An Analysis of Use-Modify-Create Pedagogical Approach's Success in Balancing Structure and Student Agency

Diana Franklin*, Merijke Coenraad†, Jennifer Palmer*, Donna Eatinger*, Anna Zipp*,

Marco Anaya*, Max White*, Hoang Pham*, Ozan Gökdemir*, David Weintrop†
*University of Chicago, Chicago, IL, USA
† University of Maryland, College Park, College Park, MD, USA
{dmfranklin,jenpalmer,azipp,manaya,hoangsp,gokdemir}@uchicago.edu;{mcoenraa,weintrop}@umd.edu

ABSTRACT

As computer science instruction gets offered to more young learners, transitioning from elective to requirement, it is important to explore the relationship between pedagogical approach and student behavior. While different pedagogical approaches have particular motivations and intended goals, little is known about to what degree they satisfy those goals.

In this paper, we present analysis of 536 students' (age 9-14, grades 4-8) work within a Scratch-based, Use-Modify-Create (UMC) curriculum, Scratch Encore. We investigate to what degree the UMC progression encourages students to engage with the content of the lesson while providing the flexibility for creativity and exploration.

Our findings show that this approach does balance structure with flexibility and creativity, allowing teachers wide variation in the degree to which they adhere to the structured tasks. Many students utilized recently-learned blocks in open-ended activities, yet they also explored blocks not formally taught. In addition, they took advantage of open-ended projects to change sprites, backgrounds, and integrate narratives into their projects.

CCS CONCEPTS

• **Social and professional topics → Computer science education**; **Computational thinking**;

KEYWORDS

Computational Thinking; K-12 education; Scratch

ACM Reference Format:
Diana Franklin, Merijke Coenraad, Jennifer Palmer, Donna Eatinger, Anna Zipp, Marco Anaya, Max White, Hoang Pham, Ozan Gökdemir, David Weintrop. 2020. An Analysis of Use-Modify-Create Pedagogical Approach's Success in Balancing Structure and Student Agency. In *Proceedings of the 2020 International Computing Education Research Conference (ICER'20), August 10–12, 2020, Virtual Event, New Zealand.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3372782.3406256

ICER '20, August 10–12, 2020, Virtual Event, New Zealand

ACM ISBN 978-1-4503-7092-9/20/08…$15.00
https://doi.org/10.1145/3372782.3406256

1 INTRODUCTION

Computer science is well on its way to becoming a subject that all K-12 students learn. Countries (e.g. Israel and United Kingdom) and major US school districts (e.g. San Francisco, New York, and Chicago), have pledged to bring CS experiences to all students in their schools through large-scale initiatives.

Several different pedagogical approaches for K-12 computer science instruction have been proposed, ranging from Constructionism (typified by learner-led activities focused on building personally-meaningful projects) to more structured approaches (puzzle-based curricula with many small tasks that gradually build skills for a particular concept). The Use-Modify-Create (UMC) pedagogical approach [18] represents a middle ground. Students are first introduced to new concepts through example code, exploring and modifying that code, and then designing and building an open-ended project. Each approach has is in widespread use and has its own rationale based on desired learning goals, but little research has been performed on UMC itself.

This work pursues the guiding research question: "To what degree does Use/Modify/Create provide structure focusing learners on computer science concepts while retaining enough open-endedness for student exploration intended to lead to engagement, identity development, and differentiation?" This question focuses on the tension between structured activities that ensure content engagement and the open-ended exploration that allows for learners to draw on prior knowledge and personalize their work, not on whether the end goals of engagement and identity development were achieved. Analyzing student artifacts from Use/Modify and Create activities in a UMC-structured curriculum from 9 different schools, we ask these specific questions.

To what extent did UMC focus learners on specific CS content:

- To what degree do students focus on and complete the assigned tasks? What influences completion?
- To what degree do students utilize blocks introduced within that or previous modules?
- To what degree do students focus on assigned tasks prior to completing extensions?

To what extent did UMC support learner exploration and personalization:

- To what degree do students complete extensions?
- To what degree do students explore blocks not yet formally introduced?
- To what degree do students personalize their projects?

2 THEORY AND PRIOR WORK

In this section, we present the major theoretical and practical influences that inform this work. This includes prior work on theories on learning, pedagogical approaches and tools in elementary CS education, and work on elementary CS learning trajectories.

2.1 Theoretical Orientation

This work is grounded in constructionist design and learning theory while also being informed by Vygotsky's notion of the zone of proximal development (ZPD), the idea that in learning there is a relationship between the activities, student skills, and student engagement. There is a beneficial range of difficulty that requires academic growth, but is still within that learner's reach [6, 32]. Designing learning experiences to keep students in their ZPD produces instructional strategies that allow the same learning activity to fit many learners at various points along a learning trajectory. This is accomplished through instructional scaffolding for some students [30, 31, 37] and additional challenges for other students.

Constructionism emphasizes empowering learners to construct personally-meaningful artifacts through learner-directed exploration [25, 27]. Constructionist learning experiences are often designed to give the learner agency in the activity, encouraging exploration throughout the process of constructing public and shareable artifacts [16, 26]. The open-endedness can be difficult to adhere to when used in a context where there are specific learning objectives or when trying to introduce new conceptual material or practices. Noss & Hoyles [24] call this the Play Paradox, an inherent tension between supporting a freedom of exploration while constraining the activity to ensure specific concepts or practices are encountered.

One way out of this pedagogical paradox that attends to both sides of this tension is the Use–>Modify–>Create instructional approach [18]. This approach moves learners from an initial, highly-scaffolded activity focused on introducing concepts and practices (the Use activity) towards a more open-ended exploratory space as learners' competence and confidence grows (the Create activity). Between the two is an interstitial step where some of the initial scaffolds fade and the learner is given increased agency to create while still have some instructional supports present to help them along (the Modify activity). For this work, ZPD is used as a theoretical orientation to motivate the design employed and the research questions being asked, rather than being employed as an analytic lens. Instead, we look more generally at whether this curricular approach allows flexibility on both ends - instructional support and additional tasks, while directing students to specific learning goals.

2.2 Pedagogy In K-12 Computing Education

Much of the curricular and programming environment design effort for younger students has been inspired by Constructionism [13, 25]. As discussed above, Constructionist learning foregrounds learner-led activities that provide the opportunity for learners to construct artifacts (e.g. Scratch programs). These constructed artifacts enable learners to share their ideas with peers and can serve as resources for teacher-led classroom discussions. Providing a public forum for students to share their ideas, experiences, and programs encourages students to become more invested in their own learning and promotes an inclusive learning environment [26]. This focus on student-driven learning, not specific technical content, has coincided with work in informal spaces [15, 22] and placed an emphasis on self-directed learning and online collaboration [9, 29] and the practices of computing [3]. The Creative Computing Curriculum is an iconic example of this approach[4].

There is also a growing library of curricula designed for early elementary learning that uses specially-designed programming environments that lead students through a more structured experience. This includes curricula design by Code.org [1], and the Foundations for Advancing Computational Thinking (FACT) curriculum [11].

