Title
Teaching Computational Thinking to Multilingual Students through Inquiry-based Learning

Authors
Jacob, Sharin
Nguyen, Ha
Garcia, Leiny
et al.

Publication Date
2020

Peer reviewed

Teaching Computational Thinking to Multilingual Students through Inquiry-based Learning

Sharin Jacob
School of Education
University of California, Irvine
Irvine, US
sharinj@uci.edu

Ha Nguyen
School of Education
University of California, Irvine
Irvine, US
thicn@uci.edu

Leiny Garcia
School of Education
University of California, Irvine
Irvine, US
leinyg@uci.edu

Debra Richardson
Bren School of ICS
University of California, Irvine
Irvine, US
djr@ics.uci.edu

Mark Warschauer
School of Education
University of California, Irvine
Irvine, US
markw@uci.edu

***Abstract*— Central to the theory of learning are inquiry-based approaches to education. Whereas there is a plethora of research on inquiry learning in the domain of science [19, 20], few studies have analyzed how inquiry-based learning can be applied to computer science education, and how different approaches to inquiry may benefit diverse learners. This is one of the first studies to analyze teacher enactment of inquiry-based learning during the implementation of an upper elementary, computational thinking curriculum, and to explore how teacher approaches to inquiry appear to support or constrain multilingual students' development of computational thinking and computer science identities. Design-based research was used to iteratively develop, test, and refine the inquiry-based curriculum, which aligns with computer science and literacy standards, provides linguistic scaffolding, and integrates culturally responsive materials. We adopt a cross-case mixed-methods design to collect data from five teachers and 149 students including detailed field notes, teacher interviews, student computational artifacts, and student identity surveys. Through analyses of teacher moves, we find that teachers adopt different approaches to inquiry that can be indexed along a continuum ranging from open to closed. Patterns in student data revealed that those who received more structured inquiry lessons developed more sophisticated computational artifacts and showed greater identification with the field of computer science. Findings from this study are being used to add more structured inquiry approaches to the next iteration of our curriculum, including integrating USE/MODIFY/CREATE models into lessons and applying metacognitive strategies from reading research to students' programming activities.**

Keywords—inquiry-based learning, computer science, computational thinking, multilingual, English learners

I. INTRODUCTION

Considerable effort has been dedicated to integrating computer science into K-12 education for students who are traditionally underrepresented in STEM (e.g., women, students of color, students with disabilities). For example, the White House's 2016 Computer Science for All (CSforAll) initiative seeks to equip all K-12 students with the computing and computational thinking skills necessary to become creators, and not just consumers, of technology [1]. To help realize this goal, the National Science Foundation developed the CSforAll program which focuses on developing research-practice partnerships (RPPs) that foster the types of theory and practice needed to bring computer science and computational thinking to all students in K-12 schools. With a focus on CSforAll, educational policy makers and stakeholders have shifted their attention to developing computer science pedagogy and materials that meet the needs of diverse learners.

While efforts to combat underrepresentation in computer science education have been numerous and laudable [2, 3], little attention has been paid to broadening participation for multilingual students [4, 5], or those who speak a language other than English at home. This is especially important for the large and growing Latinx population--which grew from 9 million (6% of U.S. population) in 1970 to 59 million (18% of population) in 2017, and is projected to reach 132 million (30% of population) by 2050, but is seriously underrepresented in CS education and achievement. For example, in California, the site of this study, Latinx students constitute 54% of the K-12 population, but only 22% of advanced placement CS test takers [6]. A number of important obstacles hinder CS study for Latinx students, including reduced access to home computers or family members who are knowledgeable about CS [7], lack of Latinx role models in CS whether through direct experience or through media representations [8], fewer course offerings in CS in Latinx-neighborhood schools [6], and decontextualized and individualized methods of CS instruction that are not a good match for the cultural values of Latinx students and families.

Inquiry-based learning has shown particular promise for engaging culturally and linguistically diverse students in STEM education [9]. Given the efficacy of inquiry-based learning for raising science achievement for multilingual students [9], inquiry-based approaches may also be effective for engaging the nation's growing Latinx student population in computer science education. Despite the promise of inquiry learning approaches, it remains unclear as to whether more structured or open inquiry approaches are more effective for engaging these student in computer science. Proponents of open inquiry argue that the freedom to construct and conduct investigations develops students' higher order thinking skills, disciplinary

978-1-7281-7172-2/20/$31.00 ©2020 IEEE

knowledge, and inductive methods of inquiry [10]. Those who prefer structured approaches claim that providing students systematic methods for conducting investigations supports the development of content knowledge, scientific skills, and a nuanced understanding of the discipline [11]. Structured approaches are further thought to prevent lost opportunities that arise from students getting stuck due to minimal guidance [12].

