Use, Modify, Create: Comparing Computational Thinking Lesson Progressions for STEM Classes

Nicholas Lytle
Veronica Cateté
Danielle Boulden
nalytle@ncsu.edu
vmcatete@ncsu.edu
NC State University
Raleigh, North Carolina

Yihuan Dong
Jennifer Houchins
Alexandra Milliken
Amy Isvik
ydong2@ncsu.edu
NC State University
Raleigh, North Carolina

Dolly Bounajim
Eric Wiebe
Tiffany Barnes
wiebe@ncsu.edu
tmbarnes@ncsu.edu
NC State University
Raleigh, North Carolina

ABSTRACT

Computational Thinking (CT) is being infused into curricula in a variety of core K-12 STEM courses. As these topics are being introduced to students without prior programming experience and are potentially taught by instructors unfamiliar with programming and CT, appropriate lesson design might help support both students and teachers. "Use-Modify-Create" (UMC), a CT lesson progression, has students ease into CT topics by first "Using" a given artifact, "Modifying" an existing one, and then eventually "Creating" new ones. While studies have presented lessons adopting and adapting this progression and advocating for its use, few have focused on evaluating UMC's pedagogical effectiveness and claims. We present a comparison study between two CT lesson progressions for middle school science classes. Students participated in a 4-day activity focused on developing an agent-based simulation in a block-based programming environment. While some classrooms had students develop code on days 2-4, others used a scaffolded lesson plan modeled after the UMC framework. Through analyzing student's exit tickets, classroom observations, and teacher interviews, we illustrate differences in perception of assignment difficulty from both the students and teachers, as well as student perception of artifact "ownership" between conditions.

CCS CONCEPTS

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

KEYWORDS

Use-Modify-Create, Computational Thinking, Lesson Design

ACM Reference Format:
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 *Innovation and Technology in Computer Science Education (ITiCSE '19), July 15–17, 2019, Aberdeen, Scotland Uk.* ACM, New York, NY, USA, Article , 7 pages. https://doi.org/10.1145/3304221.3319786

ITiCSE '19, July 15–17, 2019, Aberdeen, Scotland Uk

ACM ISBN 978-1-4503-6301-3/19/07…$15.00
https://doi.org/10.1145/3304221.3319786

1 INTRODUCTION

It is becoming increasingly necessary for every child to have experience with 21st-century Computational Thinking (CT) skills [28]. However, these skills have typically been taught within elective Computer Science classes or outside of school activities [17]. To reach *all* students, CT must be integrated into required K-12 courses, such as science and math. This CT integration will pose several challenges. First, lessons must not only focus on key CT concepts but must also include and integrate domain knowledge, though research demonstrates that CT topics can be integrated without detracting from the learning of the core domain material [3]. Further, these activities must be designed with the understanding that they may be the first introduction to programming or CT for many students and teachers. Successful integration depends on equipped and capable teachers, though many do not have the prerequisite background required to teach CT or computer science [7]. Professional development can give teachers experience with CT skills [8, 12, 21], however, to reach all students, we must develop solutions that can be readily adopted by both experienced and inexperienced teachers and that can improve student learning in both CT and course content.

This study compares two separate design implementations of a 4-day computing infused science lesson across multiple classrooms. One condition received a lesson including 3 days of coding while the other received a scaffolded curriculum modeled after Lee's Use-Modify-Create (UMC) [16]. Through our quasi-experimental design, we aim to investigate how UMC sequencing impacts:
1. Student-perceived *difficulty* of the lesson.
2. Student-perceived *ownership* of the used and developed programs.
3. Teacher-perceived *difficulty* of the lesson and *ability* to teach it.

2 RELATED WORK

As demand increases for incorporating computing into core K-12 subjects, so does the need for classroom activities that are tailored for non-computing focused teachers and students. Through adding computing activities directly into a STEM course, teachers gain improved mastery of their discipline when using new instructional approaches [26]. However, reporting suggests that teachers lose

much of the control they traditionally have over the learning process and may become uncomfortable when students pose and solve open-ended integrated STEM and Computing problems [6]. For teachers to support these students engaging in self-directed collaborative processes, they require an ability to diagnose difficulties and give hints, rather than supply solutions.