2.3 Programming Environments for K-8 Learners

While traditionally a subject for high school and beyond, there is growing demand for bringing CS into K-8 classrooms. Early work by Papert and colleagues found that programming was accessible to younger students and could serve as a powerful pedagogical strategy [12, 25, 28]. In the last decade, bringing CS to K-8 has grown in popularity and has been facilitated by programming tools designed for young students [8, 17]. An increasingly popular approach for creating engaging and accessible programming environments is the graphical, block-based programming interface [2, 33]. Visual block-based languages use a programming-command-as-puzzle-piece metaphor to visually render syntax rules and allow users to use a drag-and-drop interaction to construct programs. Block-based programming tools provide numerous scaffolds that make programming easier, including limiting syntax errors, providing visual cues on how commands can be used, and providing an easy way to browse available commands [36]. Block-based programming interfaces have been used to create an array of programming environments and tools, including museum exhibits [14], libraries for controlling robots [23, 35], and tools for creating mobile apps [7].

2.4 Prior Work

Prior work has attempted to understand what students learn within different pedagogical approaches. Constructionist approaches places more focus on student-driven learning, and less on specific technical content. It has coincided with work in informal spaces[22] and placed an emphasis on self-directed learning and collaboration [21, 29], and the practices of computing[3]. However, this open-ended approach, with its emphasis on empowerment through building an end product, can leave gaps in student knowledge[5].

While the Use–>Modify–>Create (UMC) approach has been used widely, only recently the approach been the focus of research. One study providing choices in modify tasks ("Choose") found no differences in perceived difficulty by students or teachers, despite being potentially harder to teach [20]. A quasi-experimental study on UMC found that students in the UMC treatment found Use–>Modify (UM) tasks easier than control students did. The most important result, however, was that treatment students felt more ownership over larger projects [19]. The authors hypothesized that this was because the scaffolding provided by the UM activity provided them with the skills to get past the required elements of their projects more quickly, allowing them more time and skills to implement their own ideas in their projects.

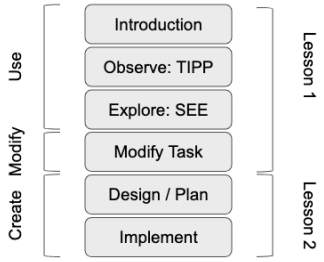

Figure 1: Common module design with two activities.

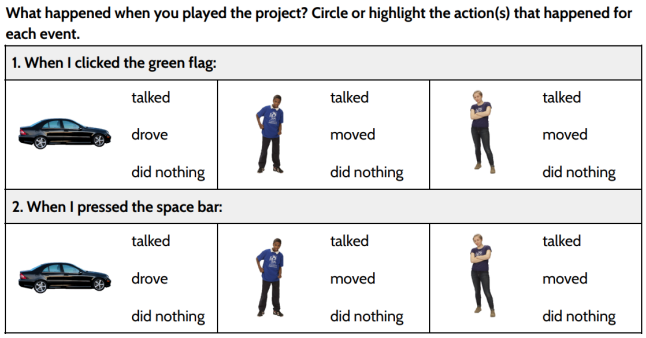

Figure 2: TIPP worksheet guides students through mindfully playing and observing the provided Scratch project.

3 ENACTING USE–>MODIFY–>CREATE

In order to illustrate how UMC was enacted, we provide an overview of Scratch Encore followed by one detailed module.

3.1 Curriculum

Scratch Encore is a 2-3 year intermediate Scratch curriculum designed for students with a year of introductory coding covering sequencing, loops, and conditional logic [10]. It has 15 UMC learning modules, each with 3-5 class sessions (Figure 1).

Each module first introduces the concept by tying the content to examples from students' everyday lives. Students then receive example code for a Use–>Modify activity, scaffolded in their exploration of the project with a worksheet, shown in Figures 2 and 3. TIPP orients students to the project and steps them through playing and recording observation of the project execution. SEE asks them to navigate the Scratch interface and find the code that performed the actions they observed. They then step through small, deliberate modifications to learn how the blocks being introduced behave. They then do a small modify task. Finally, the Create activity involves a largely open-ended challenge where students are given design prompts that lead them towards a project that can utilize the new concept.

3.2 Example Module: Conditional Loops

To more clearly illustrate the principles enacted in Scratch Encore, we provide details on Module 4: Conditional Loops.

The module begins with an introductory discussion about repeated actions and how repeated actions can end after a set number

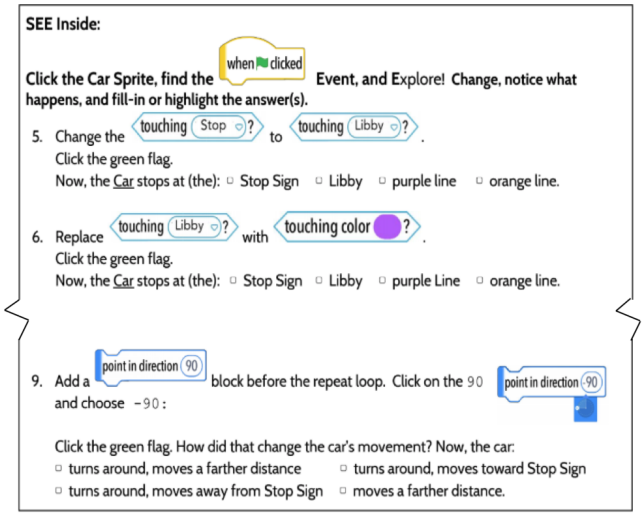

Figure 3: SEE worksheet helps students navigate Scratch and learn blocks through deliberate exploration (tinkering).

- ❑ Choose a transportation sprite
- ❑ Choose a backdrop
- ❑ Arrange the sprites on the stage
- ❑ Build a script so that `When the green flag is clicked:`
  - ❑ Your sprite moves across the stage
  - ❑ Your sprite stops when it is touching either another sprite or a color.
    - ❑ Feel free to add another sprite from the Sprite Library!
  - ❑ When your sprite stops, it should say something or make a sound.

Figure 4: Task list for a Create activity to encourage the use of conditional repeated actions.

of repetitions or based on a condition becoming true. The term conditional loop is introduced, and students brainstorm repeated actions and identify the actions and end conditions. We then provide example code from a project with a recognizable Urban setting reflecting the Youth Culture strand. The worksheets for this activity are shown in Figures 2 and 3. The example program includes scripts for three sprites: a car, a girl, and a boy. A stop sign sprite is also included in the program, but it has no scripts associated with it. When the green flag is clicked, the car sprite moves until it touches the stop sign sprite modeling how a conditional loop works.

In the Modify step, students are encouraged to individualize the project by choosing different car costumes, stopping the car at different locations, changing the speed of the car movement, and adding `say` blocks. If students have extra time, extension activities are provided including adding other sprites, having sprites perform repeated actions until a condition becomes true, and adding sounds.

In lesson 2, students create an open-ended project using conditional loops. Teachers are supplied with a starter project that is pre-filled with different transportation costumes and backgrounds representing popular transportation locations from the students' city. Students are asked to have a sprite move across the stage and stop when it touches another sprite or a specific color. A planning document is provided to help with the selection of sprites, actions, and setting to reinforce what they learned in Lesson 1 (Figure 4).

