International Journal of
STEM Education

RESEARCH Open Access

Fostering computational thinking through educational robotics: a model for creative computational problem solving

Morgane Chevalier[1,2*†], Christian Giang[2,3†], Alberto Piatti[3] and Francesco Mondada[2]

Abstract

Background: Educational robotics (ER) is increasingly used in classrooms to implement activities aimed at fostering the development of students' computational thinking (CT) skills. Though previous works have proposed different models and frameworks to describe the underlying concepts of CT, very few have discussed how ER activities should be implemented in classrooms to effectively foster CT skill development. Particularly, there is a lack of operational frameworks, supporting teachers in the design, implementation, and assessment of ER activities aimed at CT skill development. The current study therefore presents a model that allows teachers to identify relevant CT concepts for different phases of ER activities and aims at helping them to appropriately plan instructional interventions. As an experimental validation, the proposed model was used to design and analyze an ER activity aimed at overcoming a problem that is often observed in classrooms: the trial-and-error loop, i.e., an over-investment in programming with respect to other tasks related to problem-solving.

Results: Two groups of primary school students participated in an ER activity using the educational robot Thymio. While one group completed the task without any imposed constraints, the other was subjected to an instructional intervention developed based on the proposed model. The results suggest that (i) a non-instructional approach for educational robotics activities (i.e., unlimited access to the programming interface) promotes a trial-and-error behavior; (ii) a scheduled blocking of the programming interface fosters cognitive processes related to problem understanding, idea generation, and solution formulation; (iii) progressively adjusting the blocking of the programming interface can help students in building a well-settled strategy to approach educational robotics problems and may represent an effective way to provide scaffolding.

Conclusions: The findings of this study provide initial evidence on the need for specific instructional interventions on ER activities, illustrating how teachers could use the proposed model to design ER activities aimed at CT skill development. However, future work should investigate whether teachers can effectively take advantage of the model for their teaching activities. Moreover, other intervention hypotheses have to be explored and tested in order to demonstrate a broader validity of the model.

Keywords: Computational thinking, Educational robotics, Instructional intervention, Problem solving, Trial-and-error

* Correspondence: morgane.chevalier@hepl.ch
†Morgane Chevalier and Christian Giang contributed equally to this work.
[1]Haute Ecole Pédagogique (HEP) du Canton de Vaud, Avenue de Cour, 33, 1014 Lausanne, Switzerland
[2]Mobots Group of the Biorobotics Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Route Cantonale, 1015 Lausanne, Switzerland
Full list of author information is available at the end of the article

Introduction

Educational robotics (ER) activities are becoming increasingly popular in classrooms. Among others, ER activities have been praised for the development of important twenty-first century skills such as creativity (Eguchi, 2014; Negrini & Giang, 2019; Romero, Lepage, & Lille, 2017) and collaboration (Denis & Hubert, 2001; Giang et al., 2019). Due to its increasing popularity, ER is also often used to implement activities aimed at fostering CT skills of students. Such activities usually require students to practice their abilities in problem decomposition, abstraction, algorithm design, debugging, iteration, and generalization, representing six main facets of CT (Shute, Sun, & Asbell-Clarke, 2017). Indeed, previous works have argued that ER can be considered an appropriate tool for the development of CT skills (Bers, Flannery, Kazakoff, & Sullivan, 2014; Bottino & Chioccariello, 2014; Catlin & Woollard, 2014; Chalmers, 2018; Eguchi, 2016; Leonard et al., 2016; Miller & Nourbakhsh, 2016).

Nevertheless, studies discussing how to implement ER activities for CT skills development in classrooms, still appear to be scarce. The latest meta-analyses carried out on ER and CT (Hsu, Chang, & Hung, 2018; Jung & Won, 2018; Shute et al., 2017) have mentioned only four works between 2006 and 2018 that elaborated how ER activities should be implemented in order to foster CT skills in K-5 education (particularly for grades 3 and 4, i.e., for students of age between 8 and 10 years old). Another recent work (Ioannou & Makridou, 2018) has shown that there are currently only nine empirical investigations at the intersection of ER and CT in K-12. Among the recommendations for researchers that were presented in this work, the authors stated that it is important to "work on a practical framework for the development of CT through robotics." A different study (Atmatzidou & Demetriadis, 2016) has pointed out that there is a lack of "explicit teacher guidance on how to organize a well-guided ER activity to promote students' CT skills."

In the meta-analysis of Shute et al. (2017), the authors reviewed the state of the art of existing CT models and concluded that the existing models were inconsistent, causing "problems in designing interventions to support CT learning." They therefore synthesized the information and proposed a new CT model represented by the six main facets mentioned above (Shute et al., 2017). Though the authors suggested that this model may provide a framework to guide assessment and support of CT skills, the question remains whether teachers can take advantage of such models and put them into practice. In order to support teachers in the design, implementation, and assessment of activities addressing these CT components, it can be presumed that more operational frameworks are needed. Particularly, such frameworks should provide ways to identify specific levers that teachers can adjust to promote the development of CT skills of students in ER activities.

To address this issue, the present work aims at providing an operational framework for ER activities taking into consideration two main aspects of CT, computation, and creativity, embedded in the context of problem-solving situations. The objective of the present study is to obtain a framework that allows teachers to effectively design ER activities for CT development, anticipate what could occur in class during the activities, and accordingly plan specific interventions. Moreover, such framework could potentially allow teachers to assess the activities in terms of CT competencies developed by the students.

To verify the usefulness of the proposed framework, it has been used to design and analyze an ER activity aimed at overcoming a situation that is often observed in classrooms: the trial-and-error loop. It represents an over-investment in programming with respect to other problem-solving tasks during ER activities. In the current study, an ER activity has been developed and proposed to two groups of primary school pupils: a test group and a control group, each performing the same task under different experimental conditions. The students were recorded during the activity and the videos have been analyzed by two independent evaluators to study the effectiveness of instructional interventions designed according to the proposed framework and implemented in the experimental condition for the test groups.

In the following section, past works at the intersection of ER and CT are summarized. This is followed by the presentation of the creative computational problem-solving (CCPS) model for ER activities aimed at CT skills development. Subsequently, the research questions addressed in this study are described, as well as the methods for the experimental validation of this study. This is followed by the presentation of the experimental results and a discussion on these findings. The paper finally concludes with a summary on the possible implications and the limitations of the study.

Background

What three meta-analyses at the crossroads of ER and CT have shown

The idea of using robots in classrooms has a long history—indeed, much time has passed since the idea was first promoted by Papert in the late 1970s (Papert, 1980). On the other hand, the use of ER to foster CT skills development appears to be more recent—it was in 2006 that Jeannette Wing introduced the expression "computational thinking" to the educational context (Wing, 2006). It is therefore not surprising that only three meta-analyses have examined the studies conducted on this topic between 2006 and 2017 (Hsu et al., 2018; Jung & Won, 2018; Shute et al., 2017).

In the first meta-analysis (Jung & Won, 2018), Jung and Won describe a systematic and thematic review on existing ER literature (n = 47) for K-5 education. However, only four out of the 47 analyzed articles related ER to CT and only two of them (Bers et al., 2014; Sullivan, Bers, & Mihm, 2017) conveyed specific information about how to teach and learn CT, yet limited to K-2 education (with the Kibo robot and the TangibleK platform).

In a second meta-analysis (Hsu et al., 2018), Hsu et al. conducted a meta-review on CT in academic journals (n = 120). As a main result, the authors concluded that "CT has mainly been applied to activities of program design and computer science." This focus on programming seems to be common and has been questioned before—among others, it has been argued that CT competencies consist of more than just skills related to programming.

A similar conclusion was found in the third meta-analysis (Shute et al., 2017). Shute et al. conducted a review among literature on CT in K-16 education (n = 45) and stated that "considering CT as knowing how to program may be too limiting." Nevertheless, the relation between programming and CT seems to be interlinked. The authors suggested that "CT skills are not the same as programming skills (Ioannidou, Bennett, Repenning, Koh, & Basawapatna, 2011), but being able to program is one benefit of being able to think computationally (Israel, Pearson, Tapia, Wherfel, & Reese, 2015)."

According to the findings of these three recent meta-analyses, it appears that there is still a lack of studies focusing on how ER can be used for CT skills development. Except for two studies that specifically describe how to teach and learn CT with ER in K-2 education, operational frameworks guiding the implementation of ER activities for students, especially those aged between 8 and 10 years old (i.e., grades 3 and 4), are still scarce. It also emerges that in the past, activities aimed at CT development have focused too much on the programming aspects. However, CT competences go beyond the limitations on pure coding skills and ER activities should therefore be designed accordingly.

CT development with ER is more than just programming a robot

Because robots can be programmed, ER has often been considered a relevant medium for CT skill development. However, many researchers have also argued that CT is not only programming. As illustrated by Li et al. (2020), CT should be considered "as a model of thinking that is more about thinking than computing" (p.4). In the work of Bottino and Chioccariello (2014), the authors are reminiscent of what Papert claimed about the use of robots (Papert, 1980). Programming concrete objects such as robots support students' active learning as robots can "provide immediate feedback and concept reification." The programming activity is thus not the only one that is important for CT skills development. Instead, evaluating (i.e., testing and observing) can be considered equally important. In the 5c21 framework for CT competency of Romero et al. (2017), both activities therefore represent separate components: create a computer program (COMP5) and evaluation and iterative improvement (COMP6).

While the evaluation of a solution after programming appears to be natural for most ER activities, it seems that activities prior to programming often receive far less attention. Indeed, it is also relevant to explore what activities are required before programming, that is to say, before translating an algorithm into a programming language for execution by a robot. Several efforts have shown that different activities can be carried out before the programming activity (Giannakoulas & Xinogalos, 2018; Kazimoglu, Kiernan, Bacon, & MacKinnon, 2011, 2012). For instance, puzzle games such as Lightbot (Yaroslavski, 2014) can be used to convey basic concepts needed before programming (Giannakoulas & Xinogalos, 2018). In another work (Kazimoglu et al., 2012), a fine effort has been made to put in parallel the task to be done by the student (with a virtual bot) and the cognitive activity implied. This is how the authors sustained the CT skills of students before programming. The integration of such instructional initiatives prior to programming is usually aimed at introducing fundamental concepts necessary for the programming activities. Indeed, code literacy (COMP3) and technological system literacy (COMP4) have been described as two other components in the framework of Romero et al. (2017), and they have been considered important prerequisites for the use of programmable objects.

