International Journal of Computer Science Education in Schools, April 2021, Vol. 4, No. 4
ISSN 2513-8359

Developing and Assessing Computational Thinking in Secondary Education using a TPACK Guided Scratch Visual Execution Environment

Raquel Hijón Neira[1]
Miguel García-Iruela[1]
Cornelia Connolly[2]

[1]Universidad Rey Juan Carlos, Madrid, Spain
[2]National University of Ireland Galway, Ireland

DOI: 10.21585/ijcses.v4i4.98

Abstract

Effective and reliable assessment approaches to computational thinking in secondary education are in demand. This paper uses a guided technological pedagogical content knowledge (TPACK) framework, incorporating a visual execution environment (VEE) and Scratch project for secondary school students as a method to teach and assess computational thinking. The objective is to investigate if computational thinking and programming concepts can be improved upon following this method, and if the K-12 children are able to improve their computational thinking skills. The research study was conducted over 2 years in a school setting using the guided VEE and project developed following the dimensions of Computational Thinking process. The project participants came from two cohorts, an after-school programming camp and an in-school environment. Data was collected over two academic years and a quasi-experimental procedure with pre- and post-test was followed. The results demonstrate knowledge gain on computational and programming concepts and encourages us to convey how students translate (as opposed to transfer) their computational thinking experiences into reality. The results indicate the students achieved significant improvement in their computational thinking development.

Keywords: TPACK; Computational Thinking; Assessment; VEE; Scratch.

1. Introduction

Computational Thinking will be a fundamental skill used by everyone in the world (Wing, 2011) and is regarded as the thought processes, involving the formulation of problems and their solutions, characterised as computational steps and algorithms (Aho, 2012). Much research demonstrates how to incorporate computational thinking into classrooms (NRC, 2010; Weintrop et al., 2016; Yadav, Gretter, Good, & McLean, 2017) and many national curriculums are introducing computational thinking through a Computer Science curriculum at upper secondary school (Baron, Drot-Delange, Grandbastien, & Tort, 2014; Bell, Andreae, & Lambert, 2010; Brown, Sentance, Crick, & Humphreys, 2014). Coding is a key way to enable computational thinking (Lye & Koh, 2014) and development of related curriculum key in the enabling CT in secondary education and assessment.

Without attention to assessment, CT enactment in curriculum will have very little success (Grover & Pea, 2013). Assessment is intended to establish whether, and to what extent the curriculum intention has been achieved (Malone, 2011) and efforts to integrate and develop relevant CT-based assessments though developing are lacking (García-Peñalvo & Mendes, 2018; Grover & Pea, 2013). Assessment, and particularly formative assessment (Walsh & Dolan, 2009) help us to identify and bridge the gap between intended and the received curriculum. The two processes are not independent, but rather assessment follows after the curriculum and at times dictates (Hargreaves, Earl, & Ryan, 1996).

This paper evaluates a TPACK Guided Scratch Visual Executing Environment for secondary school students as a method to teach, develop and assess computational thinking. The study contributes to the body of knowledge concerning development and assessment of computational thinking of visual programming (Brennan & Resnick, 2012), having developed a TPACK Guided Scratch VEE and the CT Knowledge Gain Test based on the environment. This work conveys how students translate (as opposed to transfer) their computational thinking experiences into reality. To appreciate the positioning of assessment of computational thinking in secondary education it is valuable to introduce the context and changes in the current literature and we begin by describing the literature and policy pertaining to such developments.

2. Literature Review

In 2006 Jeanette Wing published her article "Computational Thinking" (Wing, 2006) which is understood as fundamentally an analytical skill used to coordinate and interpret knowledge or data in order to accomplish various practical goals or tasks (NRC, 2010). Computational thinking should teach students to apply common CT elements to solve problems and discover new questions to explore within and across all disciplines (Hemmendinger, 2010). It was understood that coding is a key way to enable computational thinking (Lye & Koh, 2014) but CT may be applicable to a variety of unplugged problems that do not directly involve coding tasks (Wing, 2008). In 2011 Wing revisited the topic and provided a new definition "Computational thinking is 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" (Wing, 2011, p. 3). It is important to focus on the importance of learners developing as computational creators (Resnick & Robinson, 2017) and such computational fluency involves not only an understanding of computational concepts and problem-solving strategies, but also the ability to create and express with and through digital technologies.

