Chapter 3
Teaching Coding in Kindergarten: Supporting Students' Activity with Robot Coding Toys

Jessica F. Shumway, Jody Clarke-Midura, Victor R. Lee, Deborah Silvis, Lise E. Welch Bond, and Joseph S. Kozlowski

3.1 Introduction

Teaching coding and computational thinking (CT) in schools is garnering attention in the United States, including in the early childhood grades (Kindergarten to Grade 2). Schooling as situated in the current digital and information technology society is increasingly requiring the teaching of CT, which is often considered a new kind of literacy (Bers et al., 2019; Wing, 2006). It has been argued that CT is one of the most important skills for students to learn (Kafai & Burke, 2014), and programming is an important context for enabling students to develop CT (Pea & Kurland, 1984; Shute et al., 2017). As a result, CT and coding instruction are becoming more common in K-12 classrooms. An important distinction, however, is that coding in early childhood classrooms looks much different than coding in secondary settings (Lin & Weintrop, 2021). This has led to the production and marketing of various block-based activities and robot toys for young children to use in service of learning coding and CT (Bakala et al., 2021; Hamilton et al., 2020; Papadakis, 2021).

J. F. Shumway (✉) · L. E. Welch Bond
School of Teacher Education & Leadership, Utah State University, Logan, UT, USA
e-mail: jessica.shumway@usu.edu

J. Clarke-Midura · D. Silvis
Department of Instructional Technology & Learning Sciences, Utah State University, Logan, UT, USA
e-mail: jody.clarke@usu.edu; deborah.silvis@usu.edu

V. R. Lee
College of Education, Stanford University, Stanford, CA, USA
e-mail: vrlee@stanford.edu

J. S. Kozlowski
Edith Bowen Laboratory School, Utah State University, Logan, UT, USA
e-mail: joseph.kozlowski@usu.edu

T. Keane, A. E. Fluck (eds.), *Teaching Coding in K-12 Schools*,
https://doi.org/10.1007/978-3-031-21970-2_3

One line of research on teaching early childhood CT and coding is with tangible coding toys (e.g. Angeli & Valanides, 2020; Bers et al., 2019; Muñoz-Repiso & Caballero-González, 2019). For example, Bers et al. (2019) developed the KIBO coding toy with a supplementary preschool curriculum (Bers et al., 2019). Results from their study of 3- to 5-year-old students' learning with KIBO indicated that these young children engaged in CT activities and developed coding skills of sequencing, repeats, conditionals and debugging through a robotics activity. Angeli and Valanides (2020) studied 5- and 6-year-old students' learning with the programmable Bee-Bot robot toy and found positive CT learning effects. Children were able to decompose tasks into subtasks in order to cope with the complexity of debugging a program. Muñoz-Repiso and Caballero-González (2019) conducted an investigation into 3- to 6-year-old students' CT learning after participating in seven sessions with the Bee-Bot robot. Students in the experimental group performed significantly better on the three dimensions of CT competence compared to the control group students (i.e. algorithms and sequences, action-instruction correspondence and debugging). Overall, while this emerging line of research provides contemporary evidence that young children can engage in CT and coding, there continues to be a need for research about how to best support CT learning in early childhood. The question remains: How should we teach coding to young children? Our design-based research study contributes to establishing evidence-based design elements for coding instruction in early childhood grades.

In this chapter, we share observations from a multiyear design-based research project exploring how to teach developmentally appropriate coding concepts and skills in kindergarten. We focus on coding toys that fit within a genre we call "grid-agent" robot coding toys. These are robots that are specifically for early childhood, commercially available, screen-free, tangible, moveable and programmable. Grid-agent robot toys invite children to explore mathematics through precise movements across a grid space. The movement options start with four simple codes: forward, backward, rotate left and rotate right (Clarke-Midura et al., 2019; Hamilton et al., 2020). These commands are represented in the visual block-based programming language of arrows (see Table 3.1, Code blocks). In addition to CT concepts such as algorithmic thinking, debugging and decomposition, grid-agent toys provide opportunities for students to use spatial knowledge and reasoning, early measurement concepts (e.g. dynamic units of linear movements), counting and sequencing to solve coding problems (Clarke-Midura et al., 2021b; Kozlowski, 2022; Shumway et al., 2021; Welch et al., 2022). Table 3.1 shows the coding robot toys used in our design-based research: Bee-Bot, Botley and Cubetto.

