Education and Information Technologies (2025) 30:6925–6949
https://doi.org/10.1007/s10639-024-13092-1

The importance of coding and translation between programming languages in sequential activities of pre-service teachers: an approach

Ainhoa Berciano[1] · **Astrid Cuida**[2] · **María-Luisa Novo**[2]

Received: 16 February 2024 / Accepted: 1 October 2024 / Published online: 18 October 2024

Abstract
In the last two decades, computational thinking has gained wide relevance in international educational systems. The inclusion of this new type of thinking poses educational challenges with some underlying research questions that need to be answered to meet these challenges with quality. Thus, this study focuses on analyzing the difficulties that teachers in initial training experience have, when carrying out translation tasks of programming languages used by certain educational robots, in this case, the Cubetto. For this purpose, a specific learning sequence has been designed to work with different programming languages (Cubetto, Bee-Bot, Scratch) and natural language. The work of early childhood and elementary trainee teachers in these tasks has been analyzed using a descriptive approach. The main results are: (1) some of the difficulties encountered are clearly caused by the Cubetto hardware (regardless of the language to which it is translated) and (2) the designed learning sequence has enabled coding skills to be improved remarkably. We conclude that translation tasks between programming languages are necessary in initial teacher training to improve their ability programming and their computational thinking, and for them to be able to detect the disadvantages and benefits of educational robots in their transposition to the classroom.

Keywords Coding · Computational thinking · Pre-service teachers · Early childhood and primary education

1 Introduction

The growing interest in computational thinking within educational settings has gradually led to its integration into school curricula, starting from the earliest educational stages. Following this idea, computational thinking should be developed in Early

Extended author information available on the last page of the article

Childhood Education (BOE, 2022) through "processes of observation and manipulation of objects" (p. 23) that enable children to "program sequences of actions or instructions for the resolution of analog and digital tasks, thereby developing basic computational thinking skills" (p. 25).

Therefore, considering the new skills to be developed at this educational stage and the challenge of achieving meaningful learning that fosters computational thinking (CP), it is necessary to provide teachers with training scenarios which enable them to understand what computational thinking represents. The acquisition of this knowledge implies: (1) to identify its terms, (2) to achieve a good attitude towards its learning, (3) favor the development of indispensable skills which can successfully address the development of computational thinking in school (Yadav et al., 2017) and (4) provide a pedagogy that enhances the development of computational thinking for the early educational stages (Zapata et al., 2021).

According to these considerations, teachers in initial training must be trained to be competent in all aspects related to computational thinking, including programming. Regarding programming, Bers (2017) raises seven associated big ideas: modularity, debugging, representation, hardware/software, algorithms, control structures, and design processes.

Focusing on hardware/software, we can establish a graduation in the complexity of imperative programming, depending on whether physical manipulation is required in the coding process (Martín et al., 2002). Thus, we have tangible programming, carried out through objects with tangible interfaces and with the capacity to be programmed (for example, educational robots: Bee-Bot, Code-a-pillar, Code 'n Learn Kinderbot, Cubetto, Bluebot, …) and non-tangible programming, carried out through software (computers and tablets). In this sense, some previous research suggests that, to develop computational thinking at early ages, programming must be tangible, with emphasis on understanding the type of programming involved in each educational robot (Misirli & Komis, 2023; Bers et al., 2014).

Furthermore, this dimension, hardware/software, delves into the importance of the translation processes between natural language and the language used by the programmable object, and the necessity of understanding that it is not sufficient to merely find a theoretical solution to a given problem. It is equally crucial to know how to accurately translate this solution into the respective programming environment to ensure the instructions are conveyed correctly.

Thus, given the relevance of the dimension hardware/software and the language involved, different authors have been concerned with the errors in programing and compiling costs in programing learning. In particular, the correction of the code requires a constant assistance from the teachers and the revision of it could be a better tool for teachers and students to improve their knowledge (Zeller, cited by Fernández-Medina et al., 2011). It is necessary that research goes deeper and analyzes the types of errors that students do in programming tasks to better understand this process (Fernández-Medina et al., 2014).

So, the aim of this work is to respond to the need mentioned in the previous paragraph, presenting as a novelty of research the interest in better understanding the teaching and learning processes of programming through the detection of errors made by students. Our interest is focused on exploring the difficulties that trainee teach-

ers have in imperative language translation processes, where implied language is completely determined by the hardware used, in this case, an educational robot. The final objective is to better understand the process of programming learning involved in teacher initial training to go further in their computational thinking skill development, designing better tasks that help them to be able to use educational robots in their prospective schools. The detection of errors will allow us to design a scaffolding system of tasks that will help trainee teachers to identify their errors and overcome them, improving their computational thinking.

Centering our interest in trainee teachers (Early Childhood and Primary Education), we want to analyze their errors and the determination of the nature of these errors, when they learn the use of educational robots. For this end, starting from a learning sequence specifically designed and implemented to perform translations between the educational robot Cubetto (an object with tangible programming) and Bee- bot (another with tangible programming) or Scratch (one with non-tangible programming), we intend to answer the following questions:

RQ 1)	What are the errors made by trainee teachers in the translation from Cubetto language to Bee-Bot and Scratch languages, their nature, and the extent of casuistry involved?
RQ 2)	How does the trainee teachers' ability to solve a translation task from the Cubetto language to natural language evolve throughout the learning sequence?
RQ 3)	To what extent does this learn-sequence help trainee teachers in the translation skills of inter-imperative languages?
RQ 4)	What are the errors by trainee teachers in the translation from natural language to Cubetto?

Next, the theoretical framework is described, explaining the relationship between computational thinking, programming and coding, then, its potential in Early Childhood Education is explained, as well as the need for teacher training for the development of computational thinking. Subsequently, the research methodology, the results and the main discussions and conclusions of this research are described.

2 Theoretical framework

2.1 Computational thinking, programming and coding

Computational thinking, defined as a broad spectrum of reasoning skills to formulate and solve problems algorithmically, and develop a sense of technological fluency (Garcia-Peñalvo & Mendes, 2018; Wing, 2006, 2011), which can be applied in other contexts (Lee et al., 2011), has taken on great relevance in recent years; a relevance to which education has not been oblivious.

Thus, recent studies have investigated the treatment of Computational Thinking in the educational field; showing in many cases a close relationship between computational thinking, that is, programming, and coding; where programming is the process of constructing programs, i.e. algorithms expressed in a specific language for an automatic processor (Knuth, 2005); an ability increasingly demanded in society (Ching et al., 2018), which enables effective participation in a society where there is an increasing ubiquity of digital devices (Bers, 2018; Kafai, 2016); and coding or

code-literacy is the ability to create new socially situated symbolic systems that, in turn, enable new types of expressions, as well as the extension of pre-existing forms of communication (Vee, 2017); a skill that, according to Wing (2006), is essential in the process of learning computational thinking.

Thus, a detailed analysis of these three concepts, computational thinking, programming, and coding, allows us to highlight that they are closely related, but clearly distinct; that is, computational thinking is an expressive process that enables new ways to communicate ideas while programming can be seen as a tool for teaching computational thinking and coding as writing connected to technology.

Regarding programming in the educational field, several researchers propose the need to work on it as an essential aspect involved in the development of computational thinking, because: (1) it helps "[…] children to develop as computational thinkers" (Resnick & Rusk, 2020, p.122), (2) it empowers children to solve problems (Bers, 2018; García-Valcárcel & Caballero-González, 2019), (3) it involves a practical way of teaching inquiry-based computational thinking (Ladzowska & Patterson, 2013; Halverson & Sheridan, 2014); (4) it offers a new way to express ideas (Austin et al., 2020; Bers & Sullivan, 2019; Papert, 1980).

On coding and its incorporation into education, Resnick et al. (2009) state that it provides meaningful and motivating learning opportunities. For Bers (2019) coding is a new language that children can learn through play and creativity from a very early age. Different authors suggest working on coding by starting with simple and fun tasks until achieving more complex ones, focusing the child's progress on the difficulty of the tasks and their motivating characteristics (Delacruz, 2020; DiSessa, 2001; Duin & Tham, 2019). Resnick and Rusk (2020) state that with this teaching-learning process for the literacy of computer programming languages "students are not just learning to code, they are coding to learn" (p.121) and "As students create their own stories, games, and animations with code, they start to see themselves as creators, developing confidence and pride in their ability to create things and express themselves […]" (p.123).