The purpose of this study is to analyze teacher enactment of inquiry-based learning during the implementation of an upper elementary, computational thinking curriculum, and to explore how teacher approaches to inquiry appear to support or constrain multilingual students' development of computational thinking. Whereas there is a plethora of research on inquiry learning in the domain of science [13,14], few studies have analyzed how inquiry-based learning can be applied to computer science education. The curriculum has been adapted from a grade 3-5 sequence developed by Computer Science in San Francisco, a path breaking initiative that seeks to normalize K-12 computer science instruction in schools across the San Francisco Unified School District. It has been adapted to integrate inquiry-based approaches to learning, align with computer science, English language development, and literacy standards, provide linguistic scaffolding, and integrate culturally responsive materials. We adopt a cross-case mixed-methods design to collect data from five teachers and 149 students including detailed field notes, teacher interviews, student identity surveys, and student computational artifacts. Through analyses teacher moves, we find that teachers adopt different approaches to inquiry that can be indexed along a continuum ranging from open to structured. Patterns in student data revealed that those who received more structured inquiry lessons developed more proficient computational artifacts and showed greater identification with the field of computer science. Findings from this study are being used to add more structured inquiry approaches to the next iteration of our curriculum including integrating USE/MODIFY/CREATE models into lessons and applying metacognitive strategies from reading research to students' programming activities.

We address the following research questions: (1) In what ways, if any, did teachers endeavor to teach computer science as inquiry to multilingual students in their classrooms? (2) How do differences in teachers' approaches to inquiry appear to support or constrain students' development of computational thinking skills and computer science identities?

II. THEORETICAL FRAMEWORK

A. Inquiry-based Learning and Computational Thinking

Inquiry-based learning involves engaging students in authentic scientific practices and methods for the purpose of constructing knowledge [15]. Through student engagement in exploration, experimentation, and hands-on activities, inquiry-based learning provides a powerful mechanism for providing authentic contexts for language use [16]. During open inquiry-based learning, students develop questions and participate actively in open ended interrogations to discover and construct new knowledge [17]. During structured inquiry, teachers model methods and procedures for conducting investigations [14]. Inquiry-based learning emphasizes problem solving and students apply multiple problem solving approaches, practices, and skills as they conduct their investigations [18]. Since the mid-1990s, teachers have been encouraged to better meet the needs of language learners by integrating inquiry-based approaches to make instruction more engaging, concrete, and meaningful [19].

Pedaste et al. [20] conducted a systematic literature review identifying key features of inquiry-based learning and synthesized a framework incorporating all elements of inquiry that persisted across models. This general framing of the phases of inquiry relates inquiry to the scientific method, requiring a measure of reconsideration in its application to the field of computer science. Notably, scientific inquiry focuses on identifying and evaluating hypotheses to understand a set of principles governing the physical world. This objective contrasts with inquiry in computer science, which focuses on constructing and testing logical processes to address an abstract computational problem. Instruction in computer science then seeks to develop a set of skills, practices, and dispositions collectively referred to as computational thinking, representing an ability to formulate thoughts and questions for interpretation by a computer to achieve desired results [21]. Constructing a theoretical framework for applying inquiry-based learning to the field of computer science is beyond the scope of this paper.

In its implementation, inquiry-based learning can be structured or unstructured based on the teaching and learning goals of a lesson or unit of instruction. Windschitl [14] conducted a multiple case study investigating how teachers perceived and enacted inquiry in their science classrooms and developed a continuum of inquiry demarcated by the degree of freedom students have in developing and conducting investigations. At the structured end of the continuum lies confirmatory experiences, in which students are provided a systematic method for authenticating scientific principles. Next to confirmatory experiences lies structured inquiry, in which the teacher presents a scientific concept, question, or hypothesis and students are prescribed a procedure for exploring it. The next level is guided inquiry, where the teacher provides a problem to be solved but leaves the method of investigation up to the students. The most independent form of inquiry is open inquiry, in which students identify their own concepts or questions and devise their own methods of investigation. We use this continuum to characterize participants' approaches to teaching computer science lessons to multilingual students.

B. Review of the Literature

Research on engaging students in Science, Technology, Engineering, and Mathematics (STEM) through inquiry-based learning has been well established. A recent meta-analysis conducted by Estrella at al. [22] examined the effectiveness of inquiry instruction in increasing STEM achievement for elementary language learners. An analysis of 26 articles indicated that inquiry-based instruction produced significantly greater results on measures of science achievement than traditional instruction. Furthermore, elementary students who

participated in a blended program that integrated linguistic scaffolding with science inquiry-based unit plans showed statistically significant increases on California English Language Development Test (CELDT) and California Standards Test for English Language Arts scores compared to students in a traditional program [23].

A growing body of research on using inquiry to teach computational thinking demonstrates benefits for students [24, 25], however little research specifically focuses on the types of inquiry approaches and levels of support that develop computational thinking and identity development for diverse learners. Reiser [26] acknowledges the potential of inquiry to provide authentic learning contexts for students, but also articulates the challenges inherent to inquiry learning. For example, students need to acquire sufficient foundational discipline specific knowledge to conduct investigations, and often focus on finding the right answer to a problem as opposed to identifying the principles underlying answers. To ameliorate these issues, he proposes presenting more structured problem solving activities and problematizing subject matter to promote deeper understanding of content.

C. Overview of the computational thinking curriculum

Researchers worked collaboratively with teachers to adapt an existing grade three through five curriculum created by Computer Science in San Francisco. The curriculum was adapted to meet the needs of the district's culturally and linguistically diverse students. This was achieved by 1) integrating inquiry-based approaches, 2) aligning the curriculum with computer science and literacy standards, 3) providing linguistic scaffolding, and 4) including culturally responsive pedagogy and materials.

