AU: To indicate corrections to these proofs, please use latest Adobe Acrobat Use the Commenting tool, which can be accessed on the current version by right clicking on the PDF and selecting Add Sticky Note. Position the Sticky Note where you want the correction made and insert revision text inside the Sticky Note. Text you wish to revise can also be highlighted by selecting the text, right clicking, and selecting Highlight Text. The Strikethrough Text option can be used indicated text to delete. **Please DO NOT directly edit or copy and paste corrections to the PDF**. Doing so will adversely affect formatting and mangle the text. Failure to comply to these requirements will result in corrected proofs not being accepted and returned to the author.

DO NOT EDIT ON DROPBOX. You must supply standalone files for each chapter

CHAPTER 5

LOOMING CODE

A Model, Learning Activity, and Professional Development Approach for Computer Science Educators

Heidee Vincent
Utah State University

Victor R. Lee
Utah State University

Aubrey Rogowski
Utah State University

Mimi Recker
Utah State University

ABSTRACT

As school librarians are asked to take on more responsibilities outside of their regular duties, teaching coding is one such responsibility that falls outside their area of expertise. In response to these challenges faced by school librar-

Professional Development for In-Service Teachers, pages 121–139

ians, this chapter proposes a model for designing computer science activities and instruction called Expansively-framed Unplugged. The chapter describes this model (also known as EfU) and demonstrates how it can be used to design activities such as Looming Code, a coding activity where students use Scratch to model existing patterns in a weaving as well as create new ones. This chapter describes the experience of an elementary school librarian as she attended professional development sessions to learn the activity and implemented it in two of her elementary school classes.

As providing equitable access to computer science (CS) education becomes a greater priority for many school districts throughout the United States, administrators are struggling to find additional time for scheduling learning in an already-packed school day. Because of this lack of available instructional minutes in the school day, many school librarians at the elementary and middle school levels are increasingly being asked to take on CS education responsibilities in their libraries, often in spite of their lack of training and familiarity with the subject (Martin, 2017). While school districts may sponsor professional development (PD) sessions and provide curricula, these districts are looking to librarians to teach computer coding when they have never been a student of this topic. In addition, due to budget constraints, many elementary school librarians begin working as part-time aids and are later hired as full-time librarians without ever completing a degree or certification as a school librarian or teacher. All together, these are challenging conditions for librarians to overcome in offering quality CS instruction in addition to performing their regular duties.

In this chapter, we describe a model, instructional activity, and PD approach to help school librarians or other educators with limited coding background to learn to lead coding activities in their libraries. The activities are comprised of both coding as well as "unplugged" computing activities—ones that do not require a digital computer (e.g., Bell et al., 2009). As librarians typically lack specific coding knowledge, the professional development takes a teachers-as-learners approach in which librarians are first involved as participants in the instructional activity. In this way, they first draw upon their knowledge of a familiar domain, the unplugged activity, and learn how this applies and transfers to coding concepts.

Our work begins with an observation that many of the librarians we have worked with engage in crafting as a hobby and are familiar with several different crafting media, such as paper, fabric, and paint (Lee & Vincent, 2019). Even if some librarians do not identify themselves as having a particular talent for crafting, we propose that physical crafting materials provide a concrete and tangible medium in which to explore coding concepts, and familiar media such as paper, markers, and yarn offer a low threshold to those who may be wary of learning computer programming. These aspects form the unplugged portion of our model and instructional activity.

The theoretical model underlying our approach draws upon a situated account of transfer, called expansive framing (Engle et al., 2012). This model suggests that students need extra support to fully expand the use context for new knowledge from the learning context to the transfer context, and that instructors can make that transition smoother by making multiple and frequent connections back and forth between the social context at learning and the social context at transfer. Expansive framing informed the design of the instructional activities to produce learning environments where coding is framed in real-world and authentic contexts, so that students more easily learn and retain new skills and educators (here, the librarians) are able to draw on and transfer their existing knowledge from a different yet familiar domain. Although coding connects to numerous fields and domains, the goal is for educators to make those connections explicit and frequent to both help their own learning as well as counter the risk of teaching content that their students see as only usable or applicable in one particular context.

By combining expansive framing and the unplugged approach, we propose an instructional model for introductory CS education called Expansively-framed Unplugged (EfU) (Lee & Vincent, 2019). It begins with a tangible unplugged activity, moves to paper and pencil and then digital representations of the activity (using a block-based coding language like Scratch, e.g.), and culminates in using the digital model to produce a new version of the original unplugged activity. This model informed an instantiation of an instructional activity, called Looming Code (see Table 5.1).

In the next sections, we first describe the theoretical motivation for our approach, its instantiation in Looming Code, and our approach to PD. We then describe one school librarian's experiences as she participated in our PD structured around EfU. The PD consisted of her participating in the Looming Code activities first as a student to draw upon her crafting knowledge to better understand and feel more comfortable with the specific coding activities. The librarian then implemented Looming Code with two classes of elementary school students in her school library.

BACKGROUND

Our EfU model combines ideas from both unplugged computing and the situated model of transfer, called expansive framing. Unplugged CS is a practice for teaching CS concepts and principles without a computer (e.g., Bell et al., 2009). These kinds of unplugged activities have been demonstrated in several domains, including beading (Eisenberg, 2010) and playing tabletop board games (Berland & Lee, 2011).