But even if students meet these prerequisites, there are other important processes that they should go through prior to the creation of executable code. The two following components in the framework of Romero et al. are related to these processes: problem identification (COMP1) and organize and model the situation (COMP2). However, it appears that in the design of ER activities, these aspects are often not given enough attention. In a classroom environment, robots and computers often attract students' attention to such an extent that the students tend to dive into a simple trial-and-error approach instead of developing proper solution strategies. Due to the prompt feedback of the machine, students receive an immediate validation of their strategy, reinforcing their perception of controllability (Viau, 2009), but this also causes them to easily enter in a trial-and-error loop (Shute et al., 2017). In many different learning situations, however, researchers have shown that a pure trial-and-error-approach may limit skill development (Antle, 2013; Sadik, Leftwich, & Nadiruzzaman, 2017; Tsai, Hsu, & Tsai, 2012). In the context of inquiry-based science learning, Bumbacher et al. have shown that students who were instructed to follow a Predict-Observe-Explain (POE) strategy, forcing them to take breaks between actions, gained better conceptual understanding than students who used the same manipulative environment without any instructions (Bumbacher, Salehi, Wieman, & Blikstein, 2018). The strategic use of pauses has also been investigated by Perez et al. (2017) in the context of students who worked with a virtual lab representing a DC circuit construction kit. The authors argued that strategic pauses can represent opportunities for reflection and planning and are highly associated with productive learning. A similar approach has been discussed by Dillenbourg (2013) who introduced a paper token to a tangible platform to prevent students from running simulations without reflection. Only when students gave a satisfactory answer to the teacher about the predicted behavior of the platform, they were given the paper token to execute the simulations.

However, to this day such instructional interventions have not been applied to activities involving ER. As a matter of fact, many ER activities are conducted without any specific instructional guidance.

As elaborated before, the development of CT skills with ER should involve students in different phases that occur prior as well as after the creation of programming code. While most of the time, the evaluation of a solution after programming appears to be natural, the phases required prior to programming are usually less emphasized. These preceding phases, however, incorporate processes related to many important facets of CT and should therefore be equally addressed. The following section therefore introduces a model for ER activities that allows teachers to identify all relevant phases related to different CT skills. Based on this model, teachers may accordingly plan instructional interventions to foster the development of such CT skills.

The CCPS model

Educational robotic systems for the development of CT skills

Educational robotics activities are typically based on three main components: one or more educational robots, an interaction interface allowing the user to communicate with the robot and one or more tasks to be solved on a playground (Fig. 1).

This set of components is fundamental to any kind of ER activity and has been previously referred to as an Educational Robotics System (ERS) by Giang, Piatti, and Mondada (2019). When an ERS is used for the development of CT skills, the given tasks are often formulated as open-ended problems that need to be solved. These problems are usually statements requiring the modification of a given perceptual reality (virtual or concrete) through a creative act in order to satisfy a set of

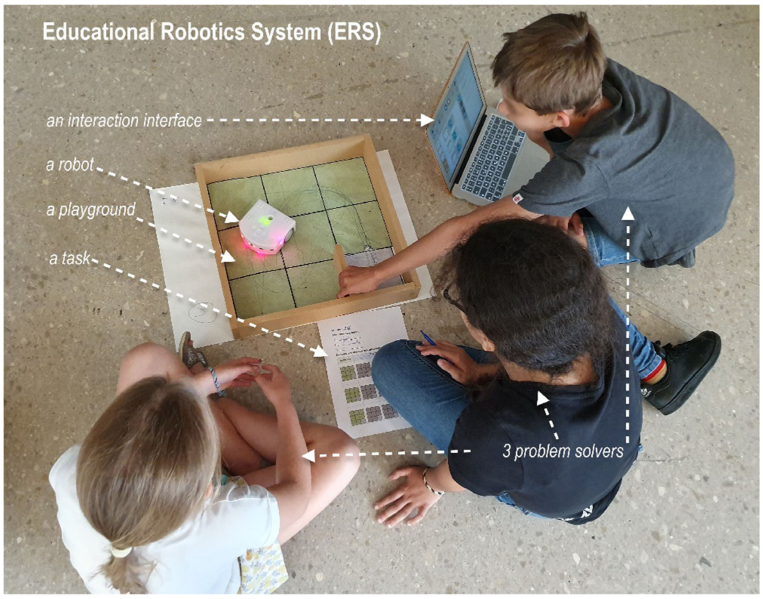


Fig. 1 Example of an educational robotics (ER) activity. The figure exemplifies a typical situation encountered in ER activities. One or more problem solvers work on a playground and confront a problem situation involving an Educational Robotics System (ERS), consisting of one or more robots, an interaction interface and one or more tasks to be solved

conditions. In most cases, a playground relates to the environment (offering the range of possibilities) in which the problem is embedded. The modification can consist in the creation of a new entity or the realization of an event inside the playground, respectively, the acquisition of new knowledge about the playground itself. A modification that satisfies the conditions of the problem is called solution. The problem solver is a human (or a group of humans), that is able to understand and interpret the given problem, to create ideas for its resolution, to use the interaction interface to transform these ideas into a behavior executed by the robot, and to evaluate the solution represented by the behavior of the robot. The language of the problem solver and the language of the robot are usually different. While the (human) problem solver's language consists of natural languages (both oral and written), graphical, or iconic representations and other perceptual semiotic registers, the (artificial) language of the robot consists of formal languages (i.e., machine languages, binary code). Consequently, the problem should be stated in the problem solver's language, while the solution has to be implemented in the robot's language. For the problem solver, the robot's language is a sort of foreign language that he/she should know in order to communicate with the robot. On the other hand, the robot usually does not communicate directly with the problem solver but generates a modification of the playground that the problem solver can perceive through his/her senses. To facilitate the interaction, the robot embeds a sort of translator between the robot's language and the problem solver's language. Indeed, graphical or text programming languages allow part of the language of the robot to be shown and written in iconic representations that can be directly perceived by the problem solver.

Combining creative problem solving and computational problem solving

It has often been claimed that CT is a competence related to the process of problem solving that is contextualized in "computational" situations (Barr & Stephenson, 2011; Dierbach, 2012; Haseski, Ilic, & Tugtekin, 2018; Perkovic, Settle, Hwang, & Jones, 2010; Weintrop et al., 2016). These processes involve in particular the understanding of a given problem situation, the design of a solution, and the implementation in executable code. At the same time, some researchers have pointed out that the development of CT competencies also involves a certain creative act (Brennan & Resnick, 2012; DeSchryver & Yadav, 2015; Repenning et al., 2015; Romero et al., 2017; Shute et al., 2017). This perspective refers to creative problem solving which involves "a series of distinct mental operations such as collecting information, defining problems, generating ideas, developing solutions, and taking action" (Puccio, 1999). In a different context, creative problem solving has been described as a cooperative iterative process (Lumsdaine & Lumsdaine, 1994) involving different persons with different mindsets and thinking modes and consisting of five phases: problem definition (detective and explorer), idea generation (artist), idea synthesis (engineer), idea evaluation (judge), and solution implementation (producer) (Lumsdaine & Lumsdaine, 1994).

The creative computational problem solving (CCPS) model presented in the current study, represents a hybrid model combining these two perspectives and adapting them to the context of ERS. Similar to the model of Lumsdaine and Lumsdaine (1994), the proposed model involves the definition of different phases and iterations. However, while Lumsdaine and Lumsdaine's model describes the interactions between different human actors, each taking a specific role in the problem-solving process, this model considers the fact that different human actors interact with one or more artificial actors, i.e., the robot(s), to implement the solution. The CCPS model is a structure of five phases, in which transitions are possible, in each moment, from any phase to any other (Fig. 2).

The first three phases of the model can be related to the initial phases of the creative problem-solving model presented in the work of Puccio (1999): understanding the problem, generating ideas, and planning for action (i.e., solution finding, acceptance-finding). While the first two phases (understanding the problem and generating ideas) are very similar to Puccio's model, the third phase in this model (formulating the robot's behavior) is influenced by the fact that the action should be performed by an artificial agent (i.e., a robot). On the other hand, the last two phases of this model can be related to computational problem solving: the fourth phase (programming the behavior) describes the creation of executable code for the robot and the fifth phase (evaluating the solution) consists in the evaluation of the execution of the code (i.e., the robot's behavior).

The phases of the CCPS model

Based on the conceptual framework of ERS (Giang, Piatti, & Mondada, 2019), the CCPS model describes the different phases that students should go through when ERS is used for CT skills development (Fig. 2).

It is a structure of five main phases that theoretically, in the most effective case, are completed one after the other and then repeated iteratively.

Understanding the problem (USTD) In this phase, the problem solver identifies the given problem (see COMP1 in the 5c21 framework of Romero et al. (2017)) through abstraction and decomposition (Shute et al., 2017) in

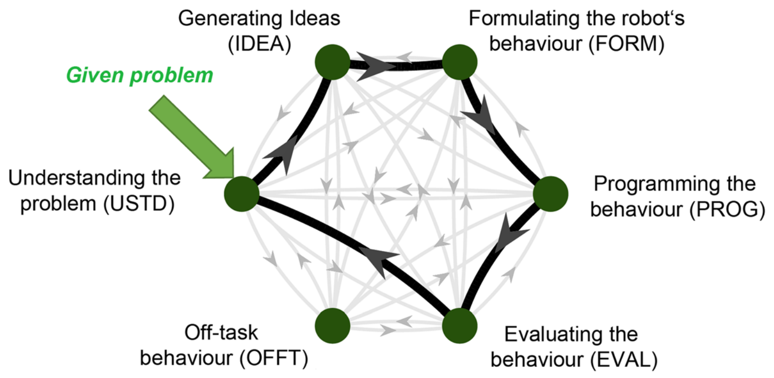

Generating Ideas (IDEA)
Formulating the robot's behaviour (FORM)
Given problem
Understanding the problem (USTD)
Programming the behaviour (PROG)
Off-task behaviour (OFFT)
Evaluating the behaviour (EVAL)

Fig. 2 Phases and transitions of the CCPS model. The graph illustrates the six different phases (green dots) that students pass through when working on ER activities and all the possible transitions between them (gray arrows). The theoretically most efficient problem-solving cycle is highlighted in black. The cycle usually starts with a given problem situation that needs first to be understood by the problem solver (green arrow)

order to identify the desired modification of the playground. Here, abstraction is considered as the process of "identifying and extracting relevant information to define main ideas" (Hsu et al., 2018). This phase takes as input the given problem situation, usually expressed in the language of the problem solver (e.g., natural language, graphical representations). The completion of this phase is considered successful if the problem solver identifies an unambiguous transformation of the playground that has to be performed by the robot. The output of the phase is the description of the required transformation of the playground.