First, researchers and teachers aligned materials with the Common Core State Standards for English Language Arts (ELA), and the California Department of Education English Language Development (ELD) Standards. Linguistic scaffolds were developed to promote and leverage academic language proficiency for language learners. Researchers and teachers developed linguistic frames to scaffold both the academic language related to computer science concepts as well as the functions of social interaction. To integrate inquiry-based approaches, we utilized the "5 E" model of inquiry to guide unit development. Bybee [27] drew from constructivist approaches to learning to construct the "5E's" model, which includes five indicators of inquiry-based instruction: Engage, Explore, Explain, Elaborate, and Evaluate.[1] While we integrated the phases of inquiry into the curriculum, we encouraged teachers to use their own judgement when determining the level of structure necessary to meet students' needs. Finally, the partnership paid special attention to integrating literacy into the computational thinking curriculum. Culturally responsive stories depicting diverse characters who pioneered the computer science and engineering fields were selected to make the content relatable to students.

III. METHOD

A. Study Context and Participants

This study took place in five upper elementary (grades 3-5) classrooms across a large urban school district. The district in which the study is situated has among the highest percentages low-income students (91%), Latinx students (96%), and English learners (63% in elementary grades) in the nation. Due to attrition, five out of the seven original teachers were selected for this study. All the students in their classes (total N=149) participated in the project and thus were part of the study. Student demographics at the classroom level broadly mirror those at the district level.

B. Data Sources & Analysis

The participating teachers piloted the year long, five-unit computational thinking curriculum in their classrooms once a week for a lesson duration of fifty minutes. The researchers conducted weekly classroom observations and took detailed field notes on the types of instructional strategies used. The McGill Inquiry Teacher Short Interview (MITSI) protocol was used to interview the five participating teachers in their classrooms. A rubric for scoring Scratch projects developed by SRI International was utilized to assess students' 1) overall proficiency in programming, 2) user experience, and 3) the use of coding and computational thinking constructs [28]. The researchers conducted interrater reliability checks on the rubric data. After each scorer completed ten projects, interrater-reliability ranged from 75% to 80. Finally, this study adapted the "Is Science Me?" [29] survey to computer science to measure student attitudes towards and identification with the field of computer science.

The generation of codes and categories for classroom observations and teacher interviews is situated within a procedural, deductive, frame of analysis [30]. This process consisted of reading the data multiple times to categorize inquiry learning phases and subcategories within each phase. Codes and categories were then compared within and across each case to determine the types of inquiry being enacted in each classroom.

To assess students' Scratch projects, a sum score for each category (e.g., overall proficiency, design mechanics, user experience) was calculated and z-score transformed. A one-way ANCOVA with post-hoc Tukey HSD test was conducted to examine whether there was a statistically significant difference among the teachers on the computational thinking criteria for the end-of-unit projects, controlling for student background information (i.e., computer access and parental education). Assumptions of normal distribution and homoscedasticity were met. When the ANOVA tests indicated that the scores across

[1] Engage involves stimulating interest on a topic. Explore refers to conducting hands-on activities in which students grapple with a problem or phenomenon. Explain means to leverage the language and conceptual understanding used by students to develop explanations. Elaborate involves providing opportunities for students to apply what they have learned. Finally, evaluate means engaging students in reflecting on their own understanding.

classes were significantly different, this study performed pairwise t-tests with Bonferroni and Holm adjustments to examine which class was substantially different from the other classes.

Finally, the *Is Science Me?* survey mostly included three-point Likert scale items. This study calculated internal consistency using McDonald's omega instead of Cronbach's alpha, as Cronbach's alpha may be less accurate when data come from ordinal items with few response options [31]. A confirmatory factor analysis using polychoric correlations matrix was conducted based on the theorized constructs on the pretest and posttest datasets. Both datasets showed good model fit according to the guideline in [32]: CFI and TLI higher than or equal to .90, RMSEA smaller than .05, SRMR smaller than .08. Because the survey items were ordinal and did not approximate a normal distribution, this study performed the Wilcoxon matched-pairs signed-rank test to measure the changes from pre to posttest for each survey item for each teacher.

IV. Findings

Based on classroom observations and detailed field notes, it became apparent that teachers enacted the curriculum differently across the five classrooms. The first four teachers used inquiry in different points along the continuum mentioned in the theoretical framework of this paper, with Ellen using open inquiry, Juanita using guided inquiry, Jenny using a combination of guided and structured inquiry, and Helen using confirmatory experiences. The fifth teacher, Sue, did not use inquiry-based instruction and instead adopted a direct, explicit approach to teaching computer science content to her students. What follows are descriptions of teaching episodes for each classroom that are representative of how teachers conducted inquiry in their classrooms.

A. Open Approaches to Inquiry

Two of the teachers, Ellen and Juanita, exemplified open inquiry approaches in mentoring their students through various aspects of computer science research. On a typical day, both teachers would orient students to computational thinking concepts through activities structured around focal phenomena, and facilitate collaborative sense-making of key computational thinking concepts and practices. During investigation, Ellen often promoted independent learning in her classroom, acting as a facilitator of the research process by equipping students with the resources and strategies necessary for conducting open-ended investigations.