As a research team, we have explored young students' grid-agent coding as a collaborative small-group student activity through the lens of Engeström's (1987) depiction of cultural-historical activity theory (CHAT; Sannino & Engeström, 2018). CHAT is a useful framework for understanding and describing the complex and dynamic activity system that occurs in classroom-based design studies and in particular with children's activity with mediating artifacts (i.e. robot coding toys as a tool for learning). From a CHAT perspective (see Fig. 3.1), learning is in the actions and activities in which the subjects (students) engage in pursuit of

Table 3.1 Screen-free, tangible robot coding toys from the grid-agent paradigm

	Bee-Bot	Botley	Cubetto
Robot			
Manufacturer	Terrapin	Learning resources	Primo toys
Movements	Forward Rotate right 90° Rotate left 90° Back	Forward Rotate right 90° Rotate left 90° Back	Forward Rotate right 90° Rotate left 90° Back
Code blocks (codes that correspond to the movement of the robot)			

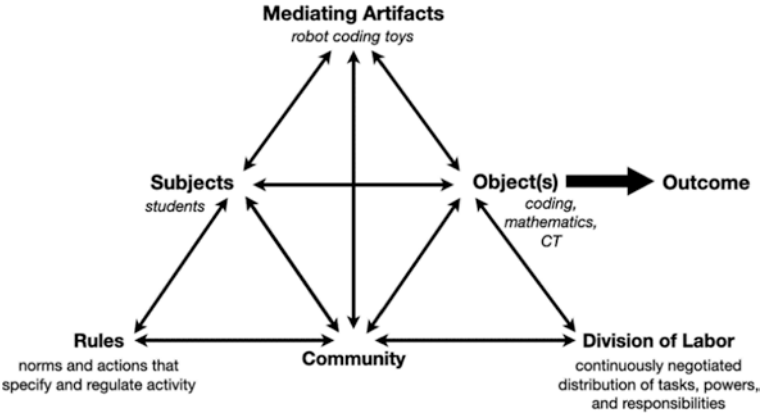

Fig. 3.1 Adapted from Engeström's (1987) image of an activity system

meaningful objects (coding, mathematics and computational thinking) with mediating artifacts (robot coding toys). This activity occurs within a community that operates with rules and division of labour, for instance, collaborative programming and negotiation of which codes to use in the context of the rules for what the codes instruct the robot toys to do. Hence, learning to code is planned and analysed as an

activity system and is situated (in coding toy tasks), distributed (among the small group of 4–5 students) and mediated with artifacts (robot coding toys).

We conducted classroom design studies (Cobb et al., 2017) that were guided by this question: *What design elements for grid-agent CT tasks and instruction are important for supporting kindergarten students' coding activity?* To answer this, we developed coding tasks for small groups of children to complete as part of teacher-facilitated classroom station rotations. We produced "design memos" to document design decisions, iterative revisions to tasks and instruction and students' responses to specific design elements of a task. In total, 48 kindergarten students across three public school sites in the western United States participated in the study. Thirty-two students participated in six lessons, and 16 students only participated in two lessons because of early termination of data collection due to the COVID-19 pandemic. Data included approximately 28 h of video of the lesson implementations and 56 design memos that tracked lesson design elements, intended learning outcomes and in-the-moment researcher observations and reflections.

In this chapter, we present the necessary design elements for coding toy tasks and supportive teacher-led instruction that emerged from iterative review and revision of our videorecorded kindergarten group activities. Before presenting these key findings, we provide a vignette that will serve as an anchor for understanding the findings.

3.2 Local Vignette of Coding in Kindergarten

3.2.1 Tasks for Introducing a Robot Coding Toy: Learning Codes and Sequencing Codes

We provide a description of a first session with 5- and 6-year-old children gathered on the rug for their technology centre in their kindergarten classroom. The teacher-researcher leading this activity introduced the Botley coding toy by asking the children what they noticed about the robot. The children—pseudonyms Larissa, Hyrum, James and Simon—discussed Botley's structure (plastic eyes, plastic wheels, blue box-like body) and remote control (arrows, trash can symbol, green `GO` button; see Table 3.1). The teacher then explained that they could tell Botley how to move by putting instructions in the remote using the arrow buttons, called codes. This process of focusing on what the children notice and introducing specific arrows is typical for our first activities.

The teacher and children then spent time programming forward and backward linear movements and observing Botley's movements across the grid-structured mat. The teacher used the grid mat to emphasize that one forward or backward arrow instructs Botley to move only one grid space.

Kindergarteners frequently struggle with turns representing rotation on a point. That was the case in the episode below when the teacher asked the children what the

turn arrow button did. James guessed that the rotation arrow on the remote meant "sideways". The teacher then offered an alternative and demonstrated the rotation: "Do you know what this actually does, *((pointing to the rotation arrow on the remote))*, instead of sideways like this *((shifting robot one square to its right while facing the same direction))* it's going to tell Botley to rotate *((rotating Botley 90-degrees right))*. Everyone say 'rotate'". In this episode, one of the students was invited to push the `ROTATE RIGHT` button and the `GO` button on the remote while all students observed Botley's 90-degree right rotation. As this transpired, Larissa used gestures (curved wrist to rotate hand) to explain the movement for the `ROTATE RIGHT` command (see Fig. 3.2, scene #1).