One of the main ideas of unplugged computing is to teach learners CS skills and concepts outside of the realm of a specific programming language. Students who shy away from actual computer coding may enjoy using those same skills and strategies in a different context, and the same skills used in coding, such as debugging and composition, can be learned and practiced in other types of tasks and contexts. This approach is similar to other long standing cognitive approaches for supporting transfer of learning through the use of analogies (Gentner, 1998) or metaphors (Videla, 2017).

For those who do enjoy actual coding, unplugged activities give them a way to practice concepts and strategies outside of a single programming language and to recognize contexts outside of the computer in which those skills could be useful. Another argument in favor of unplugged computing is that it provides a more tangible way for learning abstract CS concepts (Eisenberg et al., 2009; Kafai & Vasudevan, 2015). Students who may have a hard time understanding an abstract sorting algorithm, for example, may benefit from hands-on activities in which they sort items with paper, cards, or other tangible materials.

Unplugged activities are also appealing to teachers because, as they do not require 1–1 computing, they usually have a lower cost. However, teachers who implement unplugged activities exclusively run the risk of the learners' knowledge remaining situated in the medium of a specific unplugged task. Transfer of knowledge is problematic for instructors of all subject areas, as the learning context (environment in which the knowledge is learned) and transfer context (environment in which the knowledge will be needed) are not always similar (Bransford et al., 2000). If learners successfully practice a CS concept, such as algorithm building in a board game, they still may not be able to transfer that practice to an actual coding interface.

To address this problem, we turn to the model of expansive framing that is designed to create a more favorable environment for knowledge transfer (Engle et al., 2012). This model has been used by others to design more broadly appealing computational thinking curriculum and assessments (Grover et al., 2014).

This model posits that by making several, frequent connections back and forth between the social context of learning and the social context of transfer, teachers can help learners create an encompassing context that aids in knowledge transfer (see Figure 5.1 adapted from Engle et al., 2012). Thus, if students are able to understand the larger context into which the learning context and the transfer context fall, it will be easier for them to retain and use their knowledge in that transfer context.

Engle et al. (2012) identify five specific types of ways in which expansive framing connections can be made to create and strengthen that encompassing context. The first two involve connecting settings. First, if students understand how the skills learned will be useful for them in a future setting,

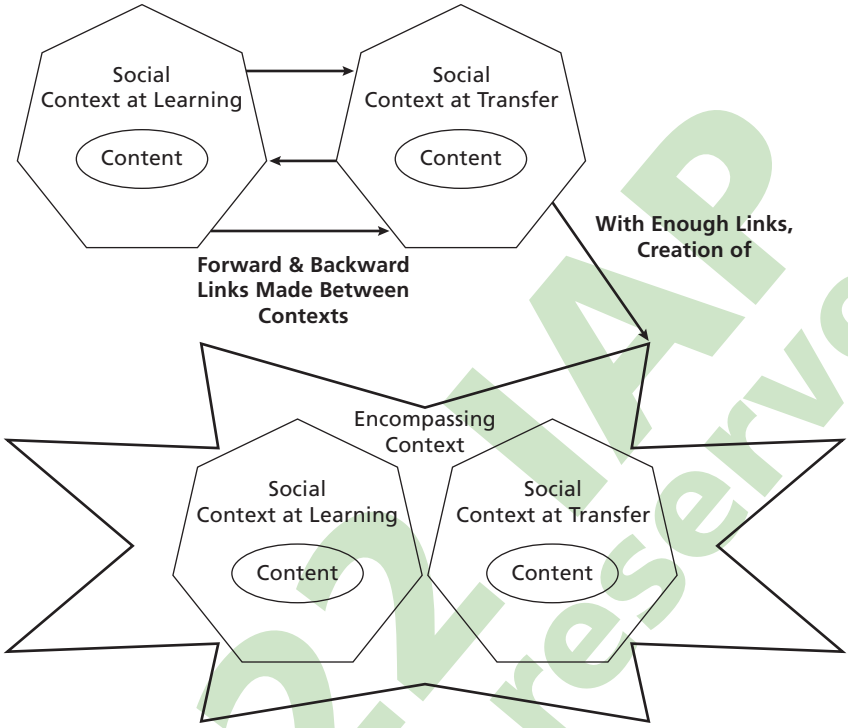

Figure 5.1 Expansive framing model. *Source:* Adapted from Engle et al., 2012.

they may learn and remember them better. Second is a focus on connecting settings in order to access prior knowledge. Besides encouraging students to look forward to future settings, this type of connection encourages students to look backwards and see how prior experience and learning may be applied to the current situation. This helps students link their current knowledge to prior knowledge, reducing the risk that current knowledge will become inert or obsolete (Bransford et al., 2000).

The next three connections involve authorship. The third type of connection suggested is that authorship leads to connecting prior knowledge in ways that support later transfer-out. This is similar to the second type in that teachers are encouraging students to use prior knowledge but emphasizes that authorship itself encourages students to do so. Engle et al. (2012) explain that students in these types of learning settings "sometimes brought in their own outside examples to form generalizations about the topics they were learning," and that "data showed that these students were also more likely to transfer certain facts, principles, and a learning strategy to a new context" (p. 224).