Ellen openly expresses her views of computer science learning as a research process in which students seek out the resources necessary to solve complex problems.

> Ellen: When you get stuck, we have resources. I am not the greatest resource because I am learning with you too. I can guide you to resources. Your peers are a resource…When you get stuck, we have resources...Am I your only resource?
> Students: No
> Ellen: You know, many of you have learned that I am not the greatest resource. I'm not wanting to be. Why, because I'm learning this with you too. Okay. So I can kind of guide you in how to be resourceful. I'm finding more and more places that I can get help when I need it and that's what you need to do as scholars.

Ellen describes herself as a being a *guide* for her students, using herself as a model to illustrate the types of habits (i.e., being resourceful, help-seeking) her students can engage in as scholars. To this end, she disrupts her traditional role as teacher to create a more horizontal, symmetrical space in which students and the teacher co-navigate computer science research. She provides the learning environment and resources necessary for students to ask and answer complex problems, and steps aside to facilitate scholarly activity.

Juanita similarly promotes independence during the investigation phases of inquiry, but unlike Ellen, models methods of problem formulation. She also disrupts the direct instruction model by encouraging students to negotiate their own learning among peers before coming to her for answers. For example, in the excerpt below Juanita modeled the first steps in a shape drawing activity designed to teach algorithms. Students were presented with written steps for drawing shapes and a picture with the desired visual outcome (i.e. a picture of a house). However, the steps did not match the shape, that is, there were errors embedded in the steps and students were tasked with debugging the algorithm so that the steps matched the desired outcome.

> Juanita: Who can read step two?
> Anita: Draw an orange equilateral triangle with one edge lined up with the of the square
> Juanita: Do you guys see that?
> Class: No
> Juanita: Okay, again let's look at this, who could tell me where my equilateral triangle is
> Roxanne: On top of the blue square
> Juanita: Remember, equilateral means it's what on all sides.
> Class: Equal.
> Juanita: Equal. Very cool. Okay, so check that it's lined up with the top of the blue square. Ok, so do you guys see it now?
> Class: Yes
> Juanita: Now talk to your table, your elbow partner. And I want you to go through...I want you to figure out with your partner the rest of the steps and let me know what you think the bug is. Once you've figured out what the bug is, I don't want to hear any ah ah ah's or ooh ooh ooh's. Figure out what it is before you raise your hand.

In this teaching episode, Juanita defined the scope of the problem for students and modeled the first few steps for solving it, then encouraged students to rely on their peer networks to finish debugging the algorithm. To this end, she established a peer learning community in which students discuss debugging techniques with their classmates before they ask the teacher. As students were provided with a problem, but not explicitly given a procedure for solving the problem, the above example

supports the guided inquiry model. After students worked to debug the sample algorithm, students were assigned to individually create their own algorithm and drawing using only their peer networks for support, further supporting the guided inquiry hypothesis.

B. Structured Approaches to Inquiry

Two of the teachers, Jenny and Helen, tended to teach inquiry learning in a structured manner by 1) providing methods for formulating and solving problems, or 2) demonstrating and confirming key computational principles. Jenny utilized a combination of guided and structured inquiry to teach computer science to her students. She typically opened with a guided inquiry lesson and then moved to more structured approaches when students had difficulty accessing abstract concepts. In the example below, Jenny initially used the guided inquiry approach to develop students' understanding of sequence and repetition. Students were provided with multiple steps of a dance and tasked with working in teams to conceptually map the dance, using loops so as to not articulate each single step. While Jenny identified the scope of the problem for students, she did not provide with them methods for solving the problem.

> Jenny: In your team, I want you to identify the actions in this dance…write a computer code, if you were to tell someone how to do this dance, what would you do, are there repeated actions, how many times do they repeat. What would an algorithm, a code for this, look like?

In this example, students were given more freedom to develop and conduct their own investigations and the teacher focused primarily on facilitating the learning process and detecting student issues as they arose. As Jenny focused in on students' needs, she moved from guided to structured inquiry to create more efficient learning environments. For example, students had difficulty characterizing the dance on their first attempt, likely because they had not committed the sequence of moves to memory. In response, Jenny used a variety of techniques to respond to this need including replaying the dance, refocusing students' attention to counting and naming the moves, using dialogic questioning to facilitate students' recall of the dance, and physically enacting the dance before the class. Each of these instructional moves points toward a more structured inquiry approach, that is, Jenny modeled alternative methods for students to investigate the key computational concepts.

Helen provided structure for her students through confirmation experiences designed to teach computing concepts. Helen's typical investigation was highly structured and relied primarily on teacher modeling and scaffolding techniques to facilitate knowledge acquisition. To illustrate a simple example, Helen drew two sprites, or characters in the Scratch interface, on the whiteboard, prepping the class to discuss what parallelism means.

> Helen: What do I do if I want the sprites to do the same thing at the same time?