Computational thinking student learning and assessment is an area of development and studies have been quite varied. Some evaluated students engineering and programming skills as they debugged prebuilt faulty e-textile projects and their deconstruction, reverse engineering, and debugging skills (Fields, Searle, Kafai, & Min, 2012). Other studies presented a more systematic assessment of CT based science learning, using CTSiMe a Computational Thinking based science learning environment (Basu, Kinnebrew, & Biswas, 2014); the identification of CT patterns which young students abstract and develop during the creation of video-games in a controlled environment (Koh, Basawapatna, Bennett, & Repenning, 2010); the development in the student use of CS literacy from engaging in computationally rich activities provides an additional instrument for measuring the growth of CT (Grover, 2011). Moreno-León et al (2015) developed a web application to analyse automatically Scratch projects and provide feedback to improve programming and computational skills. SRI Education published reports providing principled approaches to designing assessment tasks which can generate valid evidence of students' abilities to think computationally exploring CS (Bienkowski, Snow, Rutstein, & Grover, 2015; Snow, Tate, Rutstein, & Bienkowski, 2017) and studies examined students usage of CT concepts and their awareness (Bower et al., 2017).

More recently, Lui et al (2019) demonstrated that CT literacy serves as a formative assessment tool, providing students with feedback benefiting their learning. However, Fields, Lui and Kafai (2019) presented findings revealing that assessment failed to capture the process of CT learning when they were learning with electronic textiles. Nevertheless, many studies highlight the benefits of CT and developing a CT integrated curriculum (Rich et al, 2019, Sung, 2019), CT-inspired teaching and learning tools (Grover 2017) and a CT-embedded learning environment (Muniz-Repiso, Caballero-Gonzálex, 2019).

2.1 Computational Thinking in Secondary Education

Coding is a key way to enable computational thinking (Lye & Koh, 2014) and so developing computer science and programming curriculum is key in the enabling CT integration in secondary education. The introduction of computing in secondary schools has been widely researched (Deek & Kimmel, 1999; Yadav, Gretter, Hambrusch, & Sands, 2016). In many countries the focus of computer science education at post-primary level has shifted from computer and ICT applications towards a more rigorous academic discipline (Bell et al., 2010; Brown et al., 2013; Hubwieser, 2012). The pattern of interest – a basic computing in the 1970s and 1980s, followed by a shift to digital literacies in the 1980s and 1990s, with a resurgence of interest in Computer Science in the past decade seems to match what has happened in the UK for example (Brown et al., 2013; Brown et al., 2014). The English national curriculum was changed in 2014, replacing Information Communication Technology (ICT) to a new subject of Computing which has more emphasis on computer science and programming principles, facilitating computational thinking (Csizmadia et al., 2015). New Zealand, similarly, introduced Computer Science in high schools nationally in 2011 (Bell, Andreae, & Robins, 2012). The revised NZ Computer Science curriculum content focuses on programming and gives students the chance to explore a range of computer science topics beyond programming, including algorithms and complexity, human-computer interaction, encryption, artificial intelligence, formal languages, computer graphics (Bell, Andreae, & Robins, 2014). In Ireland Computer Science was introduced as part of the curriculum in 2017 (NCCA, 2017) and a major component of their upper secondary specification is computational thinking. In Spain there is a computing curriculum in secondary education (BOE, 2015) and the subject "Technology, Programming and Robotics" has been taught from the 2014/15 academic year (INTEF, 2019). In reviewing the situation in Spain regarding Computing Education in pre-university stages made by the Spanish Computing Scientific Society (SCIE), with the support of the Spanish Board of Deans of Computing Schools (CODDI), it was recommended to establish a subject titled "Informatics", which was implemented as a mandatory course offered in both primary and secondary education (Velázquez-Iturbide, 2018). In many countries the focus of visual programming is primarily at primary level (Bell, Duncan, & Atlas, 2016; Duncan, Bell, & Atlas, 2017; Sáez-López, Román-González, & Vázquez-Cano, 2016).