Thus, the inclusion of coding in education has driven: (1) the development of different types of programming interfaces (Bau et al., 2017), (2) the rise of global outreach initiatives supporting education science education such as Code.org, CSforAll, CoderDojo and Code Club, (3) the proliferation of a wide variety of programmable devices that bring flexibility to the spectrum of ways of coding that students can use such as micro: bit (Austin et al., 2020), robotics kits (Khine, 2017) and programmable toys (Yu & Roque, 2019; Clarke-Midura et al., 2019), (4) the increase in the number of teachers who manage to integrate creative and expressive approaches to coding in the classroom (Resnick & Rusk, 2020) and (5) the promotion, globally, of the development of computational thinking skills and the introduction of programming concepts and languages from early childhood education (Macrides et al., 2022).

2.2 Computational thinking, programming and coding in early childhood education

Given the potential of computational thinking, programming, and coding and, considering that quality learning opportunities from early childhood not only impact

the achievement of social, emotional, and cognitive milestones but also positively affect later learning stages (Bers, 2017; Yu & Roque, 2019), governments have been increasingly concerned with encouraging the introduction of computer programming concepts and languages from early ages (Bers et al., 2022; Strawhacker et al., 2018; Sullivan et al., 2017). These learning opportunities can be provided through employing the seven key programming ideas outlined in the introduction: algorithms, modularity, control structures, representation, hardware/software, design process, and debugging (Bers, 2017).

Regarding hardware/software, García-Peñalvo and Mendes (2018) highlight the importance of its selection in classroom contexts. Thus, their selection depends on the educational stage and on the programming associated with the hardware/software (tangible or non-tangible). Given the above, it is recommended to start with tangible and then move on to non-tangible, because tangible programming and coding environments can be more effective for introducing programming at the early ages, as can block-based programming, as "it is a practical way to encourage the child to explore coding, as well as to use languages with simple movement commands such as forward, sideways, or backward, to support spatial, visual, and cognitive skills" (Bers et al., 2022: p.26).

As examples of tangible programming it appears that for early ages educational computational kits (educational robots) are the favorites, because, among other aspects, it has been seen that educational robotics contributes to the construction of knowledge due to its versatility, providing interdisciplinarity and getting students to be active agents of their own learning (Segatto & Teixeira, 2021) and consequently educational robotics or programming are currently being included in curricula throughout the world (Bers et al., 2022; Yu & Roque, 2019). Among the many examples of educational robots, we highlight Cubetto (from Primo Toys). The Cubetto kit includes a robot named Cubetto, with a tangible interface, a set of external tangible manipulative coding blocks and parts that control the robot, a control board on which to place the coding blocks, and maps and storybooks. As children program the robot to move from place to place, they are breaking down the problem/complex into smaller/elementary parts (Pridmore et al., 2010; Angeli et al., 2016) in which children can correct errors within their programs by working debugging (Selby, 2014; Angeli et al., 2016). In addition, there are tools to measure coding ability for children aged three to six years old in tasks carried out with Cubetto (e.g., Marinus et al., 2018).

Among the examples of non-tangible programming, one such educational tool is Scratch (Resnick et al., 2009), a programming language that is part of an online community and authoring environment in which computational thinking is defined around three key dimensions including (1) computational concepts (2) computational practices and (3) computational perspectives (Brennan & Resnick, 2012); it also has an early childhood version, Scratch Jr. (Flannery et al., 2013) and has been conceived as a language that enables knowledge not only relevant to Computational Thinking or mathematics to be built but also provides opportunities for learning ideas from other disciplines and for the development of problem-solving, design, collaboration and communication skills (Resnick & Rusk, 2020; Roque & Rusk, 2019).

2.3 Initial teacher training for the development of computational thinking

In line with international approaches to incorporate computational thinking into education from an early age, teachers should know how to use them in school. In this sense, it is necessary that trainee teachers have the necessary knowledge and conditions to successfully incorporate computational thinking into classroom practices in a meaningful way; where, initial teacher training should allow them to know the variety of computing devices and programming tools (Yadav et al., 2017).

Along these lines, Estebanell et al. (2018) put forward a proposal for teacher training that they describe in four levels of computational thinking development: (1) user level (training focused on learning computational languages and increasingly complex computational problem solving strategies); (2) reflective user level (training focused on fostering reflection on solving a computational challenge); (3) teacher level (training focused on knowing how to decide what they want to teach, what they expect their students to learn about Computational Thinking); (4) reflective teacher level (training that fosters reflection on the teaching and learning processes related to Computational Thinking).

Considering these levels, we find that most of the studies conducted with teachers in initial training focus on levels 1 and 3, that is, user level and teacher level, to better understand the knowledge that this group has about computational thinking and programming.

Regarding level 1, user level, we find several studies on the programming skills of teachers in initial training. Angeli (2022), in a study focused on algorithmic thinking and robotics, claims to evidence significant improvements in algorithmic thinking and debugging skills. This is the reason which is why the author highlights the importance of preparing teachers in initial training to integrate the teaching of Computational Thinking in the classroom from an early age. Likewise, Hamilton et al. (2020) show how competencies inherent to programming (sequence, action-instruction, and debugging dimensions) become measurable skills that can be taken as a reference for teachers to use in the classroom. So, the literature review reveals certain orientations on essential aspects for initial teacher training in the field of Computational Thinking and programming, but not on coding.

Considering that our research focuses on the importance of coding, empirical research on initial teacher development in this area is limited. Research related to the analysis of the impact of coding on initial teacher education is scarce. Among them, we find those that state that: (1) in the coding process with Bee-Bot, the most frequent error made by trainee teachers in early childhood education occurs in Euclidean space programming contexts, when instructions related to spatial orientation are highlighted (Seckel et al., 2021); (2) in the resolution of programming language translation tasks, based on the use of the Cubetto robot, to verbal language, it turns out that "a notable percentage of teachers in initial training in early childhood and primary education have difficulties associated with understanding the programming language of the Cubetto robot, despite not being aware of such difficulties" (Berciano et al., 2023, p.1).

Thus, given the relevance of coding, its revision (Zeller, cited by Fernández-Medina et al., 2011) and the need to analyze the errors that students do in program-

ming tasks (Fernández- Medina et al., 2014), our research focuses on how trainee teachers do code (at user level). This would help to stablish possible links to their professional work in the future, as heads of early childhood education and primary education classrooms (levels 3 and 4 (Estebanell et al., 2018). In this work we aim to understand the difficulties that this group has in understanding the programming languages associated with both educational robots (tangible programming: Cubetto, Bee-Bot) and non-tangible programming languages (Scratch); that is, to analyze the characteristics that determine the difficulties in the coding process. To this end, the objective is to analyze the errors made by teachers in initial training in the translation processes between the three programming languages described and the nature of these errors. This will allow us to gain a better understanding of the idiosyncrasies of the process of programming learning involved in teacher initial training to go further in their computational thinking skill development, designing better tasks that help them to be able to use educational robots in their prospective schools.

3 Methodology

This research is framed within an interpretative paradigm. From this perspective, to be able to respond to our research objective, we have carried out a mixed type of research, with a qualitative component that is percentagewise more relevant than the quantitative one. This method has been chosen, sharing the ideas of Cameron (2010) who states that research with this type of design involves an excellent fusion between quantitative and qualitative orientations. The responses obtained from the pre-service teachers when individually solving four tasks typical of the first level of progression in the learning of Computational Thinking proposed by Estebanell et al. (2018) are analyzed. The tasks are focused on learning computational languages using easily programmable objects (Bee-Bot, Cubetto, Scratch), as suggested by the authors in their training module (p.29).

Accordingly, the research objectives are:

1. To identify the types of errors in Cubetto-> Bee-Bot and Cubetto-> Scratch coding, to study the specific cases and their percentages (related to RQ1, described in the introduction).
2. To determine the evolution of the trainee teachers' ability to solve a Cubetto-> natural language translation task developed throughout the didactic sequence in three moments, initial, intermediate and final (related to RQ2 and RQ3, described in the introduction).
3. To identify the types of errors in natural language-> Cubetto coding, to study the specific cases and their percentages (related to RQ4, described in the introduction).

3.1 Context and participants

This research featuring in this article takes place at a Spanish university where early childhood and primary teachers attend their induction training. Thirty-two trainees (87.5% women, 12.5% men) participated, all of whom were attending the course of School-based professional activities for early childhood mathematics education during their fourth year of their undergraduate degree course (four years into their training).

