# A Literature Review Examining Broadening Participation in Upper Elementary CS Education

Umar Shehzad
Utah State University
Logan, Utah, USA
umar.shehzad@usu.edu

Mimi Recker
Utah State University
Logan, Utah, USA
mimi.recker@usu.edu

Jody Clarke-Midura
Utah State University
Logan, Utah, USA
jody.clarke@usu.edu

## ABSTRACT

Despite proliferated efforts to integrate computer science in elementary education, there is a dearth of studies that synthesize the current state of CS education research in formal educational contexts, specifically in upper elementary classrooms. Further, while numerous studies have investigated approaches and strategies that broaden participation in computing, the majority of them focus on secondary and post-secondary settings. The present study uses a systematic literature review process to review research conducted with students in formal classroom settings in grades 4, 5, and 6 and published since 2013. We review the research through two questions: What are barriers to broadening participation in CS in upper elementary (grades 4-6)? What instructional approaches and strategies help broaden participation in CS in upper elementary (grades 4-6)? A systematic search of the literature highlighted approaches used for broadening participation, including using various teaching media, designing scaffolds in instruction, and integrating into other subject areas. We conclude by identifying gaps in the research and identifying areas for further research.

## CCS CONCEPTS

• **Social and professional topics → K-12 education**.

## KEYWORDS

Upper elementary CS, Computational thinking, CS Literature review, Broadening Participation in CS, Formal CS education

**ACM Reference Format:**
Umar Shehzad, Mimi Recker, and Jody Clarke-Midura. 2023. A Literature Review Examining Broadening Participation in Upper Elementary CS Education. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education V. 1 (SIGCSE 2023), March 15–18, 2023, Toronto, ON, Canada.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3545945.3569873

## 1 INTRODUCTION

Owing to the increasing demand for computing skills, computer science (CS) education has gained significant attention over the last few decades. As a result, research in the field of CS education has grown rapidly, bringing forth issues around the lack of equity and diversity in student participation rates in CS instruction. In response, various approaches of introducing learners to CS have been developed for use in formal education. For example, CS activities that aim to reduce the risks of the stereotypes associated with computing have been developed [10]. Others have argued for integrating CS into existing school curricula to avoid increasing the workload for teachers as well as the learners [45, 60]. At national levels, for example, the U.S. National Science Foundation (NSF) [13] and the Royal society [50] in the United Kingdom, support research on fostering equitable participation in CS education. Such efforts have resulted in a proliferation of new CS education research, creating a need to synthesize the current state of the field in terms of research findings.

## 2 RELATED WORK

There have been prior reviews of the CS education literature. Some focused on defining and elaborating computational thinking (CT) concepts [5, 53, 60]. These reviews synthesized findings of the current state of research (see Table 1) to capture the progress in this young and rapidly accelerating field [6, 20, 37]. However, the literature in the field is growing at an ever-increasing pace, renewing the need to review and synthesize recent research.

Lye & Koh [37] emphasized the need for research in naturalistic classrooms. Rich et al. [48] identified a need for grounding claims in student data. Buitrago Flórez et al. [6] recommended focusing on elementary and later grades as upper-elementary learners are especially receptive to new ideas [31]. While the present study builds on these suggestions, it also differs from existing review efforts in multiple ways. Upper elementary education presents a different set of challenges than the younger (K-3) and the later levels of schooling [47]. Furthermore, few reviews present a detailed view of CS in formal school settings, which compared to informal contexts, are less likely to be skewed towards students from more privileged backgrounds [9]. The field also lacks a systematic literature review focusing on the issues of broadening participation in CS at the K-8 level (see Table 1). Thus, the goal of the present review is to explore the research being conducted in formal CS education settings, specifically, upper elementary classrooms (grades 4-6). This review is guided by two research questions:

(1) What are barriers to broadening participation in CS in upper elementary (grades 4-6)?
(2) What instructional approaches and strategies help broaden participation in CS in upper elementary (grades 4-6)?

## 3 METHODS

In order to conduct this systematic review, we drew from Alexander [2], who defines systematic literature reviews as a search of literature that addresses an issue of importance to the field and answers an unanswered critical question. Steps in a systematic literature



*SIGCSE 2023, March 15–18, 2023, Toronto, ON, Canada*

ACM ISBN 978-1-4503-9431-4/23/03.
https://doi.org/10.1145/3545945.3569873

**Table 1: Prior Literature Reviews**

| Previous work | Age Focus | Goals of review |
|---|---|---|
| Brennan & Resnick (2012) [5] | 8-17 | Identifying dimensions of CT manifested in Scratch projects |
| Grover & Pea (2013) [20] | K-12 | Reviewing the state of discourse (in 2013) around CT |
| Lye & Koh (2014) [37] | K-12 | Categorizing literature based on Brennan & Resnick [5] |
| Weintrop et al. (2016) [60] | K-12 | Defining CT based on how it is integrated in Science and Math |
| Shute et al. (2017) [53] | K-12 | Reviewing how CT is assessed |
| Rich et al. (2017) [48] | K-8 | Reviewing CS learning goals |
| Buitrago Flórez et al. (2017) [6] | All levels | Reviewing how programming is being taught |
| Waite (2017) [58] | K-12 | Summarizing existing review papers and new studies |

review include: formulating a search approach and establishing inclusion criteria; consolidating and summarizing challenges, including charting the relevant characteristics of literature and creating informative groupings; and interpreting and communicating challenges. The final step involves identifying significant patterns and trends.

