Scaffolding Young Learners' Open-Ended Programming Projects with Planning Sheets

Jennifer Tsan
University of Chicago
Chicago, IL, USA
jennifertsan@uchicago.edu

Donna Eatinger
University of Chicago
Chicago, IL, USA
dmeatinger@uchicago.edu

Alex Pugnali
University of Maryland
College Park, MD, USA
apugnali@umd.edu

David Gonzalez-Maldonado
University of Chicago
Chicago, IL, USA
dagm@uchicago.edu

Diana Franklin
University of Chicago
Chicago, IL, USA
dmfranklin@uchicago.edu

David Weintrop
University of Maryland
College Park, MD, USA
weintrop@umd.edu

ABSTRACT

Given the increasing interest and need to teach students computer science in formal education settings, it is imperative to understand how to do so effectively and equitably. An important step of learning to program is being able to define the objective of a program and then plan out how to implement a program to produce the desired outcome. This step is particularly important in younger learners who may have little experience with programming or trying to create their own technological artifacts. In this paper, we explore how to scaffold young programmers in planning their open-ended programs as part of an intermediate Scratch curriculum for middle grade students. We analyze 203 paper and virtual planning documents from 103 5th-8th grade students. Our results reveal that the students often completed a majority of the document, which was consistent across grade levels. However, we found differences in student completion based on teacher and between physical and virtual documents. This work advances our understanding of how to support novice, young programmers in planning programs.

CCS CONCEPTS

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

KEYWORDS

computer science education, planning, K-8

ACM Reference Format:
Jennifer Tsan, Donna Eatinger, Alex Pugnali, David Gonzalez-Maldonado, Diana Franklin, and David Weintrop. 2022. Scaffolding Young Learners' Open-Ended Programming Projects with Planning Sheets. In *Proceedings of the 27th ACM Conference on Innovation and Technology in Computer Science Education Vol 1 (ITiCSE 2022), July 8–13, 2022, Dublin, Ireland.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3502718.3524769

ITiCSE 2022, July 8–13, 2022, Dublin, Ireland

ACM ISBN 978-1-4503-9201-3/22/07…$15.00
https://doi.org/10.1145/3502718.3524769

1 INTRODUCTION

In a rapidly advancing technological society, there is a need to prepare youth to be informed producers and consumers of the data and technology that surrounds them. As such, understanding effective ways to introduce young learners to foundational programming practices is important. An increasingly popular way to introduce young learners to the practice of programming and the field of computer science more broadly is block-based programming [3, 48]. Many block-based programming environments are open-ended and encourage learners to be inventive in creating their own games, interactive stories, or animations. This open-ended exploratory design is an important feature of Constructionist learning environments [28] and creates a "wide-walls" learning environment that supports many different types of projects [33].

Due to the open-ended nature of introductory programming environments, such as Scratch, it is possible for learners to envision programs beyond the capabilities of the tool or project that would require significant amount of time and skill to implement. At the same time, when used in more formal contexts where there are specific learning goals, such an open-ended context can be in tension with the goals of a teacher or curriculum that seeks to ensure learners engage with desired concepts or employ specific practices. One strategy to help address both of these challenges is the introduction of planning scaffolds to help learners (and instructors) attend to what the program will do and how they will accomplish it.

Planning scaffolds are a common instructional technique and have been explored with young children in subjects like reading [30], writing [5], math [17, 46], science [18], and engineering. In the domain of CS, researchers have explored the use of planning scaffolds with undergraduate [8, 40] and high school students [26], but relatively little work has been done to investigate ways to support younger learners' planning when learning to program.

We developed a series of scaffolds in the form of planning sheets to help students detail their vision within a structure that targets an appropriate level of difficulty and relevant technical content before starting the coding portion of their projects. We then incorporated these planning sheets as part of the Scratch Encore Curriculum, which uses the Use→Modify→Create pedagogical approach [24] to provide opportunities for both structured and open-ended programming challenges. In this paper, we present an analysis of the ways students used the planning sheets to complete their open-ended

Create projects. The Create scaffolds were inspired by "The 5Ws", which is commonly used in K-12 English Language Arts (ELA) [9].

In this paper, we seek to understand how learners complete the scaffold as they undertake open-ended Scratch challenges. We are also interested in understanding how the scaffold presentation informs the ways and the extent to which students use them. More concretely, we answer the following two research questions:

- *RQ1: How, and to what extent, do young learners use a planning sheet to scaffold their open-ended programming projects?*
- *RQ2: How does learner use of the sheet differ based on the learner and other characteristics (grade level, teacher, virtual vs. physical)?*

To answer these questions, we conducted a classroom-based study comprised of 5 teachers and 103 students over the courses of two consecutive school years. Through a mixed-methods analysis of completed planning sheets, we answer our stated research questions, and in doing so, advance our understanding of ways to scaffold young novice programmers in planning their programs and how planning sheets are used by novices.

2 PRIOR WORK

In this section, we review prior work on students learning to program in Scratch and on planning in CS and outside of CS.