Fourth, authorship promotes accountability to particular content. If a student completes a creative project for some content, as opposed to passively learning about that same content, the very nature of the task encourages students to master the content in order to create. Finally, the fifth type of connection is that authorship as a practice promotes generation and adaptation of knowledge in transfer contexts. Instead of attempting to fill students' heads with knowledge, offering them a chance at authorship encourages them to be more than just consumers of knowledge and to strive to add something to the collective body of knowledge.

Eglash and Bennet (2009), for example, use the context of cornrows as a hairstyle and students' cultural capital in that area to stimulate connections to mathematics and coding. In their Cornrow Curves activity, students learn about the history of cornrows and use a computer program to create their own design. While many such connections between unplugged tasks and other non-digital contexts are desirable for transfer, we also want students to be able to use skills learned in unplugged tasks in an actual coding environment. In order to lessen the risk of students' knowledge remaining situated in unplugged tasks, our EfU model starts with unplugged activities and then encourages expansive framing to connect those tasks to actual coding tasks.

The EfU model consists of three domains and three movements across domains (see Figure 5.2). The three domains are physical artifact, paper and pencil, and digital. The first movement across domains is the movement from physical artifact to paper and pencil, where the learner creates a paper and pencil representation of the physical artifact. The second is from paper and pencil to digital, where learners use their paper and pencil representation to inform their code and create a digital model of the physical artifact. Last is the movement from digital back to physical, where learners use customized code to create their own physical artifact. Specific instantiations of the EfU model may contain steps where learners move deeper into a single domain, called submovements.

The three domains are informed by unplugged ideas. Learners and teachers who may be intimidated by starting in a digital format are eased into the process by starting with the physical artifact, then moving to paper, and finally coding digitally. Paper and pencil are also more accessible than the other two domains; it is often easier for a novice to fix a mistake on paper than it is to undo it in a physical product or in code.

The three movements are informed by the expansive framing model and serve to make connections back and forth between domains. Although students can learn to make physical artifacts or code independently of each other, the goal is for students to achieve transfer of knowledge so that their learning does not stay accessible only in the context in which it was gained. The five aspects of expansive framing mentioned earlier can be seen in different parts of the EfU model, as seen in Columns 1 and 5 of Table 5.1

in the following section. The movements from physical artifact to paper and pencil to digital offer connections of different settings for both future transfer and access to prior knowledge. Students who have done the particular unplugged activity or a similar one have prior knowledge they can draw on, and students who feel the knowledge presented might be useful in future contexts may achieve better transfer. The movement from digital back to physical artifact uses authorship of a new artifact to support later transfer out, promote accountability to particular content, and promote the generation and adaptation of knowledge.

Although the ideas behind expansive framing are focused on a situated view of student learning and transfer, they also apply to educators with little coding background. It is important to note that, unlike the other topics they may teach, they too are learners in this content area, and every aspect of EfU designed to help students learn may benefit them as well. Work by Putnam and Borko (2000) has shown that using a teacher-as-learners model for PD helps improve teacher learning. Thus, we propose that teachers prepare to teach and lead the activity by participating in it as their students will. Using unplugged ideas to inform the medium and tasks and using expansive framing to inform the interactions and conversations, will lead to an activity with a low threshold for both teachers and learners, and will help teachers with little coding background to successfully lead and support such activities.

TABLE 5.1 Levels in Looming Code and Connections to EfU

Looming Code Level & Description	CS Concepts and Skills	EfU Model Movement	EfU Model Sub-Movement	Expansive Framing Connection
1 color grid based on physical weaving	Decomposition	Physical artifact to paper-and-pencil		FT, PK
2 make box pattern	Abstraction Loops		Paper-and-pencil	FT, PK
3 basic code in Scratch	Algorithms Abstraction	Paper-and-pencil to digital		FT, PK
4 shorter code in Scratch	Efficiency		Digital	FT, PK
5 customize design in Scratch	Debugging, Iteration		Digital	APK, APA, APG
6 create weaving from new design	Enacting code	Digital to physical artifact		APK, APA, APG

Key: FT—connecting settings for future transfer; PK—connecting settings to access prior knowledge; APK—authorship leads to connecting of prior knowledge in ways that support later transfer-out; APA—authorship promotes accountability to particular content; APG—authorship to promote generation and adaptation of knowledge.

Having teachers approach the content from the learners' perspective not only helps them become familiar with the content, but also gives them opportunities to experience firsthand the areas where students will struggle, which will help them as they lead the program themselves. This teacher-as-learner PD model has been shown to help improve teacher learning as well as shift their beliefs about teaching and learning to one more grounded in student thinking (Putnam & Borko, 2000).

EXEMPLARS

Looming Code in the Context of EfU

Looming Code consists of several levels in which learners examine example weavings, color the design on a grid, create a readable pattern of the design, code that pattern in Scratch, design a custom weaving in Scratch, and then make the design with physical materials. Table 5.1 summarizes each level and links each level to relevant CS concepts. We developed a set of Level Cards, which break down the steps of each level and offer tips and reminders, along with the student worksheet and the Scratch program used.

Level 1

In Level 1, learners are given a weaving, such as the one in Figure 5.2, and asked to examine the pattern and transfer it to a paper grid. This helps learners practice decomposition, as they move from looking at the weaving as a whole to looking at the individual parts and how they fit together. The facilitator can ask learners to think about other areas of their lives where they have to take a task or problem and divide it into smaller parts, and connect this level to coding skills by talking about decomposition explicitly.