### 3.1 Sources of Literature

Several keywords and combinations of keywords (e.g., "computing" OR "computational thinking" AND "curriculum" AND "elementary" OR "K-6") were used to find literature in the following databases: Education Source, Educational Resources Information Center (ERIC) (via EBSCOhost), and the ACM Digital Library. SCOPUS was used to search within a set of influential scholarly journal databases.

### 3.2 Analysis of Fit

Our search resulted in a total of 1250 articles across all of the databases. The appropriateness of the articles was based on the inclusion criteria (Table 2). 1025 articles were deleted after reviewing the abstracts as they did not meet our criteria and 51 were dropped because they were duplicates (see Figure 1). Of the 174 articles, another 96 were deleted after a second round of reading where it was discovered they did not meet the inclusion criteria. The first author carried out the search, read and applied inclusion criteria to abstracts, and qualitatively reviewed the 78 articles that fit the criteria. Twenty-eight articles were flagged as being borderline and not fitting the research questions. All three authors reviewed the applicability of the research questions to findings and determined the 28 articles were not relevant. In the end, 50 articles were selected for the analysis and were included in the present study.

## 4 FINDINGS

**RQ1: What are barriers to broadening participation in CS in upper elementary (grades 4-6)?**

Understanding barriers to participation in CS has implications for the design of instruction. The first RQ reviews barriers identified in studies focused on upper elementary (grades 4-6) classrooms.

### 4.1 Effect of Prior Mathematical Skills

Some studies have found that students' mathematics skills are correlated with CS learning [18, 21, 51]. A small posttest-only design study (n = 31) found that the ability to perform numerical operations had a higher correlation with CT skills for students in the lower grades than it did for the students in the upper grades [56]. Scratch, which is a popular programming environment, requires knowledge of math concepts of coordinate planes, negative numbers, percentages, and decimal numbers [23]. A quasi-experimental study that spanned over six weeks [18] (n = 74) found prior math achievement to be correlated to pre-scores on a CT assessment but not to post-scores. This suggests that baseline math skills did not predict improvement on CT assessment.

### 4.2 Effect of Prior Language and Literacy Skills

While prior language and literacy skills have been identified as a barrier to participation in CS, the findings on the effects of these skills on CS learning are not conclusive. One quasi-experimental study (n = 296) that spanned over a school year found a correlation between CS ability and reading proficiency [51]. Another study (quasi-experimental, n = 54) also found that English Language Learners (ELLs) had smaller gains in computing knowledge compared to other learners [21]. However, a group of aforementioned studies that used unplugged activities (CS activities completed without computers [10]) [56] and non-programming tasks [18, 19] before the introduction of plugged programming, found language ability did not affect CS performance. It is possible that the introductory tasks in these studies helped students establish familiarity with programming terms before moving to programming tasks. More research is needed on the relationship between language and literacy skills and programming in order to identify approaches that help less proficient readers and ELLs to equitably participate in programming. Further, identifying foundational literacy skills that are prerequisites to the development of programming skills is another understudied area of research.

### 4.3 Gender Differences

In a study that analyzed video and screen recording data (n = 26), boys were found to be more likely than girls to engage in CS activities at the elementary school level [55]. Multiple studies in the reviewed literature examined this challenge. In one study, students in grades 3-6, across twenty schools (n = 1140), participated in a 30-minute coding lesson and completed a survey [30]. Boys reported higher perceived competence in coding than girls. In a different (n = 559) year-long study, girls underestimated their achievement in CS [10]. In the same study, girls took longer to complete their tasks

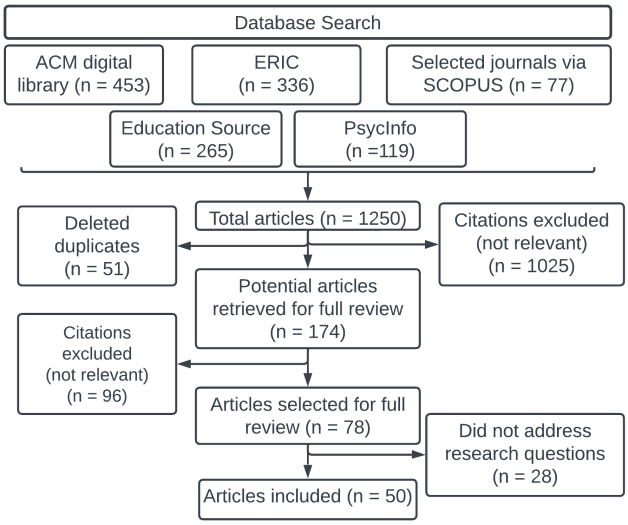


**Figure 1: Literature Search and Selection Process**

than boys. The study's authors ascribed this to girls feeling less comfortable with making coding mistakes. In a quasi-experimental study (n = 171) that spanned over 17 weeks, the CT scores of boys improved significantly both in boy-girl and boy-boy pairings [59], compared to girls whose CT scores significantly improved only in boy-girl pairings. A small (three lessons, n = 53) pre-test and post-test study [8] found that after an unplugged unit (CS activities completed without computers [10]), girls scored higher than boys on questions that were rated as more difficult. Two other aforementioned studies that used unplugged approaches also reported overall higher average scores for girls than boys, but the differences were not significant [10, 21]. In a 10-week (n = 53) study that used robotics to teach CS, girls showed more improvement in programming skills than boys [11]. One small (n = 70, four forty-minutes lessons at two-sites) study that used dress-up dolls found that gendered cultural forms of in-game characters had a positive effect on girls' interest, but a negative one on boys' interest [3]. The study's authors conclude that gendered cultural forms of game characters can provide girls with the opportunities to engage in personally meaningful learning experiences.