Simon then asked if they could program Botley to get to the adjacent green square, which required the program `ROTATE RIGHT + FORWARD`. The students negotiated which codes Simon should enter using gestures in the air and on the grid mat in front of them (see Fig. 3.2, scene #2). After running the program (`ROTATE RIGHT + FORWARD`), the teacher asked them to reflect on their program by

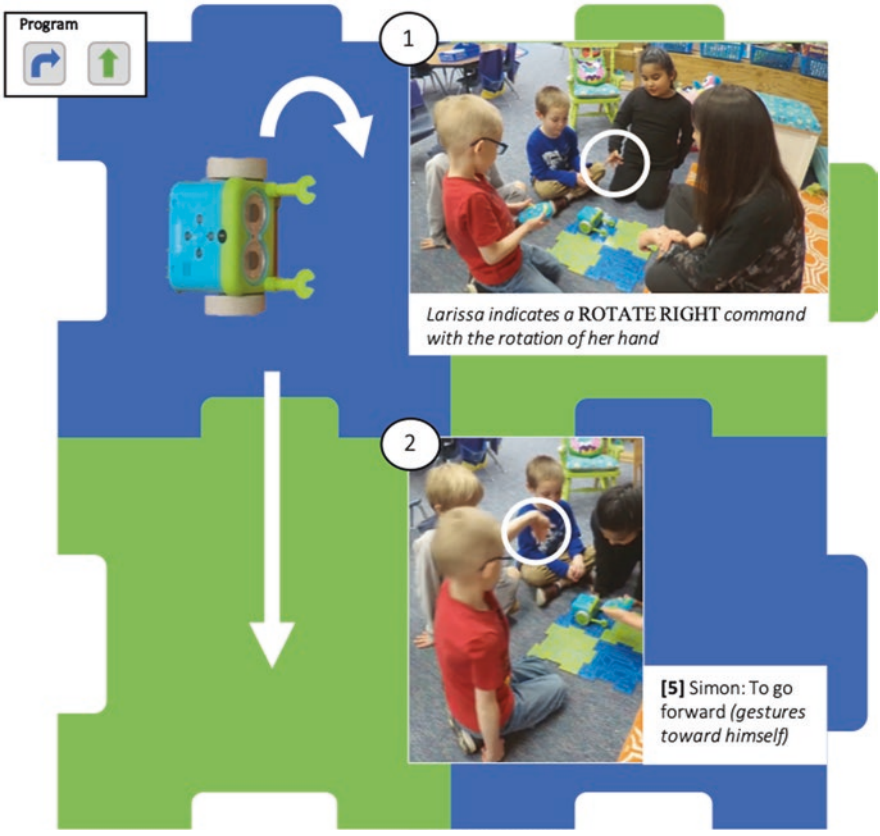

Fig. 3.2 Larissa and Simon used gestures to explain the robot's movements

describing what the robot did and what actions with the remote control (codes) led to those behaviours.

The children and teacher continued to play, programming Botley to move to other locations on the grid. The teacher encouraged students to physically move Botley and simulate its movements (i.e. move the robot with their hands) to show what they thought specific codes and sequences of codes would make Botley do *before* programming the sequence of codes in the remote and observing Botley's movements.

In summary, introductory tasks usually began with inviting children to notice a robot's controls, then asking children to observe how arrow codes corresponded to precise robot movements, next dedicating more time to turns as rotations and finally inviting children to explain what would happen when codes were run by showing their predictions through gestures or physically moving the robots.

3.2.2 Debugging a Buggy Program: What Happened?

The next Botley task in this introductory lesson is called *What Happened?* and is an activity that emphasizes the CT practice of debugging, or resolving coding errors. In this task, students were told that Botley needed to move the ball to the orange circle (see Fig. 3.3) but were given a buggy program and then asked to fix the codes. The teacher showed students an incorrect program (`ROTATE LEFT + FORWARD + FORWARD`) on the program organizer (see Fig. 3.3) and challenged them to identify and fix the bug (i.e. the bug was the direction of the rotation; correct program: `ROTATE RIGHT + FORWARD + FORWARD`). The program organizer served as an external representation of codes, since Botley does not display which codes were selected in the same way that other robot coding toys do. For this task, a repurposed baking sheet and magnets with arrows on them were used as the program organizer.

The following conversation occurred:

- Teacher: Here's what happened, there's a bug in the program. The program said, 'rotate left, forward, forward' ((*pointing to arrows on the program organizer as she names them*)). But it didn't work quite right.
- Larissa: It goes… ((*pointing toward Botley*)) It goes that way ((*rotating wrist to show right rotation*)) and then goes forwards ((*pointing in the direction Botley would move forward to get to the circle*)).
- Teacher: Ah! So, you think the program needs to go that way *((pointing to Botley's right))* and then two forwards? Hmm…What do you all think?

