Modifying Curriculum for Novice Computational Thinking Elementary Teachers and English Language Learners

Dana Saito-Stehberger
University of California, Irvine
Irvine, CA, USA
dsaitost@uci.edu

Leiny Garcia
University of California, Irvine
Irvine, CA, USA
leinyg@uci.edu

Mark Warschauer
University of California, Irvine
Irvine, CA, USA
markw@uci.edu

ABSTRACT

The demand for computational thinking (CT) problem solving abilities surge as every aspect of life becomes more dependent on complex digital technologies. Just as in math and language, a strong CT foundation needs to be established in early education in order for students to develop an instinctive CT perspective of the world. The urgent demand for CT instruction in elementary school quickly draws attention to the shortage of elementary school-level teachers qualified and interested in CT. Additionally, with a commitment to equity in the United States education system and knowledge of the high percentage of English language learning (ELL) students in schools, the obligation to create curricula that will provide access to CT knowledge, skills, and practices for elementary-level ELL students is loudly apparent. In response to these two needs, our team has adapted existing Scratch-based CT curriculum to support classroom teachers with minimal CT experience and to be more accessible to English language learners. The purpose of this paper is to share the framework that guided the curriculum adaptations, to describe the specific changes that were made, and to discuss discoveries made during the process. This journey may be helpful to anyone who is tasked with modifying a curriculum to meet the needs of novice content teachers and ELL students.

CCS CONCEPTS

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

KEYWORDS

English language learners, curriculum design, elementary education, computational thinking

ACM Reference Format:
Dana Saito-Stehberger, Leiny Garcia, and Mark Warschauer. 2021. Modifying Curriculum for Novice Computational Thinking Elementary Teachers and English Language Learners . In *26th ACM Conference on Innovation and Technology in Computer Science Education V. 1 (ITiCSE 2021), June 26–July 1, 2021, Virtual Event, Germany.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3430665.3456355

ITiCSE 2021, June 26-July 1, 2021, Virtual Event, Germany.

ACM ISBN 978-1-4503-8214-4/21/06.
https://doi.org/10.1145/3430665.3456355

1 INTRODUCTION

As the world becomes more dependent on complex digital technologies, it becomes more imperative to prepare students to be able to navigate the complexities and to attain strategies to solve multi-faceted problems. Computational thinking (CT) has gained traction as a viable framework to be incorporated in elementary school curriculum in order to provide students with these essential skills [3, 6, 8, 19–21]. As the demand for CT instruction in the United States increases, the number of elementary school-level teachers prepared to teach computer education lags behind. Only a handful of states require that pre-service teachers across all subject areas and grade levels receive exposure to computer science to prepare them with the knowledge and skills to integrate CT into their classroom practice [2]. The aim is for all students to attain CT problem solving skills, including the 3.8 million English Language Learning (ELL) students in public schools in the United States [4]. The ELL population in the United States represents more than 350 language groups and a wide array of cultural groups. Since the 1980's, when content-based language teaching emerged, teachers and researchers have acknowledged the need to articulate instructional practices to support ELL students in mastering specific academic content while simultaneously learning the target language [17]. CT is the new academic content on the block that requires attention in creating curriculum to support ELL students in their attainment of CT knowledge, skills, and practices to elementary level ELL students.

Our team has had the opportunity and challenge to adapt existing Scratch-based CT curriculum to support the groups mentioned above: (1) classroom teachers with minimal CT experience and (2) English language learners. The purpose of this paper is to share our journey of identifying aspects of the curriculum that would benefit from modification, to describe the changes that were made, and then to justify them. Throughout the paper, our insights and observations of this process will be infused.