4 METHODS

In this section, we detail the recruitment of participants in our IRB-approved study, followed by data collected and data analysis methods.

4.1 Recruitment and Participants

For this study, teachers were recruited from a large, urban school district in the United States. Twenty-seven teachers attended professional development for the Scratch Encore curriculum, of which nine were included in this study. Interested teachers were chosen for inclusion based on the number of students they taught (preferring teachers with larger numbers) and the grade levels they taught (5th-7th grade preferred). Classes of less than 10 students were omitted from the data set. In total, there were 19 5th-8th (ages 10-14) grade classes included in the study, totally 536 students. Due to teachers using the curriculum flexibly (as was intended) and classes completing different numbers of modules with the curriculum, the number of students who completed a specific project ranges from 265 to 452.

4.2 Data Collection

Several types of data were collected, including worksheets, observations, transcriptions of teacher interviews and focus groups, and student-authored Scratch programs. Researchers observed instruction for each teacher 2-3 times during the school year (same grade level, typically same classroom). Teachers were interviewed following each observation and participated in focus groups with other teachers at the end of the school year. Scratch projects were publicly available online, organized by pilot teachers into classrooms and studios for analysis.

4.3 Data Analysis

Data was analyzed to better understand two aspects of student work: completion of technical elements and variations from the structured tasks.

For completion of technical elements, the Scratch programs from classrooms were analyzed using static analysis to determine if the work met project requirements and if any module extensions were attempted. This data was then used to calculate a completion percentage for each student. This percentage represents the total number of module requirements completed by students for each activity they attempted in modules 1-4. Since some teachers within our data set taught students in multiple classes, we first analyzed intra-class correlation (ICC) for a hierarchical model predicting completion score by teacher and grade. Due to the low ICC (ICC=0.03), it was determined that a hierarchical model was not necessary and general linear regression models were used to analyze correlations between teacher, grade, and student completion. A multiple regression model was run to determine the influence of teacher and grade on completion percentage and two simple linear regressions were run to separately determine the influence of each variable (teacher and grade) on completion percentage. Assumptions of normality, linearity, homoscedasticity and independence with each regression model were examined through statistical testing and analysis of graphs. All assumptions were determined to be met. Analysis of Variance (ANOVA) were used to compare across the three models.

To understand decisions students made in the Create projects, a number of analyses were conducted. First, we analyzed the blocks used and categorized them based on when they were introduced to students (in a previous module, the current module, or a future module). Next, Scratch projects from a subset of students were inspected to understand in what ways students completed tasks beyond the requirements of the project. Researchers performed qualitative analysis, recording project characteristics such as the number of sprites, backdrops, and costumes used and how students changed their final projects as compared to the starter project that most students were given for the create tasks. These data were then analyzed for frequency. Finally, static analysis modules were coded in JavaScript to analyze the Scratch code in all student projects in a studio and produce a spreadsheet of results. These looked for not only the required elements, but also examples of the types of additional actions that had been identified through hand inspection.

Finally, when differences were identified between teachers, observation notes were inspected to discover any differences in how teachers ran their classrooms.

5 RESULTS

This paper investigates two aspects of instructional materials that utilizes the Use–>Modify–>Create instructional approach: (1) engagement with content and (2) creativity and personalization. As such, the findings section is broken down into two sections that align these two aspect of the curriculum. More specifically, we first explore the trade-offs between requirements and flexibility and the interplay between conceptual learning goals and activity structure (e.g. required tasks and intentional flexibility via extensions and open-ended activities). This includes looking at how UMC allows for different ways of attaining those goals. Second, we explore to what degree and in what ways students express the creativity that Create projects allow.

5.1 Engagement with Technical Content

In this section, we explore to what degree students both engaged with the intended technical content (the concepts taught in the module) and also engaged with additional content (through either suggested extensions or student-led exploration). Our goal is to place emphasis on the former while building in the flexibility for students to also do the latter.

5.1.1 Overall Task Completion. We begin by measuring the level of requirement completion across all classrooms of the first four modules of the curriculum. Figure 5 depicts the average percentage of requirements that students completed each required in each lesson. The lessons are ordered in the same order in which they are completed in the curriculum. We see that requirement completion varies by module, with the fewest requirements being completed by students on the Create project on the last module. In most of the modules, completion rates were between 50% and 75%, a moderate rate.

In order to better understand these results, Figure 6 presents the same data broken down by individual classrooms and ordered so as to see differences across grade level and teacher. In the figure, each bar represents a distinct classroom of students with the color indicating which teacher taught that class. Height of the bars denote

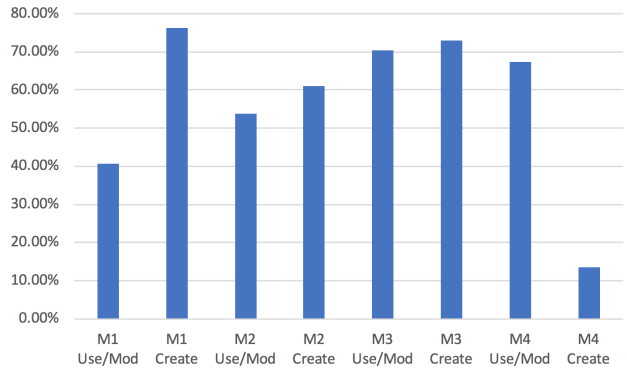

Figure 5: Requirement Completion across Activities

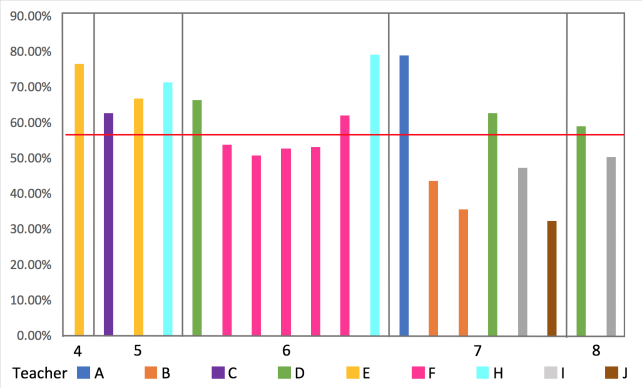

Figure 6: Requirement Completion Versus Grade and Teacher, Animation Module UM Activity

the classrooms' requirement completion rate across all lessons. Finally, each cluster of charts in the figure represents a different grade level, corresponding to 4th-8th grade, respectively.

Finding 1: Students in all grades were able to complete the activities.

Students in lower grade levels did not have more difficulty completing the tasks. In fact, students in lower grades showed slightly higher completion rates for 4th-5th grade than 6th-8th grade, and grade level is predictive in that being in any grade other than 4th grade lowers your predicted completion rate, as described in more detail below. Therefore, we can conclude that the content being presented in the first modules of the Scratch Encore is developmentally appropriate for learners as early as fourth grade (ages 10-11). The slight dip in completion rates in older grades may suggest the curriculum is too easy and therefore did not keep older learners engaged.