A prior case study by Cateté et al. [4] on infusing CT into science classrooms highlights how teachers were hesitant to lead new programming activities, and were afraid of misguiding students, even with professional development and classroom support. These concerns are also identified in a 2017 report for UK teachers facing computing infusion [23]. The most common challenges mentioned included subject knowledge, differentiation, lack of time, approaches to teaching topics, students' understanding, and ability to problem solve. This report also lists successful teacher strategies towards teaching computing such as unplugged activities[2], computational thinking, contextual learning, and scaffolding of programming tasks. In order to improve the incorporation of computing into science classrooms, we attempt to utilize the above findings to create more supportive materials for both teachers and students facing computing.

One such approach to improving CT acquisition with reduced cognitive stress comes in the form of curricular materials that follow a scaffolded intensity of interaction. Research by Lee et al, suggests that using a Use-Modify-Create (UMC) learning progression can promote the acquisition and development of CT while also limiting the anxiety from activities that teachers may have previously perceived to be "too hard" for students [15, 16]. In the first phase, **Use**, students inspect code and run existing models acting as "consumers of someone else's creation[16]." In transitioning to the **Modify** phase, students go from solely using existing code to changing the code to suit their intended desires as designers. This act of modifying also brings about a change in perceived ownership as students move toward viewing the code as their own. In the final phase, **Create**, students end up in a state where they have created a completely new model, having full ownership of the design and agency over its development[16].

Lee promoted UMC not only as a lesson scaffolding framework, but also a way to create this sense of "ownership" in learners. It has been argued that in order to realize the full benefit of CT, students must develop a sense of ownership over the models underlying the CT concepts being taught [5]. During creation, the final step of UMC, students increase engagement in learning and perceived agency of their learning, which is associated with behavioral, emotional, and cognitive engagement [22]. UMC allows students to take increasing ownership of the learning by giving them progressively more complex tasks. This increased ownership empowers students to investigate CT and underlying assumptions behind the tasks.

Researchers have used UMC as a basis for a number of CT and CT-infused activities across the K-8 curriculum [13, 15, 27]. Werner et al. employed UMC in the creation of an elective game programming course [27], finding that students demonstrated understanding of several CT and CS concepts through developing games. Furthermore, Grizioti et al. developed a game-specific adaptation in which players first play then modify/fix a "half-baked" version of the game, and then create a new version [13]. Sentance and Waite extend UMC in their PRIMM (Predict, Run, Investigate, Modify, Make) model for teaching text-based programming[24]. Initial workshops suggest that teachers are willing to adopt this model, but like Werner and Grizioti, this work is situated in a pure CS context.

We believe the UMC framework can extend into core domains, alleviating the burden of learning to program while simultaneously learning domain material. This will allow students to ease into the activity, become familiar with the programming environment, and explore how smaller changes affect the code. We assume these same benefits extend to teachers who also might not be familiar with coding and would welcome scaffolded lessons. We finally posit that as students go from users to creators in the Use-Modify-Create lessons, their sense of ownership of the project will increase to match that of students who participate in lessons where they always create code from scratch. We test out these assumptions using an A/B study across multiple classrooms as described in the section below. If these hypotheses are supported, these benefits can reduce teachers' fears of being CT novices as well as students' frustration with the difficulty of learning to program, potentially increasing adoption of materials developed using the UMC progression.

3 METHODS

3.1 Context

The study took place in two separate middle schools (School 1, School 2) in the mid-atlantic United States. The classrooms were all 6th grade (age 11-12) science classrooms taught by one of 4 teachers (1 from School 1, 3 from School 2). None of the teachers had experience instructing programing lessons. Each teacher was responsible for multiple class *periods* with different students. Teachers in School 2 had 5 class periods each while the teacher in School 1 taught 2 class periods. Each period averaged 20 - 30 unique students. A total of 394 students participated in the study, but we only analyze data from the 160 consenting students who provided data for every day. Demographic information for these students is reported in Table 1. Chi-Square tests show no significant differences in gender or race between the two populations.

Table 1: Gender and race/ethnicity by condition.