2.1 Learning to Program in Middle School

The last decade has seen a rapid growth in research on K-12 CS education [39, 45], including work focused on middle school (grades 6-8, ages 11-14) learners (e.g., [2, 15, 19]). Much of this work has focused on the use of block-based programming generally, and Scratch in particular, as the context in which introductory programming instruction takes place [33, 48]. A number of middle-school curricula have incorporated block-based programming into their instruction, such as Creative Computing [4], Scratch Encore [13], Code.org's Computer Science Discoveries, and the Learning Computer Science Concepts with Scratch Project [25]. Accompanying the design of middle school curricular materials are findings related to productive pedagogical strategies for supporting middle school students learning to program (e.g., [20, 41]) and identifying conceptual sequencing for middle school instruction [12].

2.2 Teaching Planning

One of our main goals is to create equitable CS learning experiences. This includes ensuring our curriculum meets the needs of neurodiverse students and students who are English language learners (ELL). Planning in many subjects has been shown to be effective in scaffolding tasks for all students, and are especially helpful for those who are neurodiverse [18, 21] or ELLs [17, 30]. While we do not focus on these populations of students in this paper, developing effective planning sheets will help us reach our goal.

Here we present prior work on teaching planning. We cover research on planning in CS education. Then we review research on planning in subjects outside of CS.

In CS. When teaching novice learners how to program, instructors and researchers often use Unified Modeling Language (UML) diagrams, [1, 29, 40], flow charts [16, 27], and pseudocode [14, 32]. In areas such as Human-Computer Interaction, storyboarding is often used [42, 43]. Recently, researchers have also begun to investigate new planning formats/strategies such as supporting students in decomposing and chunking programming problems [8, 36].

Very recent work has also focused on scaffolding student planning by having them identify aspects of their projects such as the backstory, actors, important scenes, mechanics, player goal, and aesthetics. These scaffolds were developed for undergraduate [6] and high school [26] students in game design.

As we continue to teach programming to more young learners, we must investigate how to best support them in planning their projects. Common formats such as UML diagrams are likely too complex for students aged 10-14. Of the work we reviewed, only two publications addressed teaching planning to K-12 students [26, 36]. Similar to previous work [6, 26], we focus on supporting students in identifying and recording elements of their programs, such as the sprites/actors, events, and actions/backstory.

Outside of CS. At the K-12 level, students often use planning and comprehension documents in reading [9, 30], writing [10, 17], math [46, 47], and science [18, 21, 35]. While we cannot cover every format that is used by students in this age group, we review ones that are commonly used. The most relevant formats are the 5Ws and story-mapping. The 5Ws (who, what, where, when, why) is a strategy that encourages students to identify elements of a story. Story-mapping is a similar strategy that has students identify and organize a story's characters, settings, actions, problems, and other elements. Both strategies have been shown to improve student reading comprehension [9, 34] and writing [17, 44]. We drew inspiration from these formats because they are appropriate for our age-group, the students are likely to be familiar with this process, and the Create projects are story-based.

3 PLANNING SHEET

The planning sheet was designed to support the open-ended *Create* activities that concluded each module of Scratch Encore and was inspired by the Story Map/5W Questions graphic organizers that are common in English Languages Arts elementary classrooms [9, 17, 34], used to scaffold literature comprehension (story element graphic organizers) and pre-writing activities (Who, What, Where, When, and Why) [9]. In order to help students visualize their Scratch "Stories," our graphic organizers asked students to attend to the Characters, Setting, and Events in their Scratch projects [17]. In doing so, the planning sheet uses something familiar and accessible - Story map/5W scaffold - in order to support learners in doing something novel - designing and implementing a Scratch program.

The planning sheet contains six sections (Figure 1). The first section (top) asks students to define the overall theme of the project. Students are given a few idea prompts to spark their imagination along with an open ended free choice. The remaining five sections align to the 5W questions and are arranged in a grid. The grid rows ask: ***Who** will be in the project?* where students define the sprites that will be used; ***What** are they doing?*, where students identify specific actions that sprites will perform; ***When?*** with suggested event blocks to cause actions where students choose events to start their scripts (note: the scaffolds in this row differ by module to direct students to events relevant to the project and recently covered content); ***Where?***, which prompts students to

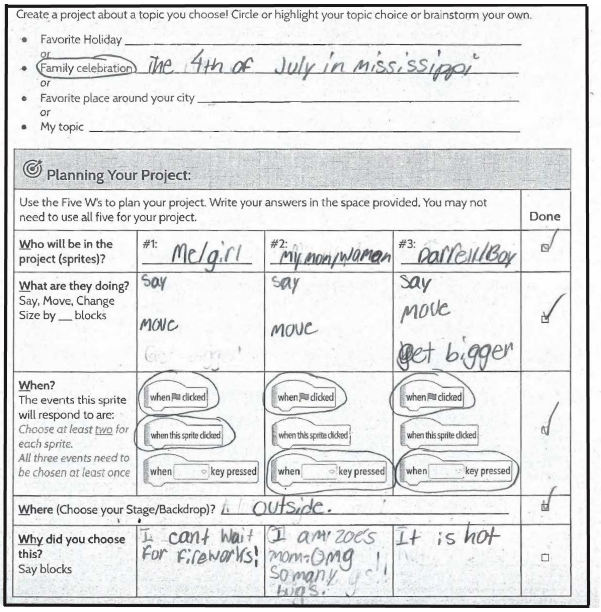


Figure 1: A completed planning sheet for Module 2, Year 1

define a setting (or background) for their project; and ***Why*** *did you choose this?* which provides an opportunity for students to explain their decisions. Each row includes a check box to help learners track whether that aspect of the project has been completed.