Generating ideas (IDEA) The problem-solver sketches one or more behavior ideas for the robot that could satisfy the conditions given in the problem, i.e., modify the playground in the desired way. This phase requires a creative act, i.e., "going from intention to realization" (Duchamp, 1967). The input to this phase is the description of the transformation of the playground that has to be performed by the robot. The phase is completed successfully when one or more behaviors are sketched that have the potential of inducing the desired transformation of the playground. The sketches of the different behaviors are the output of this phase.

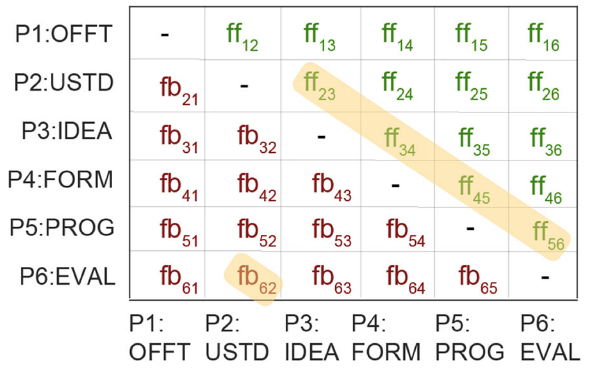
Formulating the behavior (FORM) A behavior idea is transformed into a formulation of the robot's behavior while considering the physical constraints of the playground and by mobilizing the knowledge related to the characteristics of the robot (see COMP4 in Romero et al. (2017)). To do so, the problem solver has to organize and model the situation efficiently (like in COMP2 in Romero et al. (2017)). The input to this phase is the sketch of a behavior, selected among those produced in the preceding phase. The phase is performed successfully when the behavior sketch is transformed into a complete formulation of the robot's behavior. The behavior formulation is expressed as algorithms in the problem solver's language, describing "logical and ordered instructions for rendering a solution to the problem" (Shute et al., 2017). This is considered the output of this phase.

Programming the behavior (PROG) In this phase, the problem solver creates a program (see COMP5 in Romero et al. (2017)) to transform the behavior formulation into a behavior expressed by the robot. Prerequisites for succeeding in this phase are the necessary computer science literacy and the knowledge of the specific programming language of the robot or its interface, respectively (see COMP3 in Romero et al. (2017)). Moreover, this phase serves for debugging (Shute et al., 2017), allowing the problem solver to revise previous implementations. The input to this phase is the robot's behavior expressed in the problem solver's language. The phase is performed successfully when the formulated behavior of the robot is completely expressed in the robot's language and executed. The output of this phase is the programmed behavior in the robot's language and its execution so that, once the robot is introduced to the playground, it results in a transformation of the playground.

Evaluating the solution (EVAL) While the robot performs a modification of the playground according to the programmed behavior, the problem solver observes the realized modification of the playground and evaluates its correspondence to the conditions of the problems and its adequacy in general. As described in Lumsdaine and Lumsdaine (1994), the problem solver acts as a "judge" in this phase. The input to this phase is the transformation of the playground observed by the problem solver. The observed transformation is compared with the conditions expressed in the given problem. Then, the problem solver has to decide if the programmed behavior can be considered an appropriate solution of the problem, or if it has to be refined, corrected, or completely redefined. This phase is therefore crucial to identify the next step of iteration (see Shute et al. (2017) and COMP2 in Romero et al. (2017)). As a result, the transitions in the CCPS model can either be terminated or continued through a feedback transition to one of the other phases.

Finally, an additional sixth phase, called *off-task behavior (OFFT)*, was included to account for situations where the problem solver is not involved in the problem-solving process. This phase was not considered a priori, however, the experiments with students showed that off-task behavior is part of the reality in classrooms and should therefore be included in the model. Moreover, in reality, transitions between phases do not necessarily occur in the order presented. Therefore, the model also accounts for transitions between non-adjacent phases as well as for transitions into the off-task behavior phase (light arrows in Fig. 2). In order to facilitate the presentation of these transitions, the matrix representation of the model is introduced hereafter (Fig. 3).

In this representation, a feedforward (ff) is the transition from a phase to any of the subsequent ones and feedback (fb) is the transition from a phase to any of the preceding ones. Consequently, ff_{ij} denotes the feedforward from phase i to phase j, where $i < j$ and fb_{ij} denotes the feedback from phase i to phase j, where $j < i$. With six states, there are in theory 15 possible feedforward (upper triangular matrix) and 15 possible feedback transitions (lower triangular matrix), as represented in Fig. 3. Although some of the transitions seem meaningless, they might however be observed in reality and are therefore kept in the model. For instance, it seems that the transition ff_{36} would not be possible, since a solution can only be evaluated if it has been programmed. However, especially in the context of group work, it might be possible that a generated idea is immediately followed by an evaluation phase, which was implemented by another student going through the programming phase. Finally, it can be assumed that feedback transitions usually respond to instabilities in previous phases. In this model, special emphasis is therefore placed on the cycle considering the transitions ff_{23}-ff_{34}-ff_{45}-ff_{56}-fb_{62} (highlighted in yellow in Fig. 3), which, as presented before, correspond to the theoretically most efficient cycle within the CCPS model.



	P1: OFFT	P2: USTD	P3: IDEA	P4: FORM	P5: PROG	P6: EVAL
P1:OFFT	-	ff_{12}	ff_{13}	ff_{14}	ff_{15}	ff_{16}
P2:USTD	fb_{21}	-	ff_{23}	ff_{24}	ff_{25}	ff_{26}
P3:IDEA	fb_{31}	fb_{32}	-	ff_{34}	ff_{35}	ff_{36}
P4:FORM	fb_{41}	fb_{42}	fb_{43}	-	ff_{45}	ff_{46}
P5:PROG	fb_{51}	fb_{52}	fb_{53}	fb_{54}	-	ff_{56}
P6:EVAL	fb_{61}	fb_{62}	fb_{63}	fb_{64}	fb_{65}	-

Fig. 3 Matrix representation of the CCPS model. The figure depicts all phases of the CCPS model and transitions between them using a matrix representation. The rows *i* of the matrix describe the phases from which a transition is outgoing, while the columns *j* describe the phases towards which the transition is made (e.g., ff_{23} describes the transition from the phase USTD towards the phase IDEA). In this representation, feedforward transitions (i.e., transitions from a phase to one of the subsequent ones) are on the upper triangular matrix (green). Feedback transitions (i.e., transitions from one phase to one of the preceding ones) are on the lower triangular matrix (red). Self-transitions (i.e., the remaining in a phase) are not considered in this representation (dashes). The theoretically most efficient problem-solving cycle is highlighted in yellow (ff_{23}–ff_{34}–ff_{45}–ff_{56}–ff_{62})

Research question

As presented before, one common situation encountered in ER activities is the trial-and-error loop that in the CCPS model corresponds to an over-emphasis on feedforward and feedback transitions between PROG and EVAL phases. However, this condition might not be the most favorable for CT skill development, since the remaining phases (USTD, IDEA, and FORM) of the CCPS model are neglected. Consequently, this leads to the following research question: In a problem-solving activity involving educational robotics, how can the activation of all the CT processes related to the CCPS model be encouraged? In order to foster transitions towards all phases, the current study suggests to expressly generate an USTD-IDEA-FORM loop upstream so that students would not enter the PROG-EVAL loop without being able to leave it. To do so, a temporary blocking of the PROG phase (i.e., blocking the access to the programming interface) is proposed as an instructional intervention. Based on the findings of similar approaches implemented for inquiry-based learning (Bumbacher et al., 2018; Dillenbourg, 2013; Perez et al., 2017), the main idea is to introduce strategic pauses to the students to reinforce the three phases preceding the PROG phase. However, creating one loop to replace another is not a sustainable solution. With time, it is also important to adjust the instructional intervention into a "partial blocking," so that students can progressively advance in the problem-solving process. At a later stage, students should therefore be allowed to use the programming interface (i.e., enter the PROG phase); however, they should not be allowed to run their code on the robot, to prevent them from entering the trial-and-error loop between PROG and EVAL. Based on these instructional interventions, the current study aims at addressing the following research sub-questions:

- *Does a non-instructional approach for ER activities (i.e., unlimited access to the programming interface) promote a trial-and-error approach?*
- *Does a blocking of the programming interface foster cognitive processes related to problem understanding, idea generation, and solution formulation?*
- *Does a partial blocking (i.e., the possibility to use the programming interface without executing the code on the robot) help students to gradually advance in the problem-solving process?*

The resulting operational hypotheses are as follows:

- *Compared to the control group, the students subject to the blocking of the programming interface (total then partial) will activate all the CT processes of the CCPS model.*
- *Compared to the test group, the students not subject to the blocking of the programming interface will mostly activate the PROG-EVAL phases of the CT processes of the CCPS model.*

To test these hypotheses, an experiment using a test group and a control group was set up, with test groups that were subject to blocking of the programming interface and control groups that had free access to it.

Methods

The proposed CCPS model was evaluated in a research study with 29 primary school students (for details see "Participants" subsection). In groups of 2–3 students, the participants were asked to solve the robot lawnmower mission with the Thymio robot. In this activity, the robot has to be programmed in a way such that it drives autonomously around a lawn area, covering as much of the area as possible. Based on the CCPS model and the presented instructional interventions, two different experimental conditions were implemented for the activity and the students randomly and equally assigned to each condition. The activities of all groups were recorded on video, which subsequently were analyzed by two independent evaluators.

The robot lawnmower mission

The playground of the robot lawnmower mission consists of a fenced lawn area of 45cm × 45cm size (Fig. 4).