Finding 2: Completion rates did differ by teachers.

The performance of classrooms taught by the same teacher was consistent both within and across different grade levels. For instance, classes of teachers D (green), E (yellow), I (gray), and H (light blue) performed similarly even across different grade levels. In order to determine the influence of teacher and grade on completion percentage, we conducted a series of three regressions. First, we find that teachers statistically-significantly impact completion percentage ($F(8, 527) = 31.42$, $p<0.001$, $R^2 = 0.32$). Second, we found the same to be true for grade-level impact completion percentage ($F(4,531) = 31.12$, $p<0.001$, $R^2 = 0.19$). Finally, we used a multiple linear regression to determine the influence of teacher and grade together on completion percentage, again, finding the impact to be statistically significant ($F(12. 523) = 22.03$, $p<0.001$, $R^2 = 0.34$).

An analysis of Variance (ANOVA) was used to determine the least restrictive model predicting completion percentage. Based on this analysis, teacher alone is a better predictor of completion percentage than just grade ($F(4, 527) = 25.9$, $p<0.001$). Additionally, teacher and grade together better predict completion percentage than grade alone ($F(8, 523 = 14.40)$, $p<0.001$) and teacher alone ($F(4, 523) = 2.54$, $p=0.04$). This indicates that teacher significantly impacts the completion percentage of students, especially when considered in conjunction with grade. Teacher alone accounts for 32% of the variance in student completion percentage and 34% of the variance in completion percentage when considered with student grade. Triangulating with classroom observation notes, we find that teachers had very different classroom norms with respect to task completion, which means fidelity, not the curriculum itself, was a major factor. Some teachers required students to complete initial worksheets (not analyzed for this study) prior to being given the modify task, which sets an expectation of adherence to assigned tasks. Other teachers allowed their students to work more independently, perhaps resulting in a lower completion rate. We make no judgement about which approach is better, only that they are different.

5.1.2 Extensions. The first explicit source of flexibility in Scratch Encore is through extensions. While not strictly required for a UMC curriculum, extensions serve as a point of flexibility and is a best practice for accommodating differences in student learning and implementation speeds. The intent of extensions is that students who have completed the requirements can choose additional related tasks, some meant to be purely creative (e.g. adding a new costume or choosing a new sprite) while others exercise new or existing knowledge in interesting ways (e.g. animating another sprite in a different way). In our analysis, we investigate whether students completed extensions as intended (e.g. waiting until requirements are completed before completing them) and when extensions were attempted, whether students chose non-coding extensions or coding extensions.

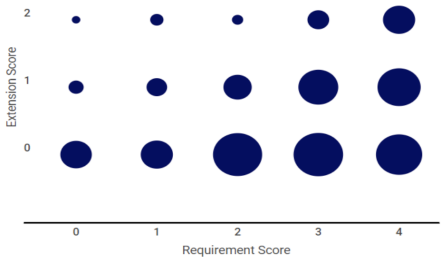

Figure 7: Requirement Completion Versus Extension Completion in Module 4, UM Activity

Figure 7 shows the distribution of students into groups based on how many requirements were completed and extension scores on

the Module 4 (Conditional Loops) UM lesson. In this module, there are four requirements (x axis) and 2 optional extensions (y axis). The size of the circle represents the number of students with that value of requirements and extensions completed.

Finding 3: Students generally complete requirements before working on extensions.

If students were completing requirements (as intended), then only the right-most column of would have entries for the extension scores of 1 or 2. While this is not strictly adhered to, there is a trend in this direction. For the most part, students complete requirements before extensions (as evidence by the fact that the largest circle in each column is the bottom one). Likewise, for each extension, the highest number of students completing it have also completed all of the requirements (as evidenced by the fact that the largest circle in each row is the right-most one). This trend suggests learners are eager to explore extensions and their inclusion in the curriculum is important as students want the flexibility that extensions provide as demonstrated by their choosing to pursue them despite not fully completing the required portions of the activity. However, this could also mean that extensions are a distraction to the intended learning. Extensions are not all directly related to the concept being exercised this project.

5.1.3 Block Usage. Having looked a the data from the more heavily scaffolded portions of the modules (Use and Modify) we now shift focus to the Create projects, looking at the blocks learners choose to use as a means to explore the balance between structured content learning and opportunity for student-driven exploration. For the Create activities, structure is provided in the form of a planning document as a means to encourage students to incorporate the blocks that were introduced in current or previous modules into their programs. However, the assignment is intentionally open-ended enough to enable students to explore new blocks that have not yet been taught. With this analysis, we explore how learners navigate this tensions between the structured aspects (e.g. encouragement to use recently-taught blocks) with the open-ended freedom of the assignment.

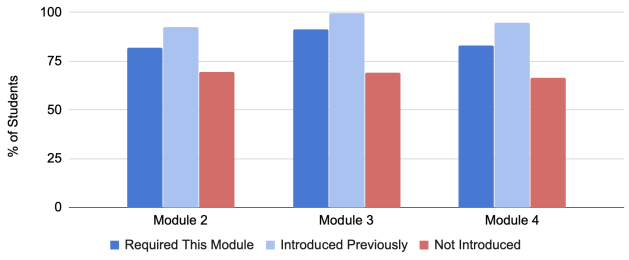

Figure 8: Block Usage across Create activities

Figure 8 presents the blocks that students chose to use in in the Create task from modules 2, 3, and 4. We did not include Module 1 because it is focused on introducing students to the Scratch interface and introduces very few blocks. The dark blue bar indicates the percentage of students who used a block that was introduced in this module, the light blue column shows the percentage of students who used blocks introduced in a previous activity, and the red column shows the percent of students who incorporated blocks that had not yet been introduced as part of Scratch Encore.

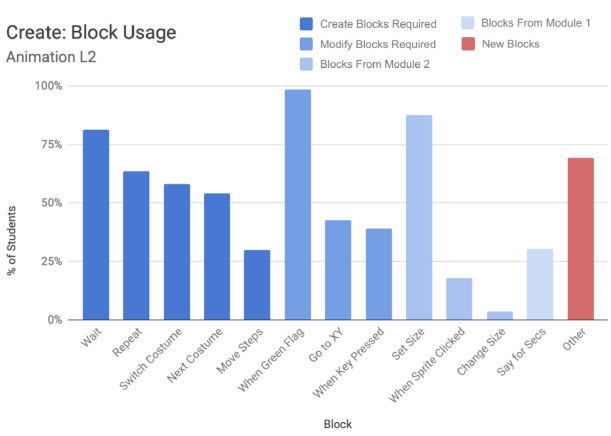

Figure 9: Block Usage in Animation Create Activity

Finding 4: Students incorporate desired computing concepts into open-ended Create projects.

We can see from Figure 8 that over 80% of students use *some* required blocks across the these three modules. Going one step deeper, Figure 9 shows students' block usage in the Create activity of Module 3, which focuses on Animation. Over half of students chose to use required blocks for the Create project, with wide variation by block.

Finding 5: Students incorporate commands from previous modules when completing open-ended Create tasks.