Next, Helen drew two columns, one for sprite 1 and one for sprite 2 and added blocks to each column. Then, she manipulated blocks in a variety of ways and asks students to make predictions.

> Helen: What's going to happen?

To test the concept, Helen hit a mock flag button to start a "test run" of parallel commands. When sprites 1 and 2 correctly executed their commands, students verified their understanding of how to use commands in Scratch to make two sprites take actions simultaneously.

> Helen: It worked!

Upon checking student understanding, Helen added additional layers of complexity to teach more advanced computational concepts. For example, she modeled more complicated multi-step commands for each sprite to execute simultaneously, except in this case the sprites are doing two different things at the same time (i.e., sprite 1 walks forward, sprite 2 jumps up and down). Furthermore, she purposefully included errors in her mock code to engage students in debugging during concept development.

In the above scenario, students were presented with a variety of scenarios in which parallelism could occur, and then these scenarios were verified by 'test runs.' Although she increased the complexity of her examples, students were not provided the opportunity to generate or conduct investigations on their own. Instead, they engaged in confirmatory experiences of key computational principles.

C. Direct Approaches to Teaching

Sue did not take an inquiry-based approach to teaching computer science to her students. Instead, she focused on rhythm and periodicity within her classroom, ensuring that students had access to stable routines and durable infrastructure to support knowledge acquisition. This allowed her to manage interaction and bring about predictability of sequential classroom activities and students' behaviors.

Sue predominantly used the mirroring technique to teach computer science concepts to her students. In the example below, she taught the concepts of sequence and order, initiating the mirroring activity, in which she would say "mirrors on", and the students would imitate her words and actions

> Sue: Okay. I liked what you said about the events, but when we're talking about sequence of events, what does that mean? Is it, are we talking about groups or am I talking about order?
>
> Class: Order
>
> Sue: Right? Mirrors on! Sequence (Sue waives hand motions, class repeats) is order (Sue waives hand motions, class repeats) or sequence is the order of events.

In this excerpt, Sue took a direct, explicit approach to teaching, enforcing targeted stimuli (i.e., teacher models

specific motions) and response (students mimic teacher's motions) behaviors, coupled with repetition to teach key concepts. She also reinforced correct verbal and motor associations with a clip up classroom management technique, using positive reinforcement as an example for the class of what types of behaviors are preferred.

D. Student Outcomes

First, we combined students' scores on items using the selected SRI rubric scale. When examining teachers' overall combined average rubric scores for complexity, design mechanics, user experience, and use of computer science constructs, we saw that Helen and Jenny's students performed better overall (See Figure 1). Results of the ANCOVA showed that overall, Helen's and Jenny's class performed substantially better in several criteria (See Figure 2). There were significant differences among the classes in overall complexity, $F(4, 107) = 17.33$, $p < .01$; user experience, $F(4, 107) = 5.27$, $p < .001$; CS constructs, $F(4, 107) = 9.11$, $p < .001$; and counts of different types of Scratch blocks, $F(4, 107) = 4.58$, $p < .01$, after accounting for home computer access, mother's education, and father's education. Tukey HSD test revealed that there was a significant difference in the overall scores for Ellen and Sue ($p < .001$), and Jenny and Sue ($p < .01$). For user experience, there was a significant difference between Helen and Ellen ($p = .03$), Juanita and Sue ($p < .01$) and Helen and Sue ($p < .001$). For CS constructs, there was a significant difference between Helen and Ellen ($p < .001$), Helen and Juanita ($p < .001$), Ellen and Sue ($p < .001$), Sue and Juanita ($p = .01$), and Helen and Jenny ($p = .02$). For block use, there was a significant difference for Juanita and Helen ($p = .01$), and Juanita and Jenny ($p < .01$).

There appeared to be positive changes from pre to posttest in three classes (Helen, Jenny, and Sue) in terms of students' ability beliefs, perceptions of support for computer science interests from family and friends, and perceptions of the usefulness and importance of computer science (See Figure 3). The effect size ranged from medium to large, Cohen's $d = (.46, 1.07)$.

V. Discussion

The idea of teaching science as inquiry has been well supported by research [33, 34] and has been found particularly effective for engaging English learners in STEM [22]. Recently, researchers have called inquiry learning into question [35, 36], finding that explicit instructional approaches better support learning and transfer [37]. This is one of the first studies to investigate how different approaches to inquiry support or constrain multilingual students' development of computational thinking skills. Preliminary findings indicated that more structured forms of inquiry appear to better support multilingual students in engaging in and identifying with computer science.