Level 2

Once they move to Level 2, the learners make a box pattern from their grid pattern, as shown in Figure 5.3. With a completed and colored grid of the design, learners can start decomposing it further. When weaving, designs are made by using different combinations of up and down for the vertical strings, called the warp. These strings are either up or down, forming a sort of binary system in which infinite designs can be created. Learners will start with the first line of the weaving and create a box pattern for that line. This is essentially a row of boxes, with each box shifted either up or down to represent the position of a particular string. Learners will create a box pattern row for each unique row of their pattern. After creating the individual rows, learners will write the sequence of rows, for example, 1 2 3 4 3 2 1. This creates a readable pattern from which the design could be

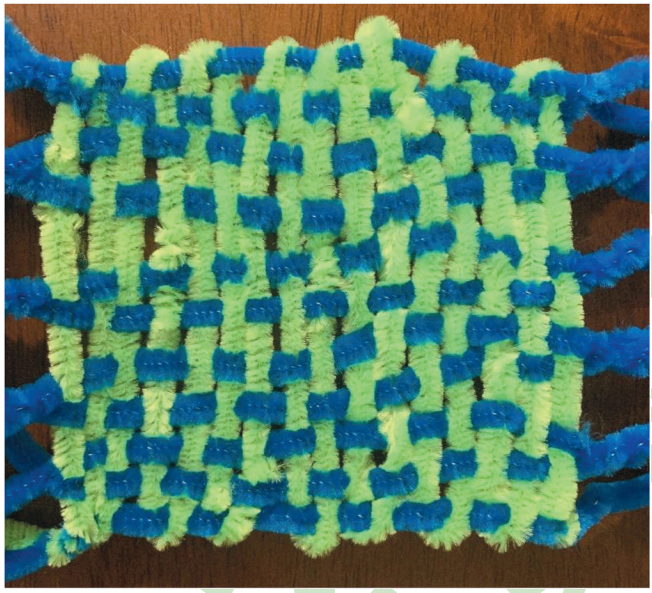

Figure 5.2 Example weaving given to students to decode.

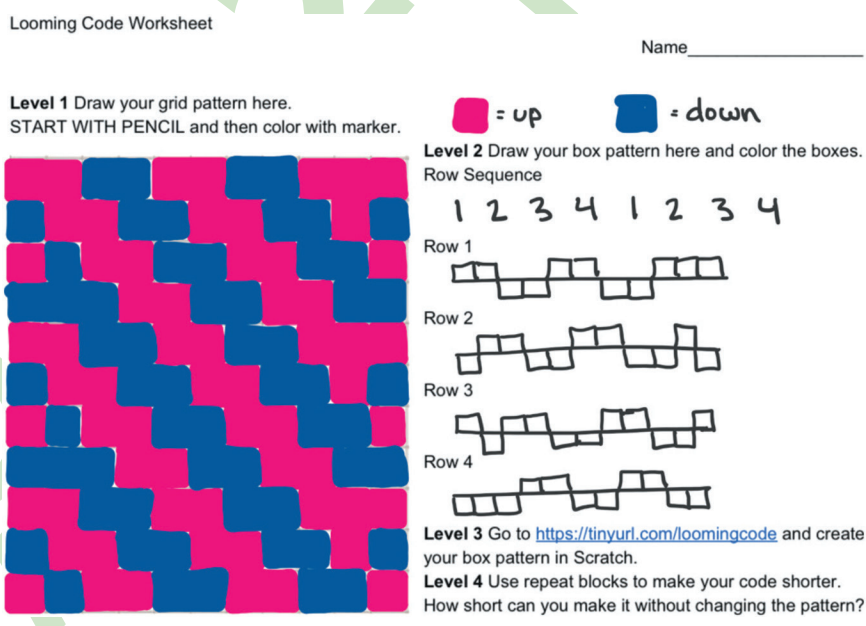

Figure 5.3 Example looming code student worksheet, levels 1 and 2.

recreated. This encourages learners to look for repeating patterns in and within rows. Some weaving designs will have a 1 2 1 2 1 2 sequence, while others include 1 2 3 4 1 2 3 4 and 1 2 3 4 3 2 1. In the classroom or library, the teacher can talk with students about abstraction as well as repeat loops.

The facilitator can also start a conversation about notation. The librarian who participated in our PD sessions stated that her students had just finished a unit in poetry, where they learned to notate poem lines by their ending rhymes, using letters to represent unique rows. Like the weaving designs, poems can have many different rhyming sequences, such as A B A B, A B A C, and others. Thus, the instructor can explain to students that it does not really matter whether they use numbers or letters or even shapes to create written notations of patterns, but it is important that they are consistent with their notation and that they make the notation clear to others who might want to understand or recreate the design and pattern.

At this point, learners can also discuss how the grid pattern from Level 1 and the box pattern from Level 2 are similar and different, and whether both are useful or necessary. Both patterns contain an accurate and complete representation of the weaving design. They both show which colored strings are visible in different parts of the weaving, but the grid pattern shows a complete visual depiction of the design as completed, whereas the box pattern is a deconstructed version. The grid pattern could be used when creating a new design to visualize the result, or to show someone else what a particular design will look like. The box pattern is more for someone who is making the design. When weaving, using a complete visual of the design requires weavers to keep track of the row they are on, as well as count boxes in a small colored grid, which could easily lead to errors. Using a more deconstructed representation, such as the box pattern allows them to see one row at a time as well as how to move their warp strings to make each row.

