Combining Models to Orchestrate an Instructional Scenario Fostering Computational Thinking in Educational Robotics

Frankie Dubois, Stephanie Burton, Patrick Wang(✉), and Morgane Chevalier

University of Teacher Education, Lausanne, Switzerland
p49499@etu.hepl.ch,
{stephanie.burton,patrick.wang,morgane.chevalier}@hepl.ch

Abstract. Computational thinking (CT) is often defined as multifaceted which, on the one hand, allows researchers to embrace its complexity but, on the other hand, blurs the possibilities of its teaching. Although many models shed light on the multiple dimensions of CT, few studies investigate the benefits of combining such models when a teacher orchestrates in-class activities aiming at developing students' CT. This position paper aims to fill this gap by describing and analysing how a teacher can base the orchestration of a pedagogical scenario on three different models: Komis et al.'s model to design ER activities in co-creative problem solving, Sentance et al.'s PRIMM model to scaffold the students' tasks, and Chevalier et al.'s CCPS model to unscaffold the learning activities.

Keywords: Educational Robotics · Computational Thinking · Teacher's Guidance

1 Introduction

The construction of knowledge in computer science (CS) teaching and, in particular, during educational robotics learning activities (ERLA), can be based on three learning theories, namely constructivism [11], constructionism [10] and socio-constructivism [17]. These theories may be too abstract for teachers to be put in practice, limiting the number of studies available on their impacts on the students' learning processes. The scientific literature is nonetheless rich in recommendations on specific aspects of the construction of the knowledge at stake, for example, during an ERLA aimed at fostering student's computational thinking (CT) skills. In this respect, some studies [5,6] highlighted teaching and learning strategies that enable teachers to structure activities and encourage students to develop solutions to problem situations that have been thought out in advance whilst avoiding trial-and-error approaches that are generally unproductive and discouraging [2]. Such studies have the advantage of providing concrete

J.-P. Pellet and G. Parriaux (Eds.): ISSEP 2023, LNCS 14296, pp. 113–125, 2023.
https://doi.org/10.1007/978-3-031-44900-0_9

evidence of how CT is developed in the classroom but they only take into account part of the complexity involved in this thought processes. Yet teachers must take into account all this complexity and address it in a planned way. This is where comes in one's classroom orchestration expertise i.e. the ability to design and conduct "multi-plane scenarios under multiple constraints" [8].

Thus, to support teachers in their lesson planning, our research question is the following: When developping and implementing educational robotics scenarios, how can teachers combine different models validated by scientific research to foster students' computational thinking? In order to achieve this, a structure of the pedagogical scenario can be considered with reference to the scenario-based approach in educational robotics [9]. This approach will be reinforced by implementing the strategies developed in the PRIMM model [15] and in the CCPS model [6].

This paper has the following structure: in Sect. 2, we present our theoretical framework i.e., the three CT dimensions and three models to foster each of these facets. In Sect. 3, based on the combination of these three models, we expose the design of our pedagogical scenario to foster CT in ERLA. Subsequently, we highlight in Sect. 4 the relations between our theoretical model and the pedagogical scenario in order to justify its design. Finally, we conclude in Sect. 5.

2 Theoretical Framework

2.1 The CT Concept and Its Three Main Dimensions

According to Brennan and Resnick [4], CT is made up of three facets:

- **Computational perspectives**, which consist of cross-disciplinary abilities that are not characteristic of problem-solving within the framework of computer science. For example, these abilities can be to identify or generalize a problem, to model, generate ideas and communicate them.
- **Computational concepts**, which encompass the notions of computer science called upon during the learning activity and therefore, according to [13], notions linked to both the machine to be programmed (e.g., knowledge of the components of the robot used) and its programming language (e.g., knowledge of syntax and semantics, but also the knowledge of the interface used to program the robot).
- **Computational practices**, which refer to the skills required in the actual act of programming. This involves being able to decompose a problem or to modularize it, to test and evaluate a solution or to debug it, as well as working iteratively towards a satisfying solution.