A. Overview of Findings

A primary aim of this study was to explore the question: "In what ways, if any, did teachers endeavor to teach computer science as inquiry in their classrooms?" One of the ways to gain

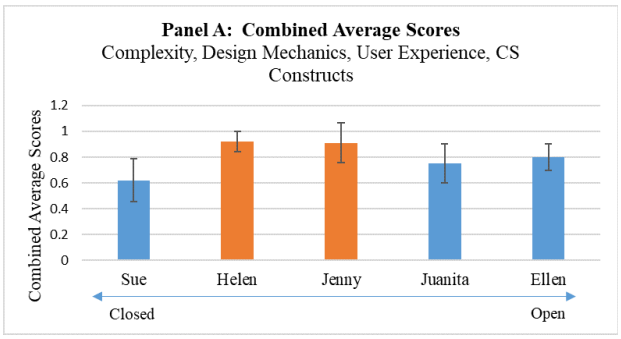

Fig. 1. Combined ratings of student Scratch projects by teacher

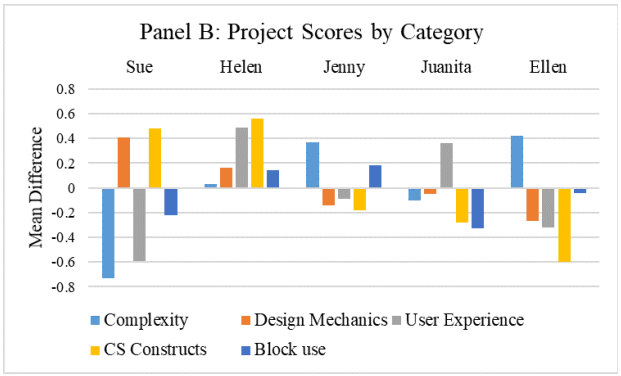

Fig. 2. Mean differences in Scratch project scores by category

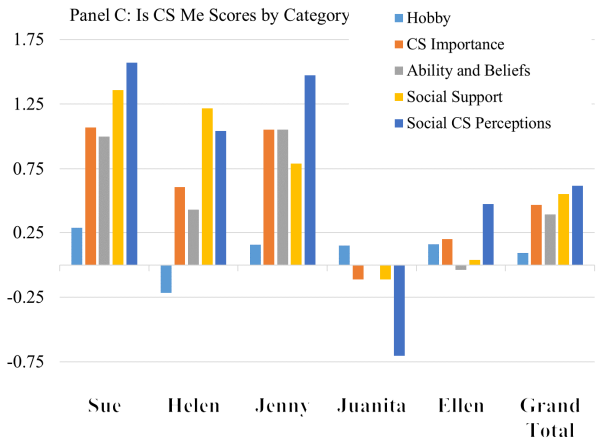

Fig. 3. Mean differences in student identity survey scores by category

insight into how teachers used inquiry approaches was to analyze how teachers enacted computer science lessons in their classrooms. By exploring the phases of inquiry that emerged from the lessons, we were able to piece together the ways in which teachers approached inquiry and index these approaches as being more open or closed along an established continuum. While the phases [20, 27] and types [14] of inquiry in science education have been well established, our study provides considerable insight into how the types of inquiry pertain to the discipline of computer science. These results extend our knowledge of the ways in which computer science can be taught

as inquiry, and how different approaches can be used to meet the needs of diverse students.

An additional aim of this study was to explore how teachers' differing approaches to inquiry appeared to support or constrain multilingual students' development of computational thinking skills. Patterns in the data revealed three clusters of teachers, those whose teaching sequences revealed more open approaches, those whose sequences revealed more structured approaches, and those whose patterns displayed direct instructional approaches. For teachers who were indexed as being more structured along the inquiry-based learning continuum (Helen and Jenny), students tended to develop more sophisticated Scratch projects. We present several conjectures for this relationship. In these classrooms, teachers used modeling techniques to illustrate methodologies for solving problems, such as simulating algorithmic processes, physically enacting computational concepts, and incrementally increasing levels of complexity. The upshot of these moves is that they facilitate schema building, in which students draw from a conceptual foundation when addressing increasingly abstract problems, thereby building knowledge from prior understandings [38].

The current literature on inquiry learning portrays students as scientists who develop hypothesis and conduct investigation, linking these practices to the development of discipline specific skills such as computational thinking [13, 14]. In this model, confirmatory experiences are frowned upon as encompassing the rote presentation of scientific facts, with little contribution to students' development of scientific practices. Yet these teachers provided worked examples to characterize conceptual and investigative principles, using strategies such as repetition, refocusing students' attention, and open ended questioning techniques to address student confusion. Studies have indicated that providing worked examples reduces the tax on working memory and opens up the resources necessary for learning [39]. It is plausible that working through examples of algorithmic processes, as observed in Helen's class, provided the scaffolding necessary for successful problem solving.

In this study, we also see that more structured and direct approaches produced better outcomes for diverse learners' identification with the field of CS. Although inquiry-based instruction engages students in authentic scientific practices, delivering unstructured inquiry without sufficient schema building and preparation can lead to disappointment and lost opportunities [40]. All four teachers who implemented inquiry-based approaches reported students getting lost, getting stuck, and jumping in without seeing the big picture. If we want to broaden participation in computing, it is important to not only give students experiences, but to give them successful experiences. We were especially impressed the work from Estelle, herself a Latina who had substantial experience with the community she served and taught large number of multilingual students and students with disabilities. Further exploration into the methodological and incremental approach she used could uncover valuable strategies that meet the cognitive and affective needs of multilingual students.

