 2 - 3 questions that helped elicit student thinking processes when approaching an assigned problem. We also collected student interaction data with the programming environments, which we will elaborate more on in further research.

After each implementation cycle was complete, the full research team discussed the findings from the in-class observations, debriefings with teachers, and cognitive interviews. The findings were then translated into actionable changes to the curriculum for later iterations. We present several of our key findings below.

Table 5: Invasive Species Curricula outline. P: programming, U: Unplugged

Day	Lab Description	Science Outcomes	CS Outcomes	CT Outcomes
1	U: Inspect/Compare State environmental data to find patterns for presence of Invasive Species	Define invasive species and how they are established. Define Abiotic factors and their influence on invasive species. Understand dependent/ independent variables.	Data Representation, Data Analysis.	Abstraction, Pattern Recognition
2	P: Visualize data to find relationships between variables and use decision trees to create predictive models.	Visualize relationships between dependent and independent variables.	Modeling, Simulation.	Abstraction, Pattern Recognition
3	P: Implement the prediction model from day 2 in NetsBlox and test the model.	Reinforce and Transfer Day 2 Concepts.	Simulation, Automation, Prediction, Problem Representation.	Algorithm, Abstraction
4	U: Calculate the precision of prediction models against the dataset.	Learn to calculate error, inspect complex relationships b/t variables.	Prediction, Validation, Modeling, Simulation.	Algorithm

5 LESSONS LEARNED

5.1 Professional Development

Teacher engagement is critical. As noted in the case studies, not all teachers were receptive to the offered professional development. We found teachers fell into three main categories: (1) Proactive: wanting to learn CS/CT skills and concepts, (2) Willing: wanting to incorporate CT but needing help, and (3) Disinterested: only wanting the research team to teach their class. The 8th grade science teachers fell into the first two categories, engaging fully in the PD and more fully in leading the integrated curricula. The 7th grade science teachers from School 2 and the three elective teachers from School 1, were in categories 2 and 3, with even “Willing" teachers being hesitant to lead the planned lessons and “Disinterested" teachers unwilling to attend even our lightweight PD. Based on our classroom observations, we found that the level of teacher buy-in seemed to be strongly related to overall student engagement, with students in “Proactive teacher classrooms" having much higher on-task than off-task behavior.

Teachers should experience new CT-integrated lessons as learners. We conducted our short professional development (PD) sessions by giving a brief overview of the planned lessons and provided teachers with links to the curricula and solutions to each activity. Nearly all teachers felt comfortable leading unplugged sections; as domain experts, science teachers already have developed schema for the scientific concepts. However, several were uncomfortable leading plugged activities. Teachers did learn CT and programming while following along with students, and their understanding of the activities became deeper as they co-taught the second class and lead the third class. However, disinterested teachers never engaged in learning, and the researchers had to lead all of the plugged activities.

Teachers need help becoming facilitators for CT. Observers found the teaching quality to be varied, even among highly engaged proactive teachers. The proactive teachers intentionally inserted their own errors and had students problem solve to in order to debug their algorithms for solutions. Instead of trying to be a person who knew everything about the curriculum, these teachers identified as students learning the curriculum together. In early development, teachers expressed the need for a solution key in the teaching guide to make them more comfortable teaching. This key then became a gold-standard for all the activities to make sure nothing went wrong in the following activities. However, even our most engaged, proactive teacher once stated, ‘if your code doesn’t look like mine, it is wrong.’ This means that the research team did not effectively communicate that student solutions could and should vary. Several instances of this issue occurred when determining the order of setting variables or the order of simple conditional statements. For a teacher new to CT and programming, it is not obvious when the ordering of tasks doesn’t matter. This is understandable, as that the lack of previous programming experience makes it difficult for the teachers to identify alternative solutions. Teachers feared that one difference between the student solution and the teacher solution might lead to errors or further differences in the subsequent activities. We believe that teacher preparation should focus on helping teachers become facilitators rather than CT experts. This could be realized through adopting strategies from POGIL (process-oriented guided inquiry learning) or problem-based learning, that focus on helping teachers learn to provide process feedback and ask good questions to help students learn.