Before trying Larissa's idea, the teacher suggested they observe the buggy program. Hyrum entered the codes in the remote while the other students called them out in unison, "rotate left, forward, forward" and the teacher pointed to the corresponding codes on the program organizer. Hyrum ran the program and Botley moved off the grid toward Larissa (see Fig. 3.3, scene #3). The children giggled and Larissa said, "I knew it!" The teacher then named the problem as a "bug":

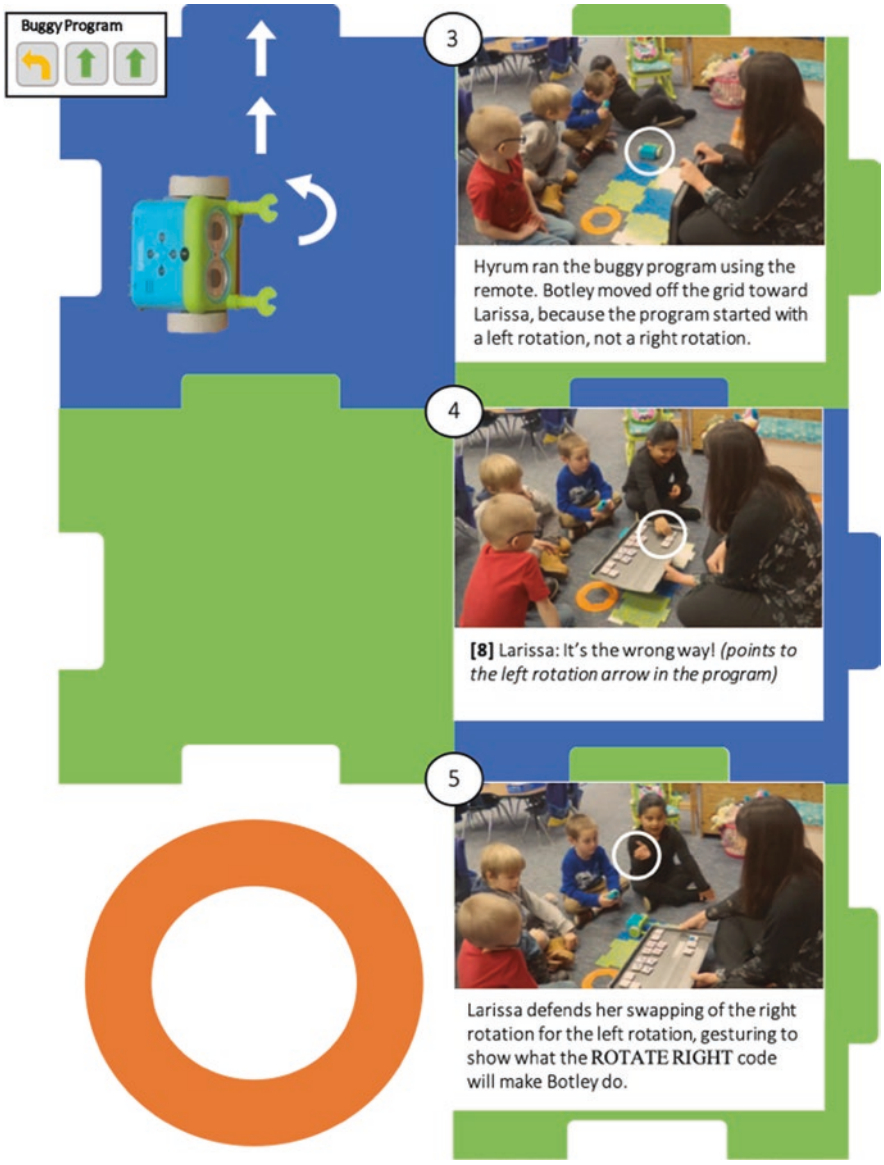

Fig. 3.3 Students' gestures for debugging the coding error in *What Happened?*

- Teacher: There's a bug in the program. This is what programmers do sometimes. They have to figure out where the bug is to fix it…Show me in this program ((*pointing to the program on the program organizer*)) where is the bug in the program ((*holding out plastic toy bug*)).
- Larissa: ((*Places the toy bug on the* `ROTATE LEFT` *code on program organizer.*))

- Teacher: Say, it's a bug!
- Students: It's a bug!
- Larissa: It's the wrong way! (see Fig. 3.3, scene #4).
- Simon: ((*reaches for a* `ROTATE RIGHT` *code*)).
- Hyrum: ((*points at the* `ROTATE RIGHT` *code*)).

After some discussion, the teacher asked Larissa to "switch it out", and Larissa switched the `ROTATE LEFT` code with a `ROTATE RIGHT` code, successfully debugging the program with a *swapping code* strategy. She then defended her debugging strategy with a rotation gesture with her hand (see Fig. 3.3, scene #5).

To summarize, this portion of the vignette shows how learning tasks can involve operating the coding toy so that the robot produces "incorrect" behaviours, using an external representation of codes (the program organizer) with the students and explicitly naming some of their CT activities (such as debugging) as they emerge. In the next section, we revisit portions of this vignette for illustrative purposes.

3.3 Key Findings

Drawing on the CHAT perspective, we analysed various factors in the activity system (i.e. mediating artifact, object, subject, community, rules, division of labour) that supported students' activity in coding. Our key findings are organized by *design elements for tasks*, *design elements for instructional practices* and *design elements for leveraging robot toys' design features*.