Both the physical and virtual versions of the planning sheet followed the same structure and used the same prompts. However, there were some differences between the two formats. For example, we revised the order of questions in the virtual versions of both modules by placing the Why question immediately after the project choice selection in response to feedback we received from teachers. Additionally, the virtual version for module 2 asked students to directly connect sprite actions to specific events (e.g "What will Sprite X do when the green flag is clicked?").

4 METHODS

4.1 Study Context and Curriculum

In this study, students learn to program in the Scratch [33] programming environment through Scratch Encore. The curriculum uses the Use-Modify-Create [24] pedagogical approach to scaffold student learning through a process of gradual release [11]. The curriculum starts with a module that introduces Scratch Basics then gradually introduces more sophisticated concepts and practices (e.g. conditional loops, and decomposition) through other modules. In each module, students first *Use* example code, designed to clearly demonstrate the focal concept of module, which students are familiarized with through the TIPP&SEE strategy [37]. Students then *Modify* the same Scratch project, altering its behavior to accomplish specific outcomes. The module ends with an open-ended *Create* task, that asks students to design and implement a Scratch project that incorporates the focal concept. The students are given a set of constraints for the project but are encouraged to design a project that draws on their interests. To facilitate the *Create* project, students are provided idea prompts (e.g., Tell about your favorite sport, tell a story that your family likes to tell at the holidays).

The planning sheets were given to students at the outset of each *Create* task and then collected at the conclusion of the module. We created two versions of the planning sheet. In year 1, the documents were distributed and completed on paper during class time. In the second year of the project, due to COVID-19 virtual learning accommodations, the planning sheets were converted to a online form that students completed on the computer.

4.2 Participants

In this IRB-approved study, we collected data from middle grade students who attended schools in a large, urban school district in the Midwestern United States. Teachers were recruited with the help of school district collaborators. Students in the participating teachers' classes were invited to participate in the study. Students completed activities spanning 4-5 modules of Scratch Encore. We collected data from two school years (2019 and 2020) and worked with five teachers. For year 1, we collected 58 students responses from two 5th grade and one 7th grade class. In year 2, 45 student responses were collected from 5th through 8th grade classrooms.

Four of the teachers were female (teachers A, B, C, E) and one was male (teacher D). All teachers were white. Only one teacher (teacher A) participated in the both years of the study; their experience as CS teachers ranged from 2-5 years, and their experience with this curriculum ranged from 1-3 years. Table 1 shows the breakdown of students, grade level, and school demographic data.

Year-Tchr	# of Stdnts	Gr Lev	Asian	Black	HSP	White	Other
1-A	23	5	2.4%	60.7%	23%	9.6%	4.3%
1-B	18	5	0%	88.9%	5.6%	2.5%	3%
1-C	17	7	0%	5.9%	92.3%	0.6%	1.4%
2-A	14	5	0%	88.9%	5.6%	2.5%	3%
2-D	13	8	9.7%	1.2%	45.8%	37.5%	6%
2-E	18	6&7	76.1%	1.2%	16.3%	5.2%	<2%

Table 1: Classroom and School Demographics

4.3 Data Collection and Analysis

This paper focuses on students' *Create* project planning sheets from modules 2 (events) and 3 (animation), over two consecutive school years. We will refer to the modules and years in the form of MxYx (e.g. Module 2, year 1 is M2Y1). In this study, we use a mixed methods design to analyze a total of 203 planning sheets from 103 students.

Each completed planning sheet was qualitatively coded. To analyze the planning sheets, 4 researchers iteratively developed a coding manual [38], which attended to the different planning sections of the documents (e.g. project choice, Who: sprites, What: actions, When: events, Where: background/ setting, and Why this project was chosen). To analyze the planning sheets, we coded each cell under the sections of the planning sheet, marking both whether

the cell was completed as well as the contents of student responses. Every planning sheet was given a completeness score based on the number of cells that were filled in. In our coding phase, we counted the Events section as completed if at least one event was chosen.

To understand the contents of each section, 4 researchers individually coded the same 20% of the documents for each pairing (e.g., M2Y1, M3Y1). We calculated the Interrater Reliability (IRR) using Cohen's Kappa (κ = 0.822-0.918), resulting in almost perfect agreement [23]). The team met and reached complete agreement through discussion. Finally, we coded the remainder of the planning sheets individually. To determine if there were any statistically significant differences in the planning sheets between the type of document (physical vs. virtual), grade level, and teacher, we used the Kruskal Wallis test (our data did not meet the assumption of normality or homogeneity so we could not use the one-way ANOVA).

Additionally, we developed a coding manual for coding the Events cells of the M3Y1 sheets to determine whether the students' open-ended answers aligned with Scratch events. The codes types included: *Scratch* (e.g. "green flag", "GF"), *real-life with sequence* (e.g. "after the coach talks"), *real-life without sequence* (e.g. "when the boards are chopped."), *action only* (where the answers did not involve an event, such as, "move to edge of screen"), *other* (e.g. "No Movement"), and *blank* (student did not fill in the cell). Two researchers individually coded all responses, resulting in an IRR of (κ = 0.837) and came to a complete agreement.

