Teaching Computational Thinking to K-12 Educators through Distance Learning

Eduardo C. Oliveira[1], **Roberto A. Bittencourt**[1]

[1] UEFS – Universidade Estadual de Feira de Santana
Av. Transnordestina, s/n, Novo Horizonte
Feira de Santana – BA, Brasil – 44036-900

educdoliveira@gmail.com, roberto@uefs.br

Abstract. *This paper reports an experience of teaching Computational Thinking (CT) to K-12 educators through an online Scratch programming short course. The meeting of CT and modern technologies is extending the use of coding in K-12 education. An essential requisite for this to prosper is the teacher preparation. However, most current teacher training programs fail to supply with pedagogical knowledge for educators to teach CT. Thus, it is critical to present CT to K-12 teachers, providing proper conditions to learn and use its concepts. In this context, this work aimed to design and implement an online Scratch programming course for K-12 educators. Results suggested that using Scratch to teach CT for K-12 educators is adequate, and analyzing educators context when presenting tutorial Scratch projects is relevant.*

Resumo. *Este artigo relata uma experiência de ensino de Pensamento Computacional (PC) através de programação em Scratch para educadores do ensino fundamental e médio. O encontro do pensamento computacional com as tecnologias modernas está ampliando o uso da programação na educação básica. Um requisito essencial para fazer este tipo de iniciativa prosperar é a preparação dos professores. No entanto, a maioria dos programas atuais de treinamento de professores em tecnologias de informação e comunicação falha em fornecer conhecimento pedagógico para o ensino de PC. Por isso, é fundamental apresentar o PC aos professores da educação básica, proporcionando condições adequadas para que aprendam e usem seus conceitos. Neste contexto, este trabalho teve como objetivo projetar e implementar um curso de programação online para educadores do ensino fundamental e médio. Os resultados sugerem que o uso do Scratch para ensinar PC para professores da educação básica é adequado e que é importante analisar previamente o contexto dos educadores ao introduzir os projetos nos tutoriais de programação.*

1. Introduction

The rising of digital technologies in the last decades has brought new challenges for society. Since modern devices provide adequate solutions to a range of day-to-day problems, scientific community has been arguing how humans can utilize these skills. In this context, Wing (2006) has established the idea of Computational Thinking, which is a skill set that involves problems solving, systems design and abstraction processes, based on the core elements of Computer Science.

Wing (2006) highlights that Computational Thinking (CT) is applicable for everyone, in spite of its concepts being relied on Computer Science (CS). In an educational scenario, for instance, regular activities such as reading, writing and solving arithmetic operations can be boosted by CT. Given that, researchers have been promoting experiences around the globe in order to assess the impact of CT on K-12 educators.

Yadav et al. (2014) have run a project in the United States indicating how Computational Thinking disciplines can be included into an educational psychology course for preservice teachers, as long as CT is firmly related to abstraction and problem solving and processes. Likewise, Kaila et al. (2018) have designed a programming course for teacher education students in Finland, determining CT as a learning goal. In Brazil, Silva et al. (2017) have implemented a CT training project for K-12 educators, seeking to broadcast knowledge about CT and Computing as an interdisciplinary science.

Since programming skills are progressively demanded, most recent research on Computational Thinking is associated to teaching Computer Science fundamentals [Kaila et al. 2018]. In current K-12 curricula, the presence of CS modules is becoming increasingly introduced. Scientists argue that it is essential to include content in K-12 education courses, considering that technology is everywhere in modern lives. That could lead students to eventually become not only consumers, but true technology makers. Gal-Ezer and Stephenson (2014) expressed that a critical requirement for this to succeed is teacher preparation. Despite some countries have formal CS educational prerequisites, they share the same concerns to ensure that teachers are provided with technical, content and pedagogical knowledge to teach Computer Science. Hence, the development of appropriate CT qualification for K-12 educators is currently an issue.

However, Gal-Ezer and Stephenson (2014) emphasized that there is indication of various countries moving effectively forward. The first evidence is the ongoing updates to standards and curricula, focusing on Computer Science fundamentals. Subsequently, the increase of CS teachers' communities also contributes to professional development and advocate for better CS education. Last, the application of new technologies in education, along with the establishment of new educational projects concerning how to make CS more appealing to all students, suggest a progress in K-12 educators qualification.

Nevertheless, Computational Thinking and its potential is not contemplated by a large section of the scientific community, specially in Brazil. Most teachers still do not know the basics of CT or how it can power not only CS modules, but various other disciplines as well. Therefore, it is critical to present CT to K-12 educators. In addition, it is also necessary to provide those teachers with suitable conditions to learn and apply CT concepts. In this context, this work aimed to design and implement an online programming course for K-12 educators.