2 CONTEXT OF THE PROJECT

The IMPACT Curriculum is an introductory Scratch curriculum that is designed to teach computational thinking to English language learners and culturally and linguistically diverse students in 4th-6th grades. It draws on the Creative Computing curriculum developed by the ScratchEd team at the Harvard Graduate School of Education. It also includes elements from Code.org, Wonder Workshop, and Bootup.org. The curriculum was then further developed by the IMPACT team – including San Francisco Unified School District's CSinSF.org; the Digital Learning Lab at the University of California, Irvine; the CANON Research Lab at the University of Chicago; and teachers and administrators in Santa Ana Unified School District

– with funding from the National Science Foundation and the US Department of Education.

The IMPACT curriculum that is discussed in this paper is part of a project that aims to develop and pilot instructional materials for teaching computational thinking in upper elementary school classrooms in a school district that is comprised primarily of Latino students (98%) and English language learners (60%). Over the course of three years, teachers who volunteered to teach the curriculum were trained in the summer, attended a monthly teacher meeting for continued support, and then taught the curriculum for one-hour a week. Few of the teachers have had previous experience working with CT or the Scratch programming language. The intention of the project is to scale up the number of teachers who teach the curriculum in the school district each year, beginning with six grade 3-5 teachers in Year One, six fourth-grade teachers in Year Two, and 10 fourth-grade teachers in Year Three. A year-long professional development and curriculum were developed as resources for this scale. Due to COVID-19, Year Three required a shift from face-to-face to a virtual environment for the delivery of the instruction as well for the summer and monthly professional development for the teachers. Being a Research Practitioner Partnership (RPP) project, teachers are involved in the evolution of the curriculum through their ideas for improvement, observations, and student feedback. The development of the curriculum was designed to be an iterative process. This paper will describe the decision-making processes in modifying components of the Year 2 curriculum for implementation in Year 3, with a focus on best practices for our ELL and novice educators.

3 INITIAL CURRICULUM: IDENTIFYING NECESSARY MODIFICATIONS

The Year Two curriculum was created by teams that understand the Scratch environment, CS pedagogy, standards, and the target elementary school population. Most key components of the curriculum were in place in the Year Two curriculum. Using Van den Akker's framework of Curriculum Components [18] as a checklist, researchers found that the Year Two curriculum satisfactorily fulfilled six of ten Curriculum Components (see Table 1). However, through teacher feedback, student and classroom observations, and specialist review of the curriculum, four of the components were identified for modification to better support teachers and ELL students: content, learning activities, materials and resources, and assessment.

4 MODIFICATIONS FOR TEACHERS

The volunteer teachers in this project are seasoned classroom teachers who are experienced in teaching ELL learners. The challenge is to acclimate these experienced ELL classroom teachers to the Scratch environment and to providing instruction for CT concepts and processes that they have not yet internalized.

4.1 Materials and Resources

The curriculum component of materials and resources answers the question: With what are the students learning with? Through classroom observations and teacher feedback the need for more curricular guidance became apparent. With teachers volunteering

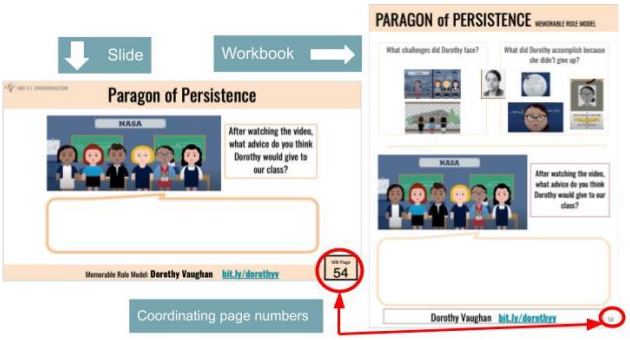

Figure 1: Sample Slide and Workbook Pages

to do extra work learning and teaching new concepts, it is a team priority to make sure the materials and supporting resources are as well-designed and accessible as possible.