Level 3

In Level 3, learners are coding their pattern in Scratch. Using their box pattern and a skeleton program in Scratch, learners code each unique row with Up or Down blocks that map to the position of the boxes in their box pattern rows. Once the learners code their rows, they then move to code the sequence of rows for their design. Learners have this already written in their box pattern, so they simply need to transfer that to a row of code with the same numbers. Once they are finished coding, learners can run their code to create a visual representation of the design. The skeleton program has code in the background that creates a colored grid based on the sequence of Up and Down blocks in each row and the sequence of rows. Learners can change the colors of the boxes to match the design they are using. Ideally, the colored grid in Scratch should match their grid pattern on paper exactly. Of course, errors can be made in the process of

transferring the design to a paper grid, then to a box pattern, and then to Scratch, but the visual model in Scratch allows students to compare their digital and physical designs to check for accuracy. If errors appear, the facilitator can talk to the learners about debugging, and how important it is to be able to identify whether or not there is an error, how to find it, and how to fix it. The facilitator can help learners debug or ask individual learners to help each other.

Level 4

Once learners debug their patterns and code and the visual model presented in Scratch is acceptable, they start looking for repeating patterns inside their code of the individual rows and sequence of rows. This constitutes Level 4. At this point, the facilitator can ask the learners to make their code as efficient as possible without changing the design. Code such as "Up Up Up" can be rewritten with a repeat block, using only two blocks instead of three. The result of this level depends on both the pattern a learner is using and how they decide to use the repeat blocks in their code.

Level 5

After the learners work through the first few levels with the given weaving design, they have the opportunity to create their own design in Level 5. Learners can modify their existing code or wipe everything and start afresh. If learners are having a hard time thinking of what to do for a new design, they can always keep the rows they have already coded and simply rearrange them into a new design.

Level 6

Once learners finish creating their design in Scratch, they begin Level 6 and read the pattern in their code to create their design with pipe cleaners. Instead of using real looms and yarn to weave, we use pipe cleaners and large combs. Real looms are either expensive to purchase for a whole class or difficult for younger children to operate, and we discovered that wide-toothed combs hold the pipe cleaners just enough to give students a sturdy base on which to start their weaving. Using pipe cleaners allows students more room for creativity, as it is cheaper to provide a wide variety of colors in pipe cleaners than it is in yarn. Students can also change the colors in the weft or the warp very easily with the pipe cleaners. Weaving with pipe cleaners is less time consuming than weaving with yarn, since it is not necessary to have a very tight weave and frustrating tangles are much less likely. At the end of Level 6, students complete the Looming Code sequence. If time permits, the facilitator can provide students an opportunity to try others' designs and create patterns that will stay in the library (or classroom) for future use.

As students move through the levels of the Looming Code activity, they also progress through the domains of the EfU model and make more and more connections between contexts. Table 5.1 lists each level of Looming Code with its associated movement or submovement. Table 5.1 shows how Looming Code involves the five types of connections suggested by the expansive framing model to promote transfer. The first four levels (Levels 1–4) focus on connecting settings for both access to prior knowledge and future transfer. The type of tasks students will be doing, such as taking apart a bigger problem into smaller pieces and writing and reading step-by-step instructions, can be applied to many situations and may be tailored to student interest and experience at the discretion of the teacher.

The last two Levels of Looming Code (Levels 5 & 6) focus on authorship. According to the expansive framing model, allowing students to use their creativity to take what they learned in the first half and create a new design will draw on prior knowledge, promote accountability to what they are learning, and give them a sense of having generated and adapted knowledge rather than passively consuming information.

As mentioned earlier, the three domains of the EfU model are informed by unplugged ideas, and the three movements are informed by expansive framing (see Figure 5.4). Some students and teachers have prior experience with Scratch or other block-based coding platforms, but many have none. If learners not familiar with Scratch start on the computer right away, trying to code a pattern as they decipher it, they would be learning new content and a new interface simultaneously. In order to reduce this cognitive

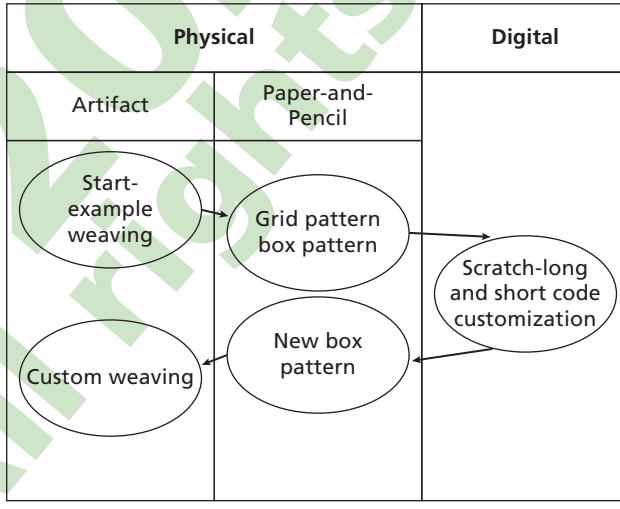

Figure 5.4 Domains and movements within EfU model with associated looming code levels.

load, learners use both physical artifacts and paper and pencil work to become familiar with the content—the weaving design—before being introduced to a new interface—Scratch. By the time learners finish their paper and pencil representations of the pattern, they are simply re-instantiating the pattern in Scratch.