3.2 Data collection

The data presented in this article were collected during four working sessions with the participants after they had signed an informed consent form.

The learning sequence was developed in four working sessions with Cubetto, each lasting 90 min. Bee-Bot was also used in the second session and Scratch in the third. The activities developed in the different sessions are detailed below. In all of them, the trainees carried out the tasks individually.

3.2.1 Sessions 1, 2 and 3

1. Explanation of the operating characteristics of the Cubetto or Bee-Bot robot with all its elements.
2. To explain the characteristics of the operation of the programming language with all its elements.
3. Several examples of paths are practiced using different commands.
4. The task sheet is provided with the task divided into three parts:

(a) The image of the sequence created on the Cubetto board is given (Fig. 1b). The trainee had to write the same sequence in natural language, Bee-Bot language (Fig. 2), or Scratch language, depending on whether it was the first, second or third session, respectively.
(b) The difficulties encountered when transcribing should be explained.
(c) With the Cubetto map in view (Fig. 1d) and the image of the created sequence, for the respective session, the trainee teacher has to deduce to which square the robot arrives and in which direction it is facing. (Fig. 1c).

In terms of teaching objectives, the three sessions aim to:

(a) Transcribe the Cubetto language into the other languages, natural, Bee-bot, or Scratch, according to the corresponding session.
(b) Note down the difficulties encountered in the transcription.
(c) Work on orientation and location based on a route on the board starting from specific coordinates.

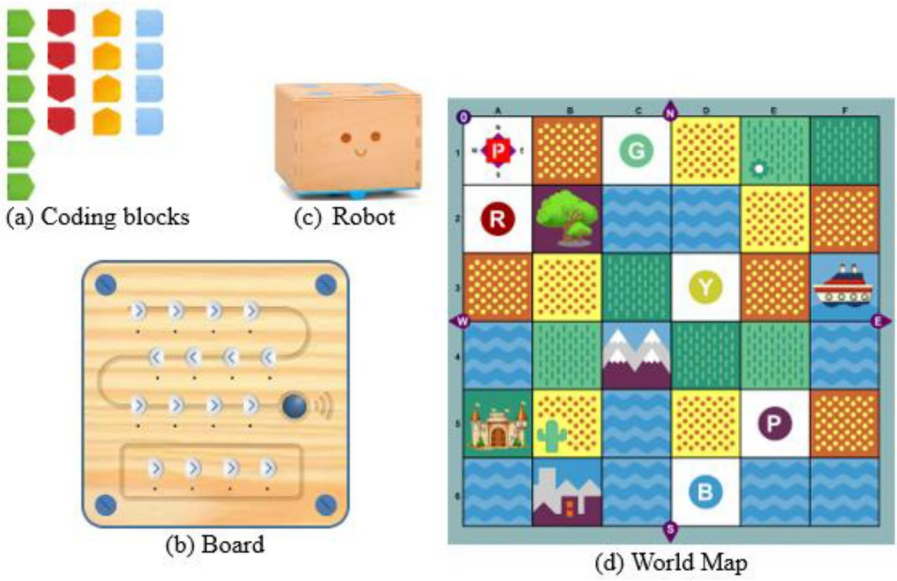

Fig. 1 Cubetto Playset

Fig. 2 Bee-Bot's directional language

3.2.2 Session 4

In the last session, the fourth one, a task was carried out that was the inverse of those performed in the previous sessions. With the Cubetto board without counters, trainees were asked to create a sequence starting from the square (5, B) facing the building (6, B) and aiming to reach the ship (3, F) without passing through squares with letters or objects with a plain background. The teaching aim of this session is to check the trainee teachers' understanding of how Cubetto works, and do translations form natural language to Cubetto language. It should be noted that, due to the lack of boards, each trainee was provided with one board and 16 printed coding blocks to work with (Fig. 1a).

The teaching objectives of this session are:

a) Transcribe into Cubetto language a resolution done in natural language of a given problem.
b) Work on orientation and location based on a route on the board with initial and final coordinates given.

Table 1 Initial categorical structure source: Berciano et al. (2023)

N.	Category designation	Explanation of interpretation
1	Good	The translation is correct.
2	One order is missing.	One command has been omitted.
3	One command is wrong.	One command has been translated incorrectly.
4	The loop is wrongly translated.	The loop has been translated incorrectly.
5	One command is missing, and the loop is wrong.	
6	One command and the loop are wrong.	
7	Several wrong commands	
8	It does not interpret the natural order of the Cubetto instruction.	The translation does not follow the order established by the Cubetto template.
9	Wrong	The translation is composed of multiple errors which makes it incomprehensible.

Table 2 Final categorical structure

N.	Category designation	Explanation of interpretation
1	Good	The translation is correct.
2	One order is missing.	One command has been omitted.
3	One command is wrong.	One command has been translated incorrectly.
4	The loop is wrongly translated.	The loop has been translated incorrectly.
5	One command is missing, and the loop is wrong.	
6	One command and the loop are wrong	
7	Several wrong commands	
8	It does not interpret the natural order of the Cubetto instruction.	The translation does not follow the order established by the Cubetto template.
9	Wrong	The translation is composed of multiple errors which makes it incomprehensible.
10	Cancelling reverse commands in the loop*	Simplification of commands (left-right).
11	Too many commands*	Excessive commands.

Source Self-generated table (*examples 1 and 2 of Sect. 4.1.1)

3.3 Data analysis

To answer RQ1, in order to examine the productions made by the participants in the didactic sequencing tasks (item 4a of sessions 2 and 3), we use the categorical analysis structure of Berciano et al. (2023), given in Table 1, as a tool to evaluate any translation errors. This tool considers any classic programming errors and some others that have emerged because of the use of Cubetto and its hardware idiosyncrasy, as described in Berciano et al. (2023). Therefore, the process of research and investigation of the errors in the tasks performed by the participants is twofold (inductive-deductive): first, using Table 1, the errors made by the participants are classified (inductively) and, second, new categories, if needed, are created (Table 2) for those errors that do not fit into the given ones (deductively).

The answers provided by the participants were individually explored and reviewed by the research team and, subsequently, the following were analyzed: (1) the alignment of the criteria used by each member of the research team in the individual analy-

ses of the participants' productions and (2) the robustness of the categorical system used, created by Berciano et al. (2023). To this end, we proceeded to calculate the inter-observation consistency index, the kappa index, obtaining a value of k = 0.936 (p = .000). This index value legitimizes, on the one hand, the usability of the categorical system used; and, on the other, the suitability and alignment of the classifications made by the research team about the trainees' productions.

To answer RQ2 and RQ3, the degree of correctness in solving the translation task described above was analyzed over three moments (item 4c of sessions 1, 2 and 3) and an inferential analysis was performed to see if the differences found were statistically significant. This study was carried out with IBM SPSS 28.0 software; non-parametric tests for repeated measures (Friedman test) were performed, due to the non-normality of the distributions.

Finally, to answer RQ4, a mixed analysis was carried out, first determining the different ways of solving the task described above (session 4), and then analyzing the number of instructions given in each of the resolutions, making a statistical description.

4 Results

To answer the research questions, we first demonstrated the degree of correctness, and the typology of errors detected in the translation task from Cubetto to Bee-Bot. Second, we analyzed the translation from Cubetto to Scratch.

4.1 Translation errors Cubetto-> Bee-Bot and Cubetto-> Scratch

In this task, given two similar sequences created on the board (Figs. 3 and 4), the trainee teachers were asked to translate them into Bee-Bot or Scratch language, respectively.

Regarding the degree of correctness of the task proposed in Session 2 (Fig. 3), it is worth noting that 28% of the participants made translation errors in this activity; these errors are categorized accordingly in Table 3. In this process, given that the kappa index was not full (k = 0.936 < 1), a triangulation of the discordant classifications was performed. This resulted in the detection of small variants in the classification of new errors, leading to the need to create two new error categories, which emerged from the qualitative analysis of the participants' productions: (1) cancellation of inverse orders in the loop and (2) orders left over. With the incorporation of these two new categories, the calculation of the Kappa index of the inter-observation differences gave a value k = 1, p-value = 0.000, implying a 100% agreement in the classification by the independent observations.

Thus, these two categories were incorporated into the initial categorical table (Table 2), giving rise to a total of 11 different types of errors that reflect the reality of participant difficulty in this type of translation task.