4.1.1 Creation of corresponding lesson slide decks . During a classroom observation, the teacher was standing in front of the classroom reading from printed pages of the lesson plan while students had their Student Workbooks open to different pages in the unit. To reduce the sole dependence on oral comprehension and to provide more visual cues to keep students on track, slide decks were created for each lesson for teachers to project to a screen. They help to build continuity in the lesson and to help students stay on track. The page number of the Student Workbook is clearly labeled on the lower left corner of corresponding presentation slides (See Figure 1.) Such synergy across materials was essential for a smooth teaching and learning process.

4.1.2 Transforming Lesson Plans into a Teacher's Guide. The Year 2 curriculum provided lesson plans with one or two pages with objectives, lesson notes, a link to language frames used within the lesson, a description and estimated time of each activity in the lesson. The 23 lesson plans were available on a website to download individually. Teachers requested more immediate access to the lessons and more detailed guidance in actualizing them. We chose to consolidate the lessons into 5 units and to house them in one downloadable document that is referred to as the Teacher's Guide.

Each unit opens with standard curricular components of the learning objectives, alignment to standards, the unit's "Big Idea", the "Unit Walk Through", and a link to the unit's online resources. For each lesson, the "Learning Objectives" are at the top, followed by the "Learning Activity Summary" where the suggested amount of time is provided. There is a list of necessary "Student Materials", and a section called "Teacher Preparation" lists what is needed to teach the lesson.

The Teacher's Guide lays out the lesson slides and provides key points, insights, and even a suggested script for navigating conversations on a concept. Images of the lesson slides are provided in the Teacher's Guide to help teachers visualize the flow of the lessons, an idea borrowed from the "Bytes of AI" curriculum[1].

[1]https://ai-4-all.org/open-learning/resources/

Table 1: Van den Akker's Framework of Curriculum Components [18]

Curriculum Components		Existing Curriculum
Rational or Vision	Why are they learning?	To introduce elementary school-level students to CS concepts, processes, and the culture of the discipline.
Aims & Objectives	Toward which goals are they learning?	To demonstrate understanding and the ability to apply the concepts of algorithm, sequence, event, loop and synchronization.
Content	What are they learning?	Each of the 5 units focus on a CT concept: 1. Algorithm 2. Sequence 3. Event 4. Loop and 5. Synchronization
Learning Activities	How are they learning?	Learning activities include Code.org unplugged activities, scaffolded exercises using Use-Modify-Create and TIPP&SEE [14].
Teacher Role	How is the teacher facilitating learning?	The teacher guided students through the steps of the curriculum, modeling, overseeing, and answering questions.
Materials and Resources	With what are they learning?	Materials and resources included a colored student workbook and online lesson plans.
Grouping	With whom are they learning?	Students are taught as a whole class by their fourth grade classroom teacher.
Location	Where are they learning?	In Years One and Two, instruction took place in the physical classroom. In Year Three, instruction took place virtually on Zoom or on Google Meet.
Time	When are they learning?	Students are taught for one hour a week during a language art class period.
Assessment	How to measure how far learning has progressed?	Student learning was assessed by a Computer Science Identify measure, through a Bebras-type measure, student interviews, and by analyzing student projects.

The question of how much detail to provide in the Teacher's Guide arose as it was being developed. When asked, some teachers strongly favored detailed description and instruction for each slide to the extent of providing scripts to say for slides that contain particularly important concepts. The drawback of providing extensive detail is the time required to digest it and the difficulty of scanning it as one teaches. Other teachers preferred a one page summary of the steps in each lesson for teachers to have in hand during the lesson to use as a guide. The trade-off is less information and guidance. Teachers have different preferences. The intention is to have two versions of guidance for teachers: one detailed, multi-page guide, which currently exists, and one brief at-a-glance version.

Another alteration of the Teacher's Guide was a direct request of a teacher. She asked for a logical mid-lesson stopping point for each lesson so that teachers could teach two 25-minute lessons instead of one 50-minute lesson. Each of the 23 lessons in the curriculum is designed to be 50 minutes long. Teachers were approved by the district to teach 50 minutes of the IMPACT curriculum each week in place of language arts instruction. A concern was raised through teacher feedback and through classroom observations that having a lesson only once a week was too infrequent and required excess time in reminding students where they had left off last time. In response, there is now a "Turn and Talk" partner discussion question in the middle of each lesson to signal a stopping point and provide teachers with a quick way to re-calibrate at the beginning of the second lesson. This discussion raises the conversation of where CT and CS can fit regularly in the elementary school day and how the frequency of instruction may affect learning outcomes.