5 RESULTS

In this section we begin by focusing on the students' use of the planning sheet. We specifically attend to how students used the planning sheets and how characteristics related to students' use of the planning sheets (teacher, grade level, virtual vs. physical).

5.1 RQ1: How do Students Use the Sheets?

We explore this question in two phases. First, we look across the full set of planning sheets to provide a quantitative picture of how they were used. Next, we present and discuss two completed planning sheets to further illustrate students' use of the sheets.

Finding 1: Students completed a majority of the planning sheets (90%+) across all modules and years. Figure 2 shows the completion percentage of each section by module and year. Of the six sections, "Actions" (χ^2=3.8788, df=3, p=0.27), "Events" (χ^2=4.3579, df=3, p=0.23), and Background (χ^2=6.524, df=3, p=0.09) are the most consistent across years and modules.

For the sections "Project Choice" (χ^2=32.443, df=3, p=4.2×10^{-7}), "Why" (χ^2=21.351, df=3, p=$8.9\text{x}10^{-5}$), and "Sprites" (χ^2=9.8214, df=3, p=0.02), the differences are statistically significant between modules and years. A post hoc test (Games-Howell Test) on the "Project Choice" revealed that the differences between M2Y1–M3Y1 (p=0.0002), M2Y2–M3Y1 (p=0.002), and M3Y1–M3Y2 (p=0.002) are statistically significant. This is likely the result of the formatting of the sheet; the question was located outside of the grid and was easy to miss (Figure 4). For the "Why" section, a post hoc test showed that the differences were between the following pairs: M2Y1–M2Y2 (p=0.002), M2Y1–M3Y2 (p=0.002), M2Y2–M3Y1 (p=0.034), and M3Y1 – M3Y2 (p=0.034). The sheets of the modules of the same year did not differ (i.e., M2Y1–M3Y1 and M2Y2–M3Y3). This suggests

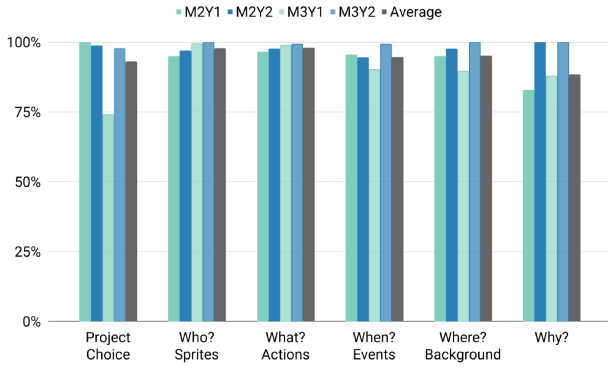


Figure 2: Completion rate in each section.

that student behavior may be related to the modality of the sheets (explored further in Section 5.2). A post hoc test on the "Sprites" revealed that the differences were not statistically significant.

Finding 2: Students' open-ended responses in the "When: Event" section for M3Y1 included Scratch (54%), Real-life (28%), and non-events (11%). One noticeable difference between the two planning sheets is how the students completed the ***When?*** portion of the plan. In the M2Y1, M2Y2, and M3Y2 plans, we provided Scratch event block choices for students to circle, but the M3Y1 plan for this portion was more open-ended. We designed the sheets this way because M2 is focused on Events, and we hypothesized that the students would be more comfortable choosing and using Scratch events by M3. However, only 54% of the M3Y1 answers included Scratch events (Figure 3). Twenty-eight percent of the answers were real-life events (7.5% with sequence and 20% without sequence). An additional 7.5% did not specify events at all and only listed actions. We hypothesize that real-life events may prove difficult to implement the way the students intended. This emphasizes the importance of scaffolding students in their planning and providing more explicit explanations and examples on CS vocabulary.

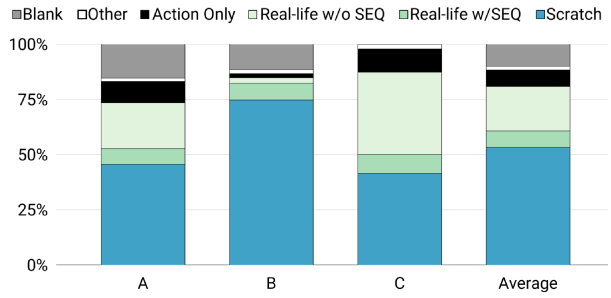
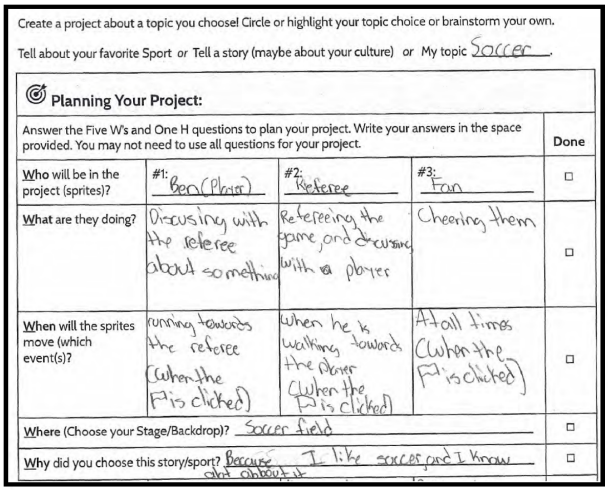


Figure 3: Types of open-ended event responses.

