Computational Thinking in Context Across Curriculum: Students' and Teachers' Perspectives

Nataša Grgurina[1,2(✉)] and Sabiha Yeni[3]

[1] Radboud University, Nijmegen, The Netherlands
natasa.grgurina@ru.nl, n.grgurina@rug.nl
[2] Teaching and Teacher Education, University of Groningen, Groningen, The Netherlands
[3] Leiden University, Leiden, The Netherlands
s.yeni@liacs.leidenuniv.nl

Abstract. Integration of computational thinking (CT) into numerous disciplines across the K-12 curriculum is gaining increased attention. In this study, based on the technology integration framework, we investigated how students' understandings, difficulties, and attitudes towards learning subject matter varied at different levels of CT integration. We implemented six different case studies by integrating CT into six different subjects: science, traffic, language, biology, geography, and physics. Two primary and four secondary school teachers, 38 primary school students, and 113 secondary school students were involved in the study. We categorized these lessons according to the technology integration model: unplugged activities are grouped as augmentation level; robotic and two modeling activities are labeled as modification level; modeling and digital storytelling activities are labeled as redefinition level. Our findings indicate that students reported a very positive attitude toward redefinition level activities. Teachers stated that compared to standard instructional activities, students can go deeper and understand the subject content better in CT integrated lessons.

Keywords: Computational thinking · SAMR model · Integration · K-12 · Students' attitude · Teachers' attitude · Computational modeling

1 Introduction

One of the increasing trends in the current educational systems is the call for the integration of digital literacy, and in particular its aspect Computational Thinking (CT), into the curricula [6, 27]. CT provides students with a new set of skills for problem-solving and thinking about the world. Computer Science courses are often considered as regular lessons for teaching CT, however there is an increasing interest to see CS as "a truly interdisciplinary undertaking" [26] for other disciplines [14]. Wing [28] defined the CT term in this context as "… the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent […]. These solutions can be carried out by

E. Barendsen and C. Chytas (Eds.): ISSEP 2021, LNCS 13057, pp. 3–15, 2021.
https://doi.org/10.1007/978-3-030-90228-5_1

any processing agent, whether human, computer, or a combination of both" [10]. While there are many definitions of the CT problem solving process, they all share a common view that it involves three steps: translating the problem under scrutiny into computational terms, constructing a computational solution, and finally, using that computational solution in the domain where the problem originates [3]. Even individuals who will not become programmers are expected to acquire some universal competencies such as the ability to solve problems in daily life, break down complex problems into components, and generalize solutions.

There are various approaches that guide the development of students' computational thinking and subject skills. We used the elements of active learning in all cases of our study. Integrating CT concepts through **digital storytelling** is an effective approach that is used particularly in language classrooms to promote subject-related skills and CT skills [11, 18, 23]. Another approach for integrating CT into an interdisciplinary field is through **computational modeling** which is used in different contexts such as science, social sciences etc. [4, 15]. Computational modeling has been used in computer-based learning environments to model learners' subject related knowledge, cognitive skills, and customize their experiences in the environment based on this information [4]. **Robotics** construction kits have been designated as a promising way to introduce CT processes to students in K-12 schools because it motivates students and scaffolds the learning of programming—novices create programs by initially manipulating visual blocks on the computer screen and can then be guided to create more complex programming [2, 13]. A meta-analysis of studies where robotics is used to promote STEM learning revealed that, generally speaking, the use of educational robotics increased students' learning of specific STEM concepts [5]. Finally, **unplugged activities** can be a good approach to introduce students, especially young learners, to CT [20]. Regarding the integration to curriculum, teaching science concepts using CT unplugged approach may provide an opportunity for the students to strengthen their foundation in CT skills [7] and improve their understanding in science concepts [24] simultaneously.

The effectiveness of the CT integrated lessons was examined in educational intervention studies with different aspects such as students' comprehension level about CT skills [8] or level of understanding about subject related objectives [25] or attitude and satisfaction of students toward CT integrated lessons. With respect to the attitude of students, for example, studies explore (increased) interest in and perception of STEM and CS [8, 17] and interest to pursue a CS degree [16]. Further examples focus on affective aspects such as enjoyment in the activity [12], learning experience [22], motivation [19], ownership [21], and difficulties [1].