Professional Development: Teacher as Learner

In addition to developing Looming Code as an instantiation of the EfU model to help both student and teacher learning, we also designed accompanying PD, using a teacher-as-learner approach. Instead of simply presenting the program materials to the librarians and talking through the program, we set up a space with all the materials needed for librarians to participate in Looming Code just like students.

This served three purposes. First, the librarian, who would later be the instructor for the program, was able to learn new and unfamiliar content she would later be required to teach. This is unlike most PD approaches where teachers are assumed to have the necessary background knowledge. Second, she had the opportunity to sit in the students' place and see the program from the learner point of view. This helps her recognize places where students will struggle and prepare for questions and scaffolding opportunities. Finally, she gets to see the instruction modeled as we teach the program. All three purposes help prepare the librarian to successfully implement the program with students and create successful learning environments.

METHODS/DATA COLLECTION

As part of a larger design-based research study in the Rocky Mountain region of the United States, one public school media and technology teacher, Rachel (a pseudonym), participated. Rachel had her elementary education teaching license, a master's degree in gifted education, and an administrative endorsement. This was Rachel's first year as the media and technology teacher, but she had several years of teaching experience. At the time of data collection, she was teaching Grade K–5 media and technology courses in the library, working with students for 90 minutes each week. Rachel was responsible for teaching both the library media and computer classes for all students in the school.

Rachel and her fifth grade students were participants in early iterations of our PD and Looming Code program. Her participation and feedback during PD sessions helped to inform our subsequent iterations and implementations of the program. Prior to participation in our Looming Code

program, Rachel had been teaching her fifth grade students Scratch so they were familiar with the basics of the Scratch interface prior to beginning this project. In addition, Rachel had pursued looming as a personal hobby and was familiar with weaving techniques and patterns.

Rachel attended two PD sessions, taught by our team, each lasting around 1 hour, in which she participated in the program as a learner, with the understanding that she, in turn, would implement the program with two fifth grade classes in the library during their media and technology class time. In addition to the PD sessions, she was provided with the necessary program materials and supplies.

Rachel implemented the Looming Code activity with two of her classes, and each class spent two class periods on the activity. Each class period was ninety minutes, however both classes experienced some interruptions that prevented them from using the entirety of their time on the activity. There were 30 students total between the two classes who participated in the activity. We observed the activity and only directly interacted with them when Rachel asked us for clarification on something she was doing. The data described in the next section was collected in the PD sessions and classroom implementations in the form of observational field notes and photographs.

RESULTS AND DISCUSSION

During our first PD session with Rachel, we gave her a copy of the level cards, which included the instructions for each level and the worksheet and had her participate in the program as a student. We asked questions and pointed out things to watch for as she worked. After completing the first two levels, Rachel suggested a small change to the format of the worksheet and had several ideas on how she would teach this portion of the activity to her students. While working in Scratch, she deleted some of the background code needed to create the visual, but then realized her mistake. She was able to successfully transfer her box pattern to Scratch to produce a digital representation of the pattern. Then she customized the code to create her own pattern and began weaving it on a loom. Due to time constraints, she was not able to finish her weaving. During the second session, we gave her all the materials and resources she would need to teach the program and reviewed the program flow. She felt comfortable with the material and was excited to start the program with two of her fifth grade classes.

Throughout Rachel's implementation of the Looming Code program in her library and computer lab, she engaged in various activities we considered to be good teaching practice, or habits developed by teachers with her level of experience. She managed the behavior and focus of her classes well, asked probing questions to check for understanding, and was able to

provide support and scaffolding for individual students or the whole class when help was needed. Besides Rachel's scattered comments on how she would teach certain sections of the program, none of these pedagogical behaviors were discussed in the PD sessions. We found that once Rachel was comfortable with the content of the program, she used knowledge of teaching and facilitating to adjust and augment the program to fit the needs of her students and space.

To begin the program, Rachel gave the students an overview of all the levels, and then started on a three-column chart, labeled, "Something I wonder about," "Something I learned," and "How I felt about the activity." She had the students share thoughts for the first column. While sharing thoughts about the "Something I wonder about" section of the chart, one student asked why they were doing this activity. Rachel responded by saying, "That's a good question. What does life have to do with code?" She let the students think about that for a moment, and then asked if there are no rules that we have to follow in life. One student pointed out that we should follow the rules but do not have to, and Rachel said that just like life has certain rules, so does code. She then asked, "What does code have to do with life?" The students thought, and she asked them if they have ever played a computer or video game and said that is a good example of code. She also pointed out that the fact that their parents' eye and hair color determine their eye and hair color is actually a form of code, genetic code. The students thought that was neat. One student shared that he wondered how they will be able to fix their work if they mess up, Rachel asked, "Even if it doesn't turn out exactly the way we wanted, is that really a failure?" and pointed out that some of the best learning experiences come from failures.