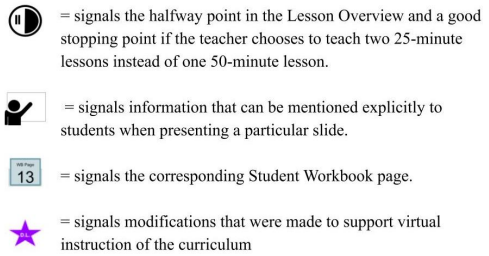

Figure 2: Symbols in the Teacher's Guide

Lastly, visual symbols were added to highlight features of the Teacher's Guide in response to a suggestion informed by a graphic design perspective. These symbols enable teachers to more efficiently scan pages to find what they are looking for. (See Figure 2.)

5 MODIFICATIONS FOR ELL STUDENTS

The pedagogy interwoven in the Year Two curriculum reflects practices supported by computer science educators and researchers, such as the 5e instructional model [1], Use-Modify-Create and TIPP & SEE [14]. An important aim of our project is to also consider how best to help ELL students engage in and to internalize the CT aims of the curriculum. We made modifications to support ELL learning in the two areas: learning activities and assessment.

5.1 Content

The curriculum component of content answers the question: What are students learning? The content in the Year Two curriculum strongly supports the aims of the CT curriculum, as it had been collectively created by educators and researchers who are at the forefront of the computer science education field. The content has been aligned with international ISTE and national CSTA Computer Science Standards, as well as with the California ELD state standards. Since our project specifically targets ELL students, we wanted to include content that ELL students could identify with. In line with asset-oriented pedagogies [10, 15], which view the diversity that students bring to the classroom as assets instead of deficits, we sought to incorporate aspects of our target population's cultural background in each unit. Already, the curriculum featured an activity that revolved around the Mexican tradition of the ofrenda. We decided to add a 15-minute "Memorable Role Model" activity to each unit. We identified role models who have made a profound contribution to the computing world and who had characteristics that our target population could identify with, such as being female and having a strong sense of imagination (Ada Lovelace), being an immigrant, a female from Mexico, and someone who says "yes" to opportunities (Margaret Zoila Dominguez), being black and an inventor (Mark Dean), and being female, being black, and being persistent (Dorothy Vaughn). The learning activities for the Memorable Role Model content includes animated 3-minute biographies, reflection, and classmate and class discussion.

5.2 Learning Activities

The curriculum component of learning activities answers the question: How are students learning? In other words, what are students doing to explore, practice, and learn the target concepts and processes? The changes that were made to the learning activities were motivated by cognitive load theory [11] and instructional strategies developed for ELL students in STEM classrooms, as CS is within the STEM domain [13]. The impetus for these changes emerged through classroom observations and curriculum review by an English language instruction specialist.

5.2.1 Modifications to reduce cognitive load. The cognitive load theory acknowledges that meaningful learning occurs when cognitive processing does not exceed the learner's available cognitive capacity. Mayer & Moreno [11] present problems that occur when the cognitive demand of multimedia learning is beyond the processing ability of the learner and they share solutions that can inform instructional designers. The cognitive demands on ELL students learning CT skills and processes are high. One modification that was made to lighten the learner's cognitive load was reducing ambiguity and wordiness of some instructions by using more common vocabulary, simpler sentence structures, and eliminating unnecessary detail. The size of the font as well as the density of the text on the Student Workbook pages were considered. Additionally, we sought to signal, or cue, learners as to how the information in the curriculum is organized. For example, we color-coded each of the five units to provide a visual cue of what unit the class was on. Consistent activity sequence and headings throughout the workbook build a sense of continuity and familiarity for learners. Lastly, we sought to reduce the learner's linguistic load by aligning images