3.3.1 Design Elements for Robot Coding Toy Tasks

The purpose of our study was to examine what design elements for grid-agent CT tasks and instruction are important for supporting kindergarteners' coding activity. We found that it was important to (1) design introductory tasks focused on preparatory knowledge we call context proficiencies; (2) design tasks focused on CT components of algorithmic thinking, debugging and decomposition and their associated strategies; and (3) design tasks focused on play-with-constraints. Below, we discuss each of these necessary design elements.

3.3.1.1 Introductory Tasks Focused on Context Proficiencies

In designing tasks to elicit engagement with specific CT skills such as algorithmic thinking, debugging and decomposition, we found there were additional knowledge and skills that were contextual to the grid-agent paradigm that students needed to understand first. Specifically, we found students needed four context-specific

grid-agent proficiencies to effectively engage in algorithmic thinking, debugging and decomposition. These four context proficiencies are listed, defined and contextualized in Table 3.2.

Drawing from the CHAT perspective, these context proficiencies emerged as *rules* for coding activity, in particular the meaning of the codes and how to use them to operate the robot. Students' activity with the mediating artifact (robot toy) to engage in the object (coding) proved difficult without first exploring the rules (constraints of the toy and learning to use codes). This exploration of the rules occurred within the community (small group) as the students divided the labour (inputting the codes, providing ideas for the code needed, gesturing predicted movements), which thereby allowed students to better use the artifact to engage in coding. Hence, we found that it was important to design the introductory tasks to elicit students' activity in the context proficiencies. This was apparent in encouraging students to notice and explore individual commands for the robot as a way of cuing context proficiencies. Students' gestures were a mode they explored and expressed their understandings of these context proficiencies.

It is worth noting that important mathematical work takes place here, too. The context proficiencies are anchored in spatial concepts. Students explored spatial orientation and reasoning as well as measurement and countable objects (grid spaces or units of movement across grid spaces) in these tasks (Shumway et al., 2021). Spatial orientation, in particular, can be a mediator of the tasks, because tasks tend to be more challenging for students who are *not* seated sharing the robot's perspective on the grid space (Clarke-Midura et al., 2021a, b).

Table 3.2 Context proficiencies as critical concepts for grid-agent coding activity

Context proficiency	Definition	Vignette example
Space-symbol coordination	Knowing how codes or parts of programs correspond to movements or paths travelled by the agent	Larissa points to a `FORWARD` code on the remote and then leans over the grid to indicate moving forward in space
Spatial code meanings	Knowing what each of the codes instructs the agent to do	James indicates that the rotation code on the remote moves the robot "sideways"
Spatial orientation	Knowing that the codes always produce the same movements but depend on the agent's orientation	Larissa takes the robot's perspective and gestures to show it must rotate, rather than move forward (from her perspective)
One code to one movement correspondence	Knowing that one code produces a single discrete linear or rotational movement	To move the robot to an adjacent square, Larissa and Simon sequence `ROTATE RIGHT + FORWARD`

3.3.1.2 Tasks for CT Strategies

Tasks can be designed to elicit specific CT competencies or component skills. For example, in the *What Happened?* portion from the above vignette, we focused on *debugging* and had a bug at the beginning of the program (a rotation bug). This task had three design elements (an intentional bug, using the rotational bug, introducing the bug at the beginning of the program) that leveraged students' sense of spatial rotation while teaching how to precisely symbolize a specific kind of rotation (right versus left). Simultaneously, this task elicited debugging strategies. When Hyrum ran the buggy program and the robot went toward Larissa, students' recognition of the problem was first expressed by giggling and Larissa moving her body so that the robot did not run into her.

To debug this, there were two important actions from the children: (1) Larissa pointed to the `ROTATE LEFT` arrow in the program organizer, identifying the bug in the program and (2) Simon placed his hand on the goal, indicating that it was the space on the grid where the robot needed to land. Together, these led to the debugging strategy of *swapping out rotations* (Silvis et al., 2021). We found that debugging in kindergarten is more than simply recognizing, finding and fixing bugs, but it also requires knowing that debugging the code will fix the problem in the robot's movements or path. Often, young students want to fix the robot's movement by physically moving the robot rather than working with the code. We also found that debugging tasks are more challenging when the strategy involves modifying the beginning or middle of the program and when the program includes rotations. Varying these design elements (e.g. embedding a bug in the middle of the code or using a different error that does not involve the turn code) changes the difficulty of the task.

Some of the design elements for eliciting specific strategies we have identified from analysing groups of kindergarten students using robot coding toys include the following:

- Create an engaging need for a solution (e.g. getting a ball to the goal, observing a robot's secret program and trying to recreate it, planning an adventure).
- Provide an intentional progression of challenges (e.g. programs with only one rotation leading to programs with multiple rotations, programs with less than four codes leading up to programs with four or more codes).
- Intentional planning of bugs (e.g. the above vignette with varying types of bugs for different strategies: swapping, wipe and start over).
- Intentional planning for decomposition strategies (e.g. challenges where the robot makes "stops" along the way to highlight segments of codes within long programs, need to break apart a program).
- Intentional planning for algorithmic thinking strategies (e.g. tasks that require coding one at a time versus observing a whole program run and writing the program in chunks).