During levels one and two, Rachel passed two example weavings around the class for the students to hold and examine. When they were coloring their grids, she passed around boxes of crayons that included only the colors of crayons they needed to match the example weavings. Before moving to Level 3, Rachel moved all the students to the computer lab. She pulled up the Scratch project on the projector and showed the students the code (see Figure 5.5).

She pointed out that all the blocks currently on the screen need to stay where they are and they should not delete any of them, something she had done the first time she used the project. Rachel also modeled for the students how to code their rows and run their program, and asked questions to check their understanding. While modeling code, she purposely coded something incorrectly, ran the program, and asked her students why it did not work. The students were able to spot the bug in the code and fix it with Rachel's help.

As the students worked on their code, Rachel walked around, monitoring their behavior and looking for opportunities to provide support to

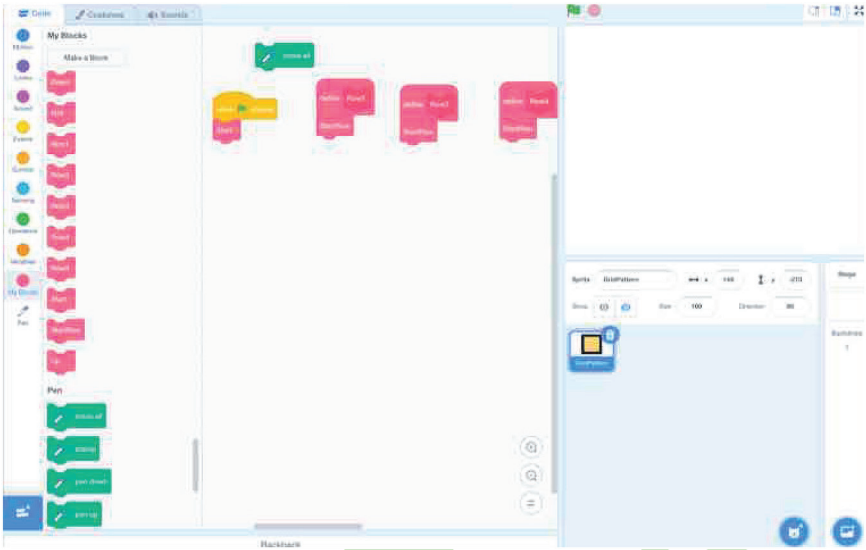

Figure 5.5 Skeleton code students used to build their patterns in scratch.

struggling students. While walking around, she was able to clarify misconceptions the students had, such as one student who believed the pattern could not be bigger than a certain length. Near the end of class, students chatted about the possibilities of coding in Scratch. Rachel mentioned that they could do some geometry work in Scratch, since they were learning about geometry in another class, and asked the students if they thought they could code a rhombus or other geometrical shape.

As the class progressed to later levels of the program, Rachel used printouts of the level cards taped to the whiteboard to help students see what they had done so far and what they would be doing next. She modeled how to set up the comb with the pipe cleaners and how to start and complete the weaving. For the rest of the time, there were students at different stages of progress. Some were coding the example pattern, some were coding their own pattern, and some were weaving their own pattern. As Rachel circled the room and talked to individual students, she was able to answer their questions no matter what step they were on. Near the end of class, she noticed an incorrect practice that had started spreading across the room and called everyone's attention to the front so that she could point it out and correct it. After explaining the correct method, she helped students who had made that error to fix their weaving.

After completing the program, we conducted a 20 minute, recorded, semi-structured interview with Rachel to find out how she felt the program had gone. We asked ten questions regarding the implementation of the

program and use of the resources and materials. She said that the biggest takeaway for the students had been that they could use Scratch to make something real, and that the biggest takeaway for herself was to make sure she could do each part of the program herself before she taught it. She had shared with us when we first started working with her that she was familiar with weaving, but during the interview explained that she had never followed any specific patterns and had done what she called free looming. She told us that the Looming Code program really got her thinking about the intentionality of executing a specific pattern in a design.

Rachel also shared with us that she appreciated having both the learner's and the teacher's perspective of the program, and that it helped her see where her students would need help. She knew how she would further tweak and improve the activity if she did it again, and said that overall, she really enjoyed it. Seeing the kids sitting on the floor weaving and chatting makes her think of a quilting bee (a social activity where people gather to make quilts).

Overall, our observations show that the EfU model, Looming Code program, and teacher as learner PD approach helped Rachel gain enough content knowledge in a new activity to be able to adapt it to her particular students and use her good teaching practices to create a fun, successful learning environment.

NEXT STEPS

This chapter described a conceptual model, instructional activity, and PD approach designed to prepare school librarians or other educators with limited coding background to learn to lead coding activities and lessons in their libraries. The conceptual model, EfU, is based on a situated account of transfer, in which backward and forward connections between known unplugged contexts and new coding contexts are made salient and authorship is emphasized. EfU informed the design on an instructional activity, Looming Code, which links weaving, as its familiar and unplugged context, to the new context, coding patterns in Scratch, and back to weaving again. Similarly, the PD design is influenced by EfU and the teacher as learner approach, in that librarians first participate in Looming Code as students, and draw upon their existing knowledge of crafting to help understand and feel comfortable with the subsequent coding activities.