An analysis of the correctness of the tasks developed in Sessions 2 and 3 (Figs. 3 and 4) allows us to verify that in the former the percentage of successes is 72%, in the latter there is a certain improvement since the jobs correctly carried out account for

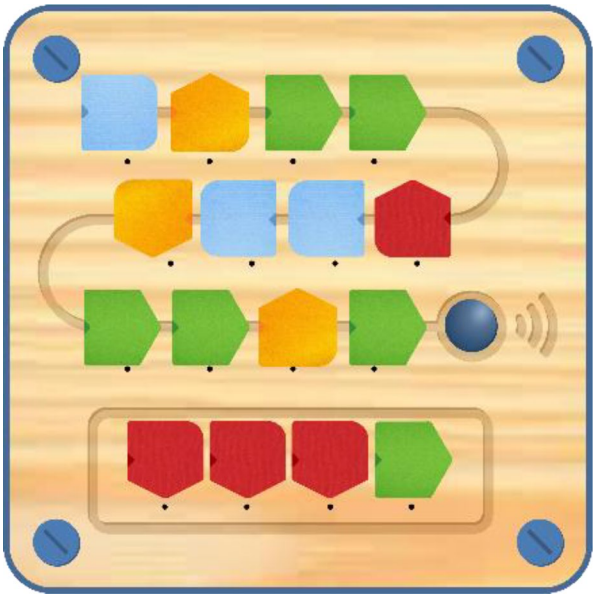

Fig. 3 Bee-Bot translation task

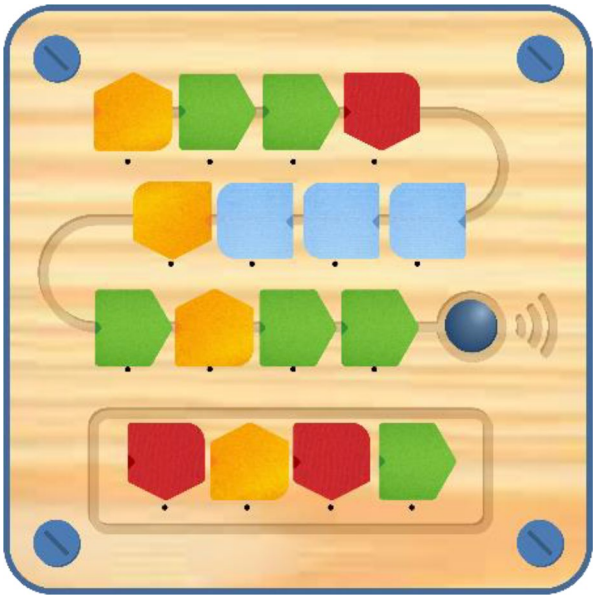

Fig. 4 Scratch translation task

Table 3 Degree of correctness of the Cubetto-Bee-Bot and Cubetto-Scratch

Translation		Cubetto–Bee-Bot	Cubetto-Scratch
Degree of correctness (%)		72	82
Degree of incorrectness (%)	N2. One order missing	3	3
	N4. The loop is wrongly translated	13	3
	N7. Several wrong commands	3	0
	N8. Failure to interpret the natural order of the Cubetto instruction	3	6
	N9. Wrong	0	3
	N10. Cancelling reverse commands in the loop	3	0
	N11. Too many commands	3	3

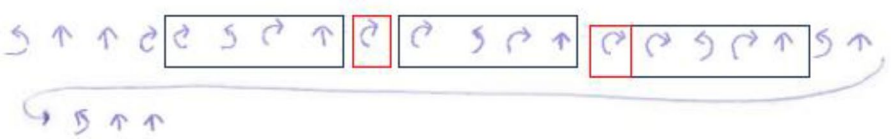

Fig. 5 Transcript of Participant A

82%. The most typical translation error is associated with the lack of understanding of the loop defined in the Cubetto board. Therefore, Table 3 incorporates the percentages of errors found.

4.1.1 Examples of errors detected in the analysis of session 2

The following examples are some of the errors detected in the translation of Cubetto into Bee-Bot by the participants (in the new categories that appeared):

1. Too many commands

   In the transcription made by Participant A (Fig. 5) we can see that when translating the loops he includes a non-existent command, a right-hand turn that is left over (framed in red), giving rise to a total of 23 commands, when only 21 are needed.

2. Cancellation of inverse commands in the loop

   In this case, in the transcription of Participant B (Fig. 6) there are, in total, 15 symbols instead of 21, due to the fact that when translating the loops he has cancelled reverse commands (he repeats 3 times “right and forward”, instead of “right, left, right and forward”).

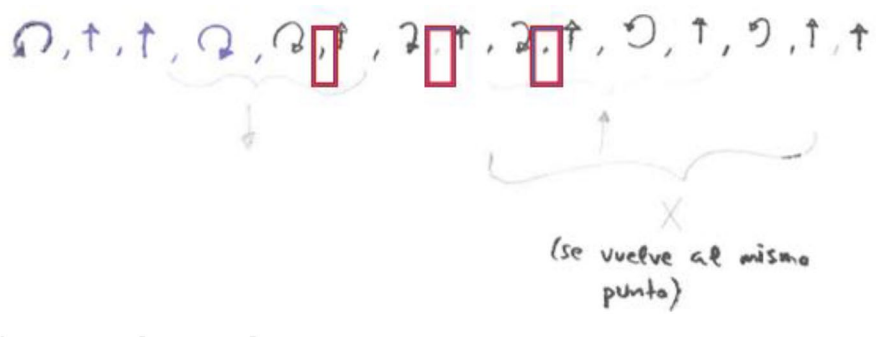

Fig. 6 Transcript of Participant B

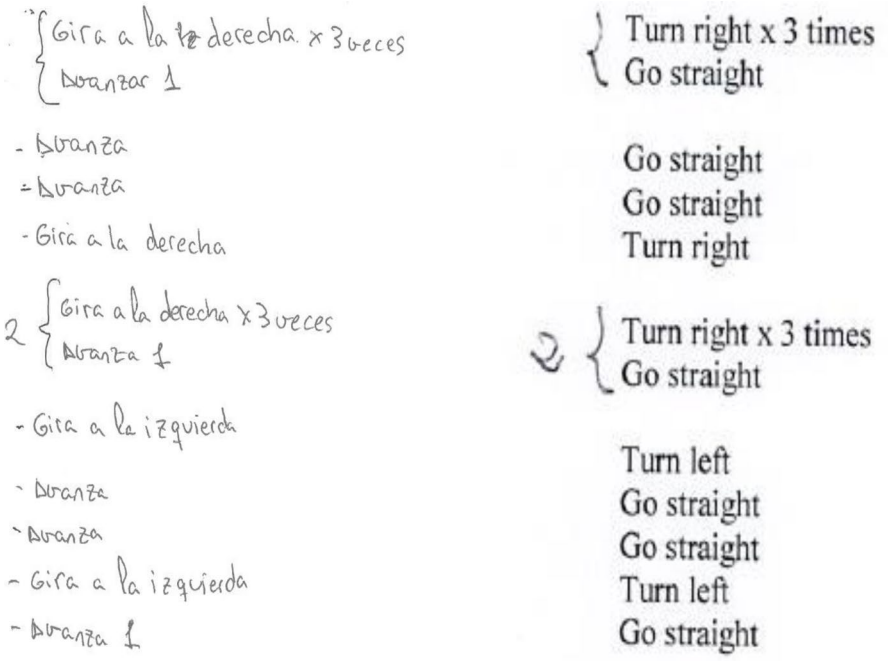

Fig. 7 Transcript of Participant C (original and its English translation, left and right respectively)

4.1.2 Examples of participant productions observed in session 3

Similarly, we show some resolutions (two incorrect and one correct) and the categorization of the error in the case it occurs is during the translation of Cubetto to Scratch:

1. Incorrect solutions of two participants C and D

   In the transcript of Participant C (Fig. 7) one command is missing and, in addition, the code is not efficient in the sense that it does not use compact commands

to reduce the number of instructions. In the transcription of Participant D (Fig. 8) there are too many commands.

Missing an order (Fig. 7).

Too many commands (Fig. 8).

2. Correct resolution
3. In the following Figure (Fig. 9), it is the correct exercise done by Participant E, who has understood and solved correctly the problem.