While some of the findings described above report that girls performed better than boys, a large number of studies (n = 28) that used novel programming approaches did not report on gender differences related to learning. This could be a result of reporting or publication bias. It is possible that studies in which boys perform better than girls are less-frequently published.

**RQ2: What instructional approaches and strategies help broaden participation in CS in upper elementary (grades 4-6)?**

Our second RQ focused on instructional approaches for broadening participation in CS at the upper elementary level.

## 4.4 Instructional Materials and Context

As reported in the previous section, some unplugged and novel approaches to CS had positive effects on girls' attitudes and performance. Accordingly, many studies focus on instructional materials

**Table 2: Literature Review Inclusion Criteria**

| Criteria | Inclusion |
|---|---|
| Timeliness | Published after Lye & Koh (2014) i.e., November 2013 – December 2020 |
| Publication type | Conference paper, Journal Article |
| Age range | Grades 4th, 5th, 6th. Ages 10 – 12 |
| Study type | Findings are based on student data |
| Intervention context | Formal education, Classroom setting |
| Intervention type | Curricular units implemented for students |
| Language | Written in English |
| Focus | Addresses broadening participation |

in designs that broaden participation. Unplugged CS activities have been found to offer advantages over plugged CS when introducing programming concepts to novices because they remove barriers associated with the use of computers [10, 34, 43]. In a study (eight hour-long lessons, n = 47) that compared students' learning in an unplugged CS unit to students who completed a 'plugged' unit in Scratch [49], the authors did not find any statistical differences in outcomes between the two groups/approaches. In a smaller study (two 90-minutes sessions, n = 11) that also compared students' learning in unplugged and 'plugged' instructional contexts [1], the unplugged group performed better on rule construction whereas the plugged group reported a better understanding of the syntax of the code, which was attributed to the programming environment's dynamic feedback.

We identified multiple examples of instructional materials that use unplugged CS: board games [4, 34, 35, 43, 56], stories and scenarios [4, 8, 35, 39, 43, 54], non-digital tangible materials such as LEGOs [15], electronic MakeMe blocks [29], and paper and pencil activities on printed pages [1, 4, 10]. Such instructional approaches have the potential to increase the participation of girls and demographics that are generally underrepresented in CS. However, there is a need for research that explores whether plugged programming environments provide affordances that the unplugged media cannot provide. Further research is also needed on how students transition from unplugged to plugged instruction and how it affects learning. We only found two studies that addressed the transition from unplugged to plugged. In one study (seven 30-minute lessons), the instructors made explicit links between the unplugged and plugged contexts in order to facilitate the transition. Using qualitative (n = 1) [52] and quantitative (n = 87) [34] analyses, the study reported positive results of the approach. In a quasi-experimental study at a middle school (n = 74, 20 sessions), students engaged in non-programming digital activities before they engaged in programming tasks in order to support their early conceptual CS skill development [18, 19]. The study reported positive results. However, supporting the transition from non-programming media to programming environments remains an understudied area.

## 4.5 Visual Block-Based Programming Environments (VBBPE)

In visual block-based programming environments (VBBPE) [61], users write programs by manipulating graphical elements (i.e.,

blocks) as opposed to using text-based syntax. VBBPEs are popular ways to introduce novices to programming [61]. They have visually different representations for events, loops, conditionals, and other programming structures [51]. VBBPEs are also better suited to introduce concepts that are considered advanced in text-based languages, such as parallelism [61], which allows computers to run multiple instructions of code in parallel.

Despite research that supports VBBPEs as a strategy for broadening participation, some programming concepts are difficult, even if they are presented via VBBPEs. For example, in a qualitative study that used printed scratch blocks on LEGO bricks [15], students struggled with the concept of the orientation of a Scratch sprite (a programmable graphical element or character in Scratch). In order to guess the direction the sprite would face, students often used their own body's orientation rather than the sprite's orientation. In the same study, students made multiple sequencing errors and struggled with loops. If there were multiple instructions inside a loop, they would repeat the first instruction multiple times before moving on to the next instruction and repeating it, and so on. In a study that reported observations from five different schools, students in the 4th grade were given pre-populated Scratch projects to modify. The students struggled with recovering elements that they accidentally deleted [24].

In VBBPEs, the real functionality of commands can differ from what the users expect [23]. For example, a multi-school (N = 1385) study reported that some students expected the placement of the script in the scripting area to influence the movement of their programmable character [23]. The study also found that students can struggle with understanding the purpose of Scratch blocks that do not make a noticeable change in the output. Some features such as painting the stage or sprites can distract students from engaging in programming tasks [24].

Research has also identified programming concepts that are not well represented in VBBPEs. For example, the wait blocks in Scratch often affect the execution time of the scripts, leading to the incorrect assumption that the speed of the computer can be manipulated [61]. Another example is the concept of initialization. [23] also found that when students were asked to reset their programmable sprites, they added code towards the end of their scripts to make them run back to the starting line, instead of initializing the variables at the start of the program. In the same study, they found that about half of the Scratch programs created by students were programmer controlled and non-interactive, which suggests that students did not consider how someone else would use their program.

If educators are aware of the challenges students face when using VBBPEs, they can address them during instruction. For example, Franklin et al. [14] suggested explicitly mentioning how the concepts in VBBPEs relate to text-based languages. Examples include linking initialization to the green flag in Scratch, and using absolute blocks and absolute values to reproduce the same output every time instead of relative blocks whose output depend on the current state. Another strategy is to explain the syntax of text-based programming [21] to students. Finally, one study recommended that teachers de-emphasize blocks that cause confusion [61]. For example, the effect of the wait command on the execution of instructions should be de-emphasized to avoid the misconception that the computer speed can be manipulated.