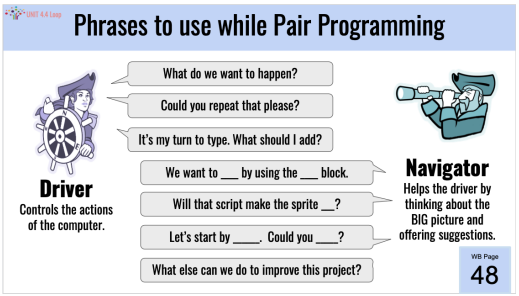

Figure 3: Linguistic Frames for Pair Programming

that represent a CT concept with the word of the particular concept. For instance, when introducing the concept of "algorithm", learners see a picture of a flowchart right above the word "algorithm", providing students with both a textual and symbiotic representation of the concept in both the slides and the Student Workbook.

5.2.2 Engaging students in disciplinary practices. Engaging ELL students into the disciplinary practices of computer science familiarizes students with what computer scientists do, the language that computer scientists use, problems that computer scientists face, and processes that computer scientists take part in. The Year Two curriculum offered many computer-science related discipline practices, such as pair programming experiences and plan and build lessons as students build unique computational artifacts. We added language frames to the pair programming activity to scaffold the interactions for ELL students [7]. By explicitly drawing attention to and discussing patterns of language used in STEM and computer science processes and discussions in the classroom, ELLs can progress in both their computer science understanding as well as in their English language development [16]. The language frames provide the linguistic support to allow students to learn the information they need from their partner; they model appropriate language to navigate social situations. (See Figure 3). For example, a common occurrence during pair programming is for the student with more programming experience to do all the work; it is often difficult for the less experienced student to speak up and to claim the role of "driver" who types in the code. The language frame "It's my turn to type. What should I add?" provides both the means to express who's turn it is and reinforces the expectation of students fulfilling the two roles of driver and navigator. Another sentence frame models a way to kindly point out a possible bug: "Will that script make the sprite . . . ?" Language frames can help all students to better understand the two roles of pair programming as well as how to communicate confidently and sensitively with each other.

The practice of self-evaluation and reflection is essential for growth in all disciplines. Upon completion of each project, students respond to the "Reflection Journal", which is a set of three questions: the first is a question about the key concept of the unit, the second is about the process of creating the project, and the third reflects on how the student feels about the project itself. These questions are supported by language frames that students can choose to use or not.

Another important computer science disciplinary practice is debugging, the process of figuring out why the program is not running

as intended and then finding a solution. Having students debug problematic code has worthwhile benefits: (1) they experience the problem solving process and have the opportunity to learn nuances of the Scratch environment, and (2) they are introduced to the reality that mistakes are normal and necessary for learning. We created exercises that lead students through the process of debugging existing code in order to draw attention to common pitfalls and to familiarize them with the concept of debugging before students begin building each of their four unit projects.

5.2.3 Engage students in discourse and interactions with others. As students engage in productive discourse and interactions with others, their English ability will improve [5, 9, 12]. The Year Two curriculum introduced language frames to the curriculum in order to help students articulate their understanding of key concepts. We wanted to integrate structured learning activities into the curriculum where students could use these language frames in authentic interactions. A tension exists between the value of structured, authentic student interaction and the amount of valuable time that can be allotted to those more time-consuming activities. The interactive activities that we created were: informal short "Turn and Talk" tasks, more structured pair programming tasks, options for sharing projects.

The "Turn and Talk" tasks have students turn to a classmate and share their responses to a question related to the curriculum, such as "Which scripts do the same thing?" or "Share a favorite joke that could be used in your next Scratch project." This task provides an opportunity for students to process the target objectives through the modalities of speaking and listening, it is an opportunity to share ideas and to brainstorm ideas for their projects, and invites a sense of community among the students.