		UMC (N=95)	Control (N=65)	Total (N=160)
Gender	Female	42.1%	41.5%	41.9%
	Male	47.4%	55.4%	50.6%
Race Ethnicity	Black	19.0%	13.9%	16.9%
	Caucasian	26.3%	26.2%	26.3%
	Hispanic	15.8%	18.5%	16.9%
	Asian	14.7%	15.4%	15.0%
	Multi-racial	3.2%	3.1%	3.1%
	N/A	21.1%	23.1%	21.9%

We further surveyed students on their previous programming experiences. Responses are shown in Table 2 ranging from Never to Daily. Chi-Square tests show no significant differences in prior programming experience between the two populations. In an open response follow up to question 1, many students reporting "Rare"

or "Occasional" described participating in an Hour of Code activity [14]. Those marking "Frequent" or "Daily" report being in a computing club or technology elective course.

Table 2: Participants' computing background self-ratings.

	Never	Rare	Occasional	Frequent	Daily
Q1	Previous participation in computing activities				
UMC	7.4%	9.5%	54.7%	11.6%	6.3%
Control	6.2%	15.4%	47.7%	20.0%	7.7%
Q2	Previous experience writing a computer program				
UMC	15.8%	21.1%	41.1%	7.4%	4.2%
Control	13.8%	23.1%	40.0%	15.4%	4.6%

3.2 Curriculum

The activity was designed to be a 4-day, CT lesson. Though teachers were trained on the material, on programming days, a research team member taught the first period as part of a "Faded Instructor Scaffolding Model" designed to help teachers understand the curriculum from a student's perspective. Each programming day took place within the Cellular environment [1], an extension of the block-based programming language Snap! [10] that provides a good method for agent-based modeling and has been used in similar initiatives with infusing CT into STEM curriculum [4].

For use in the 6th grade science classroom, the topic of "Food Webs" was chosen. In the food web curriculum, students learn about how energy is transferred from producers to primary and secondary consumers. The computing-infused activity let students explore the transfer of energy in a simplified food web developed using the block-based programming environment, Cellular [1]. We describe each daily segment of the activity below, and a breakdown by condition is visualized in Figure 1. The Use-Modify-Create (UMC) version was adapted from a previous version of the food web activity used in classrooms which acts as our control lesson in this study.

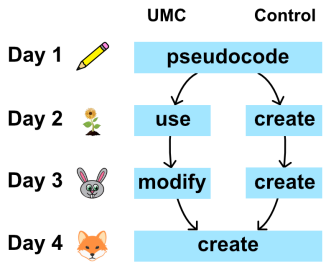

Figure 1: Programming methods for Food Web agents (plants, bunnies, foxes) by day and condition

Day 1 - Both conditions completed an "Unplugged" activity [2] in which students reviewed definitions and components of a Food Web (e.g. primary and secondary consumer, how energy is transferred through the system, etc.). This ended with completing a worksheet lead by the instructor where students described the behavior of agents in the model through pseudo-code. This was done to prepare students for developing these ideas within the programming model.

Day 2 - Day 2's focus was the "Plant" agent (the producer), which would grow based on the solar energy given by the "Sun" (code provided for both conditions). For the control condition, students had to develop code for the Plant to be able to transition between stages of its life cycle using the amount of 'Solar Energy' received over time. This was represented in code as sequential conditionals, checking both the plant's current state and energy before transitioning into the next state. The students in the UMC condition had plant code provided and instead inspected and read through the working code in order to become familiar with the different conditions. The instructor led students in exploration by changing the initial input (the solar energy intensity), the cutoff conditions (how much energy is needed to transition) and the amount of energy lost through transitioning. Students then used a worksheet to record how those changes affected the speed in which flowers changed state.

Day 3 - Day 3's focus was the "Bunny" agent (the model's primary consumer). Control condition students wrote code to add the new agent to their working model. Meanwhile, the UMC condition had Bunny code provided at the outset. However, the given bunny behavior did not conform with their idea of the actual model (e.g. bunnies never ate when they got low on energy, flowers transitioned to an incorrect state after being eaten etc.). Thus, students during this class period *modified* the existing code in order to make it conform to the existing ideas they had discussed on Day 1's activity of how the model should behave. This is similar to the activity found in prior studies of fixing the "half-baked microworld"[13].

Day 4 - The agent focus for the final lesson was on this model's secondary consumer, the "Fox". Both conditions had to develop the entire Fox code (in a sense, *creating* a new model with this additional agent) and update the "Bunny" code to react to the new agent and implement the desired final behavior in their model. Once complete, as shown in Figure 2, students were tasked with changing some functionality in their code and comparing how their new simulations behaved differently from their previous one.