The fence is constructed using wood, and the lawn area is represented by eight squares of equal size with an imprinted lawn pattern. A ninth square is imprinted with a brick pattern and placed at the bottom right corner of the area, representing a garage, i.e., the starting point of the Thymio lawnmower robot. In this activity, the students have to program a lawnmower behavior, which autonomously drives the robot out of its garage and in the best case, makes it pass over all eight lawn squares while avoiding any collision with the fence. The interest of using the Thymio robot to carry out this mission is twofold: on the one hand, this robot has many sensors and actuators (Mondada et al., 2017; Riedo, Chevalier, Magnenat, & Mondada, 2013; Riedo, Rétornaz, Bergeron, Nyffeler, & Mondada, 2012). On the other hand, among the different programming languages that can be used with Thymio, one is the graphical language VPL (Shin, Siegwart, & Magnenat, 2014). The VPL platform (Fig. 5) represents parts of the robot's language by graphical icons that can be directly interpreted by human problem solvers, particularly facilitating transitions from FORM to PROG phases. Students can

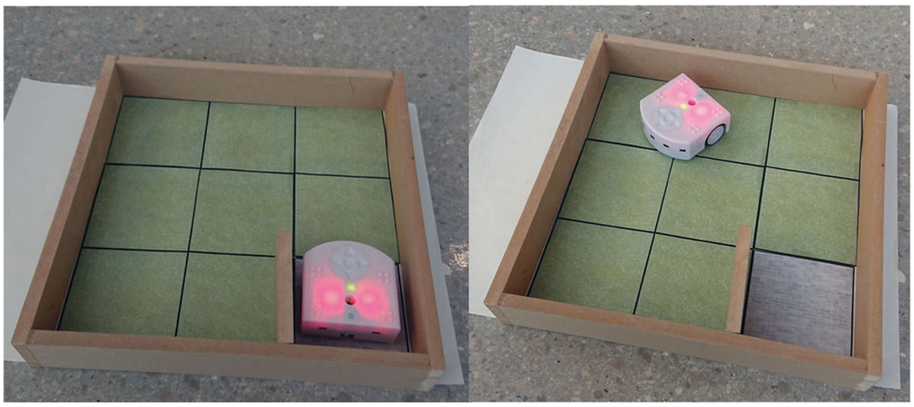


Fig. 4 Playground of the robot lawnmower mission with Thymio. The playground consists of a wooden fence surrounding an area (45 × 45cm) representing the lawn. One of the nine squares represents the garage, the starting position of the mission (left). The task is to program the Thymio robot so that it passes over all eight lawn squares while avoiding any collisions with the fence (right)

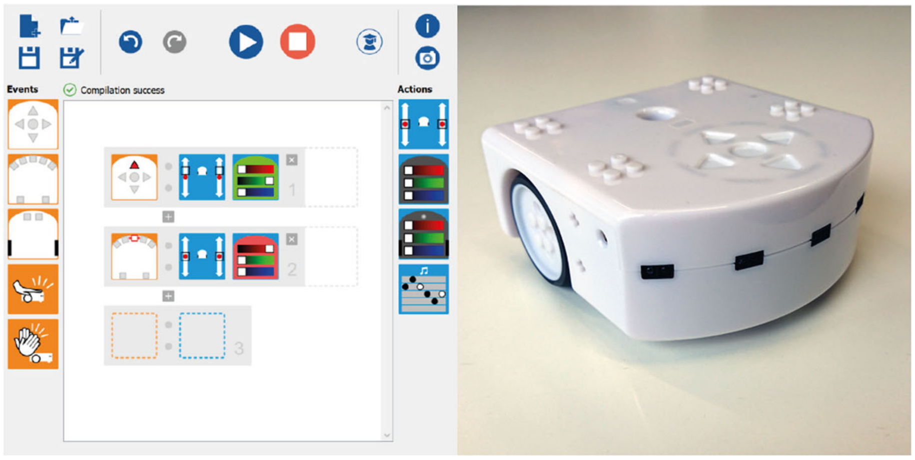


Fig. 5 Illustration of proximity between the VPL programming interface and the Thymio robot. The figure illustrates the iconic representation of programming commands in the VPL programming platform (left). The icons were designed to be as close as possible to the characteristics of the Thymio robot (right)

implement their solutions by simple drag-and-drop actions, without the need of extensive efforts on learning complex syntax beforehand. However, in contrast to sequential programming languages, the robot cannot simply be instructed to move a certain distance towards a given direction. Instead, in the event-based programming language VPL, students have to reflect on how to use the robot's sensors and actuators to generate a desired behavior. The openness and uncertainty of the task thus requires the students to leverage many competences related to computational thinking.

Participants

A total of 29 primary school students (13 girls and 16 boys between 9 and 10 years old) participated in an experimental study with the purpose of evaluating the proposed CCPS model. Prior to the study, all students have been introduced to the Thymio robot and the VPL programming interface through several school lessons (1-h per week for 12 weeks). The participation of the students in this study was approved by their guardians (parents) and class and school leaders (teachers and principal). A statement on ethics approval and consent was issued by The Coordination Committee for Educational Research in the Canton of Vaud (Switzerland).

Experimental procedure

At the beginning of the experimental session, all students were randomly assigned to groups of two or three. Each group of students was then randomly assigned to one of the two experimental conditions (test or control). The experimental procedures for the groups in each condition were different:

Control groups

The activity for the control groups started with a short introduction, where the goal and the rules of the mission were explained by one of the experimenters. The students were then given 40 min to implement their lawnmower robot. During the whole time period, they were allowed to use everything that was provided to them: the playground, the Thymio robot, and the VPL programming interface. No additional constraints were imposed.

Test groups

The experimental procedure for the test groups differed in the structure of the activity. Following the introductory speech, the activity started with 10 min of blocking of the programming interface. The students were given access to the playground and the Thymio robot, but they were not allowed to use the VPL programming platform. After this phase, the blocking was released, and the students were allowed to use everything for 10 min. This was followed by a partial blocking phase of 10 min, where the students had access to everything including the VPL platform, but they were not allowed to execute any code on the robot. For the last 10 min, the blocking was released again, and the students were allowed to use everything that was provided to them.

The study was conducted in two consecutive sessions of 45 min, one for each experimental condition. The test group (7 girls and 8 boys) started the mission first, while the control group (6 girls and 8 boys) went on a guided museum exhibition. After the completion of the first session, both groups switched. During each session, the five groups of the same experimental condition worked on the Thymio lawnmower mission simultaneously. Each group was provided a playground, a Thymio robot and a computer with the VPL platform installed. The sessions were supervised by two experimenters who provided technical support and addressed the students' questions regarding the task assignment. However, the experimenters did not provide any support regarding the solution of the lawnmower mission. Each group, as well as their interactions with the VPL platform and the playground, was recorded on video for later analysis.

Video analysis

Based on a socio-constructivist approach, this study relied upon in situ observations to capture the interactions of the students with each other as well as with the different cognitive artifacts (the robot, the interface, and the playground). The videos recorded from the experimental sessions were analyzed in several steps. Prior to individual analyses, two evaluators met to discuss and agree on appropriate observables (visual and verbal) indicating transitions towards the different phases of the CCPS model. Therefore, both evaluators first analyzed various prerecorded ER activities together. The videos were recorded from different kinds of ER activities and allowed to establish criterion standards (Sharpe & Koperwas, 2003) that are not limited to one specific ER activity. The whole procedure was aimed at streamlining the way both evaluators would perform their individual analyses. Subsequently, both evaluators performed the behavioral analysis independently, sequentially mapping the behaviors of the students during the robot lawnmower activity to the different phases of the CCPS model. The mappings were made under the assumption that a student can only be in one of the six phases of the CCPS model at a time. Each evaluator performed the mapping based on their interpretation of the behavior of the students, while considering the criterion standards that have been established beforehand. Transitions to the first three phases of the CCPS model were mainly mapped based on the students' verbalizations, such as "How can we do that?" (USTD), "Ah, I have an idea!" (IDEA) or "If this sensor detects the wall, the robot turns left" (FORM). In contrast, transitions to the last two phases were mostly based on visual observations (e.g., a student starting to use the computer (PROG) or a student watching the Thymio robot after executing the program (EVAL)). Students who were clearly not involved in the activity were mapped to the off-task behavior phase (OFFT). Two state graphs were created for each student (one by each evaluator) using a software dedicated to the creation of activity chronicles such as Actograph (SymAlgo Technologies, Paris, France) and a numerical computing tool such as Matlab (MathWorks, Natick, Massachusetts, USA). Following this step, both evaluators compared their state graphs against each other and discussed any major discrepancies between their evaluations. Major discrepancies were considered segments in the state graphs in which both evaluators did not agree on the same behavior for more than 1 min. The corresponding video scene was reviewed by both evaluators together to achieve a mutual decision. Based on this decision, the state graphs of the evaluators were modified accordingly. Subsequently, the continuous state graphs of both evaluators were discretized into equally spaced time segments of one second. Finally, Cohen's Kappa was computed for the discretized pair of state graphs of each student, in order to validate the inter-rater reliability of the performed video analyses. Therefore, confusion matrices were created for the observations made by both researchers. Agreement between both evaluators was quantified by the number of times both evaluators agreed on mapping the same phase of the CCPS model to a student's behavior. The Kappa values were then calculated for the observations made for each student, using the formula presented in (Bakeman & Gottman, 1997) and taking into account the proportion of agreement observed and the proportion expected by chance. The range of the values for Cohen's Kappa was $0.59 < k < 0.84$ (Fig. 6), which according to the literature (Landis & Koch, 1977) can be interpreted as a substantial agreement between both evaluators. Finally, the state graphs