In this study, in line with the different levels of technology integration framework, students' progress from completing challenges posed by teachers or modifying existing products to taking an active role as developers of their own designs that make use of computational tools. Puentedura [25] proposes SAMR framework that classified technology integration into learning activities into four levels. The substitution degree is the lowest step at the enhancement level, where CT concepts, models, or associated technologies are directly substituted for more traditional ones. The next degree is the augmentation degree where the technology acts as a direct technology substitute with functional improvement. At the higher level, i.e., *transformation level*, the *modification*

degree signifies that the technology use allows significant task redesign. Finally, at the highest level in this framework—the *redefinition degree*—the use of technology allows for the creation of new tasks that were previously inconceivable.

This study is part of a larger project aiming to examine the characteristics of a successful continuous learning trajectory for the integration of computational thinking into the K-12 curriculum across the disciplines of sciences, humanities, and languages. In cooperation with participating teachers, we developed lesson series and carried them out during the first phase of the project. In these lessons, students engage in active learning with the different levels of integration of the SAMR framework. The aim of this study was to characterize the perceived understanding, difficulties, and attitudes of the students participating in CT integration studies at various degrees of technology integration. We look at these issues from the students' and the teachers' perspectives.

2 Methods

2.1 Participants

In this study, the participants from both primary and all four secondary schools participating in the larger project were involved: 38 primary school students, 113 secondary school students and one teacher from each school. The teachers are male. Two primary school teachers, (T2, T4 and T5) are project leaders for digital literacy projects. The age range of students is between 9 and 14. There are 59 female students and 88 male students, four students did not mention gender.

2.2 Lesson Design

In each school, a different lesson was implemented. The participating teachers followed a workshop where they were presented with examples of lessons integrating CT with the subject matter of disciplines across the breadth of the curriculum: science, humanities, languages etc. Then each teacher individually worked with the researchers to develop a lesson series about a topic they planned to teach. These lessons were then taught instead of traditional lessons. We categorized these lessons according to the SAMR model. The lessons start with the augmentation stage.

Augmentation Degree
Climate case. In the geography lessons, the 8th grade students were learning about the Köppen climate classification[1]. During three lessons, they learned about the characteristics of the main climate groups and how to determine the climate type from the temperature and precipitation data. The teacher chose an unplugged approach and students were asked to write down the climate determination process in the form of a decision tree. Due to Covid measures, the lessons were carried out as online lessons.

Modification Degree
Self-driving cars case. In this primary school, the 5th grade students were learning about

[1] http://koeppen-geiger.vu-wien.ac.at/koeppen.htm.

self-driving cars and corresponding ethical questions. In the first lesson, the students drew a car with sensors needed to participate in traffic safely, and were asked to write a short algorithm for each sensor. Additionally, they were asked to program a proximity sensor on a mBot car by putting the provided code blocks in the correct order. In the second lesson, the students discussed the advantages of self-driving cars and a number of ethical dilemmas arising from traffic situations. Additionally, they were asked to extend their mBot program to have it follow a line and play a sound—again by putting the given program blocks in the correct order.

Gas exchange case. The 8th grade students were learning about the human body, and in particular, about gas exchange. The researchers provided a simple gas exchange model in Scratch. During three lessons, the students were using it to learn about the oxygen saturation of blood which depends on the type and quantity of the gas being inhaled and the exertion rate. Additionally, the students were asked to adjust the model's program code.

Stopping distance case. The 8th grade students were learning about the stopping distance of a moving vehicle which depends on the road conditions and the reaction speed of the driver. The first lesson was dedicated to theory. In the subsequent two lessons, the students used a simple model in Scratch provided by their teacher and researchers to explore the influence of icy roads, drinking and distraction on the stopping distance. Additionally, the students were asked to adjust the model's program code.

Redefinition Degree

Cell division case. In this primary school case, the 5th grade students learned about the cell division. During the two introductory lessons, they learned about the theory of cell division, watched videos and observed cells under a digital microscope. In the subsequent two lessons, students worked in pairs on a practical assignment: they wrote down all they knew about cell division, they described the corresponding algorithm and developed computational models—i.e. visualizations—of cell division in Scratch. During programming, their class teacher and their programming teacher were helping them when necessary. At the end of the last lesson, each pair of students presented their program in front of the class.

Digital storytelling case. In the course of English as foreign language, the 7th grade students studied to apply the acquired grammar and vocabulary in their digital stories. In the introduction lessons they learned to ask and answer questions about themselves. In the subsequent two lessons, they created their own digital stories to introduce themselves. Students are given an explanation about the goals of the digital story and they made planning. They worked in pairs to create storyboards, got feedback from the teacher about the scripts, and created their digital stories in Scratch. They recorded their voices and added them to their digital stories in order to vocalize the dialogues of the characters. During the programming, their class teacher, researcher and their friends helped where necessary.