To provide more structure and modeling to the pair programming activity, we created language frames to guide the interaction, as mentioned in Section 5.2.2, and we added a video created by Code.org[2] to model what pair programming looks like in practice.

To provide students with the opportunity to present their project and to have students practice a more formal register, the revised curriculum provides options with detailed instructions of the "Gallery Walk" (see diagram in Figure 4) with computers and through the "Inside Outside Conversation Activity" where students stand in two concentric circles and share for two minutes with a partner before the outside circle moves to create new partners. To exercise critical thinking and evaluation as well as interaction with classmates, we created the "Classmate Comments" activity where students review each other's projects based on the criteria presented in the rubric, which is discussed in the next section. We kept the "Two Stars and a Wish" task that was already in the curriculum, where students identify one aspect of the project that was done well and one way that it can be improved. Language frames are provided to help students articulate their ideas.

5.3 Assessment

The curriculum component of assessment answers the question: How to measure how far learning has progressed? The progress of students during year two of the study was measured by a CS Identity

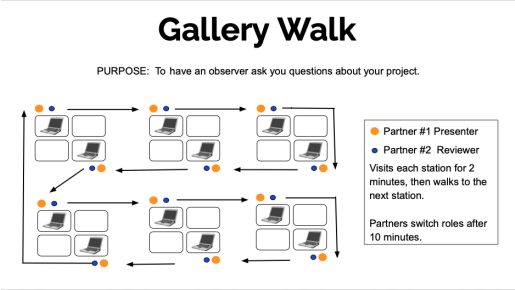

Figure 4: Gallery Walk

and a computational thinking general assessment. In addition, student projects were evaluated with an automated online assessment tool.[3] Students and teachers, however, did not receive the feedback nor the results to those measures. We wanted to add assessments to the curriculum that students and teachers could benefit from. We chose to add both informal formative assessments, looking at where students are at in the beginning, as well as a self- and peer-assessment, as mentioned in Section 5.2.2, where students reflect on their own progress and that of their classmate, for each project. The informal formative assessments include the debugging exercise also described in Section 5.2.2, the Turn and Talk tasks described in Section 5.2.3, and evaluation of the TIPP&SEE projects that students have submitted this year during virtual learning through a Google form, discussed in Section 6. The added summative assessments both revolve around rubrics that were created for each of the four projects. Students are walked through the elements of the rubric before they plan and build their projects so students have concrete criteria to aim for. The first column begins with the basic criteria that all students should achieve and the two next columns add a challenge, in order to differentiate learning for more experienced Scratch users, as shown in Figure 5. Students use the rubric to decide how they will challenge themselves as they plan and build their individual projects as well as to evaluate peer projects. The rubric provides an extra level of support for ELL students as expectations are clearly stated.

6 MODIFICATIONS FOR THE VIRTUAL CLASSROOM

These curriculum modifications took place in the winter and spring of 2020, at the start of the COVID-19 quarantine and when students began attending school 100% online. Realizing that the quarantine could continue into the following school year in September 2020, modifications were made to the curriculum to ease the transition to the virtual environment. Teachers in this project met with their students via Zoom, Google Meet, and Google Classroom. Instruction of the curriculum was challenging for imaginable reasons: unstable internet connections, distractions in the homes, sitting in front of the computer hours each day as a nine-year old, and the different nature of the teacher's presence in the online classroom. Additionally, computational thinking could not be taught in the prime morning hours when students were at their best and ready