We now present examples of completed planning sheets.

Example 1: 4th of July Figure 1 describes a family celebration "The 4th of July in Mississippi." The student completed the planning sheet for all three sprites. The ***What?*** actions and ***When?*** events listed/circled mirror the planning sheet suggestions and existing blocks within the Scratch platform. Based on this completed sheet,

it is clear that the student is planning to create a Scratch Project to illustrate a story about a family celebration by having three family sprites talking about the holiday and moving around outside.


Create a project about a topic you choose! Circle or highlight your topic choice or brainstorm your own.

Tell about your favorite Sport *or* Tell a story (maybe about your culture) *or* My topic Soccer.

Planning Your Project:

Answer the Five W's and One H questions to plan your project. Write your answers in the space provided. You may not need to use all questions for your project.

				Done
Who will be in the project (sprites)?	#1: Ben (Player)	#2: Referee	#3: Fan	☐
What are they doing?	Discussing with the referee about something	Refereeing the game and discussing with a player	Cheering them	☐
When will the sprites move (which event(s)?	running towards the referee (when the F is clicked)	When he is walking towards the player (when the F is clicked)	At all times (when the F is clicked)	☐
Where (Choose your Stage/Backdrop)?	Soccer field			☐
Why did you choose this story/sport?	Because I like soccer and I know alot about it			☐

Figure 4: A completed planning sheet for M3Y1.

Example 2: Soccer Game Figure 4 describes the student's favorite sport: soccer. Similar to the last project, this one includes a plan for three sprites. The ***What?*** actions and ***When?*** events on the other hand are more open-ended and have a less direct translation to existing Scratch programming blocks. The response still provides an indication of what the sprites will do in the project and can be feasibly completed using Scratch, like using say block to discuss soccer and sound block to cheer.

Finding 3: These examples show how the provided planning scaffold allows for differentiation in completion based on how students want to express their ideas. While the students filled out the sheets differently, they both completed all sections and portrayed a project that can be completed in Scratch and used blocks that students were taught prior to these projects.

5.2 RQ2: Completion by Characteristic

Our second research question investigates differences in students' use of the planning sheet based on their grade, teacher, and the format of the sheet: a physical sheet of paper vs. a virtual form.

Finding 4: Student plan completeness did not differ across grade level. Looking at differences in planning completion by the students' grade level, we find no difference. Although Figure 5 indicates that the 5th grade students, which were the younger students, had more variation in the completeness of their documents, the difference was not statistically significant (χ^2=5.27, df=3, p=0.15). This suggests that the scaffolding was appropriate for the students across the middle school grades.

Finding 5: Students plan completeness differed by teacher. When grouped by teacher, there is slightly more variation (Figure 6). The differences in plan completion between students taught by each teacher was statistically significant (χ^2=12.86, df=4, p=0.01). A post hoc test (Games-Howell) revealed that the differences were between teachers A and D (p=0.003) and teachers A and E (p<0.001).

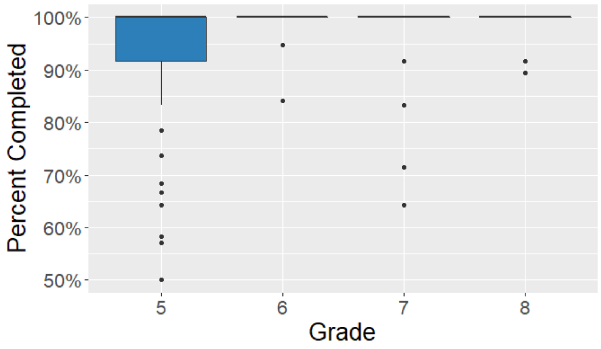


Figure 5: Completion rate by grade.

These differences are likely due to teacher fidelity in using the planning sheets in their classrooms.

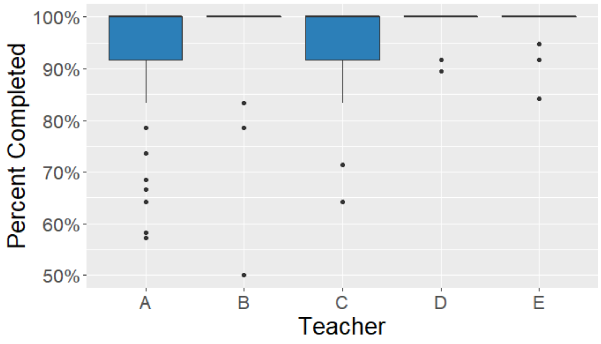


Figure 6: Completion rate by teacher.

Finding 6: Students open-ended responses to the "When: Event" section differed by teacher. Similar to the previous finding, when we conducted a Fisher's exact test on the students' categorized "When: Event" for M3Y1, we found a statistically significant difference (p=2.386^{-5}). This differences, shown in Figure 3, further supports the idea that teacher implementation affects student behaviors and outcomes.