Figure 2: Food Web in Cellular showing how plant, bunny, and fox agents appear on the grid at one time step.

Although each programming day builds on previous topics, students begin with the same starter code for their condition to reduce effects from student absences or incompletion from a prior day. The topic of Food Webs and the sequencing of our curriculum affords the exploration of our research questions for a number of reasons. First, as a life science topic found in common core guidelines, Food Web is an exemplar STEM lesson that could be taught throughout

the United States. Second, as the topic of Food Web focuses on the interaction between different actors in an environment, it lends itself nicely to a computing task focused on developing an agent-based model or simulation like prior UMC-developed lessons [16]. Finally, as we use a multi-day assignment, we can segment each programming day to be on a single Food Web actor, allowing us to frame the UMC conditions' activity to match each phase of UMC.

3.3 Data Collection

Evaluation of the initiative was recorded through a number of data collection methods. For every period, at least one research team member was present taking observation notes, focusing on the students' interactions within the environment as well as how the teacher was teaching the lesson. After the conclusion of each day's activity, students took an end-of-activity "Exit Ticket" in which they answered a series of questions about the activity. In order to study student-perceived difficulty, we asked students to rate the difficulty of the days' activity on a 1-5 Likert scale (Very Easy to Very Difficult). For student-perceived ownership, two questions were included that addressed the ability to express one's ideas and the belief the code was their own creation. These questions were on a 7-point Likert scale from Strongly Agree to Strongly Disagree.

For teacher-perceived difficulty, we first analyzed classroom observations that focused on the teacher's ability to teach the lesson. Additionally, a member of the research team conducted interviews using a semi-structured interview protocol with each of the participating teachers at the conclusion of the lesson sequence. The protocol consisted of questions targeted to elicit general teacher feedback about each of the days of the lesson sequence and their impact on students (e.g., What are the strengths of the lesson? Weaknesses?) Interviews lasted approximately twenty to thirty minutes and were audio recorded and transcribed for analysis. A constant comparison analysis [20] of the interview transcripts provided insights on the teachers' perceptions of the lessons from a pedagogical standpoint, comparing teacher interviews within groups and between groups.

4 RESULTS

4.1 Student-perceived difficulty

Each day, students completed the exit ticket question "Please use the [scale of 1 to 5 (Very Easy to Very Difficult)] to rate how difficult or easy the lesson was today." The daily average values for students in each condition are given in Table 3. A Friedman Test, similar to a parametric repeated measures ANOVA but for non-parametric data [9], was performed in order to determine if there were differences found in the average values for each of the days. For the Use-Modify-Create (UMC) condition, no significant difference was found among the four days $\chi^2_3 = 1.879, p = 0.598$. However, for the control condition, $\chi^2_3 = 9.984, p = 0.019$ showing significance. Therefore, a post-hoc Wilcoxon Signed Rank Test, a non-parametric equivalent to a paired T-Test [29], was performed between each of the pairwise groups. No significant differences were found between days 1 and 3, 1 and 4, nor 3 and 4. However, for each pairwise comparison with day 2, a significant difference was found (1-2: $V = 240, p = .002$; 2-3: $V = 770, p = .003$; and 2-4: $V = 218, p = .004$). An additional Mann Whitney U Test, a non-parametric test similar to an unpaired T-Test[19], was performed between groups for each of the 4 days to determine differences in difficulty responses between conditions for the same day. No significant differences were found in the comparisons between conditions for Days 1 ($W = 3187, p = 0.72$), 3 ($W = 3162, p = 0.79$), or 4 ($W = 3144, p = 0.84$). However, comparing Day 2 (the first coding day) between the two conditions found a significant difference with UMC being significantly easier ($W = 2238, p = .002$).