5.2 Curricular Materials

Teachers need explicit direction and help to identify computational thinking elements. When designing the curricula, the research team initially considered computational thinking as an integral mindset and set of practices to be embedded implicitly into the activities. However, the teachers seemed to view computational thinking as a set of discrete concepts based on their school or district definitions. For example, School 2 utilized the framework for computational thinking as Pattern Recognition, Abstraction, Decomposition, and Algorithms (PRADA). The teachers tended to explain and emphasize these keywords repeatedly during the activities. School 2 teachers expected these keywords to be explicitly identified in the teacher guide, and asked us to help them map the integrated activities to their PRADA CT concepts. This suggests that explicitly identifying the targeted, and even localized, computational thinking concepts within the curricular materials may help teachers become more confident and effective in helping students learn computational thinking in integrated settings.

Offer extensions and support to engage learners at different levels. Observations showed that students who moved quickly through the tasks in the activities would use their extra time to go off-task or distract other students. In order to keep the materials engaging for these more experienced/advanced students, we built extension activities with questions to guide deeper exploration of both science and CT concepts. To aid students with lower prior experience, we provided a self-paced student instruction guide with scaffolded coding segments, visual gif depictions, and text-based think-it-through activities. These multifaceted support tools allowed most students to stay on task and arrive similarly prepared for the next day's learning objectives. While the curriculum successfully built concepts up over multiple days of instruction, activities that build on previous days can prove difficult if students miss days in the sequence or are unable to complete some of a given day's activities. In the future, we should explicitly provide pre-made starting points for each day to help with this.

Scaffold and minimize switching between computer tools. Integrating STEM and CT with authentic activities for both is likely to require switching between text, paper-based activities, and multiple computer-based tools. Keeping track of where to find each tool is extraneous cognitive load that wastes valuable class time. Both teachers and students need a single place to find links for the lesson plans, online tools, and programming environments. In addition, teachers and students need enough time to feel comfortable using each tool/software. Limiting the number of tools needed for each lesson is important, and the each new tool should be carefully considered to ensure that the benefits for its use should outweigh the time and effort needed to switch contexts and learn a new interface.

Our observations showed strong student engagement in the Epidemics curriculum, with few students going off task. We provided three supports that were important in making the Epidemics curriculum effective: (1) teacher PD on the curriculum during two planning periods, (2) an in-class tutorial on block-based programming prior to the curriculum, and (3) extensions we made to the Cellular programming environment to minimize the number of new things students had to learn to achieve the targeted goals. We believe these allowed students enough time and support to learn just enough about the tools, and also helped teachers learn along with them. Teachers benefited from the same task and environment, and were comfortable with switching over to teaching by themselves by the 3rd period. We were able to fade scaffolding for both unplugged and plugged Epidemics activities. Teacher feedback was strongly positive, and the teachers agreed that they would feel comfortable doing the task again, and wished to implement it in their classrooms again next year.

The Invasive Species curriculum implementation was not as successful. Similar to Epidemics, we provided just-in-time PD during teacher planning periods, asked teachers to provide in-class time for the block-based programming tutorial, and developed scaffolds in the Netsblox environment for the activities. However, we were unable to fade the researcher classroom support completely for the plugged activities. Several factors contributed to this result. The 7th grade science teachers (either Disinterested or Willing) entered the PD with much more hesitation towards using computational tools. Not enough classroom time was provided for the block-based programming tutorial, and our curriculum included the use of two computational tools (CODAP and Netsblox). The short tutorial left some students without prior experience struggling, and the teachers felt uncomfortable teaching the plugged activities. Consequently, we could only implement gradual fading of teaching for the unplugged activities in the Invasive Species curriculum. Though it is unclear what the most important factor is, the combination of low teacher confidence, frequent switching between multiple environments, and the short duration of the block-based programming tutorial made it unlikely that teachers will implement this lesson again without researcher support.

6 DISCUSSION & CONCLUSIONS

We found that the largest differentiator in both teacher and student engagement in our integrated science-CT curricula was the degree to which teachers embraced a learner and facilitator role. Teachers who became comfortable with open-ended assignments with diverse solutions, and who acted as questioners, were able to promote student engagement in the curricula. Accepting that there are many ways that people can use computers to achieve the same results is central to computational thinking and computer science. This may be natural for computer science and education researchers comfortable with technology, but it cannot be assumed that teachers will readily adopt a facilitator role without explicit help and support to do so. As development environments and languages are constantly changing, teachers cannot be expected to become experts in all possible computational tools or programming environments. Teachers need to be able to act as facilitators for open-ended computational thinking, able to address and scaffold learning without a definitive solution guide. This might run counter to domains with specific correct solutions or styles of teaching that are more teacher-led rather than student-focused, but we believe that people must learn computational thinking by making and doing, breaking things and fixing them, through problem solving and creativity.