4.2 Evolution of the trainee teachers' ability to solve a Cubetto to natural language translation task carried out at three points during the didactic sequence (initial, intermediate and final)

As for the evolution of the final board (Objective 2), given an instruction in Cubetto and an exit point, locating it in space by giving the point of arrival and direction in which the Cubetto faces, after repeating the exercise on three different occasions, we analyzed the degree of correctness of this task in terms of the time variable, as shown in Fig. 10. In this case, the sample was reduced to 30 participants (27 women and 3 men), who each carried out the three exercises. The two participants who only completed two of the three exercises were excluded from the study.

First, we see that the results of the third exercise are (Fig. 10), a priori, better than those obtained in the first and second exercises, which leads us to assess whether this improvement is statistically significant. Similarly, we observe a slight reduction in the success rate in Exercise 2, with respect to the first and third exercises. This evo-

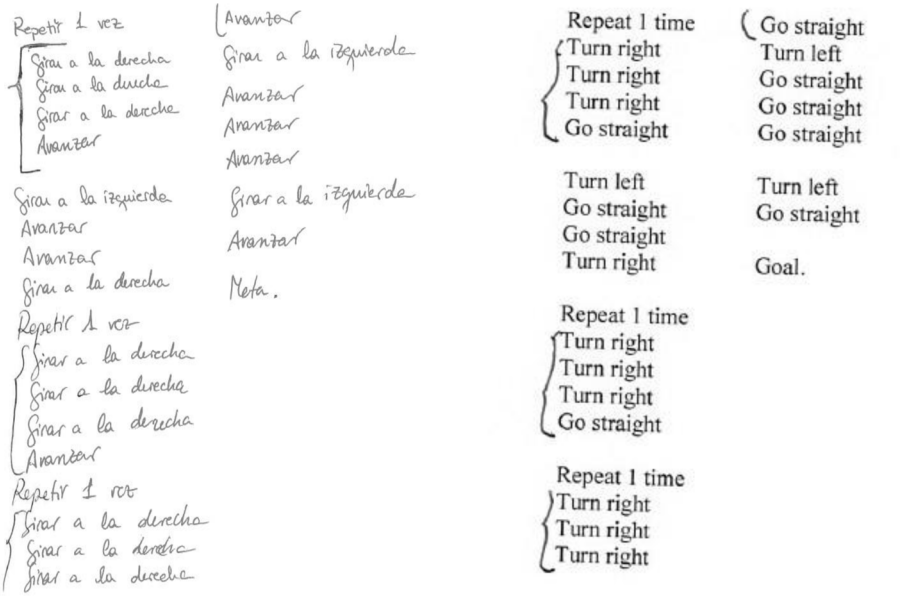

Fig. 8 Transcript of Participant D (original and its English translation, left and right respectively)

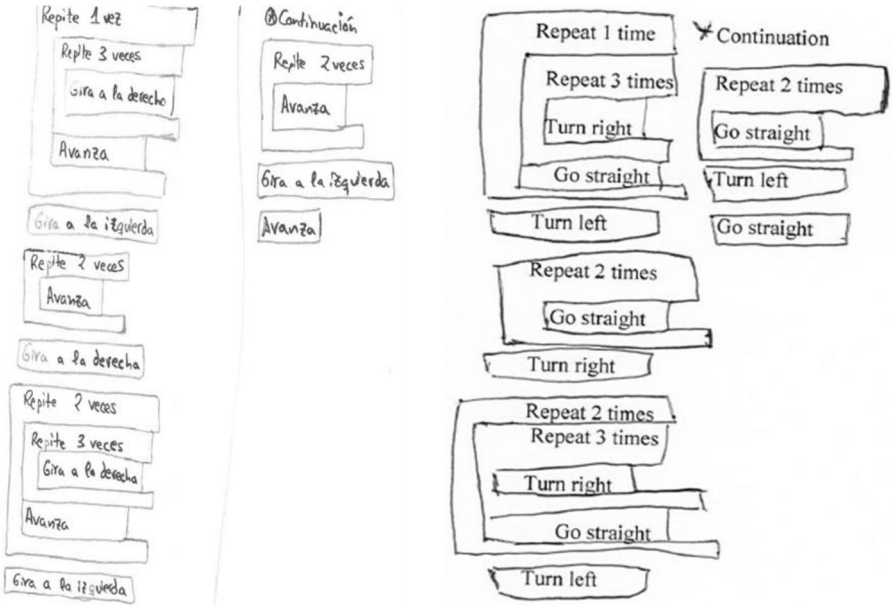

Fig. 9 Transcript of Participant E (original and its English translation, left and right respectively)

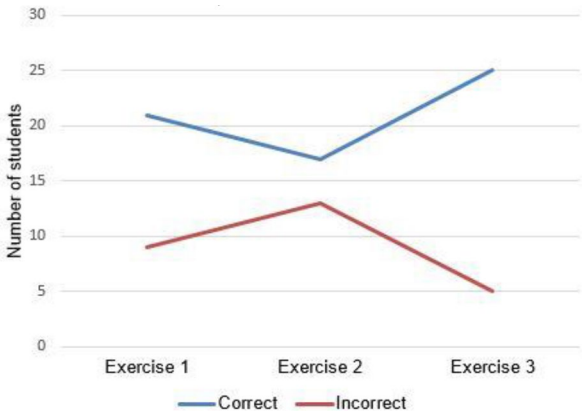

Fig. 10 Evolution of the degree of correctness of the location in the plan

lution, expressed as success rates in the resolution, gives us a clearer view of what happened: in Exercise 1, 70% solved it correctly, in Exercise 2, this percentage drops to 56.7% and in Exercise 3, the success rate is 83.3%, therefore, we see that between Exercise 1 and Exercise 3 we obtain an improvement of 13.3% in the success rate.

An inferential analysis allows us to conclude that there are statistically significant differences over time (Friedman test: Chi-square = 7.385, p-value = 0.025 < 0.05). Furthermore, the Wilcoxon test allows us to affirm that the results obtained in Exercise 3 are better than those obtained in Exercise 1 and Exercise 2, and that this improvement restricted to the comparison of Exercise 2 and Exercise 3 is statistically significant (see Table 4):

Table 4 Differences in success rates and Wilcoxon test p-value

Exercise (i)	Exercise (j)	Difference i-j	*p*-value
2	1	-0.133	0.157
3	1	0.133	0.206
3	2	0.267	0.005

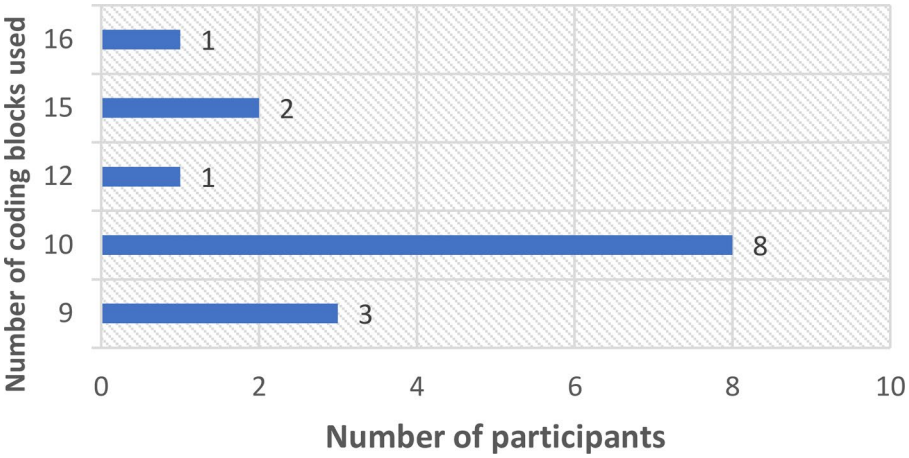

Fig. 11 Efficiency in resolution

4.3 Degree of correctness and efficiency in a task translating natural language to Cubetto

Finally, we analyze the degree of correctness in a reverse coding task, i.e., the initial position of the Cubetto (location and orientation), the final point of arrival (location and orientation) and step restrictions are given, and the participants must write the instructions on a Cubetto board, without being able to test whether their instructions are correct or not. In this case, the sample consisted of 26 participants. It is striking that only 57.60% (15 participants) solved the task correctly, while 42.30% (11 participants) made some error in its resolution.

Regarding the degree of efficiency in solving the task, understood as the ability to use the minimum number of commands, incorporating as far as possible the use of the function card, we find that only 20% of the participants who performed the task correctly do so in an optimal way (see Fig. 11), while the rest use a higher number of commands than necessary:

4.3.1 Example of optimal tasks collected from session 4

Among the optimal (and correct) examples, we show two different resolution paths, one from Participant F (Fig. 12) who places 9 tokens with two loops and another from Participant G (Fig. 13) who uses 9 tokens with three loops.