## 4.6 Scaffolding Instruction

Scaffolding (providing instructional support( is another strategy used for broadening participation in CS. An example of scaffolding is teaching how to read and interpret programs written by others [1], which is also intended to help learners write their own programs.

Another example of scaffolding in CS is "use-modify-create" sequence, a progression where learners first use existing code, then transition to modifying existing code, and then finally they create their own code [33]. In a short (n = 160, 4 lessons) quasi-experimental study [38], the use-modify-create approach led students to finish their designated tasks earlier than planned, which gave them time to add additional elements to their programs and engage in teacher-led discussions about connections of code to other class topics. As a result of scaffolded design, teachers felt more confident in their ability to teach and students were more likely to take ownership of their code. A study that analyzed video data of three focal students [46] recommended adding scaffolds that help students collaborate.

## 4.7 Sequencing CS Concepts

Another strategy used to broaden participation is to introduce computing concepts in the order of simple to more challenging. Studies identified sequences as the easiest concept [21], even for 4th grade learners [16], followed by conditionals [21, 45] and abstraction [45]. Variable manipulation [21] and initialization [16] are thought to be more difficult concepts. [19] examined the appropriateness of introductory CS concepts and recommended introducing boolean operators and repeat-until loops earlier in lessons as these are frequently used in free choice projects.

## 4.8 Integrating CS In Other Subjects

Research suggests that integrating CS activities into other subjects lowers the barrier to the adoption of CS [12]. Some findings support the idea of integration whereas others do not.

In a study (3 lessons, n = 51), CT-infused curricula raised attainment in grammar and sentence structure [28]. In a 5-day language learning camp (n = 34, pre-test and post-test), coding was integrated into activities, found that students thought more deeply about the language after the unit [7]. In the same study, students reported an increase in self-efficacy, but their interest in programming dropped and the camp did not attract female participation. A different study that integrated CS into math as well as into social studies (n = 129, four weeks), reported that integration of programming in social studies resulted in nearly double the effect size in terms of improvement in academic performance as compared to math [40].

Studies on the integration of CS into math report mixed outcomes. For example, the integration of math and geometry concepts into CS resulted in a positive outcome of students asking more questions about angles and polygons [10]. Similarly, in a study that used Scratch to teach least common multiples and greatest common divisors, students' knowledge of these concepts improved significantly [49]. On the other hand, in a study that asked 358 participants about the connection of programming to other subjects, only four participants saw a connection between math and CS [62]. This suggests that math ideas in programming are often hidden, and instruction should make this connection more explicit. Furthermore,

lessons that fail to make the connections between CS and math remain at the risk of being removed from the curriculum by the teachers [12, 27]. In a year-long study conducted in Germany, the experimental group who was taught CS integrated into physics and mathematics, performed worse than the CS-only group on a programming knowledge test [44]. The authors of the study suggested introducing students to the programming context first and moving to integration once the students have gained the necessary programming skills. Another study [27] also recommended introducing programming concepts before math-integrated programming tasks.

### 4.9 Creative Choice in Instruction

Another strategy for broadening participation is through creative choice, which we define as enabling students to make choices in the process of creating their computer programs. Multiple studies found that creative choice in tasks was favored by students [25, 31, 38, 57]. In addition, giving students a large problem space and creative choice resulted in better engineering solutions [63], supported exploration and encouraged the use of prior knowledge [15], increased student ownership of the code [41], and stimulated discussions between students [57].

There are a variety of ways to afford creative choice to students. The act of making can provide deeper understanding, reflecting, creativity, and sharing and the tangible objects can be used as "objects-to-share-with" [29]. Another way is to use sensors to collect data from the physical environment, helping students make connections between data and the underlying activity [17, 32]. Allowing students to personalize the different aspects of the learning is another way of increasing student engagement in computing activities. One study (n = 75, three 4-hour sessions) reported that students liked creating their own avatars, creating their own music, and experimenting with programming rules [25]. Enabling students to customize their in-activity characters may help them remember and understand computing concepts better [36], and improve their motivation [42]. An interesting finding related to avatar creation is that most students in the experiment created a character having the same gender and skin color as themselves [36].

Despite all the benefits of allowing students to practice creative choice, it imposes time-related, monetary, and instructional demands. In a study that used e-textiles to encourage students to use their creativity in computing tasks, the authors noted that the equipment was expensive, and teachers required competency with e-textiles [26]. Units that allow students to practice creative choice need unstructured time between activities for teachers to attend to individual student needs [22] and help students solve issues in their programs [32]. Exploratory learning also requires flexible scaffolds such as instructor-led discussions and should enable learners to ask questions and observe others [32]. Furthermore, students may also need unstructured time to get into creative headspace [10, 26].

Findings suggest that it is beneficial to limit the creative choice that students can practice. Giving students too much freedom can result in confusion for students [41]. In the study that used sensors for collecting data from the environment, the sensors put a natural constraint on what could be measured, which helped limit the scope of the unit [17]. One way of constraining the activities while allowing creative choice is to differentiate between the required and the optional tasks. This may encourage students to complete the required tasks before working on the additional tasks [15].

## 5 DISCUSSION

We set out to systematically review the recent research being conducted in formal CS education settings, specifically upper elementary classrooms. We summarized the barriers to broadening participation in CS and the instructional approaches that help broaden participation in CS. In this section, we discuss these findings in the context of implications for the design of instruction and future research.