At first glance, it appears that over 90% of students incorporate blocks from previous modules into their projects. Upon close investigation, when looking at what blocks are being incorporated from earlier projects, there is a high level of variance. The blocks When green flag clicked (98%), Set size (88%), and wait (81%) were used very frequently, suggesting that either these blocks are very useful or that students had a high level of comfort with them (or both).

We think that this outcome can be explained by the content reinforcement that is part of the structure of Scratch Encore. By this we mean the higher frequency of exposure to and using certain blocks helped the students understand how and when to use them, resulting in a higher level of comfort with them. Another possible explanation of this outcome is that blocks that were introduced in the beginning of the curriculum serve as building blocks for more complex projects. As a result, they are used frequently in later modules and lessons.

Finding 6: A majority of students are exploring new blocks in the Create activities

Overall, about 70% of students use not-yet-introduced blocks. Looking at Module 3 in particular, 70% of students used at least one block that had not yet been introduced as part of the curriculum. This finding indicates that a majority of students either had prior knowledge of the block or chose to explore and experiment with the full set of blocks. In either case, this finding show the importance of including open-ended aspects of programming activities, either to

support exploration or allow learners to draw on prior knowledge, or both.

This type of learning is aligned with the constructionist design and constructivist learning theory which emphasizes the importance of allowing learners to explore and build new knowledge upon the knowledge they already poses. Knowing that both drew on existing the content just introduced while also incorporating blocks beyond what has been introduced without being instructed explicitly to do so shows that Create tasks are successfully bridging the goals of engaging learners with specific content while also supporting exploration.

5.1.4 Use of Loops. Our final piece of data related to how flexibility and conceptual learning can coexist within a single curriculum via the UMC approach is to look at how students chose to involve loops in their Create projects. Module 3 introduces through the animation of sprites. The UM activities show how looping worked using three blocks: `next costume`, `wait`, and `move X steps`. Therefore, students creating loops with 3 blocks inside could be reasonably expected based on how the concept was introduced. In Module 3's Create task, the use of a repeat block was required, but students had flexibility about how complex the repeating logic could be (e.g. how many blocks inside the loop) and what blocks to use inside the loop. In addition, they were only required to use two loops. To understand the nature of exploration within a structured activity, we investigated students' choices with respect to the contents of loops.

As can be seen in the box plot in Figure 10 that students used many loops. Students used on average, almost 4 loops in their Create projects. In addition, their loops contained an average of almost 7 blocks inside them.

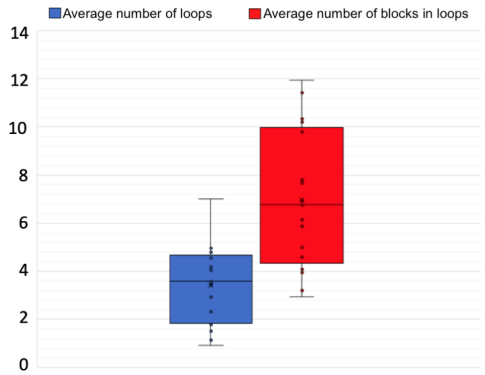

Figure 10: Loop Count and Complexity in Animation Create

Figure 11 shows the most common blocks students used inside their loops. As expected, a majority of the blocks students chose to use matched the blocks introduced by the lesson: wait, move, costume change. However, we see that students also chose to use many other blocks for their loops, including blocks that control sprite motion, sprite appearance, sounds, and controlling script execution speed via the wait block. There were even several nested loops. The fact that learners used the previously taught concept (loops in this case) in a variety of ways shows how the Use-Modify-Create progression can help address the problem of learners tightly coupling concepts with specific contexts or usages that has be identified in previous work on elementary-aged students learning Scratch [34].

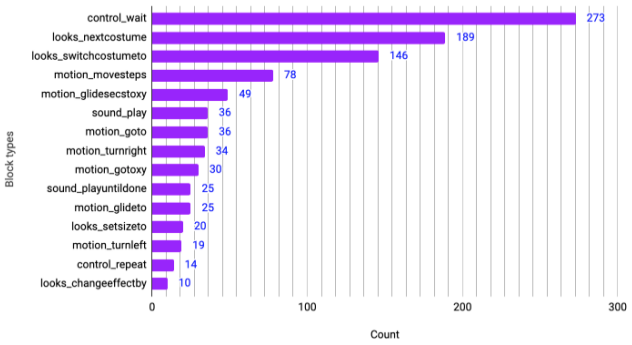

Figure 11: Loop Block Usage in animation Create Activity

5.2 Creativity in Create Activities

We now turn our attention to non-content-focused goals of the Create activities. Create activities are intentionally open-ended, not just to allow student-driven exploration of new blocks, as shown in Sections 5.1.3 and 5.1.4, but also to provide an opportunity for learners to incorporate aspects of themselves in their work, and in doing so, make activities more personally meaningful, more engaging, and to help build an identity as a person who does computing. The constructionist philosophy emphasizes this creative outlet for students' own projects, personalizing them as they desire, using them as vehicles of creativity and expression. However, these aspects of a learning experience can be difficult to measure. In this work, we have analyzed the Create projects in two ways to provide insight into whether students are taking advantage of this opportunity. We focus on Module 4, Conditional Loops, which had a transportation theme.

5.2.1 Narrative Creation. One pattern that emerged from analysis was the use of say blocks to create a narrative, something not suggested in the requirements or extensions. Many students crafted a theme or suggested a story, often using `Say` Blocks, to either create a conversation between sprites, talk about the transportation theme of this specific Conditional Loops lesson, or otherwise reference some other element they included to create a more cohesive project. For example, Figure 12 shows a project that creates an interactive story in which one sprite waits for a train, and after boarding, the user can press the arrow keys to make the train and passenger move out of the station. Students also utilized age-appropriate humor, such as the magic school bus calling the Lamborghini "Lambor-P-P" (Figure 12). One student even created a two-scene story, combining narrative, a conditional loop, and a scene change (Figure 13).

Finding 7: Say blocks are frequently used by learners to personalize projects, often through creating conversations.

Figure 14 shows a histogram of the number of say blocks used in each project. This shows that while about 94 students had no say blocks, 172 students (65%) had at least one say block, with 86 students (32%) having a dialogue containing more than one say block.

Figure 12: Short narrative examples: Catching the train (left). Potty humor in the trash talk (right).

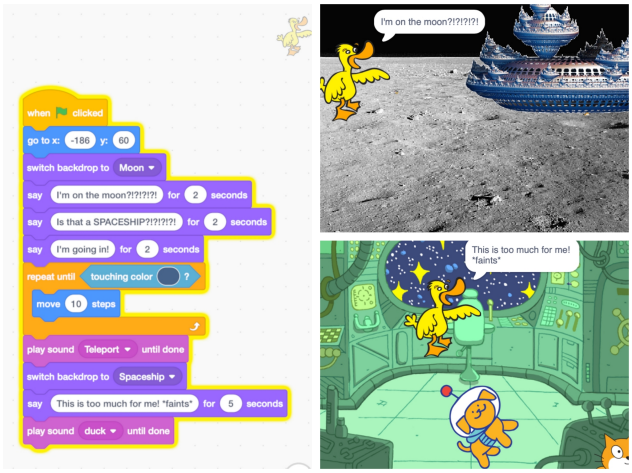

Figure 13: Two-scene Narrative: Spaceship Story

Again, these numbers are significant as the say block had not been formally introduced in the curricular materials, meaning learners either discovered the command or had prior Scratch experience. In either case, this data shows learners going beyond the curriculum either through exploration or incorporating prior knowledge.