3.3.1.3 Tasks for Play

One other design element that was useful for encouraging CT was proposing semi-structured play tasks for practicing context proficiencies, planning programs and building programs (i.e. algorithmic thinking). For example, the *Happy Dance* task was designed for play with algorithmic thinking, but with constraints that focused students' attention on context proficiencies. For example, our vignette group was later assigned to pairs and tasked with programming a Happy Dance for Botley. Larissa and Hyrum were given only forward and backward codes, while James and Simon were given only rotational codes for their dance. Students used a program organizer for planning, and then they simulated their Botley's movements to show what they thought their program would tell Botley to do. They then tested the program and discussed whether it worked as intended. This testing of the program allowed them to playfully explore algorithmic thinking with the artefact while engaging their context proficiency knowledge.

3.3.2 Design Elements for Robot Coding Toy Instructional Practices

In addition to task design, instructional practices had features that could be modified. Those modifiable features were specific design elements summarized in Table 3.3. Many of these were identified in the vignette above but are more formally named and summarized here.

3.3.3 Design Elements for Leveraging or Supplementing the Robot Coding Toy's Features

While we were able to design the tasks that children completed and design the instructional support provided by the teacher, there are some aspects of the mediated group coding activity that rely on designed features that are built into the specific robot coding toys. For example, when completing a sequence of movement codes, Botley moves continuously, with no stops after each code. Conversely, Cubetto slows to a stop between each code in its program. Cubetto's slow, segmented movements afforded a dynamic visual for seeing that one forward or backward code results in one linear movement. The segmented stops can be highlighted to show students that each code in the program is a measured unit (e.g. one grid space) and multiple linear movements are iterated units of movement. Hence, we designed algorithmic thinking tasks to leverage this feature of Cubetto.

Another robot toy design feature we found to be important for kindergarten students was the program organizer. Some coding toys had a program organizer that

Table 3.3 Design elements for robot coding toy instructional practices

Design element	Instructional practice	Vignette examples
Plan for direct instruction of code functions	Introduce each code, observe associated movement (and from various orientations of both the students and robot) and read/verbalize codes Introduce opposites (R v L and F v B) Define: A rotational movement is on a point; robot stays in square Define: A linear movement is the same length every time, from one square to the next square on a grid	The teacher explained that they can tell Botley how to move by putting instructions in the remote using the arrow buttons, called codes. The teacher and children spent time programming forward and backward linear movements and observing Botley's movements across the grid-structured mat. The teacher used the mat to emphasize that one forward or backward arrow instructs Botley to move only one grid space. The teacher anticipated that the concept of a rotation on a point would be difficult
Plan for simulating movements	Simulate codes with gestures or by physically moving the robot toy (teacher and students) Students plan a path through simulating movements; predict what code is needed or what the robot will do through simulating the robot's movements (physically or with gestures)	The teacher encouraged students to physically move Botley and simulate its movements to show what they thought specific codes and sequences of codes would make Botley do before programming the sequence of codes in the remote and observing Botley's movements
Plan for naming of concepts and strategies	Watch for CT competencies, context proficiencies and strategies during students' activity and name the activity: Matched codes with movements, debugging, swapped codes, break it into parts, noticed the robot's orientation, etc.	Teacher: There's a bug in the program. This is what programmers do sometimes. They have to figure out where the bug is to fix it Larissa: *(Places the toy bug on the* `ROTATE LEFT` *code on program organizer)* Teacher: Say, it's a bug! Students: It's a bug! Larissa: It's the wrong way!
Plan for talk, gestures and collaboration	Ask questions for probing and extending thinking; encourage explanation Attach precise vocabulary to students' gestures (e.g. forward, rotate left) Ask students to verbalize codes (e.g. read code, count movements) Vary collaboration structures: pairs work on a task; provide roles among the small group (e.g. programmer, technician, evaluator, debugger)	Simon asked if they could program Botley to get to the adjacent green square, which required the program `ROTATE RIGHT + FORWARD`. The children negotiated which codes Simon should enter in the remote, using gestures to explain or justify their choices. After running the program (`ROTATE RIGHT + FORWARD`), the teacher asked them to reflect on their program

was provided with the toy (e.g. Cubetto's programming board, which is where code tiles must be placed for Cubetto to operate). For other robot coding toys like Botley and Bee-Bot, the toy operates after buttons corresponding to each code are pressed.

This does not provide students with a record of what "program" was entered. When there was not a program organizer, we created a separate one (see the example of the baking sheet and magnetic codes in Fig. 3.3). These custom-made organizers allowed the teacher to direct students' attention to the sequence of codes as the coding toy was moving around the grid and allowed students to document their programs. We saw that students could make one-to-one connections between codes on the program organizer and the remote as well as between the organizer and the coding toy movements on the grid (i.e. the context proficiency of space-symbol coordination). Other design elements built into the grid-agent coding toys are summarized in more detail by Kozlowski (2022).