B. Limitations and Future Research

There are several limitations to this study. First, as this was an exploratory study that is part of a larger project aimed at revising, testing, and scaling instructional materials and pedagogies that meet the needs of diverse learners, we do not make causal claims about our findings. Each of the classes we observed had different compositions and grade levels and these factors could provide confounds to our claims. Furthermore, the data instruments we used to measure students' computational thinking skills may not be sensitive to the types of learning that took place in the open inquiry classes. In open inquiry classrooms, different types of learning and growth may take place, such as problem formulation, goal setting, planning strategies, and persistence. Future studies should design measures for capturing the types of learning and growth taking place in these classrooms.

Despite these limitations, this paper poses several questions to the understudied area of teaching computational thinking to multilinguals students through inquiry-based learning. As this project represents the exploratory phase of a larger project, we are currently using these findings to integrate more structured approaches in the next iteration of our curriculum. This includes integrating the Use-Modify-Create model into our lessons and applying metacognitive strategies from reading research to students' development of computational thinking. To integrate more structure, the next phase of this project will modify the curriculum to integrate a CS instructional approach known as Use-Modify-Create [41] in which students will first *use* existing programs, then work together to *modify* them, and finally *create* their own. Furthermore, during the *use* stage, we will incorporate an additional layer of scaffolding with a learning strategy borrowed from reading research known as "TIPP&SEE," developed by the Computing for ANyONe (CANON) lab at the University of Chicago and faculty at Texas State University. TIPP&SEE is derived from the reading strategy THIEVES [42], and focuses students on using context clues to better grasp intended material. Students first read the *title* of the program and make predictions based on the title. Then they analyze the *instructions* to better understand the tasks they are asked to engage in. Next, students think about the *purpose* of the program to consider the learning goals of the activity. Finally, they *play* with the program to examine its characteristics and practice documenting their observations. Students are then tasked with looking inside the program to examine the *sprites* and the *events* controlling the sprites, and then they *explore* the code. During the explore phase, students are instructed to change features of the code, test the changes, and document the results, preparing them for the MODIFY stage of the USE/MODIFY/CREATE model. This new curriculum will be scaled to three school districts and tested using randomized control trial to formally examine the impact of structured approaches to inquiry on multilingual students' development of computational thinking skills.

ACKNOWLEDGMENT

This research was supported in part by a NSF Computer Science for All grant 1738825

REFERENCES

[1] M. Smith, "Computer Science for All," Online Submission, pp. 1-13, 2016.

[2] J. Goode and J. Margolis, "Exploring computer science: A case study of school reform," ACM Trans. on Comp. Edu. (TOCE), vol. 11, pp 1-16, 2011.

[3] R. Ladner and M Israel., "For all in computer science for all," Communications of the ACM, vol. 59, pp. 26-28.

[4] S. Jacob, H. Nguyen, C. Tofel-Grehl, D. Richardson, and M. Warschauer, "Teaching computational thinking to English learners," NYS TESOL journal, vol. 5, pp. 12-24, 2018.

[5] S. Vogel, C. Hoadley, L. Ascenzi-Moreno, and K. Menken, "The role of translanguaging in computational literacies: Documenting middle school bilinguals' practices in computer science integrated units. In Proc. of the 50th ACM Tech. Symp. on Comp. Sci. Ed, pp. 1164-1170, 2019.

[6] A. Martin, F. McAlear, and A. Scott A, "Path not found: Disparities in access to computer science courses in California high schools," 2015.

[7] D. Royal and A. Swift, "U.S. minority students less exposed to computer science," Gallup, October 2016.

[8] J. Wang, H. Hong, J. Ravitz,, and S. H. Moghadam, "Landscape of K–12 computer science education in the US: Perceptions, access, and barriers," In Proc. of the 47th ACM Tech. Symp. on Comp. Sci. Edu., pp. 645–650, February 2016.

[9] G Estrella, J. Au, S. M. Jaeggi, and P. Collins, "Is inquiry science instruction effective for English language learners? A meta-analytic review," AERA Open, vol. 4, 2018.

[10] R. A. Krystyniak and H. W. Heikkinen, "Analysis of verbal interactions during an extended, open-inquiry general chemistry laboratory investigation," JRST, vol. 44, pp. 1160-1186, 2007.

[11] C. Quintana, X. Zhang, and J. Krajcik, "A framework for supporting metacognitive aspects of on-line inquiry through software-based scaffolding," Educational Psychologist, vol. 40 pp. 235-244, 2005.

[12] N. Trautmann, J. MaKinster, and L. Avery, "What makes inquiry so hard? (and why is it worth it?)," Paper presented at the annual meeting of the Nat. Assoc. for Research in Sci. Teach., Vancouver, BC. April, 2004.

[13] B. A. Crawford, "Learning to teach science as inquiry in the rough and tumble of practice," JRST, vol. 44, pp. 613-642, 2007.

[14] M. Windschitl, "Inquiry projects in science teacher education: What can investigative experiences reveal about teacher thinking and eventual classroom practice?," Science Education, vol. 87, pp. 112-143, 2003.

[15] A. Keselman, "Supporting inquiry learning by promoting normative understanding of multivariable causality, JRST, vol. 40, pp. 898-921, 2003.

[16] National Research Council, "National Science Education Standards," Washington, DC: National Academy Press, 1996.

[17] National Research Council, "National Science Education Standards," Washington, DC: National Academy Press, 2008.