CT comprises complex thought processes [1,18] that cannot be achieved directly. By conferring the status of competence to CT [7], it is then justified to approach CT in act (for example, via collaborative and creative co-construction) and in a situated way (for example, within a problem-solving situation in educational robotics). In order to ensure that we develop the full complexity of CT

(and not just one of its three dimensions), it therefore makes sense to use models validated in the literature and aimed at the same intentions formulated through the three CT dimensions according to Brennan and Resnick [4].

2.2 Scenario-Based Approach for Educational Robotics

The Scenario-Based Approach is a model designed to structure the ER activities for co-creative problem solving. Based on a constructivist-constructionist approach, it proposes a sequence of five activities to support the planning and orchestration of ER in K-12 education. This "ensure a progressive level of guidance towards the consolidation of the knowledge building process. The guidance is based on the scaffolding strategies of the Zone of Proximal Development (ZPD) [17]." [9, p.162]. In their paper, the authors identify the diversity of the ER activities that could be used in the first three activities of the scenario and claim that the last two activity types can be integrated within the first three. We therefore only describe the first three types of activity below.

Preparatory Activities: As the name suggests, this type of activity is designed to prepare learners for the ERLA: presentation of objectives, reminder of what is known about programming and robotics, presentation of the robotic tool used in the scenario. At this stage, however, students are not expected to manipulate the robot. Examples of this type of activity comprise lecture-based introduction to robotics or classroom debate about robotics.

Activities for Building Initial Knowledge: This second type of activity is designed to guide students in the use and manipulation of the educational robot leveraged in the scenario. The teacher plays a very important role at this point: to enable students to acquire the knowledge related to the robot's components and to its programming interface. Examples of this type of activity include individual guided activities or collaborative guided activities.

Activities for the Consolidation of Acquired Knowledge: This third type of activity is designed to enable students to design, manipulate and interact with their peers in a problem-solving situation. This gives students more responsibility and allows them to consolidate the knowledge built up in the previous two steps. Examples of this type of activity include individual or collaborative engineering problem, co-creative project-oriented robotic challenges.

2.3 The PRIMM Model

PRIMM [16] is a model designed to structure programming activities that avoid common difficulties found in the literature and, more specifically, the many challenges that students face when writing code. The approach followed by PRIMM is based on Vygotsky's Zone of Proximal Development [17] and consists in scaffolding the students' activities by proposing tasks that gradually progress from reading code to writing code.

PRIMM is an acronym which stands for Predict, Run, Investigate, Modify, Make. In the Predict phase, students are shown a piece of code and must work out (alone or in small groups) its outcome. In this stage of the model, students are asked to gather their code reading skills to figure out the result of the execution of the program. In the Run phase, the teacher executes the program so that students can confront their predictions with the actual outcome. In the Investigate phase, the teacher can question students regarding potential errors during the Predict phase and tries to explain these errors, or introduce new concepts to the class. The Investigate phase actually relies on the Block Model [14] to question the students' understanding of specific parts of the program, helping the acquisition or consolidation of programming knowledge. In the Modify phase, students are asked to work on the same program they had during the Predict phase and to modify it to adapt its execution based on a new set of (very similar) instructions. Finally, in the Make phase, students are asked to write a program from scratch by reusing what they have learnt throughout the process.

2.4 The CCPS Model

The CCPS model [6] is an instrument for planning and assessing whether the ERLA specifically promotes CT skills. In creative computational problem solving (CCPS), five phases (plus one to identify when the student is off-task) have been identified. All these phases are illustrated in Fig. 1. The remaining part of this section provides only a brief description of the model. For more information, we refer the readers to the original paper [6].

The first three phases of CCPS focus on three facets of CT (or the "computational perspective" dimension according to [4]): understanding the problem, generating ideas and formulating the robot's behavior. The other two phases refer to the "computational practices" dimension (ibid.): the fourth phase describes the creation of executable code (programming) by the robot, and the fifth phase focuses on executing the code to evaluate the solution. The "computational concepts" dimension (ibid.) is not identified in this model, as it is considered a prerequisite for the problem-solving task.