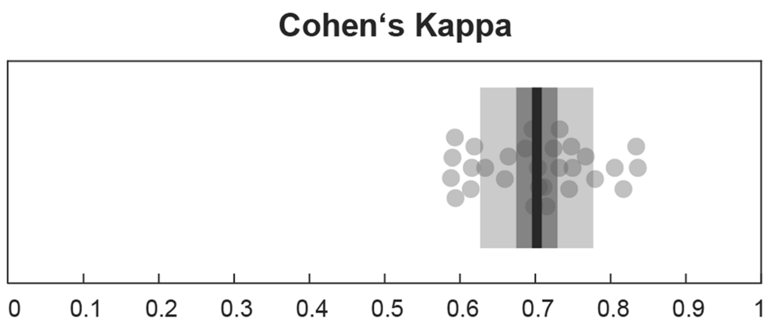


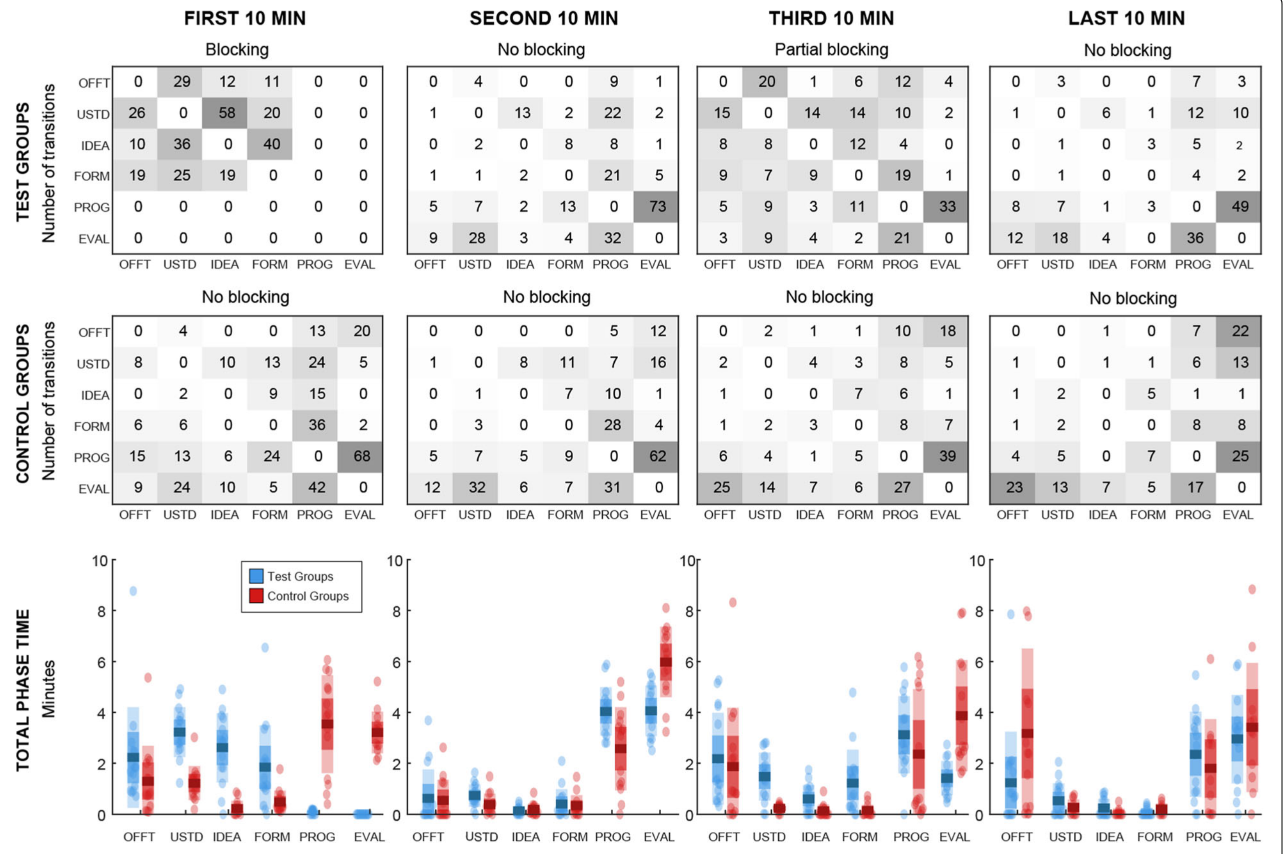
Fig. 6 Cohen's Kappa values for two independent evaluators. Dots indicate Cohen's Kappa values calculated for each student based on the independent observations of two evaluators. The dark horizontal line indicates the mean value, dark gray areas one standard deviation, and light gray areas the 95% confidence intervals

created by the first evaluator were used to perform further analyses. Based on these state graphs, the time spent in each state of the CCPS model was computed for each student, as well as the total number of transitions made between different phases. Overall, 2162 phase transitions were mapped for the total of 400 min of recordings: 1072 transitions for the test groups and 1090 for the control groups.

Results

Transition matrices were created for each student, illustrating the changes from one state of the CCPS model to another during each quarter of the activity (first, second, third, and last 10 min). All transitions made by the students in each condition (test and control) were then summed up to analyze the overall dynamics of the two experimental groups (Fig. 7). Moreover, the total time spent in each phase was analyzed for both groups during each quarter of the activity

The transition matrices for the control groups showed similar dynamics for all quarters of the activity. Transitions were found on the upper and lower triangular part of the matrices, highlighting the occurrences of both feedforward and feedback transitions. Most occurrences were found for transitions between PROG and EVAL phases. In contrast, transitions from and towards USTD, IDEA, FORM, and OFFT phases appeared to be less frequent. When looking at the total time spent in each phase, a similar trend was observed: especially in the first three quarters of the activity, students of the control group predominantly spent their time in PROG and EVAL phases (on average 22 out of 30 min), while USTD, IDEA, FORM, and OFFT phases were observed less frequently (8 out of 30 min). In the last quarter of the activity, PROG and EVAL remained more prevalent compared to USTD, IDEA, and FORM phases; however, a similar amount of time was now also spent on off-task behavior (OFFT, 3 out of 10 min).



TEST GROUPS — FIRST 10 MIN — Blocking

	OFFT	USTD	IDEA	FORM	PROG	EVAL
OFFT	0	29	12	11	0	0
USTD	26	0	58	20	0	0
IDEA	10	36	0	40	0	0
FORM	19	25	19	0	0	0
PROG	0	0	0	0	0	0
EVAL	0	0	0	0	0	0

TEST GROUPS — SECOND 10 MIN — No blocking

	OFFT	USTD	IDEA	FORM	PROG	EVAL
OFFT	0	4	0	0	9	1
USTD	1	0	13	2	22	2
IDEA	0	2	0	8	8	1
FORM	1	1	2	0	21	5
PROG	5	7	2	13	0	73
EVAL	9	28	3	4	32	0

TEST GROUPS — THIRD 10 MIN — Partial blocking

	OFFT	USTD	IDEA	FORM	PROG	EVAL
OFFT	0	20	1	6	12	4
USTD	15	0	14	14	10	2
IDEA	8	8	0	12	4	0
FORM	9	7	9	0	19	1
PROG	5	9	3	11	0	33
EVAL	3	9	4	2	21	0

TEST GROUPS — LAST 10 MIN — No blocking

	OFFT	USTD	IDEA	FORM	PROG	EVAL
OFFT	0	3	0	0	7	3
USTD	1	0	6	1	12	10
IDEA	0	1	0	3	5	2
FORM	0	1	0	0	4	2
PROG	8	7	1	3	0	49
EVAL	12	18	4	0	36	0

CONTROL GROUPS — FIRST 10 MIN — No blocking

	OFFT	USTD	IDEA	FORM	PROG	EVAL
OFFT	0	4	0	0	13	20
USTD	8	0	10	13	24	5
IDEA	0	2	0	9	15	0
FORM	6	6	0	0	36	2
PROG	15	13	6	24	0	68
EVAL	9	24	10	5	42	0

CONTROL GROUPS — SECOND 10 MIN — No blocking

	OFFT	USTD	IDEA	FORM	PROG	EVAL
OFFT	0	0	0	0	5	12
USTD	1	0	8	11	7	16
IDEA	0	1	0	7	10	1
FORM	0	3	0	0	28	4
PROG	5	7	5	9	0	62
EVAL	12	32	6	7	31	0

CONTROL GROUPS — THIRD 10 MIN — No blocking

	OFFT	USTD	IDEA	FORM	PROG	EVAL
OFFT	0	2	1	1	10	18
USTD	2	0	4	3	8	5
IDEA	1	0	0	7	6	1
FORM	1	2	3	0	8	7
PROG	6	4	1	5	0	39
EVAL	25	14	7	6	27	0

CONTROL GROUPS — LAST 10 MIN — No blocking

	OFFT	USTD	IDEA	FORM	PROG	EVAL
OFFT	0	0	1	0	7	22
USTD	1	0	1	1	6	13
IDEA	1	2	0	5	1	1
FORM	1	2	0	0	8	8
PROG	4	5	0	7	0	25
EVAL	23	13	7	5	17	0

Fig. 7 Transition matrices and total phase times. The top rows show the transition matrices for the test (first row) and control groups (second row) for the first, second, third, and last 10 min of the activity. The entries in the matrices denote the total number of transitions made between the different phases for each group. Transitions with higher occurrences are highlighted with darker colors. The last row shows the total time spent in each phase by both groups for the four quarters of the activity. Colored dots show data points for each student of the test (blue) and control (red) groups. Dark horizontal lines indicate the mean values, dark-colored areas one standard deviation, and light-colored areas the 95% confidence intervals

Also, in test groups, both feedforward and feedback transitions were observed; however, the dynamics varied during the different quarters of the activity. The behavior in the second and last quarter (no blocking conditions) was very similar to the behavior of the control groups: the great majority of transitions was observed between PROG and EVAL phases, while transitions to and from other phases were comparatively lower. However, when looking at the transition matrices for the first and third quarter of the activity, remarkable differences were found. Due to the blocking of the VPL programming interface in the first quarter, students were not able to enter any of the PROG or EVAL phases and were thus forced to shift their attention towards the remaining phases. For the third quarter, on the other hand, a more even distribution among all phases was found. Since the partial blocking condition allowed the students to work with the VPL platform (without the possibility to send the program to the robot), transitions to PROG phases could be observed. Moreover, since students have already programmed their robot during the previous quarter of the activity, they were also able to make transitions to EVAL phases.

Interestingly, there was a high number of transitions between PROG and EVAL phases, indicating a rigorous debugging of the students' previous implementation, since new implementations could not be executed and tested (i.e., trial-and-error was not possible). The blocking conditions also influenced the total time the students spent in each of the phases. Compared to the control group, there was a more even distribution among the phases for the first three quarters of the activity. On average, students spent 13 out of 30 min in PROG and EVAL phases and 12 out of 30 min in USTD, IDEA, and FORM phases. During the first three quarters, the times spent on off-task behavior (OFFT) by the test groups were very similar to the ones by the control groups. However, in contrast to the control groups, off-task behavior also remained on a similar level in the last quarter of the activity.

In order to further investigate the effect of the initial blocking condition, transition graphs were generated for the first 10 min of the activity (Fig. 8).

These graphs depict the transition probability from one phase to another for both groups as well as the total time spent in each phase. Moreover, the initial transition for each student was determined, i.e., the first phase they entered when the activity started. In the test groups, all fifteen students started the activity with the USTD phase, corresponding to the start of the theoretically most efficient cycle of the CCPS model (see Fig. 2). In the control groups, this behavior was not observed for all students. Although the majority started with the USTD phase, three of the fourteen students entered the activity by directly going to the PROG phase.

Moreover, when comparing the transition probabilities between the phases, remarkable differences were found for both groups: the results showed that the transition graph for the test groups matched well with the first part of the theoretically most effective cycle in the CCPS model. For these groups, the blocking condition hindered any transition from and towards PROG and EVAL phases. Starting from the USTD phase, students would therefore most likely continue with IDEA, then FORM phases, and then eventually return to USTD for another iteration. Although other feedforward and feedback

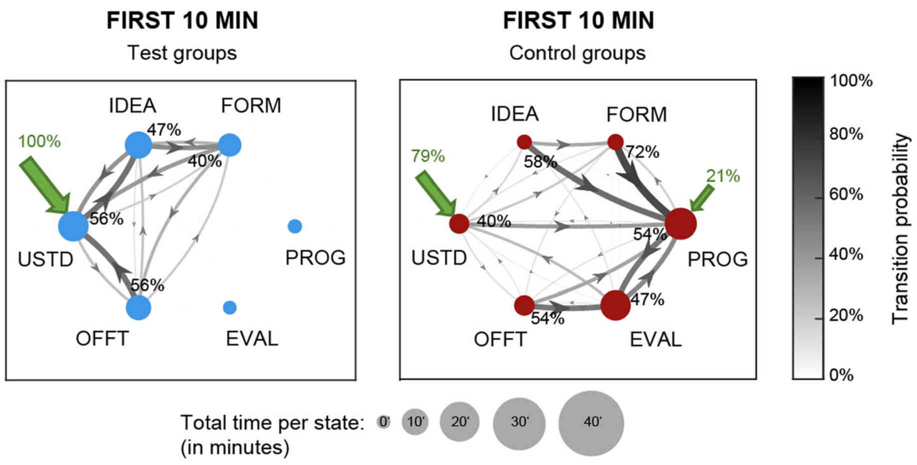


Fig. 8 Initial transition graphs for test and control groups. The figure shows the transitions graphs for both groups for the first 10 min of the activity. The green arrows indicate the probabilities for the first transitions of the activity. The size of the dots, representing the six phases of the CCPS model, is proportional to the amount of time spent by the groups in each phase. The gray arrows represent the transition probabilities between phases. Higher transition probabilities are represented by thicker and darker lines. The value for the most probable outgoing transition for each phase is given next to the corresponding arrow

transitions were observed, they appeared to be less likely. If students showed off-task behavior, they would most likely return to the activity through the USTD phase.