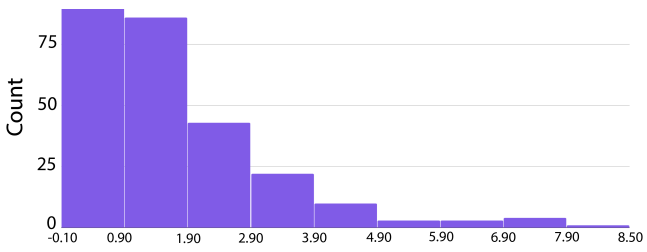

Figure 14: Say Block Usage

5.2.2 Election of Sprites and Backdrops. Sprites and backgrounds define the aesthetics of a Scratch project and are another mechanism for creativity and project personalization. Even with very similar code, students can use different sprites and backgrounds to change the theme and feel of the project. In fact, the Scratch Encore curricular design hinges on this fact - the same academic material can be situated within in very distinct contexts by changing the sprites and backgrounds involved. Therefore, one indicator of students taking ownership of their projects is the amount they personalize it by changing the sprites and backgrounds to something of their own choosing.

The conditional loops module includes a starter project within the Create task to get students started on the project. Students were given a starter project with a main sprite that had 9 vehicle costumes (e.g. train, bus, sedan, SUV), a "stop" sprite with 3 sign costumes (e.g. stop sign, bus stop), and five transportation related backdrop options (e.g. train platform, transit center, parking garage). Students were not required to use the sprites or backdrops from the starter project, but could if they desired. The default project was a train platform with a bus and a bus stop. Of the 8 teachers who taught the Conditional Loops module, 7 teachers (69.1% of students) used the starter project with their students and 1 teacher (30.9% of students) elected to have students begin with a blank project. Because the starting point of students' projects affects their inclusion of sprites and backdrops, these two groups are analyzed separately below.

Create Projects from Starter Project. To analyze how students integrated their own ideas and personality into their projects, we defined four categories the represent the degree to which they did so. At one extreme, students could have made no changes to the sprites or backgrounds. The next category is students keeping the original sprites, but choosing among the costumes and backgrounds already loaded into the project. The third category captures students who added sprites, costumes, or backgrounds from the Scratch image library. Finally, some students uploaded pictures they obtained from outside of Scratch or created their own images using the drawing tools inside Scratch.

Figure 15 breaks down student changes by category of change. The starter project has three elements: A main sprite that will be moving (originally a train but with costumes for different cars, bus, etc.), what will cause it to stop (originally a stop sign), and a background (urban scene). The stacked bar graph indicates the percentage of students who made each kind of change for that particular element. For example, the top (green) portion of the background changes bar indicates the number of students who only utilized the default background, whereas the bottom (blue) portion of backdrop changes is the percentage of students who uploaded their own backdrop. Finally, the Any stacked bar indicates what percentage of students had that high of a level of initiative in any of the three. That is, if a student used ALL initial costumes and backdrop, they would be no changes from default. However, if a student changed to a provided costume for the main sprite but went into the scratch library for a backdrop, they would be categorized as changed to Scratch library because that shows a higher level of initiative.

Finding 8: Almost all students personalized their projects by modifying sprites or backgrounds

Figure 15 breaks down student changes by category of change. Only 11.1% of students (20 students) utilized nothing more than the provided sprite costume and background, as shown in the green portion of the left-most bar. Not only did 53.0% of students (96 students) change from the default main sprite costume to another provided costume, but an additional 11.6% of students (21 students)

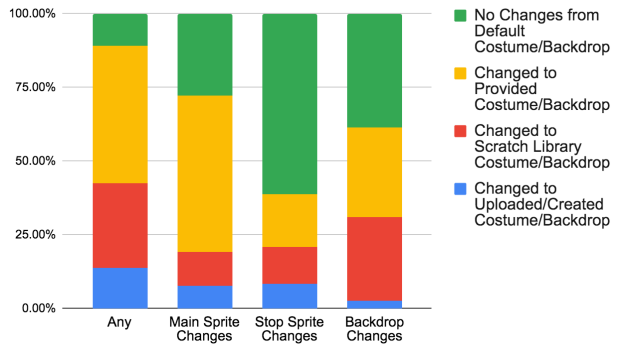

Figure 15: The degree to which students made changes to main sprite, stop sprite, and/or background

changed to a main sprite from the Scratch Sprite Library, and 7.7% (14 students) uploaded or created a costume of their own. These added sprites included food trucks, different types of cars, and car characters (i.e. Lightning McQueen from the Cars movie).

In addition, 66.9% (121 students) used new sprites or costumes that were not included in the starter project either for the main sprite, as described above, or as an additional sprite. Additional sprites were common in student projects, whether they were included within the written code or merely appeared on the stage. In total, 51.4% of student projects (93 projects) include more than 2 sprites with projects having as many as 6 sprites. This often took the form of a sprite standing at the bus stop, stop sign, or on the train platform waiting for the mode of transportation (Figure 16). In some cases, it also took the form of other cars on the road or objects in the background.

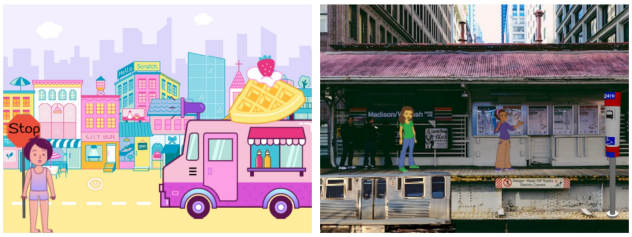

Figure 16: Sprite and Backdrop Examples: Girl in City waiting for Food Truck (left); Girls on Train Platform (right)

Of the students who were in classes that used the starter projects, most students only included one backdrop in their code, 61.3% of students (111 students) changed their backdrop from the default backdrop including 28.3% of students (51 students) adding a backdrop from the Scratch Backdrop Library, 2.8% of students (5 students) uploading or creating their own backdrop, and the rest choosing a backdrop we preloaded into the project. Often times, these backdrops were other city scenes or places where transportation is expected (Figure 16).