According to [6], during a CCPS in ER, students often rely on a trial-and-error strategy that is made possible by immediate feedback from the robots (for example, a LED activating on the robot to indicate to its user that the robot's infrared sensor has captured some information). However, this informational feedback can become less pedagogically relevant when students use it to promote task completion to the detriment of developing learning strategies [2]. For this reason, [6] suggests temporarily blocking access to the programming phase so as to promote the previous three phases of CCPS linked to the "computational perspective" (see the red "stop" sign in Fig. 1). In this way, the teacher's intervention can fade away (fading of scaffolding) during the problem-solving phase.

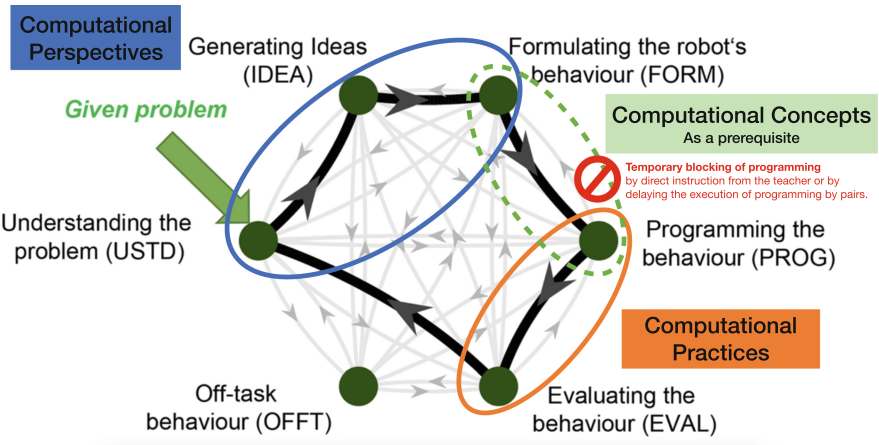

Fig. 1. The CCPS model [6] with six observable, interconnected phases (see grey arrows) and the three CT dimensions, marked in blue, green and orange. (Color figure online)

3 Design of the Instructional Scenario

Based on the state of the art and on the three models identified in Sect. 2, we designed the instructional scenario illustrated in Fig. 2 whose objective is to foster all three dimensions of CT in the context of educational robotics in primary school, for children aged 9–12 year old. The instructional scenario comprises eight activities in three different teaching stages, the latter referring to the three stages in the scenario-based approach [9] presented in Sect. 2.2. Activities #1 to #3 are "unplugged" activities [3], i.e. without the use of robots or any computing device. The following activities, on the other hand, are in plugged mode, and therefore include robots. The robot used in this proposal is the Blue-Bot[1].

The following description of the instructional scenario is also supported by the feedback from a teacher who co-designed and implemented it in a middle school classroom in Switzerland, with 10–12 year old pupils. While the design itself has not been rigorously evaluated, both the teacher and his pupils' feedback after the implementation of this scenario show that the combination of the three models selected is relevant. Indeed, the instructional scenario constructed in this way enabled the teacher to become aware of the stages to be respected in the construction of knowledge (in this case, the three CT dimensions) and to visualise more easily when he should or should not intervene with his pupils. It also enabled students to become more autonomous as the activities progressed and to develop important cross-curricular skills such as collaboration and creative thinking.

[1] https://www.tts-international.com/blue-bot-bluetooth-programmable-floor-robot/1015269.html.