[18] T. De Jong and W. R. Van Joolingen, "Scientific discovery learning with computer simulations of conceptual domains," Review of Educational Research, vol. 68, pp. 179-201, 1998.

[19] National Research Council, "A framework for K–12 science education: Practices, crosscutting concepts, and core ideas," Washington, DC: National Academies Press, 2012.

[20] M. Pedaste, M. Mäeots, L. A. Siiman, T. De Jong, S. A. Van Riesen, E. T. Kamp, , ... and E. Tsourlidaki, "Phases of inquiry-based learning: Definitions and the inquiry cycle," Educational Research Review, vol. 14, pp. 47-61, 2015.

[21] J. M. Wing, "Computational thinking. Communications of the ACM," vol. 49, pp. 33–35, 2006.

[22] G. Estrella, J. Au, S M. Jaeggi, and P. Collins, "Is inquiry science instruction effective for English language learners? A meta-analytic review, AERA open, vol. 4, 2018.

[23] S. G. Zwiep and W. J. Straits, "Inquiry science: The gateway to English language proficiency," JRST, vol. 24 pp. 1315-1331, 2013.

[24] A. Wagh, K. Cook-Whitt, and U. Wilensky, "Bridging inquiry‐based science and constructionism: Exploring the alignment between students tinkering with code of computational models and goals of inquiry," JRST, vol. 54, pp. 615-641, 2017.

[25] S. Basu, G. Biswas, P. Sengupta, A. Dickes, J. S. Kinnebrew, and D. Clark, D, "Identifying middle school students' challenges in computational thinking-based science learning," Research and Practice in Technology Enhanced Learning, vol. 11, 2016.

[26] B. J. Reiser, "Scaffolding complex learning: The mechanisms of structuring and problematizing student work," The Journal of the Learning Sciences, vol 13, pp. 273-304, 2004.

[27] R. W. Bybee, Achieving Scientific Literacy: From Purposes to Practical action, Portsmouth, NH: Heinemann, 1997.

[28] S. Basu, K. W. McElhaney, S. Grover, C. J. Harris, and G. Biswas, "A principled approach to designing assessments that integrate science and computational thinking," Intl. Soc. of the Learn. Sci., 2018.

[29] P. R. Aschbacher, M. Ing, and S. M. Tsai, "Is science me? Exploring middle school students' STEM career aspirations," Journal of Science Education and Technology, vol. 23, pp. 735-743, 2014.

[30] R. E. Boyatzis, R. E. (1998). Transforming Qualitative Information: Thematic Analysis and Code Development, Newbury, CA: Sage, 1998.

[31] A. M. Gadermann, M. Guhn, and B. D. Zumbo, "Estimating ordinal reliability for likert-type and ordinal item response data: A conceptual, empirical, and practical guide, Research & Evaluation, vol. 17, 2012.

[32] R. B. Kline, Convergence of Structural Equation Modeling and Multilevel Modeling, The SAGE Handbook of Innovation in Social Research Methods, Newbury, CA: Sage, 2011.

[33] D. Dean Jr and D. Kuhn, "Direct instruction vs. discovery: The long view. Science Education, vol. 91, pp. 384-397, 2007.

[34] C. E. Hmelo-Silver, R. G. Duncan, and C. A. Chinn, "Scaffolding and achievement in problem-based and inquiry learning: A response to Kirschner, Sweller, and Clark," Educational Psychologist, vol. 42, pp. 99-107, 2007.

[35] D. Cairns and S. Areepattamannil, "Exploring the relations of inquiry-based teaching to science achievement and dispositions in 54 countries," Research in Science Education, vol. 49, pp. 1-23, 2019.

[36] J. Jerrim, M. Oliver, and S. Sims, "The relationship between inquiry-based teaching and students' achievement. New evidence from a longitudinal PISA study in England," Learning and Instruction, vol. 61, pp. 35-44, 2019.

[37] D. Klahr and M. Nigam, "The equivalence of learning paths in early science instruction effects of direct instruction and discovery learning," Psychological Science, vol. 15, pp. 661-667, 2004.

[38] J. P. Smith III, A. A. Disessa, and J. Roschelle, "Misconceptions reconceived: A constructivist analysis of knowledge in transition," The Journal of the Learning Sciences, vol. 3, pp. 115-163, 1994.

[39] P. A. Kirschner, J. Sweller, and R. E. Clark, "Why minimal guidance during instruction does not work: An analysis of the failure of constructivist, discovery, problem-based, experiential, and inquiry-based teaching," Educational Psychologist, vol. 41, pp. 75-86, 2006.

[40] J. D. Bransford, D. J. Stipek, N. J. Vye, L. M. Gomez, and D. Lam, D, The Role of Research in Educational Improvement, Boston, MA: Harvard Education Press, 2009.

[41] I. Lee, F. Martin, J. Denner, B. Coulter, W. Allan, J. Erickson, et al., "Computational thinking for youth in practice, ACM Inroads, vol. 2, pp. 32-37, 2011.

[42] S. L. Manz, "A strategy for previewing textbooks: teaching readers to become THIEVES," The Reading Teacher, vol. 55, pp. 434-436, 2002.