2. Background

Here we describe definitions of computational thinking and some related work.

2.1. Computational Thinking

The definition of Computational Thinking (CT) has been frequently updated since its establishment in the last decade. Relied on core Computer Science (CS) principles, CT was originally defined by Jeannette Wing as an umbrella that covers from systems design

to problem solving and abstraction processes [Wing 2006]. Currently, CT is described as the thought process that includes formulating problems and their solutions. These solutions are expressed in a form that both humans or machines can accomplish.

The most critical thought process in Computational Thinking is the abstraction. According to Wing (2006), abstraction consists of a high-level thought process used to gather basic and mutual properties in a set of entities while overlooking unimportant differences among them. Once this procedure can be regularly applied, it contributes to design and implement large systems by humans, as well as the capacity to scale and deal with complexity.

Another essential aspect of Computational Thinking is that it should be reached by everyone involved in K-12 education, including teachers, parents and students [Wing 2006]. A suitable approach for this is to bring CT to the classroom. As stated by Barr and Stephenson (2011), CT could be specially applied in a range of disciplines that require reasoning, being manifested in class through problem-solving activities. This could primarily enable students to employ different mechanisms to solve problems, to feel comfortable with trial and error, and to be in an adequate atmosphere of finding out things as a group. At last, students could be able to understand that problems can be solved in multiple ways, and develop positive expectations when designing these solutions.

2.2. K-12 Teacher Training in Computing

Scientists have been arguing on the demand to prepare teachers in Computational Thinking basics. One research suggested short-term approaches to provide teachers a suitable preparation for applying CT in K-12 education [Barr and Stephenson 2011], such as building a connection between Computer Science courses with educators, along with supplying teachers with professional qualification through learning societies offered by teachers experienced in CT. These could disseminate the role of CT in non-CS disciplines and to contribute to the production of relevant materials.

In addition, Yadav et al. (2014) indicated that students could be able to learn Computational Thinking by observing their teachers, and subsequently developing their own procedures to solve problems. Previous studies also have attempted to increase teachers' perceptions about the relevance of CT and Computer Science as an educational tool to be used in problem-solving scenarios, as well as modifying their attitudes towards Computing. Regardless of the differences in pedagogical contexts, language of instruction, and terminology (Informatics, Computing, Computer Science), there are a variety of approaches towards Computing in K-12 education [Gal-Ezer and Stephenson 2014]. In various countries, such as Australia, Finland, Israel, New Zealand, the United Kingdom and the United States, curriculum analysts have reviewed and updated their standards and curricula to encourage CS and CT ideas.

2.3. Related Work

In recent years, there have been some efforts to qualify K-12 teachers in Computational Thinking and Computer Science. Yadav et al. (2014), for example, have run a project to include CT in an educational psychology course for pre-service teachers in the United States. Their goal was to evaluate the impact of CT modules on the course as well as teachers' attitudes towards Computing. The results revealed that the work could raise educators' comprehension about CT and how to integrate it into their classrooms.

In a like manner, Gal-Ezer and Stephenson (2014) have studied the advance of K-12 standards around Computer Science education in the United States and Israel, focusing on challenges such as teacher qualification. This work showed that both countries are making similar steps towards Computing in education, including the addition of specific goals in regards to Computing in the standards documents. Israel and the United States also share consonant challenges on this subject, such as the lack of projects to ensure that educators are supplied with content and pedagogical knowledge to teach CS.

Kaila et al. (2018) have designed and carried out a project for teacher education students in Finland. Their goal was to teach programming and Computational Thinking fundamentals, including approaches and tools for education. The outcomes indicated that the great majority of students completed the workshop, expressing positive feedback from the lessons and being inclined to modify their attitudes towards programming and CT.

Additionally, Santos et al. (2017) have analyzed Computing Teacher Education pre-service programs in Brazil. This study aimed to identify and discuss the major challenges faced by Computing Teacher Education students. Results from this work expressed a lack of integration between theoretical and practical approaches in terms of teacher qualification and the need to disseminate this kind of program.

Similar to our work, other studies analyze K-12 pre-service teacher qualification in Computing, including Gal-Ezer and Stephenson (2014) and Santos et al. (2017). However, we take a step ahead to design and implement an online short-term course for teachers. Furthermore, our work is also similar to the work of Yadav et al. (2014) and Kaila et al. (2018), as we also aim to teach Computational Thinking fundamentals. Our audience, however, is not pre-service teachers, but professional K-12 educators.