2.3 Data Collection

Before starting their work on the projects, the students filled in a **profile survey** where they were asked about their technology ownership, programming background, and experience regarding their familiarity and experience with specific types of programming languages. Furthermore, they were asked about their self-efficacy related to technology and programming, and finally their age and gender. The **practical assignment** consists of a series of questions that guide and document students' activities during their work on the project. The students answer the questions individually to express their problem in computational terms and to construct the computational solution. Additionally, they reflect on the project. **Exit tickets** are small questionnaires given to students at the end of each lesson or at the end of a lesson series. There, the students indicate on a three-point Likert scale whether they liked the lessons (series), whether they found the lessons interesting, whether they understood what they had to do in the lessons, whether they understood the teaching materials, and finally, whether they found the lessons difficult. Additionally, students could write comments about the lessons. **Interviews.** At the end of the project, we interviewed a number of students and all the participating teachers. During the semi-structured interviews which lasted about ten minutes, we asked about the programming practices they employed while working on their projects—which are reported elsewhere—and about their attitude towards technology use, which we report here. We conducted semi-structured interviews with each teacher individually. These interviews lasted about twenty minutes. We asked the teachers to compare their experiences teaching the lessons during the project—thus, making use of CT—to their standard manner of teaching the same content, and about their attitude towards the integration of CT in the lessons. In this report, we focus on their views on students' conceptual understanding, their attitudes and difficulties. All the interviews were recorded and transcribed verbatim.

2.4 Data Analysis

We performed quantitative and qualitative data analysis. The data from exit tickets and some questions from the practical assignments were analyzed through descriptive statistics. For other data, we used Atlas.ti software as a qualitative data analysis tool to code our data. The data were first categorized deductively according to our research goal, in line with the issues the research instruments were concerned with. Within these categories, we applied inductive coding to further characterize the data. In an axial coding process [9], the codes were grouped and merged where necessary under new categories.

3 Results

3.1 Students

Student Profiles. The most commonly used programming languages are block-based languages that 89 students (59%) indicated to use; in the second place are robotic tools that 41 students (27%) stated that they used. The least popular programming languages are text-based programming languages with 30 students (20%). The most commonly

reported programming languages are Scratch, Microbit and Python. As a whole, 119 students (79%) had programming lessons in the past, 12 students (8%) have never had programming lessons but they tried to learn programming by themselves and 20 students (13%) have never had programming lessons before. They were asked how long they had taken programming lessons. In overall, 48 students (32%) had programming lessons less than one month; 43 students (28%) had programming lessons for one year; 32 students (21%) had 2 years to 3 years, and only 3 students (2%) had programming lessons longer than 3 years. Judging their programming experience on the 10-point Likert scale from 1 (no experience) to 10 (very experienced), they score a minimum of 1, a maximum of 9 and a mean of 5,9. Overall, about half of the students (42%) feel comfortable with programming, the other (55%) of students would need at least some help, and 3% of students see themselves as a professional regarding programming (Table 1).

Table 1. The programming experience of students according to different cases

	Cases (teachers)	Programming lesson			Prog. lesson duration				Prog. experience
		Yes	No	Learn by myself	<1 month	1 m to 1 year	2 to 3 years	>3 years	1 (No exp.) to 10 (Very exp.)
Aug	Climates (T1)	95	0	5	58	26	11	5	6,2
Mod	Self-driving cars (T2)	100	0	0	63	25	0	13	4,7
Mod	Gas exchange (T3)	63	25	13	65	35	0	0	5,3
Mod	Stop distance (T4)	47	47	6	67	11	11	11	4,8
Red	Cell division (T5)	100	0	0	8	0	92	0	6,7
Red	Digital story (T6)	80	9	11	17	68	15	0	6,5

Exit Tickets. Students filled their exit tickets at the end of each lesson or at the end of the lesson series. We analyzed the following categories: Enjoyment: I enjoyed the lesson; Interesting: The lesson was interesting; Clarity: I knew what to do; Comprehension: I understood the subject matter; and Difficulty: The lesson was difficult (Table 2). The answers were on a three-point Likert scale: Yes (Y), Maybe (M) and No (N). We calculated the relative frequency of the answers for each case. According to the students'

answers, the most enjoyable lessons are the case of cell division (92%) and digital story (91%) and most of the students in these cases (88% and 77% respectively) also found these lessons interesting. The answers of students about clarity are similar to each other between groups (change between 83% and 66%); only in the case of gas exchange, 43% of students declared that they didn't know what to do. With respect to comprehension, the most remarkable answer belongs to the case of climates. While most of the students (91%) in this unplugged case state that they understand the subject (climates in geography) well, even if the number of those who have fun and find the lesson interesting is not very high compared to other cases. Regarding the difficulty of the lessons, students in the case of climates found the lesson series easy (70%).