The total time spent in each of the four phases was evenly distributed. For the control groups on the other hand, no blocking conditions were imposed. From the transition graph, it can be seen that the activities of the control groups were more centered around PROG and EVAL phases. Once the students would enter the PROG phase, the most likely transition was towards the EVAL phase and vice versa. Although transitions towards other phases were observed, the probability of leaving this PROG-EVAL loop was comparatively low. Moreover, the effect of this loop was reinforced by the fact that most transitions from USTD, IDEA, FORM, and OFFT phases were directed towards PROG or EVAL phases, resulting in an uneven distribution of the time spent in each phase: during the first 10 min of the activity, the students of the control groups spent almost 7 min in PROG and EVAL phases and less than 2 min on USTD, IDEA, and FORM phases.

In order to illustrate the dynamics at individual levels, the state graphs, transition matrices, and transition graphs for one exemplary student of each group are presented (Fig. 9).

The data is shown for the whole 40 min of the activity. It can be observed that the student from the control group immediately started the activity by jumping into the PROG phase. Throughout the activity, the student spent most of the time only in PROG and EVAL phases, sporadically transitioning to one of the other phases that were then followed by transitions back to the PROG-EVAL loop. The inclination towards these phases is highlighted in the corresponding transition graph, which clearly demonstrates that this student strongly neglected the preceding USTD, IDEA, and FORM phases. The student from the test group on the other hand showed a more balanced distribution among the five phases of the CCPS model. Indeed, the transition matrix of this student showed a more even dispersion for the transitions towards different phases. Interestingly, a high number of transitions were found for the path USTD–IDEA–FORM–PROG–EVAL–USTD, indicating an inclination towards the theoretically most efficient cycle of the CCPS model. From the state graphs, it can also be observed that the student from the test group performed more playground interactions (11 times), i.e., interactions with the robot or the lawn area, compared to the student from the control group (2 times). A similar result was observed when analyzing the overall data for playground interactions of each experimental group (in total 93 interactions for the test groups and 59 interactions for the control groups).

Finally, the performance of each group's lawnmower was quantified by the highest number of lawn squares that the robot managed to cover without collision. The results showed that three groups (two tests and one control) managed to complete the task, covering all eight lawn squares with their lawnmower robot. Five groups (three tests and two controls) covered six squares and two groups (both control) covered only 4 squares. The number of squares was only quantified for trajectories that started from the garage and that were not random.

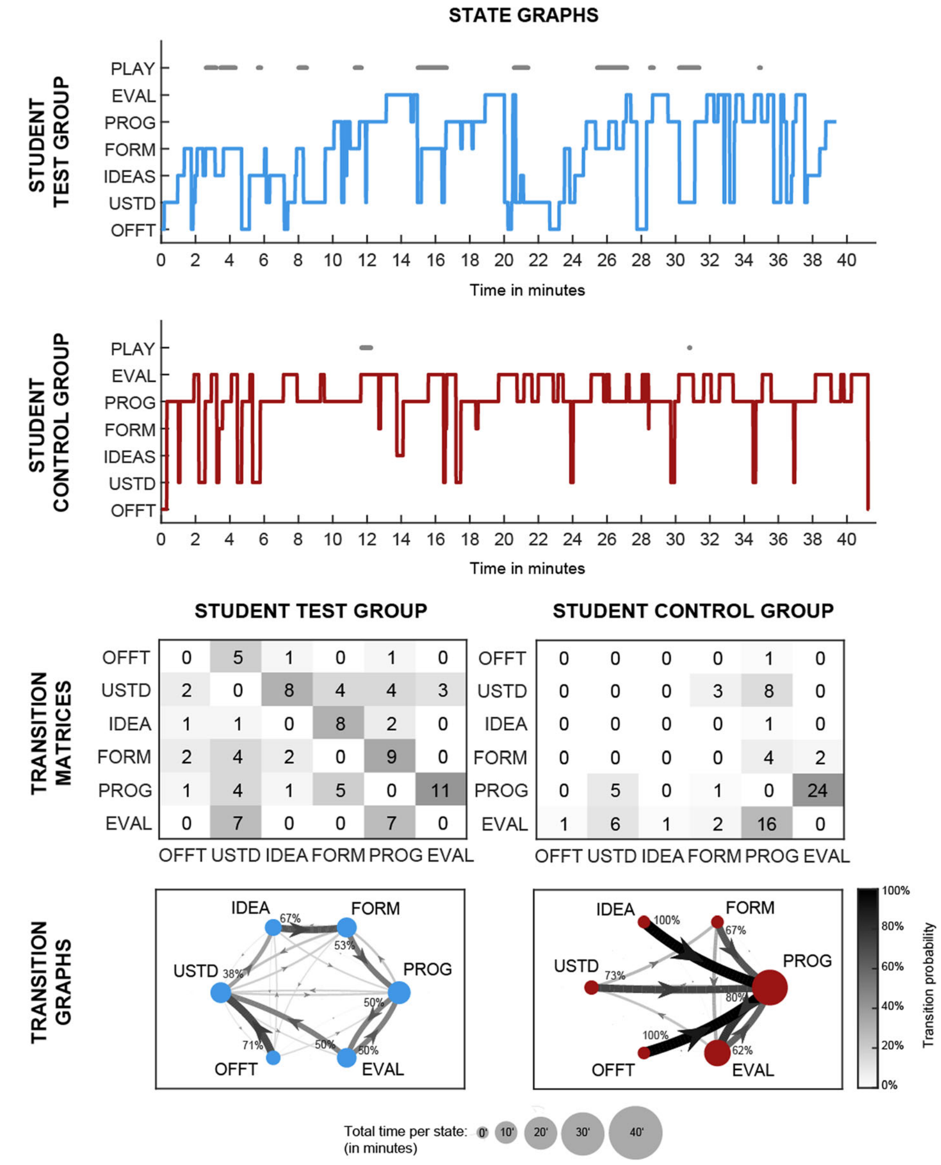
Discussion

The effect of non-instructional approaches for ER activities

Usually in educational robotics activities, students work in pairs or groups to solve one or more problems, especially when these activities are aimed at the development of computational thinking skills students are faced with open-ended problems that they have to solve in collaboration. By doing so, they benefit from the "dynamic feedback inherent in dialog and the creation of cognitive conflict" (Hoyles, 1985). In many cases, teachers let the students work in the project without any particular constraints (Buss & Gamboa, 2017; Sadik et al., 2017). In the present study, the control groups were left in this situation which corresponds to a non-instructional approach for ER activities in classrooms. However, the results of this study showed that under these circumstances, students spent most of their time in phases related to programming and evaluating. It was observed that, once entered in this loop, students would hardly change their strategies and barely work on any of the other phases presented in the CCPS model. This result suggests an answer to the first research question addressed in this study:

Does a non-instructional approach for ER activities (i.e., unlimited access to the programming interface) promote a trial-and-error approach?

Indeed, the students from the control groups spent on average almost two-thirds of their time in programming and evaluating which does not leave much time to develop other skills (understanding the problem, generating ideas, formulating a behavior). This large amount of time spent is thus a clue showing that students were plunged in a trial-and-error loop. Moreover, the results showed that the probability of leaving this PROG-EVAL loop was low and that it proves a lack of organization in the strategy: students should go from trial-and-error to systematic testing providing "evidence of problem decomposition" (Shute et al., 2017). In the current study, the population had the same background and knowledge: at the age of 9 to 10 years, such a behavior is usual while the



STUDENT TEST GROUP

	OFFT	USTD	IDEA	FORM	PROG	EVAL
OFFT	0	5	1	0	1	0
USTD	2	0	8	4	4	3
IDEA	1	1	0	8	2	0
FORM	2	4	2	0	9	0
PROG	1	4	1	5	0	11
EVAL	0	7	0	0	7	0

STUDENT CONTROL GROUP

	OFFT	USTD	IDEA	FORM	PROG	EVAL
OFFT	0	0	0	0	1	0
USTD	0	0	0	3	8	0
IDEA	0	0	0	0	1	0
FORM	0	0	0	0	4	2
PROG	0	5	0	1	0	24
EVAL	1	6	1	2	16	0

Fig. 9 State graphs, transition matrices, and transition graphs for two example students. The figure shows the data for two students, each exemplifying what was observed in the test and control groups, respectively. The top rows show the complete state graphs for the student from the test (first row) and the control group (second row), displaying in which phase each student was at each moment of the activity. Moreover, the students' interaction with the playground is highlighted (PLAY). The third row shows the transition matrices for both students for the whole 40 min of the activity. The entries in the matrices denote the total number of transitions made between the different phases for each student. The last row shows the transitions graphs for both students for the whole 40 min of the activity

students are just in the process of building proper problem-solving strategies. Indeed, as soon as the students from the test groups had access to the computer, they also spent a lot of time in the PROG-EVAL phases. Based on these results, the main conclusion is that in non-instructional classroom approaches where the teacher does not intervene in the instructional design of ER activities and does not put constraints on the students, the latter stays most of the time in a PROG-EVAL loop.

The effect of blocking and partially blocking the programming interface

In this study, the test groups had undergone an instructional intervention while they had to solve a problem involving programming. Indeed, in order to prevent them from immediately entering the PROG-EVAL loop, a blocking of the programming interface was imposed in order to require them to shift their attention to the other phases of the CCPS model. The second intervention condition that was tested was equally verified by this experimental study:

Does a blocking of the programming interface foster cognitive processes related to problem understanding, idea generation, and solution formulation?