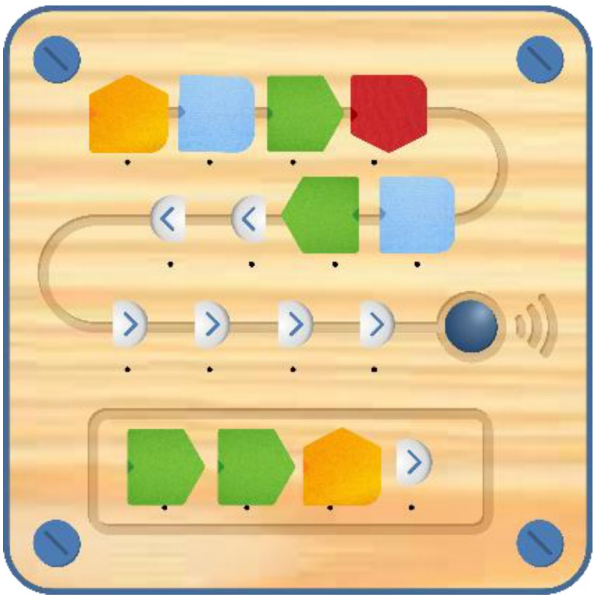

Fig. 12 Transcript of Participant F

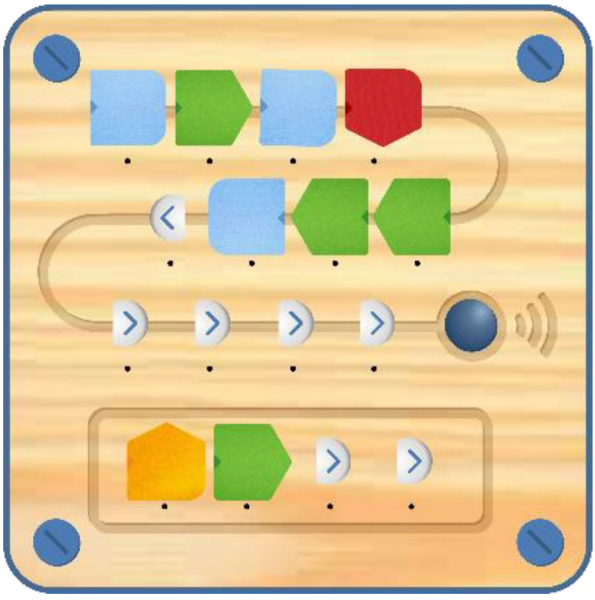

Fig. 13 Transcript of Participant F

4.3.2 Analysis of the most common types of errors

A detailed analysis of the types of errors committed reveals that in 81.81% of the erroneous cases, the errors are due to incorrect orientation of the Cubetto; that is, the Cubetto does not face the direction specified, although it does arrive at the correct

destination. Meanwhile, the other 18.18% of errors are related to both incorrect location and orientation. Below, two examples of these errors are presented:

1. Incorrect orientation and correct location

   The Participant H, on this occasion, follows all the guidelines of the task, avoiding passing through boxes with letters or boxes with objects with a plain background, but the Cubetto ends up facing east and not north (Fig. 14).

2. Incorrect location and orientation

   We highlight the exercise performed by Participant I (Fig. 15). Only 16 tiles can be placed on the panel. She places a loop on one curve of the template (blue tile) and a left turn (yellow tile) on the other curve.

5 Discussion and conclusions

Throughout this work, we have seen the importance of performing translation tasks between programming languages. The findings highlight the significance of correctly linking different coding systems through these tasks. And the results allow us to clearly answer to the research questions, described in the introduction: RQ1) What are the errors made by teachers in initial training in the translation from Cubetto language to Bee-Bot and Scratch languages, their nature, and the extent of casuistry involved?; RQ2) How does the trainee teachers' ability to solve a translation task from the Cubetto language to natural language evolve throughout the learning

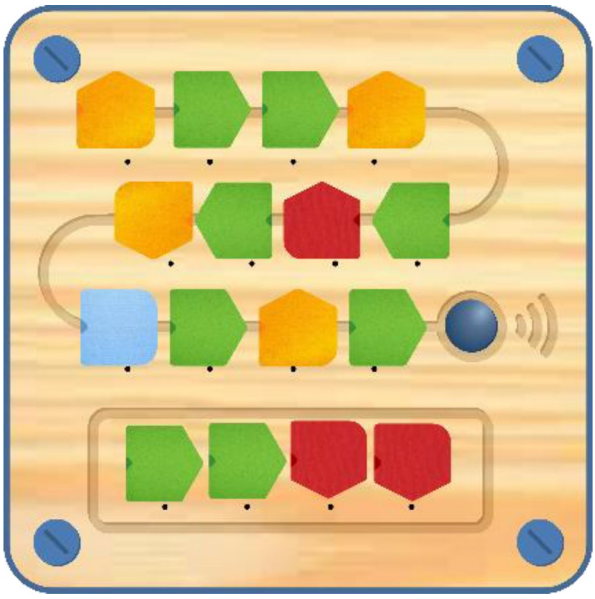

Fig. 14 Transcript of Participant H

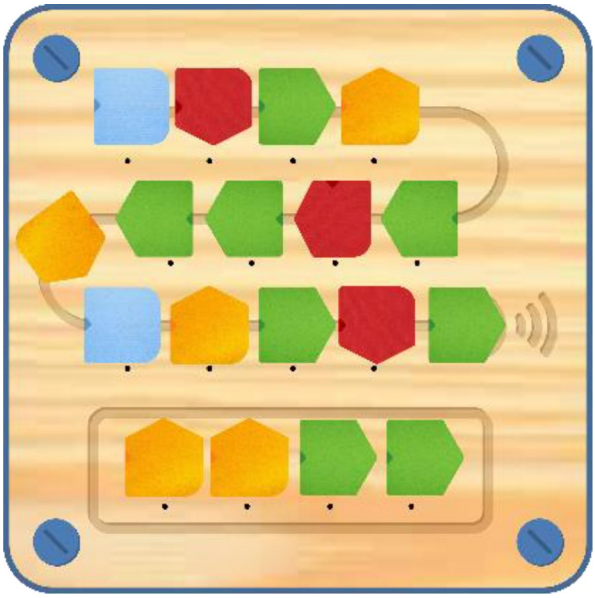

Fig. 15 Transcript of Participant I

sequence?; RQ3) To what extent does this learn-sequence help trainee teachers in the translation skills of inter-imperative languages? and RQ4) What are the errors by trainee teachers in the translation from natural language to Cubetto?

With respect to RQ1, various difficulties have been observed in the way the trainee teachers carry out the tasks. Some errors that occurred in the task resolution are related to common errors in programing (25% of total exercises in Cubetto-Bee-bot translation and 12% in Cubetto-Scratch translation), as described in Berciano et al. (2023), for example a loop incorrectly written (13%, 3%, respectively), one order missing, one command incorrectly written. This reality show how coding is one of the most difficult aspects in programing learning, and the nature of the errors should be analyzed (Fernández-Medina et al., 2014).

But another special error appears in the translation tasks (Cubetto-Bee-bot, Cubetto-Scratch), who is related to the characteristics of the Cubetto educational robot, completely defined by its hardware (It does not interpret the natural order of the Cubetto instruction) and emerges in translation to Bee-bot and to Scratch. This result was unexpected since the tasks, considered easy for adults, and the educational robot, designed for early childhood use, were supposed to be simple. Moreover, despite the small sample size, it is striking how Cubetto was found to be more complex than expected; its hardware is designed with a simple coding system for use with young children, so it was not hypothesized that trainee teachers might have difficulties to use it. The difficulties of the educational robot, linked to the peculiarities of its coding language, have become evident throughout the learning sequence, regardless of the target programming language. These difficulties complement those evidenced by Angeli (2022) in her study with another type of programmable robot (Lego Wedo); in which she concluded that teachers lacked preparation in their initial training to face the didactic challenges involved in teaching computational thinking to the very young learners. We support the need for trainee teachers to receive

adequate training to be able to develop computational thinking at the user level; to later, be able to assume tasks at the levels of teacher and reflective teacher (according to Estebanell et al., 2018).