### 5.1 Implications for the Design of CS Instruction

This review highlighted that there is a myriad of instructional materials and approaches available for teaching CS at the upper-elementary levels of schooling. It also revealed that certain materials (e.g., unplugged) and activities (e.g., non-programming activities) can positively influence the participation of girls and ELLs. Thus, educators and instructional designers should pay special attention to the choice of materials and instructional approaches. Allowing students to make their own choices during programming can improve student attitudes toward programming. To improve their experience, novice learners should be introduced to easier programming concepts (e.g., sequencing) before they are introduced to more difficult ones (e.g., conditionals, abstraction). One should also be aware that certain block-based programming environments can impose barriers to learning for students who struggle with math concepts. Instructional design should incorporate math scaffolds for these students. Furthermore, in instances where CS is integrated into math and vice versa, the instruction should draw clear connections between math and CS concepts.

### 5.2 Implications for Future Research

This review identified areas in CS education research that require more attention. We argue that research conducted in formal upper elementary classrooms should attend to issues of inequitable participation. For example, gender is an important factor that often correlates with attitudes toward and performance in CS. Thus, research presenting novel instructional approaches should also report results stratified by gender.

Further, non-computing media and activities are promising ways to familiarize novice learners with programming terms and concepts before they are introduced to programming. However, findings suggest that novice programmers can struggle when they transition from these introductory tasks to programming tasks, and there is a dearth of research that studies this phenomenon. Finally, literacy skills can affect participation in CS, however, there is also a dearth of research that aims to identify foundational literacy skills required for learning to program.

## 6 CONCLUSION

This review contributes to the CS education research by identifying underlying factors that may deter upper elementary students from participating in CS. It also comes with some limitations. Although we added contextual information such as the study's sample size

and design, other information such as learners' gender, race, appropriateness of methods, etc were not always reported. Thus, beyond the inclusion criteria, it is up to the reader to assess the claims of the research synthesized in this review. Despite the limitations, we hope that the highlighted approaches identify ways to make participation in CS more appealing to a wide array of learners in that they help inform the choice of instructional materials and media used for teaching CS, teaching strategies, CS content, and ways to integrate into non-CS subjects. Our findings also helped identify areas that are inconclusive and/or require more research. In this way, we help set the direction of future CS education research.

## 7 ACKNOWLEDGMENTS

This work was supported by the National Science Foundation, grants #1837224 and #2031382. The opinions expressed herein are those of the authors and do not necessarily reflect those of the funding organization.

## REFERENCES

[1] Ashish Aggarwal, Christina Gardner-McCune, and David S. Touretzky. 2017. Evaluating the Effect of Using Physical Manipulatives to Foster Computational Thinking in Elementary School. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, Seattle Washington USA, 9–14. https://doi.org/10.1145/3017680.3017791

[2] Patricia A. Alexander. 2020. Methodological Guidance Paper: The Art and Science of Quality Systematic Reviews. *Review of Educational Research* 90, 1 (Feb. 2020), 6–23. https://doi.org/10.3102/0034654319854352

[3] Sarah AlSulaiman and Michael S. Horn. 2015. Peter the Fashionista?: Computer Programming Games and Gender Oriented Cultural Forms. In *Proceedings of the 2015 Annual Symposium on Computer-Human Interaction in Play*. ACM, London United Kingdom, 185–195. https://doi.org/10.1145/2793107.2793127

[4] Christian P. Brackmann, Marcos Román-González, Gregorio Robles, Jesús Moreno-León, Ana Casali, and Dante Barone. 2017. Development of Computational Thinking Skills through Unplugged Activities in Primary School. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*. ACM, Nijmegen Netherlands, 65–72. https://doi.org/10.1145/3137065.3137069

[5] Karen Brennan and Mitchel Resnick. 2012. New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 annual meeting of the American educational research association, Vancouver, Canada*, Vol. 1. 25.

[6] Francisco Buitrago Flórez, Rubby Casallas, Marcela Hernández, Alejandro Reyes, Silvia Restrepo, and Giovanna Danies. 2017. Changing a Generation's Way of Thinking: Teaching Computational Thinking Through Programming. *Review of Educational Research* 87, 4 (Aug. 2017), 834–860. https://doi.org/10.3102/0034654317710096

[7] Yesheng Chen, Zhen Chen, Shyamala Gumidyala, Annabella Koures, Seoyeon Lee, James Msekela, Halle Remash, Nolan Schoenle, Sarah Dahlby Albright, and Samuel A. Rebelsky. 2019. A Middle-School Code Camp Emphasizing Digital Humanities. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, Minneapolis MN USA, 351–357. https://doi.org/10.1145/3287324.3287509

[8] Havva Delal and Diler Oner. 2020. Developing Middle School Students' Computational Thinking Skills Using Unplugged Computing Activities. *Informatics in Education* 19, 1 (Jan. 2020), 1–13. https://doi.org/10.15388/infedu.2020.01 Publisher: Informatics in Education.

[9] Jennifer DeWitt and Louise Archer. 2017. Participation in informal science learning experiences: the rich get richer? *International Journal of Science Education, Part B* 7, 4 (Oct. 2017), 356–373. https://doi.org/10.1080/21548455.2017.1360531 Publisher: Routledge.

[10] Caitlin Duncan and Tim Bell. 2015. A Pilot Computer Science and Programming Course for Primary School Students. In *In Proceedings of the Workshop in Primary and Secondary Computing Education*. 48. https://doi.org/10.1145/2818314.2818328

[11] Hatice Yildiz Durak, Fatma Gizem Karaoglan Yilmaz, and Ramazan Yilmaz. 2019. Computational Thinking, Programming Self-Efficacy, Problem Solving and Experiences in the Programming Process Conducted with Robotic Activities. *Contemporary Educational Technology* 10, 2 (Jan. 2019), 173–197. https://doi.org/doi.org/10.30935/cet.554493 Publisher: Contemporary Educational Technology.