Practical Assignments. Students from three schools turned in their practical assignments: those working on the stopping distance project, on the cell division projects, and on the digital storytelling projects. We report on students' reflecting on their projects.

Table 2. The mean percentages of exit tickets

	Cases	Enjoyment			Interest			Clarity			Compr			Difficulty		
		Y	M	N	Y	M	N	Y	M	N	Y	M	N	Y	M	N
Aug	Climates	55	43	2	51	40	4	66	30	4	91	6	2	4	26	70
Mod	Self-dr. cars	85	13	3	72	26	3	77	18	5	69	28	3	8	38	54
Mod	Gas exchange	57	35	9	46	45	10	49	28	15	45	41	14	16	43	40
Mod	Stopping dist	50	25	25	38	44	19	69	19	19	69	19	6	19	25	44
Red	Cell division	92	8	0	88	13	0	83	13	4	79	21	0	0	63	33
Red	Digital story	91	9	0	77	16	8	75	21	3	70	25	5	11	25	64

Regarding the satisfaction with their projects, the majority of the students reported they were satisfied. One student is satisfied with some parts of the project but also mentions aspects they are not satisfied with. Some students offer suggestions to improve their project, for example, by testing their program more thoroughly, and some comment they had not finished their projects We asked students what they enjoyed most in the lessons. The most frequently reported issue is programming or editing an existing program, mentioned by more than half of the students. Only students working on the cell division project mentioned subject matter, such as, for example, using the microscope to look at cells. In each class, there were students reporting they enjoyed cooperating with each other. Two students found nothing to be enjoyable in these lessons and found them to be boring. When asked what they enjoyed least, 15 students reported they actually enjoyed everything. Overall, about half of the students actually report they disliked something

in the lessons, and about half of those refer to organizational aspects of the lessons such as the quality of the instruction, but also having to wait for too long before they were allowed to begin with programming. Furthermore, the students mention programming issues: six students reported they disliked programming while another three reported had too few opportunities to program Eleven students reported disliking subject-related issues, such as working with a microscope or having to record their voices. Finally, one student dislikes having to articulate and write down their thoughts.

Regarding what the students found to be the most difficult, one student reported everything was difficult and ten students found nothing to be difficult. Out of 24 students who found programming to be difficult, 11 consider it to be the most enjoyable aspect of the project as well. A total of 10 students experienced difficulties related to subject matter—for example, working with a microscope. Eleven students report difficulties related to their creativity (for example, coming up with sentences for their stories) and to thinking (for example, implementing an idea exactly as imagined in their mind).

We asked the students whether they think working on this type of a project (i.e., with computational thinking embedded in the context of the subject matter) helped them to learn subject matter better. Students told us that, for example, this work helped them because learning this way is more fun and so they worked better; but also, that it did not help, "because it was totally different than a school subject". Finally, we asked whether they would like to work on such a project in the future and if so, what subject matter they consider suitable. The students mention math, languages, physical education, and a few believe it would suit all subjects. The elementary school students show a more positive attitude on both of these issues.

Interviews. After the lesson series finished, we conducted 24 student interviews with students from five schools: three from the self-driving cars project, nine from the gas exchange project, three from the stopping distance project, two from the cell division project, and seven from the digital storytelling project. In the data analysis, we focused on their own perspective on the lesson activities and their learning. A common theme many students mention is their appreciation of collaboration with other students. The primary school students (those working on the self-driving cars and the cell division projects) unanimously agreed they *liked* this type of lesson and believed they *learned subject matter better*. One of the students explained that he *liked* working this way and therefore was *more motivated to learn* about cell division. Another student said they *gained more insight* into the inner workings of a self-driving car after having programmed a sensor for such a car. The students working on the stopping distance project are divided on the issue. While one student said, "Now I *understand* the stopping distance, because you *learn in a fun way* and then you remember it better", another one was *critical about using a Scratch model* to help them learn: "Of course I know you shouldn't look at your phone while driving, but *I haven't learned much.*"