As an added value in this work, we emphasize that the detection of difficulties with this educational robot has allowed us to establish a categorical system consisting of eleven different types of errors. This categorial system provides a solid basis for analyzing the difficulties of any student using this hardware. Furthermore, it can be used to propose complementary activities that help to overcome these difficulties successfully, measuring their efficiency by the number and type of errors done. In this sense, the detection of errors can be used to design a scaffolding system of tasks that will help trainee teachers to identify their errors and overcome them, improving their computational thinking. Indeed, this categorical system complements and extends the one developed by Berciano et al. (2023) and can be used as a tool to measure coding ability for everyone, not only for children aged three to six years old in tasks carried out with Cubetto (e.g., Marinus et al., 2018).

With respect to RQ2, the performance of the inverse translation task (Cubetto to natural language), the results show that most of the trainee teachers performed this task with a huge number of commands, highlighting certain difficulties in coding optimization in a context of basic complexity. This result complements the results obtained by Berciano et al. (2023), which detailed the difficulties encountered by the trainee teachers in the inverse coding process (Cubetto-natural language). So, this work opens new lines of research associated with (1) investigating whether the lack of code optimization occurs in more sophisticated coding contexts, and (2) categorizing the reasons that lead some trainee teachers to perform simple translation tasks with a huge number of commands.

With respect to RQ3, considering the favorable results of this research, we should highlight that the learning sequence has served to significantly improve the trainee teachers' ability to translate between programming languages. These results are in line with research evidence showing the positive effects of learning computer programming on the development of students' algorithmic thinking and debugging skills (Bers et al., 2014; Resnick et al., 2009; Resnick & Rusk, 2020; Roque & Rusk, 2019), and are completely new for the case of coding. In our case, we add the idiosyncrasy of working with coding as one of the dimensions of the learning of computer programing, because of this dimension had so far not been valued until now, even though it is considered as a new skill to understand the languages associated to computational thinking and its development (Bers, 2018; Delacruz, 2020; DiSessa, 2001; Duin & Tham, 2019; Resnick & Rusk, (2020).

With respect to RQ4, only the 57.60% of participants do correctly the exercise. With respect to the errors detected, the main type is related to incorrect orientation of the Cubetto (81.81% of the erroneous cases). This result is aligned with the main error detected by Seckel et al. (2021) for the analogous case with the Bee-Bot. In the same lane, it is notably that we find that only 20% of the participants who performed the task correctly do so in an optimal way; so, these results open new research lines to analyze which are the reasons involved.

Finally, like previous research that has highlighted the need for more in-depth studies regarding the ability of trainee teachers, to correctly perform algorithms and debugging and the difficulties encountered in these tasks (Angeli et al., 2016; Bers

et al., 2022; Yadav et al., 2017). It is necessary that research goes deeper and analyzes the types of errors that students do in programming tasks to better understand this process (Fernández- Medina et al., 2014). So, the aim of this work has been to respond to the need mentioned, presenting as a novelty of research the interest in better understanding the teaching-learning processes of programming through the detection of errors made by students. In this sense, we have seen how the learning sequence specifically designed to work on the translation between the programming languages of Cubetto, Bee-Bot, Scratch and natural language has helped teachers in initial training to improve their problem solving skills in the Cubetto-natural language case, where the improvement is remarkable and coincides with emerging skills in other studies (Bers et al., 2014; Khine, 2017; Yu & Roque, 2019; Clarke-Midura et al., 2019), although there is still a long way to go, as pointed out by Bers et al. (2023).

We would like to emphasize the importance of pursuing this type of research in the teaching and learning of computational thinking through programming. More specifically, research should focus on translation tasks between programming languages where coding is crucial and the establishment of links between equivalent representation systems is valued.

5.1 Limitations and future research lines

As limitations of the study, we must point out that the sample was small and that, therefore, as a future improvement and line of research, we would like to extend this study to a larger sample. Likewise, the aim is to investigate whether the lack of code optimization occurs in more sophisticated coding contexts, as well as to categorize the reasons that lead some students to carry out simple translation tasks with a very high number of commands.

Funding Open Access funding provided thanks to the CRUE-CSIC agreement with Springer Nature.

Data availability The datasets generated during and/or analyzed during the current study are available on request from the corresponding author A. B. The data are not publicly available because they contain information that could compromise research participant privacy.

Declarations

Ethical approval This paper has not been published before; it is not under consideration for publication anywhere else; it has been approved by all co-authors at the university where the work has been carried out.

Conflict of interest The authors have no financial or non-financial interests to disclose.

References

Angeli, C. (2022). The effects of scaffolded programming scripts on pre-service teachers' computational thinking: Developing algorithmic thinking through programming robots. *International Journal of Child-Computer Interaction*, *31*, 100329. https://doi.org/10.1016/j.ijcci.2021.100329

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework: Implications for teacher knowledge. *Journal of Educational Technology & Society*, *19*(3), 47–57.

Austin, J., Baker, H., Ball, T., Devine, J., Finney, J., De Halleux, P., & Stockdale, G. (2020). The BBC micro: Bit: From the UK to the world. *Communications of the ACM*, *63*(3), 62–69. https://doi.org/10.1145/3368856

Bau, D., Gray, J., Kelleher, C., Sheldon, J., & Turbak, F. (2017). Learnable programming: Blocks and beyond. *Communications of the Acm*, *60*, 6, 72–80. https://doi.org/10.1145/3015455

Berciano, A., Cuida, A., & Novo, M.L. (2023). Translation errors between sequential programming languages in Cubetto activities. *RED. Revista de Educación a Distancia, 23*(76), 1–23. https://doi.org/10.6018/red.552581

Bers, M. U. (2017). *Coding as a playground: Programming and computational thinking in the early childhood classroom*. Routledge. https://doi.org/10.4324/9781315398945

Bers, M. U. (2018). Coding and computational thinking in early childhood: The impact of ScratchJr in Europe. *European Journal of STEM Education*, *3*(3), 8. https://doi.org/10.20897/ejsteme/3868

Bers, M. U. (2019). Coding as another language: A pedagogical approach for teaching computer science in early childhood. *Journal of Computers in Education*, *6*(4), 499–528. https://doi.org/10.1007/s40692-019-00147-3

Bers, M., & Sullivan, A. (2019). Computer Science Education in Early Childhood: The case of ScratchJr. *JITE: IIP*, *18*(1), 113–138. https://doi.org/10.28945/4437

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. https://doi.org/10.1016/j.compedu.2013.10.020

Bers, M. U., Strawhacker, A., & Sullivan, A. (2022). The state of the field of computational thinking in early childhood education, *OECD Education Working Papers*, No. 274, OECD Publishing, Paris. https://doi.org/10.1787/3354387a-en

Bers, M. U., Blake-West, J., Kapoor, M. G., Levinson, T., Relkin, E., Unahalekhaka, A., & Yang, Z. (2023). Coding as another language: Research-based curriculum for early childhood computer science. *Early Childhood Research Quarterly*, *64*, 394–404. https://doi.org/10.1016/j.ecresq.2023.05.002

Brennan, K., & Resnick, M. (2012). Using artifact-based interviews to study the development of computational thinking in interactive media design. *Annual Meeting of the American Educational Research Association*, Vancouver, B.C, 2012.

Cameron, R. (2010). Mixed methods in VET research: Usage and quality. *International Journal of Training Research*, *8*(1), 25–39. https://doi.org/10.5172/ijtr.8.1.25

Ching, Y. H., Hsu, Y. C., & Baldwin, S. (2018). Developing computational thinking with educational technologies for young learners. *Tech Trends*, *62*, 563–573. https://doi.org/10.1007/s11528-018-0292-7

Clarke-Midura, J., Lee, V. R., Shumway, J. F., & Hamilton, M. M. (2019). The building blocks of coding: A comparison of early childhood coding toys. *Information and Learning Sciences*, *120*(7/8), 505–518. https://doi.org/10.1108/ILS-06-2019-0059

Cubetto Universe – New Maps and Adventures (2023). (n.d.). https://www.kickstarter.com/projects/primotoys/cubetto. Accessed December 11.