[12] Janet Fofang, David Weintrop, Margaret Walton, Andrew Elby, and Janet Walkoe. 2020. Mutually Supportive Mathematics and Computational Thinking in a Fourth-Grade Classroom, Vol. 3. International Society of the Learning Sciences, Nashville, Tennessee, 1389–1396. https://doi.org/10.22318/icls2020.1389

[13] National Science Foundation. 2021. Computer Science for All (CSforAll: Research and RPPs). https://beta.nsf.gov/funding/opportunities/computer-science-all-csforall-research-and-rpps

[14] Diana Franklin, Charlotte Hill, Hilary A. Dwyer, Alexandria K. Hansen, Ashley Iveland, and Danielle B. Harlow. 2016. Initialization in Scratch: Seeking Knowledge Transfer. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. ACM, Memphis Tennessee USA, 217–222. https://doi.org/10.1145/2839509.2844569

[15] Diana Franklin, Jean Salac, Cathy Thomas, Zene Sekou, and Sue Krause. 2020. Eliciting Student Scratch Script Understandings via Scratch Charades. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, Portland OR USA, 780–786. https://doi.org/10.1145/3328778.3366911

[16] Diana Franklin, Gabriela Skifstad, Reiny Rolock, Isha Mehrotra, Valerie Ding, Alexandria Hansen, David Weintrop, and Danielle Harlow. 2017. Using Upper-Elementary Student Performance to Understand Conceptual Sequencing in a Blocks-based Curriculum. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. ACM, Seattle Washington USA, 231–236. https://doi.org/10.1145/3017680.3017760

[17] Alexandra Gendreau Chakarov, Mimi Recker, Jennifer Jacobs, Katie Van Horne, and Tamara Sumner. 2019. Designing a Middle School Science Curriculum that Integrates Computational Thinking and Sensor Technology. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, Minneapolis MN USA, 818–824. https://doi.org/10.1145/3287324.3287476

[18] Shuchi Grover, Nicholas Jackiw, and Patrik Lundh. 2019. Concepts before coding: non-programming interactives to advance learning of introductory programming concepts in middle school. *Computer Science Education* 29, 2-3 (July 2019), 106–135. https://doi.org/10.1080/08993408.2019.1568955 Publisher: Routledge.

[19] Shuchi Grover, Patrik Lundh, and Nicholas Jackiw. 2019. Non-Programming Activities for Engagement with Foundational Concepts in Introductory Programming. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*. ACM, Minneapolis MN USA, 1136–1142. https://doi.org/10.1145/3287324.3287468

[20] Shuchi Grover and Roy Pea. 2013. Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher* 42, 1 (Jan. 2013), 38–43. https://doi.org/10.3102/0013189X12463051 Publisher: American Educational Research Association.

[21] Shuchi Grover, Roy Pea, and Stephen Cooper. 2015. Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education* 25, 2 (April 2015), 199–237. https://doi.org/10.1080/08993408.2015.1033142

[22] Alexandria K. Hansen, Eric R. Hansen, Hilary A. Dwyer, Danielle B. Harlow, and Diana Franklin. 2016. Differentiating for Diversity: Using Universal Design for Learning in Elementary Computer Science Education. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16*. ACM Press, Memphis, Tennessee, USA, 376–381. https://doi.org/10.1145/2839509.2844570

[23] Danielle B. Harlow, Hilary A. Dwyer, Alexandria K. Hansen, Ashley O. Iveland, and Diana M. Franklin. 2018. Ecological Design-Based Research for Computer Science Education: Affordances and Effectivities for Elementary School Students. *Cognition and Instruction* 36, 3 (July 2018), 224–246. https://doi.org/10.1080/07370008.2018.1475390 Publisher: Routledge.

[24] Charlotte Hill, Hilary A. Dwyer, Tim Martinez, Danielle Harlow, and Diana Franklin. 2015. Floors and Flexibility: Designing a Programming Environment for 4th-6th Grade Classrooms. In *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, Kansas City Missouri USA, 546–551. https://doi.org/10.1145/2676723.2677275

[25] Daniel Hug, Serge Petralito, Sarah Hauser, Anna Lamprou, Alexander Repenning, Didier Bertschinger, Nadine Stüber, and Markus Cslovjecsek. 2017. Exploring Computational Music Thinking in a Workshop Setting with Primary and Secondary School Children. In *Proceedings of the 12th International Audio Mostly Conference on Augmented and Participatory Sound and Music Experiences*. ACM, London United Kingdom, 1–8. https://doi.org/10.1145/3123514.3123515

[26] Cristyne Hébert and Jennifer Jenson. 2020. Making in schools: student learning through an e-textiles curriculum. *Discourse: Studies in the Cultural Politics of Education* 41, 5 (Sept. 2020), 740–761. https://doi.org/10.1080/01596306.2020.1769937

[27] Maya Israel and Todd Lash. 2020. From classroom lessons to exploratory learning progressions: mathematics + computational thinking. *Interactive Learning Environments* 28, 3 (April 2020), 362–382. https://doi.org/10.1080/10494820.2019.1674879 Publisher: Routledge.