The students working on the gas exchange project are divided on the issue as well and place a number of thoughtful remarks. Several of them said *it was nice to do something new* for a change; however, some warned that this sort of game-like activities could be a *distraction.* Regarding their learning, one student commented that *this way of learning helps* because they get to examine the issue at hand from various perspectives. Another student adds that it was nice to be able to change the values of the parameters in the

model and see the effect on the outcomes. Yet, as one of the students notices, while the *model helped them to learn* about gas exchange, the visual representation of gas molecules and blood cells in the model was not realistic.

Finally, the students working on the storytelling project reported that they *liked working this way*, for example, because working from books only is boring. One of the students said programming a conversation in Scratch *helped them learn English better* because it made them speak in English; however, another student remarked their English was already good and they *did not learn much.*

3.2 Teachers

After the lesson series finished, we interviewed all six teachers. We present the findings regarding students' conceptual understanding, their attitudes and difficulties. The results regarding all of the teacher's replies regarding all pedagogical content knowledge (PCK) elements are reported in another extended study [29].

Students' Conceptual Understanding. This refers to teachers' knowledge and understanding of students' conceptual knowledge and knowledge gaps. Four teachers [T1, T3, T4, T5] reflect on their knowledge regarding the conceptual understanding of students. Regarding the students' understanding of the subject related concepts, the common view of four teachers is that students understood the subject more deeply in CT integrated lessons than in the traditional lessons.

For the augmentation level of the climate case, the teacher said, "*The students generally have quite a lot of trouble with that [to understand climate related learning objectives] The penny does not drop quickly, and that often takes a lot of time. The learning objective was actually to do that in a special way [CT integration], and we did to make it clear."* [T1] For the modification level of the "gas exchange" case, the teacher commented, *"I think most of them, 80 to 90% knew what they were doing […] it was not only Scratch but also biology."* [T3] In the case of stopping distance, *"A whole group actually got to work very fanatically to adapt it [model]* [T4]. For the redefinition level of the "cell division" case, the teacher remarked, *"If you look at the grade 5 level [of the cell division], I think they have learned enough for this year […] I thought this [CT integration] might be a way to make this indeed more understandable for the kids [Regarding the CT concepts] Everyone had programmed something in Scratch, they could do that quite well. No matter how basic it was. Programming in Scratch and making the connection between the steps you have to take in Scratch, well I thought that went pretty quickly."* [T5].

Students' Reaction and Perception. This refers to the teachers' knowledge and awareness of students' reactions and perceptions toward CT integration lessons. The teachers involved in the lesson of all integration levels stated that students' perceptions were encouraging and positive, and they said that with these terms: like, enjoy, enthusiastic, exciting, interesting, fun, educational, voluntary [T1, T2, T3, T4, T5, T6]. Only the teacher of the climate case (the augmentation level) [T1] noticed that two girls who went to the same primary school, and they indicated that they didn't like it. The teachers said,

"I think they [students] liked it. […] I think what I saw was that they were working on it with enthusiasm." [T5] *"Actually, in my experience, kids always like it. They enjoy working with such a robot. It is often new to them too […] They often get excited about that anyway."* [T2] *"A lot of them said that they liked it a lot. They thought it was hard but they said that was interesting in their opinions […] I think they did enjoy it."* [T6] *"You can really see that they have a lot of work on it but they also seemingly had a lot of fun doing it"* [T1] *"A whole group actually got to work very fanatically to remix it."* [T4] *"I had the idea that they [students] liked it. […] I do have the feeling that it has been educational and fun."* [T3].

Students' Difficulties. According to teachers, students experience various sorts of problems during the lessons, including those related to: understanding the programming concepts (conditions, design of algorithm, decision tree) [T1, T2, T5], requiring a lot of skills/knowledge to accomplish task (such as subject-matter knowledge, programming knowledge, collaboration skills, planning skills etc.) [T6], creating new things [T4], and understanding instruction [T3].