Finding 7: Students plan completeness differed between planning format. Our response to the COVID-19 pandemic required us to transition our paper materials into an electronic form to accommodate virtual learning. Figure 7 shows the differences in how complete the students' submissions were between the two modalities. This difference is statistically significant (χ^2=12.8, df=1, p<0.001). This finding suggests that the virtual scaffolding was more effective than the paper scaffolding.

6 DISCUSSION

This study investigated ways to support young programmers in planning their programs. In designing scaffolded planning sheets for students to complete before starting on their programs and aligning those sheets with the programming environment they were going to use (i.e., asking about sprites and events before writing a Scratch program), the planning sheet was able to support novices

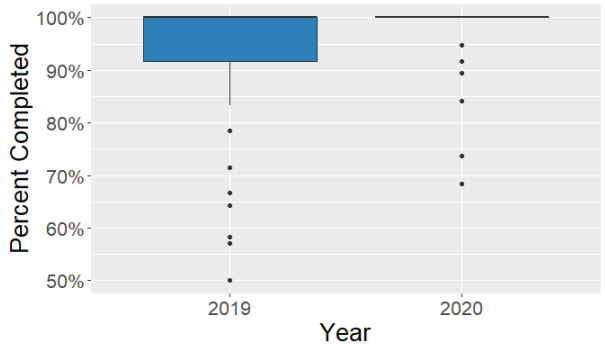


Figure 7: Completion rate by year/modality.

in planning achievable programs that still support the creativity and flexibility that is central to the design of Scratch.

6.1 Drawing Inspiration from Other Disciplines

The design challenge we pursued in this work, scaffolding the planning process, was one that had not been directly addressed in the computing education literature for the intended audience. However, this was a question that has long been a focus in other disciplines. We drew from a well-established pedagogical approach from the field of English Language Arts: Story Maps/5W Questions, which computing education researchers rarely draw from; although there are some notable examples of computing education researchers looking to ELA, reading strategies, and narrative composition for inspiration to support CS learning (e.g., [31, 37, 49]). Given the deep connections between introductory computing and storytelling (e.g., [19]), this work contributes to the scholarship drawing inspiration from ELA for improving computing instruction.

6.2 Connecting to Prior Knowledge and Interest

A second noteworthy feature of the chosen approach for scaffolding planning was the way in which the planning sheet remained true to one of the central design goals of Scratch: supporting personal expression [33]. By presenting learners with open prompts that invited them to incorporate themes that could draw on prior knowledge or cultural resources (e.g., "Favorite place around your city?", "Tell a story (maybe about your culture)"), the planning sheet provided a clear pathway for learners to incorporate their own interests and cultures into their *Create* projects. This can be seen in the two highlighted projects, which tell the stories of young learners celebrating holidays with their families and engaging in their favorite sports. By beginning the planning activity with prompts that allow learners to frame their programs in familiar and/or favored contexts, the remainder of the planning activity is situated within a space of comfort and can support a final program that reflect who the leaner is and what they value and enjoy.

6.3 Open-ended vs. Constrained Programming

A final noteworthy topic is related to how this planning sheet can help resolve tensions between structured curricula and more open-ended, exploratory learning. The Use→Modify→Create pedagogical strategy serves as a way to introduce learners to computing content in accessible ways en route to them creating their own projects that adhere to the constructionist ethos of learner-directed exploration [11]. The planning sheets introduced in this work serve as an additional, complementary scaffold to incorporate into the Use→Modify→Create structure as they can help learners design *Create* projects that are both achievable and support personal expression and creativity. We also envision planning sheets that build off of what is presented above to include additional prompts or constraints to ensure the focal concepts or practices are also included in the *Create* project to further reinforce the learning experience. By layering scaffolded planning sheets into the Use→Modify→Create framework, we show yet another way that the tension between open-ended, exploratory learning and more constrained, content-drive forms of instruction can be resolved through design.

Although our findings are promising, the differences between student performance based on teachers require further investigation. Notable work on planning revealed that novice CS students often struggle with idea development and program planning and would likely benefit from explicit instruction [7, 22]. Together, this suggests that teacher implementation fidelity has a great affect on how students fill out the planning sheets, which could also affect their final projects. This is an important avenue for future work.

Limitations. As with all qualitative studies, there is a potential of researcher bias. We worked to minimize those biases through discussion. Additionally, one out of five teachers that participated in the study both years and was more familiar with the curriculum than some of the other teachers. However, this reflects a real school environment where teachers have varying levels of experience and skills. Finally, some teachers taught Module 3 around February/March of 2020; the pandemic and switch to digital documents likely affected the students' work.

7 CONCLUSIONS

With the increase in students being introduced to programming and computer science, it is important to ensure that curricula include age- and content-appropriate scaffolds. This will lead to a more equitable learning experiences for all students. Planning has shown to be important in many K-12 subjects and has also benefited students who have been historically marginalized. We developed a planning sheet for middle grade (5th-8th) CS learners based on scaffolds that are commonly used for this age group in English Language Arts. We found promising results about how students use the sheets, however these results differ depending on the students' teachers. This suggests that in addition to improving our sheets, we should investigate teacher implementation fidelity of the sheets. Additionally, we will investigate the relationship of the planning sheets' completion to how the students implement the projects.