[2]https://bit.ly/drivernavi

[3]http://people.cs.uchicago.edu/ dmfranklin/assessment/

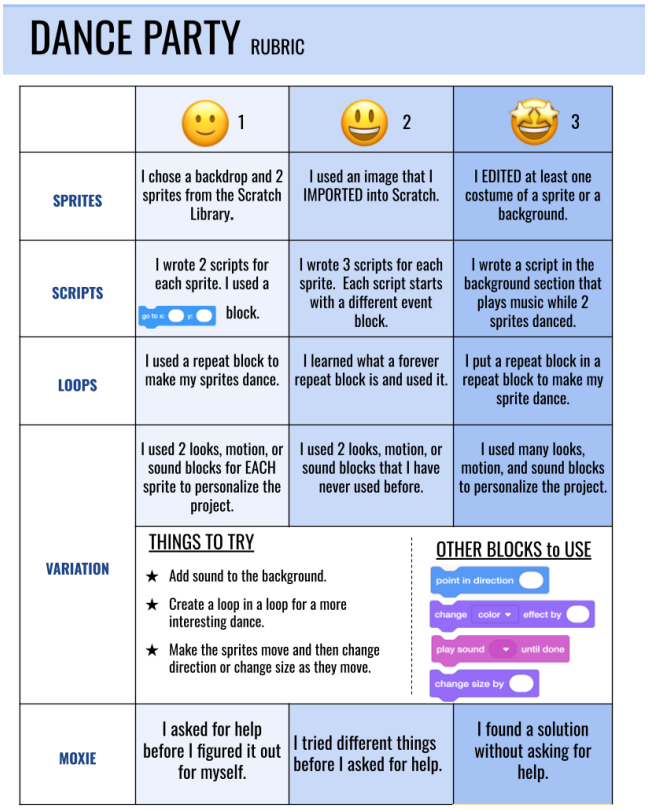

Figure 5: Example Rubric

to learn, as those hours were reserved for the core subjects of math and language arts. We made our best effort to help teachers transfer the instruction of our curriculum to the online environment, as discussed in the next session.

6.0.1 Google Forms. In order for teachers to monitor student progress, which would normally happen by walking around the classroom and looking at completed activities in the Student Workbook, Google forms were created so students could input their ideas and so teachers could conveniently access the responses for some of the activities in the curriculum. The Canon Lab in the University of Chicago created Google forms for the TIPP&SEE learning activities, which breaks down a project and walks students deliberately through each aspect of the project to increase student understanding. Google forms were also created for the Memorable Role Model and Reflection Journal activities. During the 2020-2021 school year, schools involved in this study met entirely online. The Google form assignments have been a convenient way to view student work and to give students a sense of accountability. Because of the ease of collecting student responses and because of students' familiarity and comfort with responding to online prompts, the Google forms assignments may have a place in the curriculum even when classes are held in face-to-face environments.

6.0.2 Unplugged Activities. The Year Two curriculum incorporated three unplugged activities in order to present CT concepts in another context. While teaching in an online environment, teachers found creative, yet somewhat complicated, ways to use breakout rooms and chat to have students carry out the Code.org Scratch Charades and the Big Event unplugged activities.

7 CONCLUSIONS

We have described how curricular improvement needs were identified, the changes that were made, and the rationale for why they changed in that way. There were a number of challenges that we faced in doing so. One of the challenges was deciding on which instructional strategies to integrate into the curriculum to support ELL students who are learning CT concepts. Our changes were informed by a variety of theories that have proven to be effective in ELL settings, but have not been integrated into a framework specifically for the instruction of CT. A second challenge was finding consensus on which changes to make and how to make them, once the need was determined through teacher and student feedback, observations, and specialist review of the curriculum. In the midst of various opinions about the direction of the curriculum, we primarily resorted to our teacher and student feedback to align with our goals for this research practitioner partnership.

This paper describes our experience of modifying an established CT curriculum for upper-elementary-aged students to support elementary school teachers with little CT-teaching experience and to support ELL students. We hope the details of our process and decisions about the curriculum can provide insight and direction to others who have the opportunity and responsibility to adapt curriculum to support teachers who are teaching new content and ELL students. Our team will continue to iterate and to improve the curriculum until Year 4. The implementation of the curriculum is the next area of focus. We will continue to support the teachers in the logistics of teaching what we have created.