Create Projects from Blank Screen. Student creativity within Create projects that did not begin with a starter project was more pronounced when compared against student projects from classes that used the starter projects. Students used an average of 2.8 sprites and an average total of 8.3 costumes in their projects; some projects had as many as 14 sprites. Almost half of the projects (48%; 39 students) incorporated the transportation theme from the modify project. Some did this using sprites or costumes from the Scratch Library (e.g., sailboat, food truck, car). Other projects continued the transportation theme with images students had downloaded from the internet or created on their own (e.g., a Lamborghini, a hand-drawn car, Jeffree Star's custom pink BMW i8, the Magic Schoolbus, a hoverboard, Thomas the Tank Engine). The other half of the projects (52%; 42 students) diverged from the transportation theme and represented aspects of youth culture ranging from video game characters (e.g, Kirby, Pokemon), popular music or television (e.g., Jojo Siwa, Why Don't We), or memes (e.g., Wanna Sprite Cranberry featuring LeBron James, You Got No Jams featuring Kpop stars Rap Monster and Jimin). Over half of the students (54.3%; 44 students) uploaded or created their own costume for use as the main sprite within their project rather than simply using sprites from the Scratch Sprite Library. Backdrops also varied between student projects. While most students still only utilized one backdrop, 70.4% of students (57 students) selected a backdrop from the Scratch Backdrop Library and 29.6% of students (24 students) uploaded or created their own backdrop.

6 DISCUSSION

The goal of this paper is to understand to what degree a Use–Modify–Create curriculum, as enacted through Scratch Encore, provides *structure* that encourages content learning while maintaining flexibility and *open-endedness* that encourages constructionist ideals of student-driven learning and engagement.

6.1 Structure

Overall, the Use –> Modify (UM) activities within the curriculum promoted student learning as evidenced by their completion of project requirements. Although the average completion rate for required tasks was just over 50%, for over half the modules, completion rates were greater than 65%, reaching as high as 76% and students at all grade levels were able to successfully complete the Scratch Encore activities (Finding 1). Yet, variance was very high between teachers, larger than between grade levels (Finding 2).

To provide differentiation for students, Scratch Encore includes both required elements and optional extensions for those who finish early. While these are not specifically part of the UMC framework, they are an application of Universal Design for Learning on UMC. Students complete more requirements than extensions, in general, but some students complete at least one extension prior to finishing all of the requirements (Finding 3). While the presence of extensions, a feature that lies between the structural and open-ended goals of UMC, could distract students from the project requirements that are designed to demonstrate achievement of learning goals, students did, in general, complete more requirements than extensions (Finding 3). The varied completion rate could mean that individual teachers choose how they want to use the curriculum, indicating a level of flexibility in curricular use.

Interviews with teachers revealed challenges in keeping students focused on completing project requirements before moving on to extensions, so teacher fidelity may not be the dominant factor. To address this challenge within the curriculum, materials were revised in two ways: checkboxes were added for each project requirement and extensions were moved so that they were farther from the project requirements.

The UMC structure promotes student learning of specific computing concepts by introducing blocks and concepts within the UM lessons. We found that this structure successfully introduced the desired concepts and that, overall, students were able to use the same blocks taught during the UM lesson when working on their Create project (Finding 4). Additionally, students incorporated blocks and computing content learned during previous modules within the Create tasks of later modules, showing an evolving understanding of CS concepts (Figure 5). Yet, as students are using the blocks that they were taught within the UM structure, they also experiment and do so in new ways that they have not yet been taught, such as nesting loops and using more loops than the requirements mandate in module 3. This demonstrates that although the UM structure allows for the introduction of blocks and CS content, it does not limit students in how they use those blocks and concepts.

Overall, the UM portion of UMC provides a structure that aligns with ZPD [32], scaffolding students' learning by moving through a series of tasks to move toward mastery of new concepts and formally learn CS skills and concepts while providing extensions for students who need more of a challenge.

6.2 Open-Endedness

UM and Create projects allowed for exploration due to their open-ended nature and connection to constructionist goals. For all students, the block exploration exhibited within Create projects demonstrates an opportunity to experiment with constructionist ideas and functionality in Scratch programs. A majority (about 70%) of students incorporated blocks into each Create project that had not been previously introduced (Finding 6). This demonstrates that, although the UM sequence formally introduces blocks to students, they openly explored other available blocks and used blocks that were not yet introduced. One block that students explored to promote their self-expression and individualize their projects was the `say` block (Finding 7). While not novel, the `say` block allowed students to integrate their own commentary within their projects and relate themes, even those provided by the UM or C starter projects, to their own selves, interests, and knowledges.

The open-ended nature of the Create projects gave students the opportunity to incorporate their own elements of self-expression and creativity in their projects. Within the Conditional Loops Create project, nearly all of the students express their own ideas or creativity by changing sprites and backdrops including adding new sprites and backdrops from the sprite libraries and uploading their own images relating to their interest (Finding 8). Especially when students were given a blank Create project encouraging them to develop a project with requirements for the CS content but not the aesthetics of the program, students incorporated aspects of their cultures ranging from popular memes to their vehicles of choice. This trend demonstrates that using the UMC pedagogical approach does not limit the exploration and self-expression that students are able to exhibit through their projects.

When taken as a whole, utilizing the Use–>Modify–>Create pedagogical approach has largely succeeded in providing a balance between structure and student-driven, open-ended exploration and expression. One perspective on this sequence is that Use–>Modify tasks provide additional scaffolding so that students can better spend their time during student-driven Create projects on endeavors related to the conceptual learning goals. Recent research showed that students with UMC had positive affective outcomes in both the UM and Create projects[19]. This paper provides additional support for that finding, showing students utilizing what they learned in UM activities in their Create activities while at the same time making different levels of changes to personalize their projects.

7 LIMITATIONS

While we had a large sample size overall, these students were spread over a wide variety of grade levels and teachers. In addition, we analyzed computational artifacts, but we have not presented analysis of worksheets, classroom observations, and teacher interviews. Such analysis might show correlations between teacher styles and student behaviors. Finally, we do not have a control group using a more structured or a more open-ended curriculum. Therefore, while we have gained knowledge as to how learners experience a UMC-structured curriculum, we cannot make any claims as to how this may differ from other pedagogical strategies. Rather, the contribution of this paper is in documenting how the UMC approach helped bridge the open-ended nature of constructionist learning with the content learning goals of classroom instruction.

8 CONCLUSIONS

Our analysis reveals that the UMC pedagogical approachprovided students a productive balance between structure and open-ended exploration. During open-ended tasks, many students followed guidance in reinforcing new knowledge by utilizing blocks they had just learned. They also explored new blocks not formally introduced, as well as combined existing blocks in new ways. At the same time, students also took advantage of the open-endedness afforded by the Create tasks to explore new blocks as well as to personalize backgrounds, sprites, and narratives to take ownership of their projects and express themselves creatively. Collectively, this work shows one way forward to incorporate the powerful ideas of constructionism learning into the constraints of classrooms, providing one potential path out of the Play Paradox. In doing so, this work helps provide guidance as to one productive way to help bring computing to all learners.

9 ACKNOWLEDGEMENTS

We would like to thank the teachers and students who piloted Scratch Encore. This material is based upon work supported by the National Science Foundation under Grant No. 1738758.

REFERENCES

[1] [n. d.]. Code.org CS Curricula. https://curriculum.code.org/
[2] David Bau, Jeff Gray, Caitlin Kelleher, Josh Sheldon, and Franklyn Turbak. 2017. Learnable programming: blocks and beyond. *arXiv preprint arXiv:1705.09413* (2017).