6.1 Limitations

Though we were able to provide instructional assistance on programming days, we did not have consistent researcher instructors in the classroom during these multi-day activities. Nuances in teaching style among these researchers could have created differences in both student and teacher engagement, as well as student and teacher learning of the material. While our scaffolded teaching model in the classroom was effective for some teachers, it is not feasible for large-scale implementation, so more work needs to be done to create scalable training strategies.

There are also limitations in the chosen schools for our pilot implementation. Both schools are magnet schools specifically focused on integrating digital sciences and CT. Integrating CT will likely be even more challenging in general classrooms. We only report on the integration of CT in the context of a life science class. However, we have piloted additional curricula with both Mathematics and Art classes and have found that these classrooms and topics present different challenges and considerations. Math teachers also wanted help identifying CT concepts, but this is more difficult because of the considerable overlap between math and CT concepts. The art teacher was used to open-ended tasks and connecting concepts to

diverse student artifacts, so she already had the facilitator mindset important for integrating CT.

6.2 Future Work

We plan to address these lessons learned through multiple strategies moving forward. First, we plan to redesign our professional development to provide teachers with experience as CT learners, observers, and teachers [15]. The faded structure of our initial model allowed teachers to observe teaching by the research team, and to act as teachers in their disciplinary area first and later in CT. However, only those teachers who actively engaged in the PD and helped students during the CT activities could actually experience the materials as "learners". In a week-long summer PD, we plan to provide teachers time to experience these lessons with peers as learners, and build in sufficient time for them to discuss what aspects of each activity relate to disciplinary concepts and which relate to CT. There is no pre-set right answer to this question - many CT concepts can apply at once to a single activity. However, we believe that through discussion, the research team and teachers together can discover ways to identify productive instances of CT and highlight how they are different from, but contribute to, meaningful learning both for CT and for STEM disciplines. The focus of the summer PD will also differ from the PD sessions prior to the implementations, as we will not be training these teachers in specific content. Instead, we plan to train teachers to facilitate learning in various CT lessons and teach strategies for integration, teaching, and development of their own lessons, learning how to match CT concepts to their own content. Through this, we hope to develop ways that help teachers become fluent in computational thinking and in integrating computing ideas into their specific domains, classrooms, and lessons.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant numbers 1640141 and 1742351. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] Bernd Meyer Aidan Lane and Jonathan Mullins. 2012. *Simulation with Cellular A Project Based Introduction to Programming* (first ed.). Monash University. Online: https://github.com/MonashAlexandria/snapapps.

[2] Tim Bell, Jason Alexander, Isaac Freeman, and Mick Grimley. 2009. Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology* 13, 1 (2009), 20–29.

[3] Brian Broll, Akos Lédeczi, Peter Volgyesi, Janos Sallai, Miklos Maroti, Alexia Carrillo, Stephanie L Weeden-Wright, Chris Vanags, Joshua D Swartz, and Melvin Lu. 2017. A visual programming environment for learning distributed programming. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, 81–86.

[4] Philip S. Buffum, Megan Hardy Frankosky, Kristy Elizabeth Boyer, Eric N Wiebe, Bradford W Mott, and James C Lester. 2016. Empowering All Students: Closing the CS Confidence Gap with an In-School Initiative for Middle School Students. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 382–387.

[5] Philip S. Buffum, Allison G Martinez-Arocho, Megan Hardy Frankosky, Fernando J Rodriguez, Eric N Wiebe, and Kristy Elizabeth Boyer. 2014. CS principles goes to middle school: learning how to teach Big Data. In *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 151–156.

[6] Veronica Cateté, Kathleen Wassell, and Tiffany Barnes. 2014. Use and development of entertainment technologies in after school STEM program. In *Proc. of the 45th ACM technical symposium on Computer science education*. ACM, 163–168.

[7] CE Coburn, WR Penuel, and K Geil. 2013. Research-practice partnerships at the district level: A new strategy for leveraging research for educational improvement. *Berkeley, CA and Boulder, CO: Univ. of California and Univ. of Colorado* (2013).

[8] K-12 Computer Science Framework Steering Committee et al. 2016. K-12 computer science framework. (2016).

[9] The Concord Consortium. 2014. Common online data analysis platform. *@Concord* 18, 1 (apr 2014), 16.

[10] National Research Council et al. 2011. *Successful K-12 STEM education: Identifying effective approaches in science, technology, engineering, and mathematics*. National Academies Press.

[11] Jan Cuny, Larry Snyder, and M Jeannette. 2010. Wing. *Demystifying Computational Thinking for Non-Computer Scientists, work in progress* (2010).