We note that while the example described in this chapter uses crafting, specifically weaving, as its unplugged activity, the EfU model is not limited in its scope of possible unplugged activities. In current work, we are using EfU to inform an instructional unit that uses computationally-rich board games as the familiar context. Students play these games, learn

computational thinking concepts, and then create their own board games in a digital coding environment (Lee et al., 2020). In using the EfU conceptual approach, a key goal is to identify familiar contexts for students as well as teachers that serve as useful unplugged "funds of knowledge" (Moll & Greenberg, 1990) to support learning in the digital environment. A key assumption is that a well-chosen unplugged context may offer a means to broadening participation in coding to students who may otherwise lack interest. In order to avoid replicating patterns of participation in CS that exclude certain groups of students, we seek contexts (e.g., weaving, board games) that are both rich in computation and offer many trajectories for more inclusive participation. We believe that many such fruitful contexts still remain to be explored.

Future work should investigate to what extent this provides enough of a bridge for teachers who are new to coding to support and scaffold coding activities in their instructional context. This model is not proposed as a way to create content experts in CS but rather to enable non-experts to feel comfortable in hosting activities in which the students become engaged and interested in coding. While this instantiation was implemented with one educator, the model as well as the Looming Code program are certainly scalable. Similarly, future work should examine to what extent this approach moves beyond simply sparking initial interest in students and encourages a growing level of involvement and persistence in CS.

ACKNOWLEDGMENTS

This work was supported by the Institute of Museum and Library Services grant number RE-31-16-0013-16. We thank our partnering librarians, their students, and the school district.

Resources for the Looming Code activity can be found at https://slli.usu.edu/looming-code/

REFERENCES

Bell, T., Alexander, J., Freeman, I., & Grimley, M. (2009). Computer science unplugged: School students doing real computing without computers. *The New Zealand Journal of Applied Computing and Information Technology, 13*(1), 20–29.

Berland, M., & Lee, V. R. (2011). Collaborative strategic board games as a site for distributed computational thinking. *International Journal of Game-Based Learning, 1*(2), 65–81.

Bransford, J. D., Brown, A. L., & Cocking, R. R. (2000). *How people learn* (Vol. 11). National Academy Press.

Eglash, R., & Bennett, A. (2009). Teaching with hidden capital: Agency in children's computational explorations of cornrow hairstyles. *Children, Youth and Environments, 19*(1), 58–73.

Eisenberg, M. (2010). Bead games, or, getting started in computational thinking without a computer. *International Journal of Computers for Mathematical Learning, 15*(2), 161–166.

Eisenberg, M., Elumeze, N., MacFerrin, M., & Buechley, L. (2009, June). Children's programming, reconsidered: Settings, stuff, and surfaces [Conference session]. In P. Paolini (Chair), *The 8th International Conference on Interaction Design and Children* (pp. 1–8). ACM.

Engle, R. A., Lam, D. P., Meyer, X. S., & Nix, S. E. (2012). How does expansive framing promote transfer? Several proposed explanations and a research agenda for investigating them. *Educational Psychologist, 47*(3), 215–231.

Gentner, D. (1998). Analogies. In W. Bechtel, G. Graham, & D. A. Balota (Eds.), *A companion to cognitive science* (pp. 107–113). Blackwell.

Grover, S., Pea, R. D., & Cooper, S. (2014, June 23–27). Expansive framing and preparation for future learning in middle-school computer science [Conference session]. In J. L. Polman, E. A. Kyza, D. K. O'Neill, I. Tabak, W. R. Penuel, A. S. Jurow, K. O'Connor, T. Lee, & Laura D'Amico (Eds.), *11th International Conference of the Learning Sciences* (pp. 992–996). International Society of the Learning Sciences.

Kafai, Y., & Vasudevan, V. (2015, June 21–24). Hi-Lo tech games: Crafting, coding and collaboration of augmented board games by high school youth [Conference session]. In M. Umaschi Bers & G. Revelle (Co-Chairs), *The 14th International Conference on Interaction Design and Children* (pp. 130–139). ACM.

Lee, V. R., & Vincent, H. (2019, March). An expansively-framed unplugged weaving sequence to bear computation fruit of the loom [Conference session]. In P. Blikstein & N. Holbert (Co-Charis), *FabLearn Conference 2019* (pp. 124–127). Association for Computing Machinery.

Lee, V. R., Poole, F., Clarke-Midura, J., Recker, M., & Rasmussen, M. (2020). Introducing coding through tabletop board games and their digital instantiations across elementary classrooms and school libraries [Conference session]. In J. Zhang & M. Sherriff (Co-Chairs), *ACM Technical Symposium on Computer Science Education* (pp. 787–793). ACM. https://doi.org/10.1145/3328778.3366917

Moll, L. C., & Greenberg, J. B. (1990). Creating zones of possibilities: Combining social contexts for instruction. In L. Moll (Eds.), *Vygotsky and education: Instructional implications and applications of sociohistorical psychology* (pp. 319–348). Cambridge University Press.

Martin, C. (2017). Libraries as facilitators of coding for all. *Knowledge Quest, 45*(3), 46–53.

Putnam, R. T., & Borko, H. (2000). What do new views of knowledge and thinking have to say about research on teacher learning? *Educational Researcher, 29*(1), 4–15.

Videla, A. (2017). Metaphors we compute by. *Communications of the ACM, 60*(10), 42–45.