[3] K. Brennan. 2013. Learning computing through creating and connecting. *Computer* 46, 9 (2013), 52–59.

[4] K. Brennan, C Balch, and M. Chung. 2014. Creative computing. http://creativecomputing.gse.harvard.edu/guide/

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

[6] Seth Chaiklin et al. 2003. The zone of proximal development in Vygotsky's analysis of learning and instruction. *Vygotsky's educational theory in cultural context* 1 (2003), 39–64.

[7] E. Spertus D. Wolber, H. Abelson and L. Looney. 2011. *App Inventor: Create Your Own Android Apps.* O'Reilly Media, Sebastopol, California.

[8] Caitlin Duncan, Tim Bell, and Steve Tanimoto. 2014. Should Your 8-year-old Learn Coding?. In *Proceedings of the 9th Workshop in Primary and Secondary Computing Education (WiPSCE '14)*. ACM, New York, NY, USA, 60–69.

[9] Deborah A Fields, Michael Giang, and Yasmin Kafai. 2014. Programming in the wild: trends in youth computational participation in the online scratch community. In *Proceedings of the 9th workshop in primary and secondary computing education*. ACM, 2–11.

[10] Diana Franklin, David Weintrop, Jennifer Palmer, Merijke Coenraad, Melissa Cobian, Kristan Beck, Andrew Rasmussen, Sue Krause, Max White, Marco Anaya, and Zachary Crenshaw. 2020. Scratch Encore: The Design and Pilot of a Culturally-Relevant Intermediate Scratch Curriculum. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education (SIGCSE '20)*. Association for Computing Machinery, New York, NY, USA, 794–800.

[11] S. Grover, R. Pea, and S. Cooper. 2015. Designing for deeper learning in a blended computer science course for middle school students. *Comput. Sci. Educ* 25, 2 (4 2015), 199–237.

[12] I. Harel and S. Papert. 1990. Software design as a learning environment. *Interactive Learning Environments* 1, 1 (1990), 1–32.

[13] Idit Ed Harel and Seymour Ed Papert. 1991. *Constructionism.* Ablex Publishing.

[14] Michael S. Horn, Corey Brady, Arthur Hjorth, Aditi Wagh, and Uri Wilensky. 2014. Frog Pond: A Codefirst Learning Environment on Evolution and Natural Selection. In *Proceedings of the 2014 Conference on Interaction Design and Children (IDC '14)*. ACM, New York, NY, USA, 357–360.

[15] Yasmin Kafai, Kylie Peppler, and Robbin Chapman. 2009. *The computer clubhouse: A place for youth.* eScholarship, University of California.

[16] Yasmin B Kafai and Mitchel Resnick. 2012. *Constructionism in practice: Designing, thinking, and learning in a digital world.* Routledge.

[17] Caitlin Kelleher and Randy Pausch. 2005. Lowering the Barriers to Programming: A Taxonomy of Programming Environments and Languages for Novice Programmers. *ACM Comput. Surv.* 37, 2 (June 2005), 83–137.

[18] 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.

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

[20] Nicholas Lytle, Veronica Catete, Amy Isvik, Danielle Boulden, Yihuan Dong, Eric Wiebe, and Tiffany Barnes. 2019. From "Use" to "Choose": Scaffolding CT Curricula and Exploring Student Choices While Programming (Practical Report). In *Proceedings of the 14th Workshop in Primary and Secondary Computing Education (WiPSCE'19)*. Association for Computing Machinery, New York, NY, USA, Article 18, 6 pages.

[21] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. 2010. The Scratch Programming Language and Environment. *Trans. Comput. Educ.* 10, 4, Article 16 (Nov. 2010), 15 pages.

[22] John H. Maloney, Kylie Peppler, Yasmin Kafai, Mitchel Resnick, and Natalie Rusk. 2008. Programming by Choice: Urban Youth Learning Programming with Scratch. *SIGCSE Bull.* 40, 1 (March 2008), 367–371.

[23] Amon Millner and Edward Baafi. 2011. Modkit: Blending and Extending Approachable Platforms for Creating Computer Programs and Interactive Objects. In *Proceedings of the 10th International Conference on Interaction Design and Children (IDC '11)*. ACM, New York, NY, USA, 250–253.

[24] Richard Noss and Celia Hoyles. 1996. *Windows on mathematical meanings: Learning cultures and computers.* Vol. 17. Springer Science & Business Media.

[25] S. Papert. 1980. *Mindstorms: Children, Computers, and Powerful Ideas.* Basic Books, Inc.

[26] Seymour Papert. 1993. *The children's machine: Rethinking school in the age of the computer.* ERIC.

[27] Seymour Papert and Idit Harel. 1991. Situating constructionism. *Constructionism* 36, 2 (1991), 1–11.

[28] S. Papert, D. watt, A. diSessa, and S. Weir. 1979. *Final report of the Brookline Logo Project: Project summary and data analysis (Logo Memo 53).* Technical Report. MIT Logo Group, Cambridge, MA.

[29] Ricarose Roque, Yasmin Kafai, and Deborah Fields. 2012. From Tools to Communities: Designs to Support Online Creative Collaboration in Scratch. In *Proceedings of the 11th International Conference on Interaction Design and Children (IDC '12)*. ACM, New York, NY, USA, 220–223.

[30] Bruce Sherin, Brian J Reiser, and Daniel Edelson. 2004. Scaffolding analysis: Extending the scaffolding metaphor to learning artifacts. *The Journal of the Learning Sciences* 13, 3 (2004), 387–421.

[31] C Addison Stone. 1998. The metaphor of scaffolding: Its utility for the field of learning disabilities. *Journal of learning disabilities* 31, 4 (1998), 344–364.

[32] Lev Semenovich Vygotsky. 1980. *Mind in society: The development of higher psychological processes.* Harvard university press.

[33] David Weintrop. 2019. Block-based Programming in Computer Science Education. *Commun. ACM* 62, 8 (July 2019), 22–25. https://doi.org/10.1145/3341221

[34] David Weintrop, Alexandria K Hansen, Danielle B Harlow, and Diana Franklin. 2018. Starting from Scratch: Outcomes of early computer science learning experiences and implications for what comes next. In *Proceedings of the 2018 ACM Conference on International Computing Education Research*. 142–150.

[35] David Weintrop, David C Shepherd, Patrick Francis, and Diana Franklin. 2017. Blockly goes to work: Block-based programming for industrial robots. In *2017 IEEE Blocks and Beyond Workshop (B&B)*. IEEE, 29–36.

[36] David Weintrop and Uri Wilensky. 2015. To Block or Not to Block, That is the Question: Students' Perceptions of Blocks-based Programming. In *Proceedings of the 14th International Conference on Interaction Design and Children (IDC '15)*. ACM, New York, NY, USA, 199–208.

[37] David Wood, Jerome S Bruner, and Gail Ross. 1976. The role of tutoring in problem solving. *Journal of child psychology and psychiatry* 17, 2 (1976), 89–100.