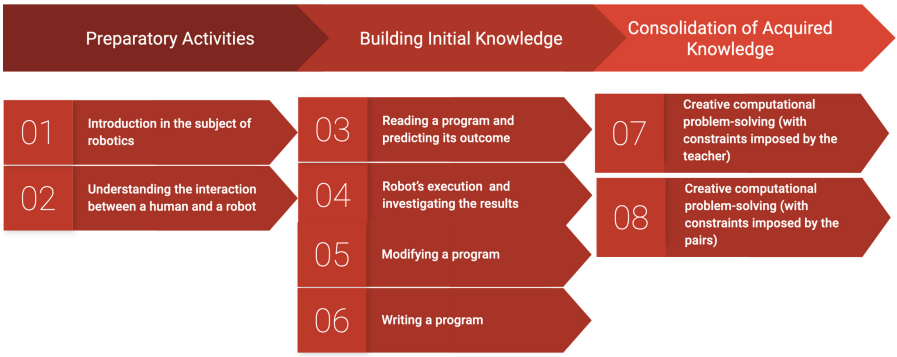

Fig. 2. Description of the instructional scenario composed of eight activities divided in three stages. (Color figure online)

3.1 Phase 1: Preparatory Activities

As shown in Fig. 2, the first phase concerns preparatory activities and is made up of Activity #1 and Activity #2, described hereafter.

Activity #1 aims to engage students in the subject of robotics, and thus introduces it. The chosen working method is a class debate. This approach enables all students to get involved in the subject. The aim is to define what a robot is, so the teacher successively asks students the following questions: "What is a robot?", "What does it do?", "What does it need to function?" For each of these questions, students write a keyword anonymously on a piece of paper. The papers are then collected by a classmate, who reads them while the teacher notes them on the board. When they are pooled on the board, the students are asked if any of the words could be grouped together. This pooling then generates a class discussion and debate for each question.

Activity #2 aims for students to understand that the robot machine executes a program defined in a univocal language understood by both the programmer and the machine. In order to make them aware of this, students work in freely formed pairs. First, Student A is given the following instruction in front of the whole class: "Using only your voice, guide your blindfolded classmate to the classroom library. As his vocabulary is limited to verbs, you can't name the various obstacles". So, initially, Student B is blindfolded and the other students modify the configuration of the classroom tables to create obstacles. Student A gives instructions to Student B to solve the given problem (i.e., enables him to progress to the library while avoiding the obstacles). The other students in the class observe and identify what does and does not solve the problem. The pooling of these identifications then leads to the emergence of the need for a shared, unambiguous language. Finally, all the students in the class pair up and carry out the activity in turn, to test this need.

3.2 Phase 2: Building Initial Knowledge

As shown in Fig. 2, the second phase concerns the building of initial knowledge and is made up of four activities (#3, #4, #5 and #6).

Activity #3 involves reading a program and predicting its outcome. For this purpose, the chosen working method is mainly individual work. Sheet 1 (Fig. 3, left) is distributed to each student and the following question is asked: "Following the instructions in the program, can you tell which square the robot should arrive in?" Individually, students read the program and trace the robot's path and then compare their results with their neighbors. In the end, they try to identify similarities and possible errors. After pooling their results, they make a joint prediction of the outcome.

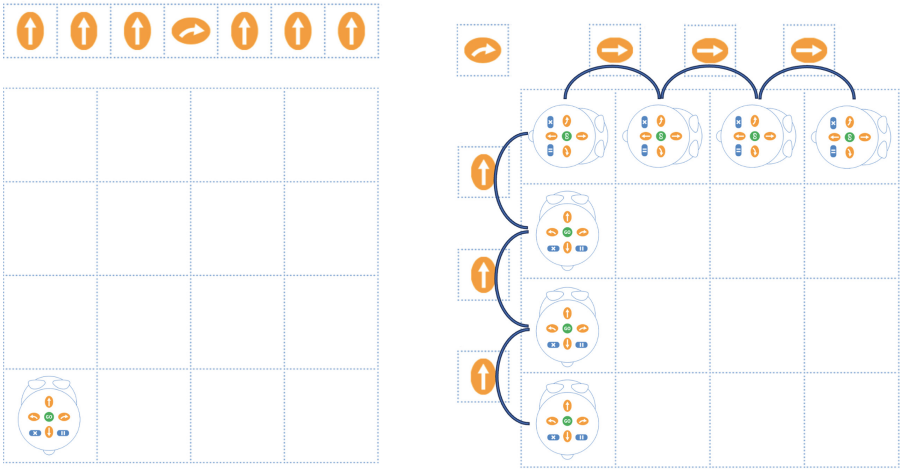

Fig. 3. Left: Example of a Predict task. Right: Guidance and pooling for the Investigate phase. (Color figure online)