This curriculum is publicly shared under a Creative Commons License. It can be downloaded at our website to use for non-profit purposes or to look closer at aspects of the curriculum mentioned in this paper.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation through Grants #1923136 and #1660871 and by the United States Department of Education through Grant #U411C190092. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the United States Department of Education.

REFERENCES

[1] Rodger W Bybee. 2014. The BSCS 5E instructional model: Personal reflections and contemporary implications. *Science and Children* 51, 8 (2014), 10–13.
[2] Code.org, CSTA, & ECEP Alliance. 2020. *2020 State of Computer Science Education: Illuminating Disparities.* https://advocacy.code.org/stateofcs
[3] National Science Technology Council. 2018. *Charting a course for success: America's strategy for STEM education.* https://www.whitehouse.gov/wp-content/uploads/2018/12/STEM-Education-Strategic-Plan-2018.pdf
[4] National Center for Education Statistics. 2020. *English Language Learners in Public Schools.* https://nces.ed.gov/programs/coe/indicator_cgf.asp#f2

[5] Susan M Gass and Alison Mackey. 2007. Input, interaction, and output in second language acquisition. *Theories in second language acquisition: An introduction* 175199 (2007).
[6] Shuchi Grover. 2018. *The 5th 'C'of 21st century skills? Try computational thinking (not coding).* https://www.edsurge.com/news/2018-02-25-the-5th-c-of-21st-century-skills-try-computational-thinking-not-coding
[7] Lisa Hoffman. 2013. Talk Like a Scientist! Simple" Frames" to Scaffold the Language of Science. *Online Submission* (2013).
[8] Enoch Hunsaker. 2020. Computational thinking. *The K-12 educational technology handbook* (2020).
[9] Michael H Long. 1981. Input, interaction, and second-language acquisition. *Annals of the New York academy of sciences* (1981).
[10] Francesca A Lopez. 2017. Altering the trajectory of the self-fulfilling prophecy: Asset-based pedagogy and classroom dynamics. *Journal of Teacher Education* 68, 2 (2017), 193–212.
[11] Richard E Mayer and Roxana Moreno. 2003. Nine ways to reduce cognitive load in multimedia learning. *Educational psychologist* 38, 1 (2003), 43–52.
[12] Sarah Michaels and Cathy O'Connor. 2012. Talk science primer. *Cambridge, MA: TERC* (2012).
[13] National Academies of Sciences, Engineering, and Medicine and others. 2018. *English learners in STEM subjects: Transforming classrooms, schools, and lives.* National Academies Press.
[14] Jean Salac, Cathy Thomas, Chloe Butler, Ashley Sanchez, and Diana Franklin. 2020. TIPP&SEE: A Learning Strategy to Guide Students through Use-Modify Scratch Activities. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 79–85.
[15] Martin Scanlan. 2007. An asset-based approach to linguistic diversity. *Focus on Teacher Education* (2007).
[16] Mary J Schleppegrell. 2013. The role of metalanguage in supporting academic language development. *Language learning* 63 (2013), 153–170.
[17] Marguerite Ann Snow, Myriam Met, and Fred Genesee. 1989. A conceptual framework for the integration of language and content in second/foreign language instruction. *TESOL quarterly* 23, 2 (1989), 201–217.
[18] Jan Van den Akker. 2007. Curriculum design research. *An introduction to educational design research* 37 (2007).
[19] Sara Vogel, Rafi Santo, and Dixie Ching. 2017. Visions of computer science education: Unpacking arguments for and projected impacts of CS4All initiatives. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 609–614.
[20] Joke Voogt, Petra Fisser, Jon Good, Punya Mishra, and Aman Yadav. 2015. Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies* 20, 4 (2015), 715–728.
[21] Aman Yadav, Chris Mayfield, Ninger Zhou, Susanne Hambrusch, and John T Korb. 2014. Computational thinking in elementary and secondary teacher education. *ACM Transactions on Computing Education (TOCE)* 14, 1 (2014), 1–16.