8 ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 1738758.

REFERENCES

[1] Sohail Alhazmi, Charles Thevathayan, and Margaret Hamilton. 2021. Learning UML sequence diagrams with a new constructivist pedagogical tool: SD4ED. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 893–899.

[2] Ashok R Basawapatna, Kyu Han Koh, and Alexander Repenning. 2010. Using scalable game design to teach computer science from middle school to graduate school. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*. 224–228.

[3] David Bau, Jeff Gray, Caitlin Kelleher, Josh Sheldon, and Franklyn Turbak. 2017. Learnable programming: blocks and beyond. *Commun. ACM* 60, 6 (2017), 72–80.

[4] K Brennan, M Chung, and J Hawson. [n. d.]. Creative computing: A design-based introduction to computational thinking. *https://creativecomputing.gse.harvard.edu/guide/* ([n. d.]).

[5] Marjorie Brown. 2011. Effects of Graphic Organizers on Student Achievement in the Writing Process. *Online Submission* (2011).

[6] Alexander Card, Wengran Wang, Chris Martens, and Thomas Price. 2021. Scaffolding Game Design: Towards Tool Support for Planning Open-Ended Projects in an Introductory Game Design Class. In *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)*. IEEE, 1–5.

[7] Francisco Enrique Vicente Castro and Kathi Fisler. 2016. On the interplay between bottom-up and datatype-driven program design. In *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*. 205–210.

[8] Umberto Costantini, Violetta Lonati, and Anna Morpurgo. 2020. How plans occur in novices' programs: A method to evaluate program-writing skills. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 852–858.

[9] Tim Crabtree, Sheila R Alber-Morgan, and Moira Konrad. 2010. The effects of self-monitoring of story elements on the reading comprehension of high school seniors with learning disabilities. *Education and Treatment of Children* (2010), 187–203.

[10] Laura Nicole Delrose. 2011. Investigating the use of graphic organizers for writing. (2011).

[11] Diana Franklin, Merijke Coenraad, Jennifer Palmer, Donna Eatinger, Anna Zipp, Marco Anaya, Max White, Hoang Pham, Ozan Gökdemir, and David Weintrop. 2020. An Analysis of Use-Modify-Create Pedagogical Approach's Success in Balancing Structure and Student Agency. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*. 14–24.

[12] Diana Franklin, Gabriela Skifstad, Reiny Rolock, Isha Mehrotra, Valerie Ding, Alexandria Hansen, David Weintrop, and Danielle Harlow. 2017. Using upper-elementary student performance to understand conceptual sequencing in a blocks-based curriculum. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*. 231–236.

[13] Diana Franklin, David Weintrop, Jennifer Palmer, Merijke Coenraad, Melissa Cobian, Kristan Beck, Andrew Rasmussen, Sue Krause, Max White, Marco Anaya, et al. 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*. 794–800.

[14] Stuart Garner. 2007. A program design tool to help novices learn programming. *ICT: Providing choices for learners and learning* (2007), 321–324.

[15] Shuchi Grover, Roy Pea, and Stephen Cooper. 2015. Designing for deeper learning in a blended computer science course for middle school students. *Computer science education* 25, 2 (2015), 199–237.

[16] Dee Gudmundsen, Lisa Olivieri, and Namita Sarawagi. 2011. Using visual logic©: three different approaches in different courses-general education, CS0, and CS1. *J. Comput. Sci. Coll* 26, 6 (2011), 23–29.

[17] Salem Saleh Khalaf Ibnian. 2010. The Effect of Using the Story-Mapping Technique on Developing Tenth Grade Students' Short Story Writing Skills in EFL. *English Language Teaching* 3, 4 (2010), 181–194.

[18] Elizabeth M Jackson and Mary Frances Hanline. 2020. Using a concept map with RECALL to Increase the comprehension of science texts for children with autism. *Focus on Autism and Other Developmental Disabilities* 35, 2 (2020), 90–100.

[19] Caitlin Kelleher, Randy Pausch, and Sara Kiesler. 2007. Storytelling alice motivates middle school girls to learn computer programming. In *Proceedings of the SIGCHI conference on Human factors in computing systems*. 1455–1464.

[20] Jordana Kerr, Mary Chou, Reilly Ellis, and Caitlin Kelleher. 2013. Setting the scene: Scaffolding stories to benefit middle school students learning to program. In *2013 IEEE Symposium on Visual Languages and Human Centric Computing*. IEEE, 95–98.

[21] Victoria F Knight, Fred Spooner, Diane M Browder, Bethany R Smith, and Charles L Wood. 2013. Using systematic instruction and graphic organizers to teach science concepts to students with autism spectrum disorders and intellectual disability. *Focus on autism and other developmental disabilities* 28, 2 (2013), 115–126.

[22] Kyungbin Kwon. 2017. Novice programmerâ€™ s misconception of programming reflected on problem-solving plans. *International Journal of Computer Science Education in Schools* 1, 4 (2017), 14–24.

[23] J Richard Landis and Gary G Koch. 1977. The measurement of observer agreement for categorical data. *biometrics* (1977), 159–174.

[24] 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.