Activity #4 aims to have a robot run the program of activity #3 and observe/investigate the results. The execution is carried out as many times as necessary to understand each step of the program. Again, students work in freely-formed pairs so as to encourage discussion when observing the results. With this plugged activity (at least for the "run" part), we need to prepare the following materials for each pair: a mat with 16 squares (15 cm by 15 cm), one Blue-Bot robot. The instructions to be given are as follows: "First, enter the program given on the robot and press the GO button to make the robot execute your program. Then, investigate each of the steps performed by the robot". A pooling of the investigation crystallizes the robot's programming language. The teacher's guidance can be based on the illustration on the right of Fig. 3.

Activity #5 aims to modify the program of activity #3. The working method chosen is once again in pairs, but the pairings must be different from the previous activity. The material is the same as in Activity #3, but now with

Sheet 2 (Fig. 4, left). The instructions to be given are as follows: "The robot's starting position has changed. How can this program be modified so that the robot arrives at the same position as before?" Paper programming cards are then used to support communication within the pairs. A final pooling of the results reveals both the difficulties encountered by the pairs and the strategies adopted to overcome them.

Activity #6 involves writing a program. The chosen working method is evenly matched pairs (in the same near-development zone). The materials required for each pair is the same as in Activity #4, but now with Sheet 3 (Fig. 4, right). The instructions to be given are as follows: "Create a program that allows the Blue-Bot robot to arrive at the garage". Students solve the given problem collaboratively, with the teacher intervening more with pairs who still need guidance.

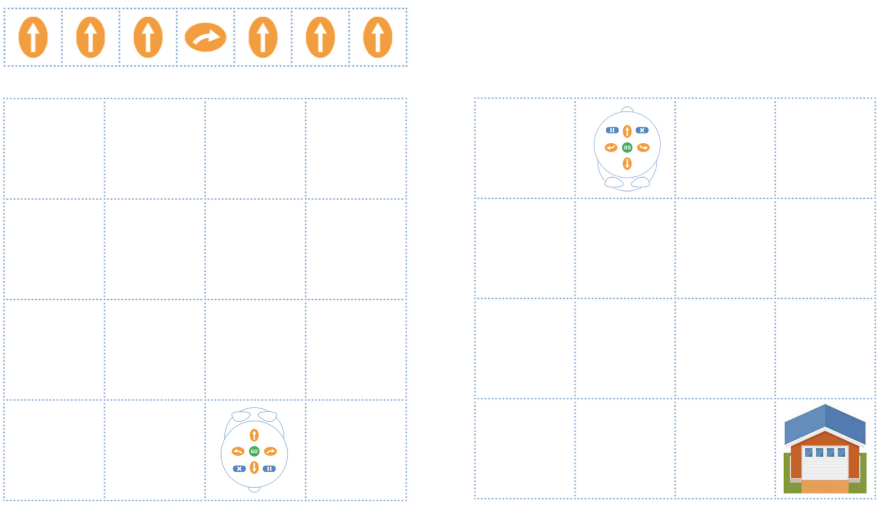

Fig. 4. Left: Example of a Modify activity. Right: Example of a Make activity. (Color figure online)

3.3 Phase 3: Consolidation of Acquired Knowledge

As shown in Fig. 2, the third and last phase concerns the consolidation of the acquired knowledge and is made up of Activity #7 and Activity #8.