[28] Craig Jenkins. 2015. A Work in Progress Paper: Evaluating a Microworlds- based Learning Approach for Developing Literacy and Computational Thinking in Cross-curricular Contexts. In *In Proceedings of the Workshop in Primary and Secondary Computing Education*. 61–64. https://doi.org/10.1145/2818314.2818316

[29] Rose Johnson, Venus Shum, Yvonne Rogers, and Nicolai Marquardt. 2016. Make or Shake: An Empirical Study of the Value of Making in Learning about Computing Technology. In *Proceedings of the The 15th International Conference on Interaction Design and Children*. ACM, Manchester United Kingdom, 440–451. https://doi.org/10.1145/2930674.2930691

[30] Ilkka Jormanainen and Markku Tukiainen. 2020. Attractive Educational Robotics Motivates Younger Students to Learn Programming and Computational Thinking. In *Eighth International Conference on Technological Ecosystems for Enhancing Multiculturality (TEEM'20)*. Association for Computing Machinery, New York, NY, USA, 54–60. https://doi.org/10.1145/3434780.3436676

[31] Anna Lamprou, Alexander Repenning, and Nora A Escherle. 2017. The Solothurn Project — Bringing Computer Science Education to Primary Schools in Switzerland. In *Proceedings of the 2017 ACM Conference on Innovation and Technology in Computer Science Education*. 218–223. https://doi.org/10.1145/3059009.3059017

[32] Susan Lechelt, Yvonne Rogers, and Nicolai Marquardt. 2020. Coming to your senses: promoting critical thinking about sensors through playful interaction in classrooms. In *Proceedings of the Interaction Design and Children Conference*. ACM, London United Kingdom, 11–22. https://doi.org/10.1145/3392063.3394401

[33] Irene Lee, Fred Martin, Jill Denner, Bob Coulter, Walter Allan, Jeri Erickson, Joyce Malyn-Smith, and Linda Werner. 2011. Computational thinking for youth in practice. *ACM Inroads* 2, 1 (Feb. 2011), 32–37. https://doi.org/10.1145/1929887.1929902

[34] Victor R. Lee, Frederick Poole, Jody Clarke-Midura, Mimi Recker, and Melissa Rasmussen. 2020. Introducing Coding through Tabletop Board Games and Their Digital Instantiations across Elementary Classrooms and School Libraries. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 787–793. https://doi.org/10.1145/3328778.3366917

[35] Luzia Leifheit, Katerina Tsarava, Korbinian Moeller, Klaus Ostermann, Jessika Golle, Ulrich Trautwein, and Manuel Ninaus. 2019. Development of a Questionnaire on Self-concept, Motivational Beliefs, and Attitude Towards Programming. In *WiPSCE'19: Proceedings of the 14th Workshop in Primary and Secondary Computing Education*. 1–9. https://doi.org/10.1145/3361721.3361730

[36] Lorraine Lin, Dhaval Parmar, Sabarish V Babu, Alison E Leonard, Shaundra B Daily, and Sophie Jörg. 2017. How Character Customization Affects Learning in Computational Thinking. In *In Proceedings of the ACM Symposium on Applied Perception*. 1–8. https://doi.org/10.1145/3119881.3119884

[37] Sze Yee Lye and Joyce Hwee Ling Koh. 2014. Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior* 41 (Dec. 2014), 51–61. https://doi.org/10.1016/j.chb.2014.09.012

[38] Nicholas Lytle, Veronica Cateté, Danielle Boulden, Yihuan Dong, Jennifer Houchins, Alexandra Milliken, Amy Isvik, Dolly Bounajim, Eric Wiebe, and Tiffany Barnes. 2019. Use, Modify, Create: Comparing Computational Thinking Lesson Progressions for STEM Classes. In *Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Aberdeen Scotland Uk, 395–401. https://doi.org/10.1145/3304221.3319786

[39] Tracy Mensan, Kamisah Osman, and Nazatul Aini Abdul Majid. 2020. Development and Validation of Unplugged Activity of Computational Thinking in Science Module to Integrate Computational Thinking in Primary Science Education. *Science Education International* 31, 2 (June 2020), 142–149. https://doi.org/10.33828/sei.v31.i2.2

[40] Jesús Moreno-León, Gregorio Robles, and Marcos Román-González. 2016. Code to Learn: Where Does It Belong in the K-12 Curriculum? *Journal of Information Technology Education: Research* 15 (Jan. 2016), 283–303. Publisher: Journal of Information Technology Education: Research.

[41] Kirsten Mork, John Wilcox, and Zoë Wood. 2020. Creative Choice in Fifth Grade Computing Curriculum. In *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education*. ACM, Trondheim Norway, 252–258. https://doi.org/10.1145/3341525.3387405

[42] Simon P. Rose, M. P. Jacob Habgood, and Tim Jay. 2020. Designing a Programming Game to Improve Children's Procedural Abstraction Skills in Scratch. *Journal of Educational Computing Research* 58, 7 (Dec. 2020), 1372–1411. https://doi.org/10.1177/0735633120932871 Publisher: SAGE Publications Inc.

[43] Jungho Park. 2019. The Development and Application of Computational Fairy Tales for Elementary Students. *International Journal of Higher Education* 8, 3 (May 2019), 159. https://doi.org/10.5430/ijhe.v8n3p159

[44] Arno Pasternak. 2016. Contextualized Teaching in the Lower Secondary EducationLong-term Evaluation of a CS Course from Grade 6 to 10. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education - SIGCSE '16*. ACM Press, Memphis, Tennessee, USA, 657–662. https://doi.org/10.1145/2839509.2844592

[45] A. Peel, J. Fulton, and E. Pontelli. 2015. DISSECT: An experiment in infusing computational thinking in a sixth grade classroom. In *2015 IEEE Frontiers in Education Conference (FIE)*. IEEE, Camino Real El Paso, El Paso, TX, USA, 1–8. https://doi.org/10.1109/FIE.2015.7344240

[46] Ashlyn E. Pierson, Corey E. Brady, and Douglas B. Clark. 2020. Balancing the Environment: Computational Models as Interactive Participants in a STEM Classroom. *Journal of Science Education and Technology* 29, 1 (Feb. 2020), 101–119. https://doi.org/10.1007/s10956-019-09797-5

[47] Siwarak Promraksa, Kiat Sangaroon, and Maitree Inprasitha. 2014. Characteristics of Computational Thinking about the Estimation of the Students in Mathematics Classroom Applying Lesson Study and Open Approach. *Journal of Education and Learning* 3, 3 (2014), 11.