This difference in difficulty is backed by classroom observations. Researchers found that in the UMC classrooms, students were often able to finish their designated tasks more quickly, especially on the third day where the UMC group modified a bunny while the control group coded one from scratch. This time difference means that the UMC group had additional time to add elements or engage in teacher-led discussions about connections to class topics. Further, researchers found in some control condition classrooms (especially on Day 2 and to some degree Day 3) that students had difficulty finishing the task. As a result, teachers in the control group on Day 2 either forged ahead leaving many students behind, or slowed the lesson so all students could keep up, but were unable to complete the full lesson to add flowers. During the follow-up interviews, teachers in the control condition commented on the need for more scaffolding, while teachers in the UMC condition did not express this concern. Teachers in the UMC condition indicated that the progression of the curriculum served as an effective scaffold for students' conceptual understanding of the programming environment that better prepares them for creating their own program models. One of the teachers articulates this below:

"...like day [two], when we were on the computer. You really understand the beginning part. And then day [three] it builds a little bit more and you're building the code. You're playing with it. And day [four] is really copying the bunny code, just tweaking it a tiny bit. So at that point they've done so much with it already. They've got it. I mean, they were playing with all kinds of things."

4.2 Student-perceived ownership

Two questions were added to the coding day exit tickets (Days 2, 3, and 4) to address student-perceived ownership. These were: "To what extent do you agree with the following statement: I was able to express my ideas in the model today" and "To what extent do you agree with the following statement: The code I ended the lesson with is my own creation" both on a scale from 1 (Strongly Agree) to 7 (Strongly Disagree). The average values for these answers are shown in Table 3. A Friedman Test was performed in order to determine if there were differences found in the average values for each of the days. No significant difference was found among the three days for 'expressing ideas' for both the control group, $\chi^2_2 = 0.0231, p = 0.989$ and for the UMC group $\chi^2_2 = 1.1421, p = 0.565$. However, in performing Mann Whitney U Tests between the 2 groups, significant differences were found in Days 3 ($W = 3389, p = 0.03$) and 4 ($W = 3425, p = 0.04$) though not for Day 2 ($W = 3683, p = 0.21$). For the statement "The code I ended the lesson with is my own creation", a Friedman Test finds no significant difference in the control condition answers: $\chi^2_2 = 0.562, p = 755$. However, for the UMC condition, the Friedman Test found a significant difference among the three: $\chi^2_2 = 9.637, p = 0.008$. A follow-up pairwise Wilcoxon-Signed Rank Test was performed between each of the

Table 3: Reporting of Average and Standard Deviation for student responses to Exit Ticket questions.

	Use-Modify-Create Condition (N=95)				Control Condition (N=65)			
Likert Questions	Day 1	Day 2	Day 3	Day 4	Day 1	Day 2	Day 3	Day 4
Rate how difficult or easy the lesson was today: (Very Easy) 1 - 5 (Very Difficult).	$\overline{\chi} = 2.11$ $\sigma = 0.93$	$\overline{\chi} = 2.04$ $\sigma = 1.18$	$\overline{\chi} = 2.15$ $\sigma = 1.03$	$\overline{\chi} = 2.25$ $\sigma = 1.09$	$\overline{\chi} = 2.03$ $\sigma = 0.85$	$\overline{\chi} = 2.58$ $\sigma = 1.13$	$\overline{\chi} = 2.11$ $\sigma = 1.03$	$\overline{\chi} = 2.25$ $\sigma = 1.09$
"I was able to express my ideas in the model today." (Strongly Agree) 1 - 7 (Strongly Disagree)	N/A	$\overline{\chi} = 2.89$ $\sigma = 1.45$	$\overline{\chi} = 2.64$ $\sigma = 1.48$	$\overline{\chi} = 2.68$ $\sigma = 1.54$	N/A	$\overline{\chi} = 3.13$ $\sigma = 1.18$	$\overline{\chi} = 2.98$ $\sigma = 1.32$	$\overline{\chi} = 2.99$ $\sigma = 1.32$
"The code I ended the lesson with is my own creation." (Strongly Agree) 1 - 7 (Strongly Disagree)	N/A	$\overline{\chi} = 3.54$ $\sigma = 1.87$	$\overline{\chi} = 2.79$ $\sigma = 1.64$	$\overline{\chi} = 2.93$ $\sigma = 1.72$	N/A	$\overline{\chi} = 3.39$ $\sigma = 1.42$	$\overline{\chi} = 3.24$ $\sigma = 1.44$	$\overline{\chi} = 3.21$ $\sigma = 1.43$

days and while no significant difference was found between Days 3 and 4 ($V = 769, p = 0.38$), significant differences were found between Days 2 and 3 ($V = 1973, p < 0.001$) and Days 2 and 4 ($V = 1000, p = 0.006$) within the UMC group. Additional Mann-Whitney U Tests were performed between the two conditions. While no difference was found between Day 2 ($W = 4207, p = 0.8$) or Day 4 ($W = 3485, p = 0.07$) between the conditions, significant differences were found between the Day 3 ($W = 3284, p = 0.02$) responses between groups.