Activity #7 involves creative computational problem-solving. The chosen working method is freely-formed pairs. The materials required for each pair are as follows (Fig. 5, left): a mat with nine green squares (15 cm by 15 cm), one Blue-Bot robot, a pencil-case for Blue-Bot, paper programming cards, paper and pencil. The instruction is: "The robot must mow the lawn symbolized by the nine green squares. The problem is considered solved if a pencil mark can be found

in each square. You have 20 min to solve this problem but with the following constraint: during the first five minutes, you cannot execute the program (by pressing the robot's Go button), you will be able to do so during the next five minutes; then, you will again be prevented from running the program for five minutes; finally, you will be able to do so during the last five minutes". The pooling of all the possible program scripts is held to validate the problem solution. A collective discussion should help identify the benefits of the constraints experienced during this resolution. The teacher explains the need to communicate within the group to clearly formulate the behavior of the robot to be programmed.

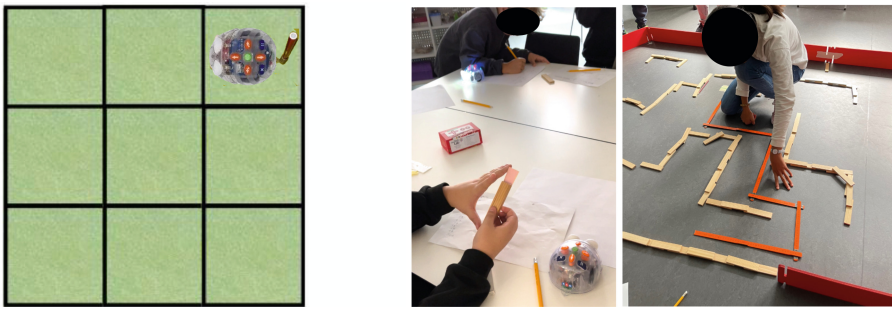

Fig. 5. Left: Material used for the lawnmower activity. Right: Materiel used for the "Theseus and the Minotaur" activity. (Color figure online)

Activity #8 involves creative computational problem-solving in a contextualized problem: Theseus and the Minotaur. The chosen working method is evenly matched pairs. The materials required for each pair are as follows (see Fig. 5, right): a maze (2 m by 3 m) located in another room, 1 Blue-Bot robot, paper programming cards, paper and pencil, conventional (ruler) and non-conventional (chablon) measuring tools. The instruction is: "The robot must get out of the maze (without destroying the walls) by executing the shortest possible program. You will only have two opportunities to come to the room where the maze is". Each member of a pair plays the role of either the programmer who programs the robot or the measurer who measures the number of necessary moves to reach the end of the maze. Finally, a pooling of all the possible program scripts is held to validate the shortest ones. A discussion brings out the different ways of thinking about computational problems in ER.

4 Design Rationale and Links with the Models

Orchestrating [8] an ERLA in the classroom involves, on the one hand, planning/designing it and, on the other, implementing it. Since the learning objective of this ERLA is CT, the three dimensions of CT [4] need to be taken into account: a) computational perspectives, b) computational concepts, and c) computational practices. The "Scenario-Based Approach for Educational Robotics"

model [9] enables structured planning of the construction of the knowledge at stake (in this case, CT). By proposing eight activities divided into three phases, we aim to support teachers in this process (see Fig. 6).