For the augmentation level of the climate case, the teacher said, *"They all found it easy. That is unusual, you always have many questioners among students… Questions were asked about the content, and that was quite unusual. They really had some difficulty with the concept of the decision tree, because I had introduced it fairly quickly."* [T1]. For the modification level of the self-driving car case, the teacher commented, *"They found the assignment more difficult to design a short algorithm for the sensors. They just had to write that down […] They had a lot of trouble with that."* [T1] For the redefinition level of the digital storytelling project, the teacher said, *"There are a lot of required skills you need to know besides the English stuff. You also need to make a plan, to be able to work together and also then the whole Scratch such as how you program those things. I guess that was the hardest part for them"* [T6]. For the cell division case, the teacher remarked, *"They [students] found it very difficult to get conditions for cell division to actually get there […] They did not progress beyond one condition to achieve cell division."* [T5].

4 Conclusion and Discussion

In this study, we explored the perceived understanding, difficulties and attitude of the students participating in CT integration studies at various degrees of technology integration, from the students' and teachers' perspective. Regarding students' understanding and difficulties related to the subject matter concepts, the teachers indicated that students understood the subject more deeply in CT integrated lessons than in the traditional lessons. The students themselves indicated that, for the most part, they did not find the lessons difficult and very few of them indicated having had difficulties understanding the subject matter. We conclude that in the lessons where CT was integrated with subject matter, the students did not experience a lot of difficulties and that participating in this type of lessons was beneficial for the deeper understanding and learning of the subject matter.

Regarding the students' understanding and difficulties related to the CT, the teachers reported about their students' specific difficulties related to programming concepts. The students themselves indicated having found programming to be difficult in less than half of the cases. Furthermore, some students experience difficulties related to other issues such as understanding instruction, planning their work and cooperation.

Regarding the students' attitude and their perception of these types of lessons, the teachers report a positive attitude of their students, and the students themselves corroborate this finding. The students reported enjoying these lessons and being satisfied with their work. Notably, a number of students realized that they were better at programming than what they initially thought, or that they actually enjoyed programming. A detailed analysis of all findings reveals that the students who actively engaged in programming (i.e. those working on the projects involving self-driving cars, cell division and storytelling) enjoyed the lessons more and found the lessons more interesting than the students working on other projects. Yet, this outcome is not reflected in the students' opinion on the issue of whether these types of lessons help them to learn the subject matter better. While the primary school students (those working on the cell division project) almost unanimously believe that it does, and would like to engage in similar projects in the future, the students in secondary schools (those working on the stopping distance and storytelling projects) are divided and a majority of those answering this question disagrees.

Relating our findings to the integration levels of the SAMR framework [25] reveals that student engaging in lessons at the transformation level—i.e., modification and redefinition degrees—taken as a whole, tend to have a positive attitude towards this type of lessons. Again, in the projects where students actively engage in programming, they tend to express a positive attitude more often, and these students are exactly the ones who more frequently indicate that they have had programming lessons in the past. Their teachers were aware of the students' programming experience and prepared the lessons accordingly. This finding confirms the importance for students to learn computational thinking: the students who are adept at it have better chances to profit from the integration of CT with the learning of subject matter and to gain deeper understanding of subject matter. Indeed, such students engage in active learning and seem to experience multiple benefits when studying subject matter: expressing the problem under scrutiny in computational terms requires deep understanding of the problem, as does the construction of a computational solution too. Finally, using that computational solution in the original domains allows students to gain new insights [3].

In this study, we saw that integration of CT with subject matter benefits students' learning. Future research within the larger project where this study is a part of is going to expand the focus on measuring learning gains in learning of both subject-matter related and CT-related learning objectives.

Acknowledgements. This work is supported by The Netherlands Organisation for Scientific Research grant nr. 40.5.18540.153.