3. Methodology

Here we describe participants, our intervention, and the process of data collection and analysis.

3.1. Participants

The workshop was disseminated on social media, inviting K-12 teachers from Brazil to register in the course by filling in a document on Google Forms[1]. At first, course members read an informed consent form and, if voluntarily accepted to participate in the research, they were required to provide demographic data, such gender and age, in addition to information related to their education and employment, including college major and education level they taught. The first 20 submissions with full approval to participate in the research were enrolled in the course.

The intervention had a total of 70% female and 30% of male entries. The course members' average age was 38.8, with a standard deviation of 9.13. Furthermore, 40% of participants held undergraduate degree in Education, while degrees in Biology, History and Mathematics totalled 10% each. The greater part of participants were elementary and middle school educators (65%), whilst 15% were high school teachers, and 20% worked on both educational stages.

[1] https://forms.google.com

3.2. Intervention

The intervention was carried out in 2018 through a distance education course for K-12 teachers. Using Scratch 2.0[2] as the programming platform and Google Classroom[3] as the virtual learning environment, the five-week workshop was divided into three units, consisting primarily of uploaded video tutorials, assignments on Scratch and group conversations on Google Classroom. The course planning is shown in Table 1.

Unit 1 ran in the first week and introduced our goals in this project and the tools to be used. Participants were also presented to the core Computational Thinking fundamentals, including the steps used to solve problems. Lastly, learners were asked to discuss the application of CT principles in K-12 education.

Unit 2 focused on Scratch assignments. In weeks two, three and four, tutorial lessons were uploaded and sorted by the existing command groups on Scratch. The lessons in the second week were dedicated to the creation of interactive stories using mainly *Events*, *Looks* and primary *Control* blocks. In the third week, video lessons indicated how to produce animations by using *Data*, *Motion*, *Operators* and advanced *Control* blocks. In the fourth week, instructive videos covered the development of a simple game, applying *Sensing* and advanced *Events* blocks. The fourth week also brought tutorials indicating how to implement specific pieces of scripts through the use of *More Blocks* tab. At the end of each week in Unit 2, course members were required to produce a Scratch project, incorporating the blocks just learned into their activities, or to answer a quiz concerning topics seen in current week.

Unit 3 gave an introduction of Computer Science basics, linking these to Computational Thinking fundamentals, as well as to Scratch blocks used in previous weeks. Hence, in the fifth week, participants were able to comprehend programming principles, such as logical and relational operations, sequences, parallelism, selection and control structures. At last, they were presented to the concept of variables and its close association with *Data* blocks on Scratch, as well as the use of functions and its connection to individual blocks on Scratch.

3.3. Data Collection and Analysis

To build the narrative of our experience, we analyzed the following data: participants' discussions in Google Classroom forums, participants' Scratch projects and instructor's notes. Participants also answered a pre-intervention demographic survey. Survey data led to descriptive statistics of the participants, and the other data suffered simple qualitative analysis procedures in order to generate a temporal narrative of the intervention.

4. Our Experience

The workshop was conducted in the second semester of 2018 and lasted five weeks. After enrolling 20 participants in the course on Google Classroom, we uploaded five video lessons for Week 1, which started on September 24th. In Video 1.1, we introduced ourselves and made a brief preface about the course.

[2]https://scratch.mit.edu

[3]https://classroom.google.com

Table 1. Course Planning

Class	Activity	Goals	Content
Week 1	Introducing the course and promoting a group discussion about CT and Scratch	Have a prior understanding about CT applications in K-12 education	CT principles; CT in K-12 education; Scratch download and installation
Week 2	Designing an interactive story on Scratch with two sprites having a dialogue	Comprehend the essential of Scratch navigation; Use basic Scratch blocks to build a project	Sprite insertion and change; Background insertion and change; Select and repeat blocks; Question and answer blocks
Week 3	Producing animations on Scratch illustrating real-world objects	Implement moderate-level projects using logical operations; Become fully familiar with Scratch	Motion blocks; Logical operation blocks; Advanced select and repeat blocks; Variables
Week 4	Creating a game on Scratch including time and score features	Grasp proper knowledge of advanced Scratch blocks; Combine blocks to design a high-level project	Sensing blocks; Event blocks in parallel; Broadcast communication; Creation of specific blocks
Week 5	Introducing programming basics	Learn programming fundamentals through a comparison between pseudocode and the examples seen on Scratch	Sequences; Conditions; Iterations; Logical and relational operations; Parallelism; Variables; Functions

In Video 1.2, we exposed additional information regarding the workshop, including tools to be used, course planning and activities to be performed by course members. The major activities required were answering a quiz and implementing Scratch projects, as well as watching video lessons and participating in group discussions on Google Classroom.