Indeed, in the test groups, students were forced to shift their attention towards other phases since their possibilities to enter PROG and EVAL phases were limited by the given constraints. It was observed that given the same amount of time, the test groups better distributed their cognitive efforts in total (measured by a similar amount of time spent in USTD-IDEA-FORM compared to PROG-EVAL). Although no specific instructions were given to the students, their behavior tended to converge towards the theoretically most efficient cycle of the CCPS model. Students started the activity by trying to understand the problem, and they then generated ideas and subsequently suggested formulations of the behavior of the robot. These iterations can be explained by the feedback inherent to dialog which occurs during collaborative situations (Hoyles, 1985). Whereas it is not an unusual behavior to go directly for programming (Buss & Gamboa, 2017; Sadik et al., 2017), no results were found a priori in state-of-the-art literature considering the behavior observed under a blocking condition of the programming interface. The results of this study therefore raise the question of how this blocking condition effectively influences the learning outcomes. It is assumed that the fact that students perform more transitions towards USTD, IDEA, and FORM phases would help them on the development and reflection of their problem-solving process. As shown in previous work on inquiry-based learning (Bumbacher et al., 2018; Dillenbourg, 2013; Perez et al., 2017), introducing these kinds of strategic pauses may help students to better reflect prior to taking actions. Applying this principle to ER activities could substantially enhance the learning outcomes, especially with regard to the development of CT skills. Indeed, in the present study, students from the test group iterated the USTD-IDEA-FORM loop (performing both feedforward and feedback transitions between those phases) in the first 10 min of the activity, arguing and anticipating what could happen afterwards. This cognitive state in which they dived into seems to allow students to distance themselves from the programming act to better reflect on the "creative act" (Duchamp, 1967).

Another finding related to the effect of this instructional intervention is that test groups seemed to interact more with the playground and the robot than the control groups. As the latter favored a PROG-EVAL loop, they were more likely to be immersed in the programming interface. In contrast, since the test groups did not have access to the computers at the beginning, they appeared to be more inclined towards using the playground and the robot as means to express their thoughts and findings. This mediation is a key element on which it is then possible to intervene. In fact, this is what happened when the experimental condition was altered in a partial blocking at the beginning of the second half of the experiment. The findings from the study allowed verification of the third intervention hypothesis:

Does a partial blocking (i.e., the possibility to use the programming interface without executing the code on the robot) help students to gradually advance in the problem-solving process?

The transition from a full blocking of the programming interface to a partial blocking can be considered as a way to provide scaffolding. Thanks to this scaffolding, during half of the time, the students were able to build a well-settled strategy to solve the problem and they were therefore mostly able to iterate the theoretically most efficient cycle of the CCPS model (USTD–IDEA–FORM–PROG–EVAL–USTD). Moreover, during this partial blocking, the high number of transitions between PROG and EVAL phases suggests a rigorous debugging of previous implementations, an element which has been considered important for the development of CT competencies (Bers et al., 2014; Shute et al., 2017). The students iteratively worked on the commonly identified issues and still had to predict possible behaviors because the partial blocking prevented them from the execution of the new program code. This condition could be considered beneficial to help students develop skills related to CT. For instance, among the test groups, during the partial blocking, two students decided to set up a writing strategy for programming the robot. While they were told that they could use the programming interface without executing their program, they decided to keep their paper strategy arguing that "it's the same." This example shows the effect of fading from a blocking to the partial blocking, and these experimental conditions can therefore be considered an interesting scaffolding tool for teachers.

The overall performances of the implemented lawnmower robots illustrated higher task completion rates for the test groups. All five test groups managed to cover at least six lawn squares and two of them completed the mission covering all eight squares. In the control groups instead, there were two groups who did not manage to cover more than four squares and two groups not more than six. Although one group managed to finish the mission using a pure trial-and-error approach, the effective CT skills development for this group might be questionable. Indeed, previous work has argued that trial-and-error strategies may not be considered optimal, since they may "support task completion but not skills development" (Antle, 2013).

Off-task behavior as part of the activity

While designing the CCPS model, it was initially not obvious to include off-task as a separate phase of the model. However, while observing the students during the experiment, it appeared that the off-task behavior (OFFT) was indeed part of the reality in classrooms. Consequently, it was decided to include it as an additional phase of the model. It appeared that the dropout from the activity was a residue that was found in both the test and control groups. However, the distribution of this residue was not equivalent between the two types of groups. In contrast to the test groups, off-task behavior increased significantly during the last quarter of the activity (after 30 min) for control groups. It seems that the working modalities in the test groups (i.e., blocking and partial blocking) may foster engagement in the task in a longer term, compared to the unconstrained modality for the control groups. As described, the scaffolding in the access to programming allows students to have a progression over time. This may facilitate their immersion in the activity and thus results in more effective learning time. In this sense, it seems that the implementation of the blocking conditions can also minimize off-task behavior in classroom situations, which could possibly lead to more efficient learning activities.

Conclusion

The findings reported in this article have provided empirical evidence that (i) a non-instructional approach for educational robotics activities (i.e., unlimited access to the programming interface) can promote a trial-and-error behavior; (ii) a scheduled blocking of the programming interface can foster cognitive processes related to problem understanding, idea generation, and solution formulation; (iii) a progressive adjustment of the blocking of the programming interface can help students in building a well-settled strategy to approach educational robotics problems and therefore may represent an effective way to provide instructional scaffolding. Taking these findings into account, this study provides initial evidence on the need for specific instructional interventions on ER activities and illustrates how teachers could use the proposed model to design ER activities aimed at CT skill development. The findings of this study thus allow to make a transition from theoretical to more operational frameworks as recommended by Ioannou and Makridou (2018). The CCPS model is indeed inspired by existing CT models (Romero et al., 2017; Shute et al., 2017), but it makes a distinct contribution regarding the transfer to the classroom by providing teachers with explicit guidance on the implementation, as previously recommended by Atmatzidou and Demetriadis (2016). Indeed, this study offers to teachers and researchers a conceptualization of five cognitive states (USTD IDEA−FORM−PROG−EVAL) which is adapted to ER activities and K-5 students. In the present work, the main pedagogical lever that has been manipulated was the blocking of the programming interface. This intervention proved to be an effective way to help students cover a more complete spectrum of CT competencies, in contrast to a non-instructional modality, in which they mainly focus on their programming skills. As a matter of fact, this intervention can be easily implemented by teachers regardless of the type of robot used. Consequently, this study also addresses the lack of research on CT for K-5 classrooms, particularly grades 3 and 4, i.e., students of age between 8 and 10 years old. However, the presented findings are not limited to this age range and may possibly be extended to younger and older students. Finally, the establishment of this model and especially of its mechanics could appear as a step forward in the implementation of the CT in the classroom through ER activities.

Limitations and future work

Although the results of this study appear to be promising, further studies are needed to draw more substantial conclusions. Due to school regulations, access to classrooms is limited for research purposes, hence in this study, the experiments were conducted with a small sample size. Nevertheless, considering the 2162 mapped transitions, this size could be considered sufficient for the purpose of this research, which is namely to verify the present model. However, as the main goal of the CCPS model is to support teachers in the design, implementation, and evaluation of ER activities, future work should investigate whether teachers really perceive an added value of the model for their teaching activities. Moreover, other intervention hypotheses could be explored and tested, in order to demonstrate more exhaustive validity of the model. Furthermore, in order to present evidence for the effectiveness as a reference model for ER activities, future longitudinal studies should investigate the effective learning gains evoked by the interventions proposed by the model. In this regard, the findings of the present study may provide a good starting point for the design and executions of such studies.

Acknowledgements

The authors would like to thank the Coordination Committee for Educational Research of the Canton of Vaud (Switzerland) as well as the school administration and all teachers, students, and their parents for the participation in the study. Moreover, the authors would like to thank Melissa Skweres for proofreading the manuscript.

Authors' contributions

M.C. designed the model, carried out experiments, analyzed data, and wrote the paper; C.G. designed the model, carried out experiments, analyzed data, and wrote the paper; A.P. designed the model, carried out experiments, and wrote the paper; F.M. designed the model and wrote the paper. The author(s) read and approved the final manuscript.

Funding
This work was partially supported by the Swiss National Science Foundation NCCR Robotics.

Availability of data and materials
The data sets generated and analyzed during the current study are not publicly available due to the sensitivity of the data of the under-age participants but are available from the corresponding author on reasonable request.

Competing interests
The authors declare that they have no competing interests.

Author details
[1]Haute Ecole Pédagogique (HEP) du Canton de Vaud, Avenue de Cour, 33, 1014 Lausanne, Switzerland. [2]Mobots Group of the Biorobotics Laboratory, Ecole Polytechnique Fédérale de Lausanne (EPFL), Route Cantonale, 1015 Lausanne, Switzerland. [3]Department of Education and Learning (DFA), University of Applied Sciences and Arts of Southern Switzerland (SUPSI), Piazza S. Francesco 19, 6600 Locarno, Switzerland.