References

1. Almeida, R., et al.: iProg: getting started with programming: pilot experiment in two elementary schools. In: 2017 International Symposium on Computers in Education (SIIE), pp. 1–5 (2017)
2. Baek, Y., et al.: Revisiting second graders' robotics with an understand/use-modify-create (U2MC) strategy. Eur. J. STEM Educ. **4**, 1 (2019)
3. Barendsen, E., Bruggink, M.: Het volle potentieel van de computer leren benutten: over informatica en computational thinking (2019)
4. Basu, S., Biswas, G., Kinnebrew, J.S.: Learner modeling for adaptive scaffolding in a computational thinking-based science learning environment. User Model. User-Adap. Inter. **27**(1), 5–53 (2017). https://doi.org/10.1007/s11257-017-9187-0
5. Benitti, F.B.V.: Exploring the educational potential of robotics in schools: a systematic review. Comput. Educ. **58**(3), 978–988 (2012)
6. Bocconi, S., et al.: Developing computational thinking in compulsory education. Eur. Comm. JRC Sci. Policy Rep. (2016)
7. Brackmann, C.P., et al.: Development of computational thinking skills through unplugged activities in primary school. Presented at the (2017)
8. Burgett, T. et al.: DISSECT: analysis of pedagogical techniques to integrate computational thinking into K-12 curricula. In: 2015 IEEE Frontiers in Education Conference (FIE), pp. 1–9. IEEE (2015)
9. Cohen, L., et al.: Research Methods in Education, 6th edn. Routledge, Abingdon (2007)
10. Cuny, J., et al.: Demystifying computational thinking for non-computer scientists. Work in Progress (2010)
11. Curzon, P., et al.: Introducing teachers to computational thinking using unplugged storytelling. In: Proceedings of the 9th Workshop in Primary and Secondary Computing Education, pp. 89–92 (2014)
12. Djurdjevic-Pahl, A., Pahl, C., Fronza, I., El Ioini, N.: A pathway into computational thinking in primary schools. In: Wu, T.-T., Gennari, R., Huang, Y.-M., Xie, H., Cao, Y. (eds.) SETE 2016. LNCS, vol. 10108, pp. 165–175. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-52836-6_19
13. Grover, S., Pea, R.: Computational thinking in K–12 a review of the state of the field. Educ. Res. **42**(1), 38–43 (2013)
14. Guzdial, M.: Computing for Other Disciplines. Cambridge, UK (2019)
15. Arastoopour Irgens, G., et al.: Modeling and measuring high school students' computational thinking practices in science. J. Sci. Educ. Technol. **29**(1), 137–161 (2020). https://doi.org/10.1007/s10956-020-09811-1
16. Jawad, H.M., et al.: Integrating art and animation in teaching computer programming for high school students experimental study. In: 2018 IEEE International Conference on Electro/Information Technology (EIT), pp. 0311–0317. IEEE (2018)
17. Kafai, Y.B., et al.: A crafts-oriented approach to computing in high school: introducing computational concepts, practices, and perspectives with electronic textiles. ACM Trans. Comput. Educ. TOCE. **14**(1), 1–20 (2014)
18. Kordaki, M., Kakavas, P.: Digital storytelling as an effective framework for the development of computational thinking skills. In: EDULEARN 2017, pp. 3–5 (2017)
19. Lévano, M., et al.: Methodological framework for the development of computational thinking and programming through experiential learning: case study from schools Juan Seguel and Allipen in Freire, Chile. In: 9th IADIS International Conference, p. 107 (2016)
20. Looi, C.-K., et al.: Analysis of linkages between an unplugged activity and the development of computational thinking. Comput. Sci. Educ. **28**(3), 255–279 (2018)

21. Lytle, N., et al.: Use, modify, create: comparing computational thinking lesson progressions for STEM classes. In: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education, pp. 395–401 (2019)
22. Nygren, E., et al.: Quantitizing affective data as project evaluation on the use of a mathematics mobile game and intelligent tutoring system. Inform. Educ. **18**(2), 375–402 (2019)
23. Parsazadeh, N., et al.: Integrating computational thinking concept into digital storytelling to improve learners' motivation and performance. J. Educ. Comput. Res. **59**(3), 470–495 (2021)
24. Peel, A., et al.: Learning natural selection through computational thinking: unplugged design of algorithmic explanations. J. Res. Sci. Teach. **56**(7), 983–1007 (2019)
25. Puentedura, R.R.: Learning, technology, and the SAMR model: goals, processes, and practice. http://www.hippasus.com/rrpweblog/archives/2014/06/29/LearningTechnologySAMRModel.pdf
26. Tedre, M., et al.: Changing aims of computing education: a historical survey. Comput. Sci. Educ. **28**(2), 158–186 (2018)
27. Thijs, A., Fisser, P., Van der Hoeven, M.: 21e-eeuwse vaardigheden in het curriculum van het funderend onderwijs [21st century skills in the curriculum of fundamental education] (2014)
28. Wing, J.M.: Computational thinking. Commun. ACM. **49**(3), 33–35 (2006)
29. Yeni, S., Grgurina, N., Hermans, F., Tolboom, J., Barendsen, E.: Exploring teachers' PCK for computational thinking in context. In: The 16th Workshop in Primary and Secondary Computing Education, 18–21 October 2021 (2021, accepted)