In Video 1.3, we presented Computational Thinking fundamentals, from the initial to recent definitions reviewed by Wing [Wing 2006] and the academic community. Particularly, we focused on abstraction processes and the elementary aspects of CT according to Wing, such as (i) CT is not only coding, (ii) it is designed for everyone, everywhere and (iii) it is how humans think, not machines. At last, we raised a question regarding how CT can contribute to K-12 education in Brazil. Most participants replied to this topic, resulting in a valuable discussion.

In Videos 1.4 and 1.5, we demonstrated how to download and install Scratch 2.0 desktop version, along with the use of Scratch online community. Participants could realize that it was possible to see, create, merge and share Scratch projects. Once participants were K-12 teachers, we showed three projects developed by Scratch users concerning

traditional K-12 educational topics in Geography (Brazil's States and Capitals), Biology (Human Body) and Physics (Parabolic Motion).

Week 2 started on October 1st, and its lessons were split into six videos. In Video 2.1, we introduced Scratch 2.0, giving a short demonstration about its use and main features, such as sprite, costumes and backdrop selection and change. In Video 2.2, we kicked off the use of Scratch blocks. Through designing a step-by-step cat animation, we used basic Looks blocks, including *say ()*, *switch costume to ()* and *switch backdrop to ()*, along with *play sound () until done*, from Sound tab.

In Video 2.3, we exhibited Event blocks and how to execute basic actions, such as opening and saving projects. Resuming the cat animation, we revealed how to create parallel events in projects by using blocks such as *when this sprite clicked*, *when () key pressed* and *when backdrop switches to ()*.

In Video 2.4, we illustrated the application of basic Control and Operators blocks. An animation involving a dog and a cat was created in a step-by-step mode, using blocks like *forever*, *repeat ()*, *if () then* and *stop ()*. These commands were used to handle when, how often and what the sprites were supposed to do, including sounding and switching costumes.

In Video 2.5, we explained the use of more complex Control and Sensing blocks. For this, we exposed a short Q&A animation – it included only one question and answer. To raise the question, we employed *ask () and wait* block, and worked with *repeat until ()* and *if () then/else* blocks from Control tab to manipulate the animation flow.

The last video (2.6) in Week 2 presented a more advanced Q&A project, in which users were supposed answer the questions with (T)rue or (F)alse, and questions should be repeated until the correct answer was submitted. We aimed to guide participants to their first Scratch assessment: create a Q&A (containing at least 2 questions) project, including topics from disciplines they taught and employing blocks previously demonstrated. Course members had two weeks to complete this task, and explored group conversations on Google Classroom to discuss issues and possible solutions.

The first of three videos uploaded in Week 3 (October 8th) concerned the application of Motion and Sensing blocks. Through implementing an animation of a ball being kicked and colliding with a rock, we exposed blocks such as *turn left/right () degrees*, *point in direction ()* and *change x/y by ()*, which are widely used to produce sprite motion.

In Video 3.2, we introduced Variables, from Data tab. For this, we designed a step-by-step animation emulating an analog clock, which had hands representing hours, minutes and seconds. Each hand of the clock had its value stored in a specific variable, and could be updated by applying *set (var) to ()* and *change (var) by ()* Data blocks.

However, the analog clock project was only completed in Video 3.3. In this video lesson, we showed how to create a communication among sprites, using *broadcast ()* and *when I receive ()* Event blocks. Finally, at the end of Week 3, we uploaded a quiz on Google Forms to measure participants' comprehension about Scratch blocks.

We uploaded four videos in Week 4, starting on October 15th. In video 4.1, we returned to the Q&A project from the second week to illustrate the development of personal blocks, from More Blocks tab. We explained how this can be used to group a set

of blocks in a specific point, and later reuse it by calling the new block as many times as necessary.

In Videos 4.2, 4.3 and 4.4, we presented step-by-step a high-level game designed on Scratch. In this game, the user controlled a fish in the sea and needed to survive during 20 seconds by collecting sea vegetables and dodging human trash. For this, we employed blocks taught in previous lessons, as well as included elements such as random value generation, sprite collision and clone creation. The game served as a tutorial for the final assessment: implement a game, including scoring and timing features, using primarily blocks seen in Weeks 3 and 4, related to content they lectured in their K-12 classes.