[48] Kathryn Rich, Carla Strickland, and Diana Franklin. 2017. A Literature Review through the Lens of Computer Science Learning Goals Theorized and Explored in Research. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education (SIGCSE '17)*. Association for Computing Machinery, New York, NY, USA, 495–500. https://doi.org/10.1145/3017680.3017772

[49] José Antonio Rodríguez-Martínez, José Antonio González-Calero, and José Manuel Sáez-López. 2020. Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. *Interactive Learning Environments* 28, 3 (April 2020), 316–327. https://doi.org/10.1080/10494820.2019.1612448 Publisher: Routledge.

[50] Royal Society. 2017. *After the reboot: Computing education in UK schools*. Technical Report.

[51] Jean Salac, Cathy Thomas, Bryan Twarek, William Marsland, and Diana Franklin. 2020. Comprehending Code: Understanding the Relationship between Reading and Math Proficiency, and 4th-Grade CS Learning Outcomes. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. ACM, Portland OR USA, 268–274. https://doi.org/10.1145/3328778.3366822

[52] Umar Shehzad, Jody Clarke-Midura, Victor R. Lee, and Mimi Recker. 2022. A Student's Access to Practice-Linked Resources in an Elementary Unplugged-to-Plugged Computer Science Unit. In *International Conference of the Learning Sciences*. International Society of Learning Sciences, Hiroshima, Japan.

[53] Valerie J. Shute, Chen Sun, and Jodi Asbell-Clarke. 2017. Demystifying computational thinking. *Educational Research Review* 22 (Nov. 2017), 142–158. https://doi.org/10.1016/j.edurev.2017.09.003

[54] Cristina Sylla, Ana Carina Figueiredo, Ana Lúcia Pinto, Pedro Branco, and Nelson Zagalo. 2014. Merging Physical and Digital White Canvas to Unleash Children's Creativity. In *Proceedings of the 2014 Workshops on Advances in Computer Entertainment Conference - ACE '14 Workshops*. ACM Press, Funchal, Portugal, 1–9. https://doi.org/10.1145/2693787.2693807

[55] Jennifer Tsan, Fernando J Rodríguez, Kristy Elizabeth Boyer, and Collin Lynch. 2018. "I Think We Should…": Analyzing Elementary Students' Collaborative Processes for Giving and Taking Suggestions. In *SIGCSE '18: Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. Association for Computing Machinery, 622–627. https://doi.org/10.1145/3159450.3159507

[56] Katerina Tsarava, Luzia Leifheit, Manuel Ninaus, Marcos Román-González, Martin V. Butz, Jessika Golle, Ulrich Trautwein, and Korbinian Moeller. 2019. Cognitive Correlates of Computational Thinking: Evaluation of a Blended Unplugged/Plugged-In Course. In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education*. ACM, Glasgow Scotland Uk, 1–9. https://doi.org/10.1145/3361721.3361729

[57] Christiane Gresse Von Wangenheim, Nathalia Cruz Alves, Pedro Eurico Rodrigues, and Jean Carlo Hauck. 2017. Teaching Computing in a Multidisciplinary Way in Social Studies Classes in School" - A Case Study. *International Journal of Computer Science Education in Schools* 1, 2 (May 2017), 3. https://doi.org/10.21585/ijcses.v1i2.9

[58] Jane Waite. 2017. Pedagogy in teaching computer science in schools: A literature review. *London: Royal Society* 253 (2017).

[59] Xuefeng Wei, Lin Lin, Nanxi Meng, Wei Tan, Siu-Cheung Kong, and Kinshuk. 2021. The effectiveness of partial pair programming on elementary school students' Computational Thinking skills and self-efficacy. *Computers & Education* 160 (Jan. 2021), 104023. https://doi.org/10.1016/j.compedu.2020.104023

[60] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2016. Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology* 25, 1 (2016), 127–147. Publisher: Springer.

[61] David Weintrop, Alexandria K. Hansen, Danielle B. Harlow, and Diana Franklin. 2018. Starting from Scratch: Outcomes of Early Computer Science Learning Experiences and Implications for What Comes Next. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. ACM, Espoo Finland, 142–150. https://doi.org/10.1145/3230977.3230988

[62] Gary Ka-Wai Wong and Ho-Yin Cheung. 2020. Exploring children's perceptions of developing twenty-first century skills through computational thinking and programming. *Interactive Learning Environments* 28, 4 (May 2020), 438–450. https://doi.org/10.1080/10494820.2018.1534245 Publisher: Routledge.

[63] Ningyu Zhang, Gautam Biswas, Kevin W. McElhaney, Satabdi Basu, Elizabeth McBride, and Jennifer L. Chiu. 2020. Studying the Interactions Between Science, Engineering, and Computational Thinking in a Learning-by-Modeling Environment. In *Artificial Intelligence in Education (Lecture Notes in Computer Science)*, Ig Ibert Bittencourt, Mutlu Cukurova, Kasia Muldner, Rose Luckin, and Eva Millán (Eds.). Springer International Publishing, Cham, 598–609. https://doi.org/10.1007/978-3-030-52237-7_48