3.4 Evidence of Mastery

In the moment of a group coding activity, mastery could be demonstrated by student gestures or the coding of one of the robot toys. However, there is a need in the early childhood CT research community for more formal assessment tools and resources (Weintrop et al., 2021). In addition to developing and refining tasks with student groups for use in the classroom, we have also been designing assessments of CT with this toy genre. The assessment we have created, through an evidence-centred design approach (Clarke-Midura et al., 2021a, b) involved a 10″ × 10″ 2-D grid and toy agents that students could hold and move along the grid with their hands (see Fig. 3.4). The assessment items involved storylines about the agent moving from

Fig. 3.4 Kindergarten student working on an assessment item

one location to another (e.g. moving the toy robot to the charging outlet on the grid). The students were asked to sequence, debug or enact programs using command arrows to indicate responses. The assessment was administered one-on-one in a standardized interview format to account for kindergarten children's emerging literacy.

We have designed and piloted summative assessment items as part of larger project work. After each instructional implementation, we administered the assessment and conducted an evaluation of the assessment items based on how students performed on them. We identified variable features within assessment items that, when varied, determined the difficulty of the task (Clarke-Midura et al., 2021a, b). Some variable features such as orientation of the agent aligned to the context proficiencies. Other variable features such as presence of rotations and distance travelled aligned to the mathematics concepts. We are currently reporting on the results of a large-scale validity and reliability study.

Currently, we are also completing development of formative assessment approaches that could be integrated into instruction. We have identified "indicators" of what we refer to as student's *knowledge in refinement* and are specifying some versions of group coding tasks that would allow teachers to gauge students' understanding in-the-moment (see Clarke-Midura et al., 2022).

3.5 Facilitating Resources

The above is a summary of iterative work that has led to an early childhood computational thinking (ECCT) competency model that captures our CT operationalization for grid-agent genre and with an eye toward mathematical concepts that these toys support (Fig. 3.5). Many of the key ideas, including those that can be especially challenging for students, are represented in this model.

The ECCT and the summaries provided above are emerging products from our larger endeavour to understand the sorts of tasks that can be designed to support kindergarten students' CT development through commercial coding toys and ways to assess that development. The various task elements and design features identified above are aspects that can be included or excluded from a given task, lesson interaction or assessment item and represent some of the design space that is available to educators and researchers who are working with this age group and with this grid-agent genre.

Our expectation is that a portion of the CT education ecosystem is going to remain commercial products (Lee, 2021) and that there is a need for the educational research community to explore that ecosystem and offer useful options and ideas for how educators can best leverage it (Papadakis, 2022). Curriculum materials for kindergarten teachers and other early childhood educators with the tasks and activities described above, along with others, are available online at http://cik.usu.edu. At that website, information about the assessment items and assessment approaches to see young children's CT development are also available.

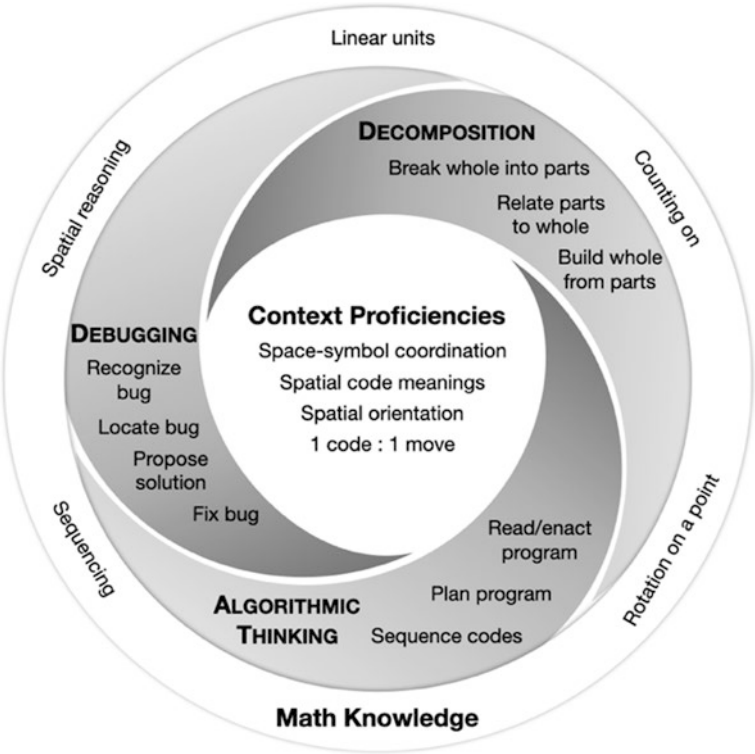

Fig. 3.5 The early childhood CT competency model resulting from design-based research

References

Angeli, C., & Valanides, N. (2020). Developing young children's computational thinking with educational robotics: An interaction effect between gender and scaffolding strategy. *Computers in Human Behavior, 105*(2020), Article 105954. https://doi.org/10.1016/j.chb.2019.03.018

Bakala, E., Gerosa, A., Hourcade, J. P., & Tejera, G. (2021). Preschool children, robots, and computational thinking: A systematic review. *International Journal of Child-Computer Interaction, 29*, 100337.