While not as direct as the student-perceived difficulty differences, there were key classroom observational differences between the conditions that corroborate the findings from the exit tickets. First, as stated before, students in the UMC condition were often able to finish tasks faster and were therefore able to explore more within the code, and add their own additional features. It is possible that adding their own touches after the guided part of the lesson led to an increased sense of artifact 'ownership'. Second, researchers observed (and teachers commented during follow-up interviews) that students in the UMC condition seemed more engaged in the activity, but it was actually difficult to keep students engaged in the control condition. This disconnect with the material and the monotony of the tasks in the control condition might have contributed to the students' sense that the code was not their own.

4.3 Teacher-perceptions

Teacher interview transcripts were analyzed using a constant comparative method [20] that entailed searching for themes amongst the teachers within each condition and then comparing data from the teachers across the two conditions. Results reflected the difficulties that teachers in the control group faced. The two teachers implementing the coding-intensive control version of the curriculum expressed concerns that their students needed more scaffolding to complete the lessons and concerns about their students' daily engagement. When asked about potential improvements to the curriculum, both teachers suggested giving students more time to "explore" and "play" with the code prior to creating their own programs. UMC provides this opportunity for students. As one of the teachers in UMC condition explains, "the kids [during the 'use' day] were understanding the coding; they were understanding why it was changing and they were starting to play around with some of that as well." Additionally, results from teacher interviews demonstrated that teachers in the control condition perceived a decrease in student engagement each day. Teachers themselves suggested changing the approach each day, e.g. "I got a lot of comments that they were bored with it because it was the same thing day after day. So I can't really think of it right now, but if there is, like some kind of way to mix it up and still have the same information, but maybe have them do it in a different way." Our data and teacher interviews suggest that approaching coding through a variety of tasks, as the UMC approach does, can improve student engagement. It is also possible that the structured UMC sequence makes the Create day more purposeful and engaging for students. One teacher explains her students' reactions to the Create day, "They're like 'day three was very fun'. They really got a chance to understand how the whole thing is connected."

The teacher interviews also corroborated that a UMC sequence offers benefits to teachers, as it supports their learning and confidence with the materials. One of the teachers revealed to us that, researcher support "wasn't needed the last day because I knew it. At this point I was like, 'I'm comfortable. I know where you're going with this.'" Both of the teachers in the code creation sequence commented that the lessons were "exhausting," as one teacher described that implementing the lessons entailed "standing in front of a room and talking and basically having you being the first person [they're] gonna ask questions to for five hours."

The findings above are supported by observations of teachers working through the curriculum. Like the teacher above noted, researchers observed that teachers gained confidence in teaching the lessons in both conditions, but this was more marked for UMC teachers. The two UMC teachers, in addition to following the guided material provided to them, were able to add new tasks to class periods where time was still available, and led students in guided discussions about the connections between the CT concepts and the scientific concepts modeled within the environment. As previously discussed, teachers in the control condition had difficulty engaging and keeping all students on task as observed by the researchers, and often needed to pause (especially on Days 2 and 3) to check which task they needed to be doing or what the code was supposed to look like. While this was also observed with one teacher of the UMC condition, this behavior was occurring later in the assignment sequence (on Days 3 and 4) and not to the same frequency.

5 DISCUSSION

For student-perceived difficulty, **students perceived the introduction to coding on Day 2 as significantly harder in the control group than any other day**. While there was the same procedure for each of the coding days in this condition (i.e. students had to create an agent on each day) having to do this for the first time might have been difficult without prior knowledge or experience in the environment. Days 1, 3, and 4 having similar reported levels of perceived difficulty, suggests that while the Day 2 activity was harder, getting through it and understanding the procedure prepared students for the Day 3 and 4 coding activities. No difficulty spike, however, was found in the sequence for the UMC condition. The benefits of this sequence can be explained using James Paul Gee's principles of good learning [11]. It could be that first "using" the new Cellular environment was "pleasantly frustrating" – challenging but perceived to be easily done, and then modifying it and then creating a new agent results in "well-ordered problems" that allow students to develop mastery [11]. In contrast, Day 2 for the control condition combined the need to learn the new environment while also learning the basics of programming and how they work within the environment, potentially increasing the cognitive load of the students [25], which could result in a higher perceived difficulty on Day 2.