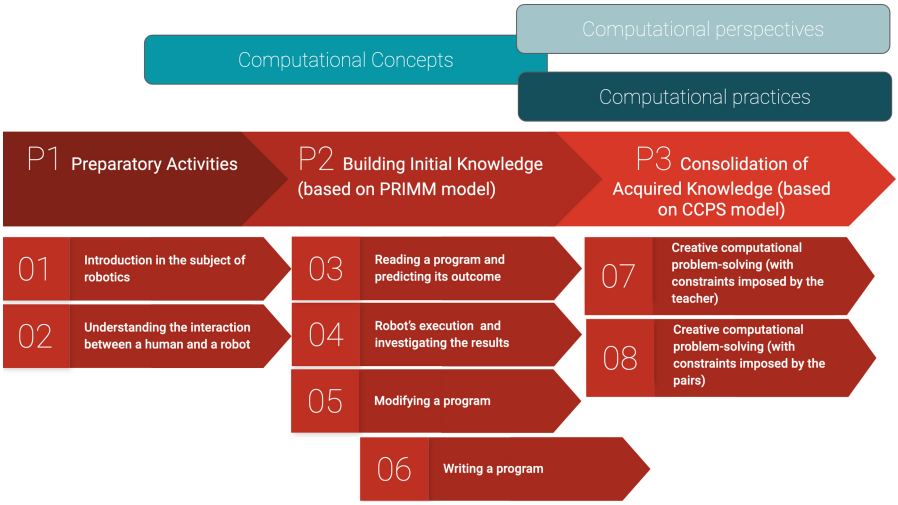

Fig. 6. Combining three models to orchestrate the three CT dimensions. (Color figure online)

Firstly, in phase 1, activities #1 and #2 are unplugged, so as to not burden the students' cognitive load with too much new knowledge (such as "computational concepts" like knowing the robot's components and programming interface). The priority here is to engage the students and challenge their initial representations. This phase also enables us to establish the principles of collaboration within the class. This basis will then be used as a lever during problem-solving in phase 3.

Secondly, in phase 2, we focus on building initial knowledge about CT. In this case, as identified by [6], the "computational concepts" dimension is a prerequisite for computational problem-solving (see dashed line in Fig. 1) since it must be possible to call on it when formulating the behaviour of the to be programmed robot and when programming it. To ensure the acquisition of these concepts, and more specifically of the programming language used by the robot, activities #3 to #6 are based on the PRIMM model [15] to support this learning through the following of five tasks: Predict, Run, Investigate, Modify, and Make. This last activity is a problem-solving task that indicates the CT competence level at this point in the scenario (end of phase 2). In this respect, we consider this to be a breaking point in the teacher's guidance. Up to this point, the teacher provided guidance and help to his students, leveraging the PRIMM model to scaffold the students' activities towards becoming more competent in the "computational concepts" dimension of CT. However, at this point in the scenario,

the students should have acquired a certain amount of autonomy regarding both the concepts and material in use, suggesting that a transfer of responsibility (from teacher to students) for learning [12] can occur. As a direct consequence, the teacher can now adapt and reduce his interventions, leaving the door open to the unscaffolding of learning activities.

Finally, in phase 3, higher-level learning should be encouraged [9]. With regard to CT, the aim is to reinforce the "computational practices" dimension while ensuring that the "computational perspectives" dimension takes root. This means proposing CCPS situations and planning a block to encourage all the phases of CT involved in the CCPS model, hence preventing students from going back and forth between programming and evaluating (see orange circle in Fig. 1).

The CCPS model is thus implemented in activities #7 and #8 with the goal of preventing unproductive trial-and-error strategies from occurring. For example, in activity #7, students are forced to work first on generating ideas and formulating expected behaviors without having any access to the programming environment (thus developing the "computational perspective" dimension of CT). After a few minutes, students can engage in the "computational practice" dimension by implementing and evaluating their solutions. This constraint is enforced by the teacher in activity #7 but then left to the responsibility of students during activity #8 (an additional rule can be added, stating that students can only test their solutions twice). This transfer of responsibility eventually participates in the unscaffolding strategy carried out in this third phase of the scenario.

An expectation of the CCPS model use is that students will engage in a virtuous cycle (as represented by the black arrows in Fig. 1) instead of getting stuck in the PROG-EVAL loop. With this structure of an instructional scenario, students should have developed all three facets of CT and, in the meantime, improved their CT skills since [6] shows that engaging in such a cycle seems to be beneficial with regards to the actual learning outcomes.

Finally, as indicated in Sect. 3, the scenario outilned was implemented by a teacher. His feedback is in line with the effects expected in this study. However, these effects still need to be measured more formally.