Bers, M. U., González-González, C., & Armas–Torres, M. B. (2019). Coding as a playground: Promoting positive learning experiences in childhood classrooms. *Computers & Education, 138*, 130–145. https://doi.org/10.1016/j.compedu.2019.04.013

Clarke-Midura, J., Lee, V. R., Shumway, J. F., & Hamilton, M. (2019). The building blocks of coding: A comparison of early childhood coding toys. *Information and Learning Science, 120*(7/8), 505–518.

Clarke-Midura, J., Kozlowski, J. S., Shumway, J. F., & Lee, V. R. (2021a). How young children engage in and shift between reference frames when playing with coding toys. *International Journal of Child-Computer Interaction, 28*, 100250.

Clarke-Midura, J., Silvis, D., Shumway, J. F., Lee, V. R., & Kozlowski, J. (2021b). Developing a kindergarten computational thinking assessment using Evidence Centered Design: The case of algorithmic thinking. *Computer Science Education, 31*(2), 117–140.

Clarke-Midura, J., Shumway, J. F., Lee, V. R., Silvis, D., Welch, L., & Kozlowski, J. (2022). *A model for integrated mathematics and computational thinking in kindergarten* (Manuscript submitted for publication). Utah State University.

Cobb, P., Jackson, K., & Sharpe, C. D. (2017). Conducting design studies to investigate and support mathematics children's and teachers' learning. In J. Cai (Ed.), *Compendium for research in mathematics education* (pp. 208–233). National Council of Teachers of Mathematics.

Engeström, Y. (1987). *Learning by expanding: An activity-theoretical approach to developmental research.* Orienta-Konsultit.

Hamilton, M., Clarke-Midura, J., Shumway, J. F., & Lee, V. R. (2020). An emerging technology report on coding toys and computational thinking in early childhood. *Technology, Knowledge, and Learning, 25,* 213–224. https://doi.org/10.1007/s10758-019-09423-8

Kafai, Y. B., & Burke, Q. (2014). *Connected code: Why children need to learn programming*. MIT Press.

Kozlowski, J. S. (2022). *Children's mathematical engagement based on their awareness of different coding toys' design features* (Publication No. 8420). Doctoral dissertation, Utah State University. All Graduate Theses and Dissertations. https://digitalcommons.usu.edu/etd/8420

Lee, V. R. (2021). Let's cut to commercial: Where research, evaluation, and design of learning games should go next. *Educational Technology Research and Development, 69*(1), 145–148. https://doi.org/10.1007/s11423-020-09865-3

Lin, Y., & Weintrop, D. (2021). The landscape of block-based programming: Characteristics of block-based environments and how they support the transition to text-based programming. *Journal of Computer Languages, 67*, 101075.

Muñoz-Repiso, A. G. V., & Caballero-González, Y. A. (2019). Robotics to develop computational thinking in early childhood education. *Comunicar, 27*(59), 63–72. https://doi.org/10.3916/C59-2019-06

Papadakis, S. (2021). The impact of coding apps to support young children in Computational Thinking and Computational Fluency. A literature review. *Frontiers in Education*. https://doi.org/10.3389/feduc.2021.657895

Papadakis, S. (2022). Can preschoolers learn computational thinking and coding skills with ScratchJr? A systematic literature review. *International Journal of Educational Reform*. https://doi.org/10.1177%2F10567879221076077

Pea, R. D., & Kurland, D. M. (1984). On the cognitive effects of learning computer programming. *New Ideas in Psychology, 2*(2), 137–168. https://doi.org/10.1016/0732-118X(84)90018-7

Sannino, A., & Engeström, Y. (2018). Cultural-historical activity theory: Founding insights and new challenges. *Cultural-Historical Psychology, 14*(13), 43–56.

Shumway, J. F., Welch, L., Kozlowski, J., Clarke-Midura, J., & Lee, V. R. (2021). Kindergarten students' mathematics knowledge at work: The mathematics for programming robot toys. *Mathematical Thinking and Learning*, 1. Advance online publication. https://doi.org/10.1080/10986065.2021.1982666

Shute, V. J., Sun, C., & Asbell-Clarke, J. (2017). Demystifying computational thinking. *Educational Research Review, 22*, 142–158.

Silvis, D., Clarke-Midura, J., Shumway, J. F., & Lee, V. R. (2021). Objects to debug with: How young children resolve errors with tangible coding toys. In *Proceedings of the International Society of the Learning Sciences (ISLS) annual meeting.* ISLS.

Weintrop, D., Rutstein, D. W., Biendowski, M., & McGee, S. (2021). Assessing computational thinking: An overview of the field. *Computer Science Education, 2*, 113–116. https://doi.org/10.1080/08993408.2021.1918380

Welch, L. E., Shumway, J. F., Clarke-Midura, J., & Lee, V. R. (2022). Exploring measurement through coding: Children's conceptions of a dynamic linear unit with robot coding toys. *Educational Sciences, 12*(2), 143. https://doi.org/10.3390/educsci12020143

Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35.