Two questions were assessed to measure student perceived *ownership* of the models each day. While there are differences between conditions for the question regarding expression of ideas, the most pronounced differences between days and conditions is found for the question on whether the code was their "own creation". While the difference was not significant, the average value for the response was higher in Day 2 of the UMC condition, indicating that UMC students did not feel as much ownership over the final code as students in the control condition. This was expected, as Day 2 represents the "Use" day where students changed parameters and input variables, but did not create any new code. It is only on Day 3, where students in UMC were "Modifying" existing Bunny code, that there is a difference in perceived ownership between groups. We were somewhat surprised that students in the UMC condition felt significantly more ownership of their code, since they made fewer programmatic changes (code adds, deletes, edits) than the control group. It could be that the framing of the activity as modifying existing code to make it perform the 'correct' behaviors played into this mindset. In addition, the large number of program edits needed in the control condition may have made more creative activities, such as augmenting the model behavior, blend in with more mechanical changes, like dragging in pre-specified blocks. In some cases, these creative activities may not have even occurred, since teachers struggled to help students complete the Day 3 and Day 4 activities in the control group. The strengthening of artifact ownership in the UMC condition that began on Day 3 carried into Day 4, with the **UMC group agreeing significantly more than the control group that their final code was their own, despite both conditions doing the exact same task**.

In addition to student reported data, reflections from teachers also suggest that they would benefit from and prefer the UMC Condition. Teacher expertise is in supporting student learning, and their perceptions confirm that a strict code creation approach is not as effective for their classrooms. Since many teachers are novices to programming and CT, and their courses are focused on other topics, it is not realistic to expect disciplinary K-12 teachers to be able to support such an intensive coding approach to integrating CT. The UMC model helps teachers gradually learn how programs represent their disciplinary knowledge, enabling them to make those connections just in time with students. As stated by teachers, having time to be able to "explore" the environment by first reading and understanding code, then performing minor edits, and finally being ready to add independent features, gives both students and teachers an easier progression of tasks. This means that teachers can adopt integrated CT curricula more readily, letting them learn CT and programming along with their students.

6 CONCLUSION

In this paper, we present results from an A/B study of Use-Modify-Create (UMC) versus a control group implementation of a CT-infused Food Web activity. With student reports corroborated by classroom observations and teacher interviews, we were able to confirm previous research results showing that UMC sequencing provides students a natural progression to learn computational thinking within a science course, while giving students more ownership over the artifacts they create. We also found that teachers using our UMC curriculum felt it was easy to teach, and that it promoted student engagement and exploration, while teachers using a code-intensive control curriculum desired more scaffolding and features to improve student engagement.

Limitations of this work include potential population bias, instructor effects, and our interpretation of the UMC model. Participants are from two middle schools where many students have prior exposure to learning computing. This population bias could have improved students' ability to go through the curriculum and the ease in which they learned and experienced the topics. However, as the daily difficulty ratings are discussed in relative terms, we assume that a middle school with less access to CT and computing education would show even greater differences in perceived difficulty between the UMC and control conditions. It is not clear how more or less programming experience would impact student ownership ratings. Four teachers participated in the study, with two leading instruction in each condition. As such, there is no way for us to separate instructor effects from the curricular content/sequencing. Though Kruskal-Wallis Tests [18] and Mann-Whitney U Tests find no significant difference in student perceived difficulty by teacher group, it is still possible that instructors played a role in the student perception of difficulty and ownership. Finally, while Lee's original paper on the UMC model defined "Creation" as students making their own designs [16], we interpret it specifically to mean that students should develop all of their own code for an agent. In future studies, we hope to design activities that facilitate more open-ended student exploration and creativity.

ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under grant number 1742351. Any opinions, findings, and conclusions expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

REFERENCES

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

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

[3] Acey Kreisler Boyce, Antoine Campbell, Shaun Pickford, Dustin Culler, and Tiffany Barnes. 2011. Experimental evaluation of BeadLoom game: how adding game elements to an educational tool improves motivation and learning. In *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*. ACM, ACM, New York, NY, 243–247.

[4] Veronica Cateté, Nicholas Lytle, Yihuan Dong, Danielle Boulden, Bita Akram, Jennifer Houchins, Tiffany Barnes, Eric Wiebe, James Lester, Bradford Mott, and Kristy Boyer. 2018. Infusing Computational Thinking into Middle Grade Science Classrooms: Lessons Learned. In *Proceedings of the 13th Workshop in Primary and Secondary Computing Education (WiPSCE '18)*. ACM, New York, NY, USA, Article 21, 6 pages. https://doi.org/10.1145/3265757.3265778

[5] Bob Coulter, Irene Lee, and Fred Martin. 2010. Computational Thinking for Youth.

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

[7] Jan Cuny. 2012. Transforming high school computing: a call to action. *ACM Inroads* 3, 2 (2012), 32–36.

[8] Yihuan Dong, Veronica Catete, Robin Jocius, Nicholas Lytle, Tiffany Barnes, Jennifer Albert, Deepti Joshi, Richard Robinson, and Ashley Andrews. 2019. PRADA: A Practical Model for Integrating Computational Thinking in K-12 Education. In *Proceedings of the 50th ACM Technical Symposium on Computer Science Education (SIGCSE '19)*. ACM, New York, NY, USA, 906–912. https://doi.org/10.1145/3287324.3287431

[9] Milton Friedman. 1937. The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the american statistical association* 32, 200 (1937), 675–701.

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

[11] James Paul Gee. 2007. *Good video games+ good learning: Collected essays on video games, learning, and literacy*. Vol. 27. Peter Lang, New York, NY.

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

[13] Marianthi Grizioti and Chronis Kynigos. 2018. Game modding for computational thinking: an integrated design approach. In *Proceedings of the 17th ACM Conference on Interaction Design and Children*. ACM, New York, NY, USA, 687–692.

[14] Filiz Kalelioğlu. 2015. A new way of teaching programming skills to K-12 students: Code. org. *Computers in Human Behavior* 52 (2015), 200–210.

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

[16] 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 (2011), 32–37.

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

[18] Patrick E McKnight and Julius Najab. 2010. Kruskal-Wallis Test. *The corsini encyclopedia of psychology* 4 (2010), 1–1.

[19] Patrick E McKnight and Julius Najab. 2010. Mann-Whitney U Test. *The Corsini encyclopedia of psychology* 4 (2010), 1–1.

[20] Matthew B Miles, A Michael Huberman, and Johnny Saldana. 2014. *Qualitative data analysis*. Sage, Washington DC, USA.

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

[22] Johnmarshall Reeve and Ching-Mei Tseng. 2011. Agency as a fourth aspect of studentsâĂŹ engagement during learning activities. *Contemporary Educational Psychology* 36, 4 (2011), 257–267.

[23] Sue Sentance and Andrew Csizmadia. 2017. Computing in the curriculum: Challenges and strategies from a teacherâĂŹs perspective. *Education and Information Technologies* 22, 2 (2017), 469–495.

[24] Sue Sentance and Jane Waite. 2017. PRIMM: Exploring pedagogical approaches for teaching text-based programming in school. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*. ACM, ACM, New York, NY, 113–114.

[25] John Sweller. 1988. Cognitive load during problem solving: Effects on learning. *Cognitive science* 12, 2 (1988), 257–285.

[26] David Weintrop, Elham Beheshti, Michael Horn, Kai Orton, Kemi Jona, Laura Trouille, and Uri Wilensky. 2014. Defining computational thinking for science, technology, engineering, and math.

[27] Linda Werner, Shannon Campe, and Jill Denner. 2012. Children learning computer science concepts via Alice game-programming. In *Proceedings of the 43rd ACM technical symposium on Computer Science Education*. ACM, ACM, New York, NY, 427–432.

[28] Jeannette M Wing. 2006. Computational thinking. *Commun. ACM* 49, 3 (2006), 33–35.

[29] RF Woolson. 2007. Wilcoxon signed-rank test. *Wiley encyclopedia of clinical trials* (2007), 1–3.