[12] Jill Denner, Linda Werner, and Eloy Ortiz. 2012. Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts? *Computers & Education* 58, 1 (2012), 240–249.

[13] Google (Firm) Gallup (Firm). 2016. Diversity gaps in computer science: exploring the underrepresentation of girls, Blacks and Hispanics. (2016).

[14] Dan Garcia, Brian Harvey, and Tiffany Barnes. 2015. The beauty and joy of computing. *ACM Inroads* 6, 4 (2015), 71–79.

[15] Joanna Goode, Jane Margolis, and Gail Chapman. 2014. Curriculum is not enough: The educational theory and research foundation of the exploring computer science professional development model. In *Proceedings of the 45th ACM technical symposium on Computer science education*. ACM, 493–498.

[16] Shuchi Grover and Roy Pea. 2013. Computational thinking in K–12: A review of the state of the field. *Educational Researcher* 42, 1 (2013), 38–43.

[17] Shuchi Grover, Roy Pea, and Stephen Cooper. 2016. Factors influencing computer science learning in middle school. In *Proceedings of the 47th ACM technical symposium on computing science education*. ACM, 552–557.

[18] Mark Guzdial. 1994. Software-realized scaffolding to facilitate programming for science learning. *Interactive Learning Environments* 4, 1 (1994), 001–044.

[19] Mark Guzdial and Barbara Ericson. 2012. Listening to linked lists: Using multimedia to learn data structures. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, 663–663.

[20] Peter B Henderson, Thomas J Cortina, and Jeannette M Wing. 2007. Computational thinking. *ACM SIGCSE Bulletin* 39, 1 (2007), 195–196.

[21] I Lee. 2016. Reclaiming the roots of CT. *CSTA Voice: The Voice of KâĂŞ12 Computer Science Education and Its Educators* 12, 1 (mar 2016), 3–4.

[22] Irene Lee, Fred Martin, and Katie Apone. 2014. Integrating computational thinking across the K–8 curriculum. *Acm Inroads* 5, 4 (2014), 64–71.

[23] Jane Margolis. 2010. *Stuck in the shallow end: Education, race, and computing*. MIT Press.

[24] Thomas W Price, Jennifer Albert, Veronica Catete, and Tiffany Barnes. 2015. BJC in action: Comparison of student perceptions of a computer science principles course. In *Research in Equity and Sustained Participation in Engineering, Computing, and Technology (RESPECT), 2015*. IEEE, 1–4.

[25] Thomas W Price, Veronica Cateté, Jennifer Albert, Tiffany Barnes, and Daniel D Garcia. 2016. Lessons Learned from BJC CS Principles Professional Development. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, 467–472.

[26] Susan H Rodger, Jenna Hayes, Gaetjens Lezin, Henry Qin, Deborah Nelson, Ruth Tucker, Mercedes Lopez, Stephen Cooper, Wanda Dann, and Don Slater. 2009. Engaging middle school teachers and students with alice in a diverse set of subjects. In *ACM SIGCSE Bulletin*, Vol. 41. ACM, 271–275.

[27] Cary Sneider, Chris Stephenson, Bruce Schafer, and Larry Flick. 2014. Computational thinking in high school science classrooms. *The Science Teacher* 81, 5 (2014), 53.

[28] Jennifer Tsan, Kristy Elizabeth Boyer, and Collin F Lynch. 2016. How Early Does the CS Gender Gap Emerge?: A Study of Collaborative Problem Solving in 5th Grade Computer Science. In *Proceedings of the 47th ACM technical symposium on computing science education*. ACM, 388–393.

[29] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2014. Defining computational thinking for science, technology, engineering, and math. In *American Educational Research Association Annual Meeting, Philadelphia, Pennsylvania*.

[30] David Weintrop and Uri Wilensky. 2015. To block or not to block, that is the question: students' perceptions of blocks-based programming. In *Proceedings of the 14th International Conference on Interaction Design and Children*. ACM, 199–208.

[31] Linda Werner, Jill Denner, Shannon Campe, Eloy Ortiz, Dawn DeLay, Amy C Hartl, and Brett Laursen. 2013. Pair programming for middle school students: does friendship influence academic outcomes?. In *Proceeding of the 44th ACM technical symposium on Computer science education*. ACM, 421–426.

[32] Uri Wilensky and Kenneth Reisman. 2006. Thinking like a wolf, a sheep, or a firefly: Learning biology through constructing and testing computational theories-an embodied modeling approach. *Cognition and instruction* 24, 2 (2006), 171–209.