Lastly, we introduced programming basics in Week 5 (October 22nd). To perform this, we reviewed the major concepts seen in previous weeks on Scratch, such as conditions, iterations, variables and functions. At the end, we compared Scratch blocks with a standard pseudocode, enabling students to identify and learn how to create their first informal computer programs.

We found encouraging results from the quiz submissions and projects developed by the course members. The outcomes revealed that the average of correct replies was 60% on the quiz in Week 3. Additionally, half the participants was able to complete either Scratch assessment from Week 2 or Week 4, and 75% of them felt comfortable to fill in a post-workshop form, in which they estimated their attitudes towards Computing, as well as their motivation during the course.

5. Lessons Learned

Using Scratch to teach Computing for K-12 educators is suitable. Since Scratch has an appealing and intuitive design, participants could quickly feel comfortable with the tool. Additionally, its block-based programming character means that, unlike various traditional programming languages, learners did not need to spend time typing specific commands, and could focus on Computational Thinking and Computer Science principles.

Analyzing educators context when presenting tutorial Scratch projects is relevant. Course members naturally attempted to associate and apply new content to their background knowledge. For K-12 teachers, this could be reached by demonstrating tutorial projects covering topics related to disciplines they taught, such as Biology and Geography.

Designing step-by-step video lessons is a proper way to introduce Computer Science fundamentals for novices in Computing. Particularly when teaching Programming concepts, step-by-step tutorials can gradually demonstrate how to use these principles as well as how to merge them. Once learners are able to run their first programs after following steps from video lessons, they feel progressively confident to use the language and create new projects.

Providing regular feedback is essential. Primarily at beginning of the learning process, it is essential to support each course member distinctively, finding and contributing to solve issues. Since coding can be not an elementary activity, a proper feedback is able to guide learners about content and strategies to be adopted at the initial stages.

Google Classroom can be combined with other tools provided by Google

(YouTube, Forms, Sheets) and play an important role to support design and application of online courses. Google Classroom is a notable virtual learning environment, mostly due to its usability. Furthermore, once it was developed by Google, it can be efficiently integrated to other demanded tools for an online course.

6. Conclusion

The main goal of this work was to design and implement an online programming course, merged with Computational Thinking aspects, for K-12 educators. Since CT is not properly explored in Brazil, the five-week intervention was an attempt to present and disseminate it to K-12 teachers in a distinct form, including the use of virtual learning environments and block-based programming.

From this work, it can be inferred that using Scratch to teach Computing for K-12 educators is suitable, specially by combining projects with content from traditional K-12 disciplines. It means that analyzing educators context is also relevant when designing Scratch projects. Furthermore, designing step-by-step video lessons is a proper way to introduce Computer Science principles for novices in Computing. At last, Google Classroom can be merged with other tools provided by Google (YouTube, Forms, Sheets) and play an important role to support design and application of online courses.

Hence, this kind of intervention can lead to the progress of Computational Thinking in schools and undergraduate Teacher Education courses. As a consequence, it can eventually contribute to the rise of Computational Thinking in K-12 education. As future work, it is recommended the introduction of Computational Thinking courses in undergraduate degree programs in Education, supported by this work.

References

Barr, V. and Stephenson, C. (2011). Bringing Computational Thinking to K-12: What is Involved and What is the Role of the Computer Science Education Community? *ACM Inroads*, 2(1):48–54.

Gal-Ezer, J. and Stephenson, C. (2014). A Tale of Two Countries: Successes and Challenges in K-12 Computer Science Education in Israel and the United States. *ACM Trans. Comput. Educ.*, 14(2):8:1–8:18.

Kaila, E., Laakso, M., and Kurvinen, E. (2018). Teaching future teachers to code : Programming and computational thinking for teacher students. In *2018 41st International Convention on Information and Communication Technology, Electronics and Microelectronics (MIPRO)*, pages 0677–0682. IEEE.

Santos, W. O., Silva, C., and Hinterholz, L. (2017). Licenciatura em computação: Desafios e oportunidades na perspectiva do estudante. In *Anais do Workshop de Informática na Escola*, page 885.

Silva, V., da Silva, L. L., and França, R. (2017). Pensamento computacional na formação de professores: Experiências e desafios encontrados no ensino da computação em escolas públicas. In *Anais do Workshop de Informática na Escola*.

Wing, J. M. (2006). Computational thinking. *Commun. ACM*, 49(3):33–35.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S., and Korb, J. T. (2014). Computational thinking in elementary and secondary teacher education. *Trans. Comput. Educ.*, 14(1):5:1–5:16.