Delacruz, S. (2020). Starting from scratch (Jr.): Integrating code literacy in the primary grades. *The Reading Teacher*, *73*(6), 805–812. https://doi.org/10.1002/trtr.1909

DiSessa, A. (2001). *Changing minds: Computers, learning, and literacy*. MIT Press. https://doi.org/10.7551/mitpress/1786.001.0001

Duin, A. H., & Tham, J. C. K. (2019). Cultivating code literacy: Course redesign through advisory board engagement. *Communication Design Quarterly Review*, *6*(3), 44–58. https://doi.org/10.1145/3309578.3309583

Estebanell, M., López, V., Peracaula, M., Simarro, C., Cornellà, P., Couso, D., González, J., Alsina, A., Badillo, E., & Heras, R. (2018). *Pensament Computacional en la formació de mestres.* Guia didàctica. Servei de Publicacions UdG.

Fernández-Medina, C., Pérez-Pérez, J. R., Paule-Ruiz, M. P., & Álvarez García, V. M. (2011). Assistance in computer programming learning using educational data mining and learning analytics COLMENA: Collaborative knowledge and user classification environment based on programming experience. In *Proceedings of the VIII Multidisciplinary Symposium on Design and Evaluation of Digital Content for Education* (50–58).

Fernández-Medina, C., Pérez-Perez, J. R., Paule-Ruiz, M. P., & Álvarez-García, V. M. (2014). Aprendizaje de la programación guiado por los errores de compilación. *JENUI 2014. XX Jornadas de Enseñanza Universitaria de la Informática.* Oviedo: Universidad de Oviedo. Escuela de Ingeniería Informática, 371–378. https://hdl.handle.net/2099/15498.

Flannery, L. P., Silverman, B., Kazakoff, E. R., Bers, M. U., Bontá, P., & Resnick, M. (2013). Designing ScratchJr: Support for early childhood learning through computer programming. *In Proceedings of the 12th international conference on interaction design and children* (1–10). https://doi.org/10.1145/2485760.2485785

García-Peñalvo, F. J., & Mendes, A. J. (2018). Exploring the computational thinking effects in preuniversity education. *Computers in Human Behavior*, *80*, 407–411. https://doi.org/10.1016/j.chb.2017.12.005

García-Valcárcel, A., & Caballero-González, Y. (2019). Robótica para desarrollar El pensamiento computacional en Educación Infantil. *Comunicar: Revista científica Iberoamericana De comunicación Y educación*, *27*(59), 63–72. https://doi.org/10.3916/C59-2019-06

Halverson, E. R., & Sheridan, K. (2014). The maker movement in education. *Harvard Educational Review*, *84*(4), 495–504. https://doi.org/10.17763/haer.84.4.34j1g68140382063

Hamilton, M., Clarke-Midura, J., Shumway, J. F., & Lee, V. R. (2020). An Emerging Technology Report on Computational toys in early childhood. Tech. *Know Learn*, *25*(1), 213–224. https://doi.org/10.1007/s10758-019-09423-8

Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM*, *59*(8), 26–27. https://doi.org/10.1145/2955114

Khine, M. S. (2017). Robotics in STEM Education. Springer. https://doi.org/10.1007/978-3-319-57786-9

Knuth, D. E. (2005). *The art of computer programming*. Pearson Education.

Ladzowska, E., & Patterson, D. (2013). Students of All Majors Should Study Computer Science. *Chronicle of Higher Education*. https://bit.ly/3Oj55j4

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J., & Werner, L. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*(1), 32. https://doi.org/10.1145/1929887.1929902

Macrides, E., Miliou, O., & Angeli, C. (2022). Programming in early childhood education: A systematic review. *International Journal of Child-Computer Interaction*, *32*, 100396. https://doi.org/10.1016/j.ijcci.2021.100396

Marinus, E., Powell, Z., Thornton, R., McArthur, G., & Crain, S. (2018, August). Unravelling the cognition of coding in 3-to-6-years old: The development of an assessment tool and the relation between coding ability and cognitive compiling of syntax in natural language. In *Proceedings of the 2018 ACM Conference on International Computing Education Research* (133–141). https://doi.org/10.1145/3230977.3230984

Martín, G., Toledo, G., & Cerverón, V. (2002). *Fundamentos De Informática Y Programación*. Universidad de Valencia.

Misirli, A., & Komis, V. (2023). Computational thinking in early childhood education: The impact of programming a tangible robot on developing debugging knowledge. *Early Childhood Research Quarterly*, *65*, 139–158. https://doi.org/10.1016/j.ecresq.2023.05.014

Papert, S. (1980). *Mindstorms: Children, computers and powerful ideas*. Harvester.

Pridmore, L., Lardieri, P., & Hollister, R. (2010). National Cyber Range (NCR) automated test tools: Implications and application to network-centric support tools. *In 2010 IEEE AUTOTESTCON* (pp. 1–4). IEEE.

Real Decreto 95 (2022). / de 1 de febrero, por el que se establece la ordenación y las enseñanzas mínimas de la Educación Infantil. *Boletín Oficial del Estado*, *BOE, núm. 28*, 2 de febrero de 2022, 2022 – 1654. https://www.boe.es/buscar/pdf/2022/BOE-A-2022-1654-consolidado.pdf

Resnick, M., & Rusk, N. (2020). Coding at a crossroads. *Communications of the ACM*, *63*(11), 120–127. https://doi.org/10.1145/3375546

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., & Kafai, Y. (2009). Scratch: Programming for all. *Communications of the ACM*, *52*(11), 60–67. https://doi.org/10.1145/1592761.1592779

Roque, R., & Rusk, N. (2019). Youth perspectives on their development in a coding community. *Info Learning Sci*. https://doi.org/10.1108/ILS-05-2018-0038

Seckel, M. J., Vásquez, C., Samuel, M., & Breda, A. (2021). Errors of programming and ownership of the robot concept made by trainee kindergarten teachers during an induction training. *Education and Information Technologies*, *27*(3), 2955–2975. https://doi.org/10.1007/s10639-021-10708-8

Segatto, R., & Teixeira, A. C. (2021). Utilização do Robô Cubetto em um Processo De Formação Docente para Professores Da Educação Básica na Área Da Robótica Educacional. *Ensino De Ciências E Tecnologia em Revista–ENCITEC*, *11*(1), 219–236. https://doi.org/10.31512/encitec.v11i1.390

Selby, C. (2014). *How can the teaching of programming be used to enhance computational thinking skills?* (Doctoral dissertation, Thesis (Doctoral)–Southampton Education School, University of Southampton, Hampshire).

Strawhacker, A., Lee, M., & Bers, M. U. (2018). Teaching tools, teachers' rules: Exploring the impact of teaching styles on Young Children's programming knowledge in ScratchJr. *International Journal of Technology and Design Education*, *28*(2), 347–376. https://doi.org/10.1007/s10798-017-9400-9

Sullivan, A., Bers, M., & Pugnali, A. (2017). The impact of user interface on Young Children's computational thinking. *J Inf Tech Educ Innov Pract*, *16*(1), 171–193. https://doi.org/10.28945/3768

Vee, A. (2017). *Coding literacy: How computer programming is changing writing*. MIT Press. https://doi.org/10.7551/mitpress/10655.001.0001

Wing, J. M. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33–35. https://doi.org/10.1145/1118178.1118215

Wing, J. M. (2011). *Research notebook: Computational thinking—what and why? The link magazine*. Pittsburgh: Spring. Carnegie Mellon University. Retrieved from: https://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Yadav, A., Stephenson, C., & Hong, H. (2017). Computational thinking for teacher education. *Communications of the ACM*, *60*(4), 55–62. https://doi.org/10.1145/2994591

Yu, J., & Roque, R. (2019). A review of computational toys and kits for young children. *Int'l J Child-Computer Interaction*, *21*(C), 17–36. https://doi.org/10.1016/j.ijcci.2019.04.001

Zapata, J. M., Jameson, E., Ros, M. Z., & Merrill, D. (2021). El Principio De Activación en El Pensamiento Computacional, las Matemáticas Y El STEM: Presentación Del número especial. *Revista De Educación a Distancia (RED)*, *21*(68), 1–9. https://doi.org/10.6018/red.498531

Authors and Affiliations

Ainhoa Berciano[1] · Astrid Cuida[2] · María-Luisa Novo[2]

✉ Ainhoa Berciano
ainhoa.berciano@ehu.eus

Astrid Cuida
mariaastrid.cuida.gomez@uva.es

María-Luisa Novo
marialuisa.novo@uva.es

[1] Department of Didactics of Mathematics, Experimental and Social Sciences, University of the Basque Country (UPV/EHU), Leioa, Spain

[2] Department of Didactics of Experimental, Social Sciences and Mathematics, University of Valladolid (UVa), Valladolid, Spain