5 Conclusion and Future Work

In this position paper, we argued for the combination of three research models from the literature (scenario-based approach [9], PRIMM [16], and CCPS [6]) to support the design and orchestration of instructional scenarios aimed at fostering CT in ERLA. We back this claim by highlighting how these models cover each and every facet of CT as described in [4].

We also identify three possible leads for future work. First, we hope to evaluate how both expert and novices teachers implement this scenario in their classrooms and how it helps them plan and orchestrate the learning activities. Second, we wish to study the progression from a sequential-based robot (e.g., Blue-Bot) towards an event-based one (e.g., Thymio). And finally, we plan to assess the effects of such instructional scenarios on the students' learning outcomes regarding all three CT dimensions.

References

1. Aho, A.V.: Computation and computational thinking. Comput. J. **55**(7), 832–835 (2012)
2. Antle, A.N.: Exploring how children use their hands to think: an embodied interactional analysis. Behav. Inf. Technol. **32**(9), 938–954 (2013)
3. Bell, T., Vahrenhold, J.: Cs unplugged-how is it used, and does it work? Adventures between lower bounds and higher altitudes: essays dedicated to Juraj Hromkovič on the occasion of his 60th birthday, pp. 497–521 (2018)
4. Brennan, K., Resnick, M.: New frameworks for studying and assessing the development of computational thinking. In: Proceedings of the 2012 Annual Meeting of the American Educational Research Association, Vancouver, Canada, vol. 1, p. 25 (2012)
5. Chevalier, M., et al.: The role of feedback and guidance as intervention methods to foster computational thinking in educational robotics learning activities for primary school. Comput. Educ. **180**, 104431 (2022)
6. Chevalier, M., Giang, C., Piatti, A., Mondada, F.: Fostering computational thinking through educational robotics: a model for creative computational problem solving. Int. J. STEM Educ. **7**(1), 1–18 (2020)
7. Chevalier, M.S.D.: Mediating computational thinking through educational robotics in primary school. Technical report, EPFL (2022)
8. Dillenbourg, P., Prieto, L.P., Olsen, J.K.: Classroom orchestration. Int. Handb. Learn. Sci. 180–190 (2018)
9. Komis, V., Romero, M., Misirli, A.: A scenario-based approach for designing educational robotics activities for co-creative problem solving. In: Alimisis, D., Moro, M., Menegatti, E. (eds.) Edurobotics 2016 2016. AISC, vol. 560, pp. 158–169. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-55553-9_12
10. Papert, S., Harel, I.: Situating constructionism. Constructionism **36**(2), 1–11 (1991)
11. Piaget, J.: Six Etudes de Psychologie. Genéve: Editions Gonthier (1964)
12. Van de Pol, J., Volman, M., Beishuizen, J.: Scaffolding in teacher-student interaction: a decade of research. Educ. Psychol. Rev. **22**, 271–296 (2010)
13. Romero, M., Lepage, A., Lille, B.: Computational thinking development through creative programming in higher education. Int. J. Educ. Technol. High. Educ. **14**(1), 1–15 (2017)
14. Schulte, C.: Block model: an educational model of program comprehension as a tool for a scholarly approach to teaching. In: Proceedings of the Fourth International Workshop on Computing Education Research, pp. 149–160 (2008)
15. Sentance, S., Csizmadia, A.: Computing in the curriculum: challenges and strategies from a teacher's perspective. Educ. Inf. Technol. **22**(2), 469–495 (2017)
16. Sentance, S., Waite, J., Kallia, M.: Teaching computer programming with PRIMM: a sociocultural perspective. Comput. Sci. Educ. **29**(2–3), 136–176 (2019)
17. Vygotsky, L.S.: Psychology and localization of functions. Neuropsychologia **3**(4), 381–386 (1965)
18. Wing, J.M.: Computational thinking. Commun. ACM **49**(3), 33–35 (2006)