Received: 18 March 2020 Accepted: 6 July 2020
Published online: 03 August 2020

References

Antle, A. N. (2013). Exploring how children use their hands to think: An embodied interactional analysis. *Behaviour & Information Technology*, *32*(9), 938–954.
Atmatzidou, S., & Demetriadis, S. (2016). Advancing students' computational thinking skills through educational robotics: A study on age and gender relevant differences. *Robotics and Autonomous Systems*, *75*, 661–670.
Bakeman, R., & Gottman, J. M. (1997). *Observing interaction: An introduction to sequential analysis*, (2nd ed.,). New York: Cambridge University Press.
Barr, V., & Stephenson, C. (2011). Bringing computational thinking to k-12: What is involved and what is the role of the computer science education community? *Inroads*, *2*(1), 48–54.
Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers & Education*, *72*, 145–157.
Bottino, R., & Chioccariello, A. (2014). Computational thinking: Videogames, educational robotics, and other powerful ideas to think with. In T. Brinda, N. Reynolds, R. Romeike, & A. Schwill (Eds.), *Key Competencies in Informatics and ICT (KEYCIT), 7*, (pp. 301–309). Potsdam: Universitätsverlag Potsdam Available at http://nbn-resolving.de/urn:nbn:de:kobv:517-opus4-70325. Accessed 6 June 2020.
Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *Paper presented at the annual meeting of the American Educational Research Association (AERA)*. Vancouver: Available at https://web.media.mit.edu/~kbrennan/files/Brennan_Resnick_AERA2012_CT.pdf. Accessed 6 June 2020.
Bumbacher, E., Salehi, S., Wieman, C., & Blikstein, P. (2018). Tools for science inquiry learning: Tool affordances, experimentation strategies, and conceptual understanding. *Journal of Science Education and Technology*, *27*(3), 215–235.
Buss, A., & Gamboa, R. (2017). Teacher transformations in developing computational thinking: Gaming and robotics use in after-school settings. In P. Rich, & C. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking*, (pp. 189–203). Cham: Springer. https://doi.org/10.1007/978-3-319-52691-1.
Catlin, D., & Woollard, J. (2014). Educational robots and computational thinking. In M. Merdan, W. Lepuschitz, G. Koppensteiner, R. Balogh, & D. Obdrzalek (Eds.), *Robotics in Education : current research and innovations*, (pp. 144–151). Cham: Springer. https://doi.org/10.1007/978-3-030-26945-6.
Chalmers, C. (2018). Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction*, *17*, 93–100.
Denis, B., & Hubert, S. (2001). Collaborative learning in an educational robotics environment. *Computers in Human Behavior*, *17*(5-6), 465–480.
DeSchryver, M. D., & Yadav, A. (2015). Creative and computational thinking in the context of new literacies: Working with teachers to scaffold complex technology-mediated approaches to teaching and learning. Journal of Technology and Teacher Education, 23(3), 411–431. Waynesville: Society for Information Technology & Teacher Education. Available at https://www.learntechlib.org/primary/p/151572/. Accessed 6 June 2020.
Dierbach, C. (2012). *Introduction to computer science using python: A computational problem-solving focus*. Hoboken: Wiley Publishing.
Dillenbourg, P. (2013). Design for classroom orchestration. *Computers & Education*, *69*, 485–492.
Duchamp, M. (1967). *The creative act [audio recording]*. New York: Roaring Fork Press Available at https://www.youtube.com/watch?v=lqLSZdX0IbQ (min.4: 20). Accessed 6 June 2020.
Eguchi, A. (2014). Robotics as a learning tool for educational transformation. In D. Alimisis, G. Granosik, & M. Moro (Eds.), *4th International Workshop Teaching Robotics, Teaching with Robotics & 5th International Conference Robotics in Education*, (pp. 27–34). Padova: RIE ISBN 978-88-95872-06-3. Available at http://www.terecop.eu/TRTWR-RIE2014/files/00_WFr1/00_WFr1_04.pdf. Accessed 6 June 2020.
Eguchi, A. (2016). Computational thinking with educational robotics. In G. Chamblee, & L. Langub (Eds.), *Proceedings of society for information technology & teacher education international conference*, (pp. 79–84). Savannah: Association for the Advancement of Computing in Education (AACE) Available at https://www.learntechlib.org/p/172306. Accessed 6 June 2020.
Giang, C., Chevalier, M., Negrini, L., Peleg, R., Bonnet, E., Piatti, A., & Mondada, F. (2019). Exploring escape games as a teaching tool in educational robotics. *Educational Robotics in the Context of the Maker Movement*, *946*, 95.
Giang, C., Piatti, A., & Mondada, F. (2019). Heuristics for the development and evaluation of educational robotics systems. *IEEE Transactions on Education*.
Giannakoulas, A., & Xinogalos, S. (2018). A pilot study on the effectiveness and acceptance of an educational game for teaching programming concepts to primary school students. *Education and Information Technologies*, *23*(5), 2029–2052.
Haseski, H. I., Ilic, U., & Tugtekin, U. (2018). Defining a new 21st century skill-computational thinking: Concepts and trends. *International Education Studies*, *11*(4), 29–42.
Hoyles, C. (1985). What is the point of group discussion in mathematics? *Educational studies in mathematics*, *16*(2), 205–214.
Hsu, T.-C., Chang, S.-C., & Hung, Y.-T. (2018). How to learn and how to teach computational thinking: Suggestions based on a review of the literature. *Computers & Education*, *126*, 296–310.
Ioannidou, A., Bennett, V., Repenning, A., Koh, K. H., & Basawapatna, A. (2011). Computational thinking patterns. In *Paper presented at the annual meeting of the American Educational Research Association (AERA)*. New Orleans: Available at https://files.eric.ed.gov/fulltext/ED520742.pdf. Accessed 6 June 2020.
Ioannou, A., & Makridou, E. (2018). Exploring the potentials of educational robotics in the development of computational thinking: A summary of current research and practical proposal for future work. *Education and Information Technologies*, *23*(6), 2531–2544.
Israel, M., Pearson, J. N., Tapia, T., Wherfel, Q. M., & Reese, G. (2015). Supporting all learners in school-wide computational thinking: A cross-case qualitative analysis. *Computers & Education*, *82*, 263–279.
Jung, S. E., & Won, E.-s. (2018). Systematic review of research trends in robotics education for young children. *Sustainability*, *10*(4), 905.
Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2011). Understanding computational thinking before programming: Developing guidelines for the design of games to learn introductory programming through game-play. *International Journal of Game-Based Learning (IJGBL)*, *1*(3), 30–52.
Kazimoglu, C., Kiernan, M., Bacon, L., & MacKinnon, L. (2012). Learning programming at the computational thinking level via digital game-play. *Procedia Computer Science*, *9*, 522–531.
Landis, J. R., & Koch, G. G. (1977). The measurement of observer agreement for categorical data. *Biometrics*, *33*, 159–174.
Leonard, J., Buss, A., Gamboa, R., Mitchell, M., Fashola, O. S., Hubert, T., & Almughyirah, S. (2016). Using robotics and game design to enhance children's self-efficacy, stem attitudes, and computational thinking skills. *Journal of Science Education and Technology*, *25*(6), 860–876.
Li, Y., Schoenfeld, A. H., diSessa, A. A., Graesser, A. C., Benson, L. C., English, L. D., & Duschl, R. A. (2020). Computational thinking is more about thinking than computing. *Journal for STEM Education Research*, *3*, 1–18. https://doi.org/10.1007/s41979-020-00030-2.
Lumsdaine, E., & Lumsdaine, M. (1994). Creative problem solving. *IEEE Potentials*, *13*(5), 4–9.
Miller, D. P., & Nourbakhsh, I. (2016). Robotics for education. In B. Siciliano, & O. Khatib (Eds.), *Handbook of robotics*, (2nd ed., pp. 2115–2134). Cham: Springer. https://doi.org/10.1007/978-3-319-32552-1_79.

Mondada, F., Bonani, M., Riedo, F., Briod, M., Pereyre, L., Rétornaz, P., & Magnenat, S. (2017). Bringing robotics to formal education: The thymio open-source hardware robot. *IEEE Robotics & Automation Magazine*, *24*(1), 77–85.

Negrini, L., & Giang, C. (2019). How do pupils perceive educational robotics as a tool to improve their 21st century skills? *Journal of e-Learning and Knowledge Society*, *15*(2). https://doi.org/10.20368/1971-8829/1628.

Papert, S. (1980). *Mindstorms: Computers, children, and powerful ideas*. New York: Basic Books.

Perez, S., Massey-Allard, J., Butler, D., Ives, J., Bonn, D., Yee, N., & Roll, I. (2017). Identifying productive inquiry in virtual labs using sequence mining. In E. André, R. Baker, X. Hu, M. Rodrigo, & B. Du Boulay (Eds.), *International conference on artificial intelligence in education*, (pp. 287–298). Wuhan: Springer. https://doi.org/10.1007/978-3-319-61425-0_24.

Perkovic, L., Settle, A., Hwang, S., & Jones, J. (2010). A framework for computational thinking across the curriculum. In A. Clear, & L. Dag (Eds.), *Proceedings of the 15th annual conference on innovation and technology in computer science education (ITiCSE)*, (pp. 123–127). New York: Association for Computing Machinery (ACM). https://doi.org/10.1145/1822090.1822126.

Puccio, G. (1999). Creative problem solving preferences: Their identification and implications. *Creativity and Innovation Management*, *8*(3), 171–178.

Repenning, A., Webb, D., Koh, K., Nickerson, H., Miller, S., Brand, C., … Repenning, N. (2015). Scalable game design: A strategy to bring systemic computer science education to schools through game design and simulation creation. *ACM Transactions on Computing Education (TOCE)*, *15*(2), 11. https://doi.org/10.1145/2700517.

Riedo, F., Chevalier, M., Magnenat, S., & Mondada, F. (2013). Thymio II, a robot that grows wiser with children. In *Proceedings of the 2013 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO)*, (pp. 187–193). Tokyo: IEEE. https://doi.org/10.1109/ARSO.2013.6705527.

Riedo, F., Rétornaz, P., Bergeron, L., Nyffeler, N., & Mondada, F. (2012). A two years informal learning experience using the Thymio robot. In U. Rückert, S. Joaquin, & W. Felix (Eds.), *Advances in Autonomous Mini Robots*, (pp. 37–48). Berlin: Springer. https://doi.org/10.1007/978-3-642-27482-4_7.

Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, *14*(1), 42.

Sadik, O., Leftwich, A.-O., & Nadiruzzaman, H. (2017). Computational thinking conceptions and misconceptions: Progression of preservice teacher thinking during computer science lesson planning. In P. Rich, & C. Hodges (Eds.), *Emerging research, practice, and policy on computational thinking*, (pp. 221–238). Cham: Springer. https://doi.org/10.1007/978-3-319-52691-1_14.

Sharpe, T. L., & Koperwas, J. (2003). *Behavior and sequential analyses: Principles and practice*. Sage Publications, Inc. https://doi.org/10.4135/9781412983518.

Shin, J., Siegwart, R., & Magnenat, S. (2014). Visual programming language for Thymio II robot. In *Paper presented at the Conference on interaction design and children (idc'14)*. Aarhus: Available at http://se.inf.ethz.ch/people/shin/publications/shin_idc14.pdf. Accessed 6 June 2020.

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

Sullivan, A., Bers, M., & Mihm, C. (2017). Imagining, playing, and coding with kibo: Using robotics to foster computational thinking in young children. In *Proceedings of the International Conference on Computational Thinking Education*. Wanchai: Available at https://ase.tufts.edu/devtech/publications/Sullivan_Bers_Mihm_KIBOHongKong%20.pdf. Accessed 6 June 2020.

Tsai, M.-J., Hsu, C.-Y., & Tsai, C.-C. (2012). Investigation of high school students' online science information searching performance: The role of implicit and explicit strategies. *Journal of Science Education and Technology*, *21*(2), 246–254.

Viau, R. (2009). *La motivation en contexte scolaire*, (2nd ed.,). Bruxelles: De Boeck ISBN ISBN: 978-2-8041-1148-9.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology*, *25*(1), 127–147.

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

Yaroslavski, D. (2014). *How does lightbot teach programming?* Lightbot.com Available at https://lightbot.com/Lightbot_HowDoesLightbotTeachProgramming.pdf. Accessed 6 June 2020.

Publisher's Note