[25] Orni Meerbaum-Salant, Michal Armoni, and Mordechai Ben-Ari. 2013. Learning computer science concepts with scratch. *Computer Science Education* 23, 3 (2013), 239–264.

[26] Alexandra Milliken, Wengran Wang, Veronica Cateté, Sarah Martin, Neeloy Gomes, Yihuan Dong, Rachel Harred, Amy Isvik, Tiffany Barnes, Thomas Price, et al. 2021. PlanIT! A New Integrated Tool to Help Novices Design for Open-ended Projects. In *Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*. 232–238.

[27] Isaac Nassi and Ben Shneiderman. 1973. Flowchart techniques for structured programming. *ACM Sigplan Notices* 8, 8 (1973), 12–26.

[28] Seymour Papert. 1980. " Mindstorms" Children. *Computers and powerful ideas* (1980).

[29] Marian Petre. 2013. UML in practice. In *2013 35th international conference on software engineering (icse)*. IEEE, 722–731.

[30] Sam D Praveen and Premalatha Rajan. 2013. Using Graphic Organizers to Improve Reading Comprehension Skills for the Middle School ESL Students. *English language teaching* 6, 2 (2013), 155–170.

[31] Chris Proctor and Paulo Blikstein. 2017. Interactive fiction: Weaving together literacies of text and code. In *Proceedings of the 2017 Conference on Interaction Design and Children*. 555–560.

[32] Haider Ali Ramadhan. 2000. Programming by discovery. *Journal of Computer Assisted Learning* 16, 1 (2000), 83–93.

[33] Mitchel Resnick, John Maloney, Andrés Monroy-Hernández, Natalie Rusk, Evelyn Eastmond, Karen Brennan, Amon Millner, Eric Rosenbaum, Jay Silver, Brian Silverman, et al. 2009. Scratch: programming for all. *Commun. ACM* 52, 11 (2009), 60–67.

[34] D Ray Reutzel. 1985. Story maps improve comprehension. *The Reading Teacher* 38, 4 (1985), 400–404.

[35] Veronica Roberts and Richard Joiner. 2007. Investigating the efficacy of concept mapping with pupils with autistic spectrum disorder. *British Journal of Special Education* 34, 3 (2007), 127–135.

[36] Jean Salac. 2020. Diagramming as a Strategy for Primary/Elementary-Age Program Comprehension. In *Proceedings of the 2020 ACM Conference on International Computing Education Research*. 322–323.

[37] Jean Salac, Cathy Thomas, Chloe Butler, Ashley Sanchez, and Diana Franklin. 2020. TIPP&SEE: A Learning Strategy to Guide Students through Use-Modify Scratch Activities. In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education*. 79–85.

[38] Johnny Saldaña. 2021. *The coding manual for qualitative researchers*. SAGE Publications Limited.

[39] Sue Sentance, Erik Barendsen, and Carsten Schulte. 2018. *Computer Science Education: Perspectives on Teaching and Learning in School*. Bloomsbury Publishing.

[40] Michael Striewe and Michael Goedicke. 2014. Automated assessment of UML activity diagrams. In *Proceedings of the 2014 conference on Innovation & technology in computer science education*. 336–336.

[41] Addison YS Su, Chester SJ Huang, Stephen JH Yang, Ting-Jou Ding, and YZ Hsieh. 2015. Effects of Annotations and Homework on Learning Achievement: An Empirical Study of Scratch Programming Pedagogy. *J. Educ. Technol. Soc.* 18, 4 (2015), 331–343.

[42] Jakita O Thomas. 2018. The Computational Algorithmic Thinking (CAT) Capability Flow: An Approach to Articulating CAT Capabilities over Time in African-American Middle-school Girls. In *Proceedings of the 49th ACM Technical Symposium on Computer Science Education*. 149–154.

[43] Khai N Truong, Gillian R Hayes, and Gregory D Abowd. 2006. Storyboarding: an empirical determination of best practices and effective guidelines. In *Proceedings of the 6th conference on Designing Interactive systems*. 12–21.

[44] Kayo Tsuji. 2017. Implementation of the Writing Activity Focusing on 5W1H Questions: An Approach to Improving Student Writing Performance. *LET Journal of Central Japan* 28 (2017), 1–12.

[45] Jan Vahrenhold, Quintin Cutts, and Katrina Falkner. 2019. *Schools (K–12)*. Cambridge University Press, 547–583. https://doi.org/10.1017/9781108654555.019

[46] Delinda van Garderen and Amy M Scheuermann. 2015. Diagramming word problems: A strategic approach for instruction. *Intervention in School and Clinic* 50, 5 (2015), 282–290.

[47] David W Walker and James A Poteet. 1990. A Comparison of Two Methods of Teaching Mathematics Story Problem-Solving with Learning Disabled Students.. In *National Forum of Special Education Journal*, Vol. 1. ERIC, 44–51.

[48] David Weintrop. 2019. Block-based programming in computer science education. *Commun. ACM* 62, 8 (2019), 22–25.

[49] Robert Whyte, Shaaron Ainsworth, and Jane Medwell. 2019. Designing for Integrated K-5 Computing and Literacy through Story-making Activities. In *Proceedings of the 2019 ACM Conference on International Computing Education Research*. 167–175.