Understanding the impact of a block-based programming environment in high school classrooms has been researched (Weintrop & Wilensky, 2017) and the work by Armoni et al (2015) focused on the transition from learning CS in middle school with Scratch, to learning CS in secondary school using a "real" programming language and a professional software development environment. Results demonstrated evidence to justify learning CS in middle schools, although there were no significant differences in achievements compared to students who had not studied Scratch (Armoni, Meerbaum-Salant, & Ben-Ari, 2015). This is consistent with the results of Levy et al. (2003), who showed that the use of the Jeliot program animation system primarily benefited the students who are capable of learning but not outstanding.

2.2 Serious Games and MaKey MaKey as a Teaching Resources

Game is understood as a playful action without a concrete, free and voluntary purpose. The win-lose dynamic is intrinsic. In this study, we use educational games (which have a specific purpose) where losing is a new opportunity to learn. Through the game, skills are developed to study the environment or specific problems and be creative looking for solutions (Granic, Lobel, & Engels, 2014), in this case computational thinking. Structuralist theory considers that the game establishes the way of seeing the world and thinking of the child as CT will be a new concept for many to discover. The absence of this "learn to think" prevents further learning from having depth (they are not reflexive) and therefore do not activate the emotional part that enables long-term learning (Piaget & Inhelder, 1999). On the other-hand the Fogg model (2009) designed to change human behaviour, establishes that three elements are necessary to modify the behaviour – motivation, skills and a trigger. An educational game facilitates dynamics in which these three components converge simultaneously, being an optimal method for teaching-learning dynamics of new concepts, in our case concepts related to programming and the development of computational thinking.

MaKey MaKey was developed at MIT and is a simple hardware platform for improvising tangible user interfaces (Collective & Shaw, 2012). The Guided Scratch Visual Executing Environment (VEE) in this study can be used with a MaKey-MaKey device. The use of Makey-Makey is closely related to the constructivist conception of education, since it corresponds to the user, the design, construction and, where appropriate, modification of the controls to be used. Interactive controls have proven to be a good tool for the promotion of class interactivity (Álvarez Martínez & Llosa Espuny, 2010). In addition, the inclusion of the design of tangible game controls which develop inventiveness in Makey-Makey broadens learning opportunities (Lee et al., 2014). Martinez and Stager (2013) detail some of the main ideas that underlie the didactic use of this device:

- Learn by doing: You learn more when learning is part of something that is interesting. You learn best when we use what we learn to do something we really want.
- Technology as a building material: If you can use technology to do things, you can do more interesting things. And you can learn much more by doing them.
- Fun is not easy: We learn and work better if we enjoy what we are doing. But enjoying is not synonymous with easy. The greatest enjoyment occurs when the challenge is difficult.
- Learn to learn: The belief that you can only learn when someone teaches you is widespread. You don't always have someone who can teach you what you want to learn.
- Take the time to do the job: It is important to learn how to manage time for yourself to achieve the desired objectives.
- You cannot do well if you have not made a mistake: Complex things do not work at first.
- The only way to get it right is to analyze the problems that produced the previous failures.

Another important aspect of the MaKey-MaKey device is that it allows for easy adaptation to any need, and this was particularly important for our study in interfacing with the VEE and assessing computational thinking.

3. Research Design

This paper evaluates a TPACK Guided Scratch Visual Executing Environment for secondary school students as a method to teach, develop and assess computational thinking. The two research questions are as follows: Can computational thinking and programming concepts be improved with a TPACK Visual Execution Environment and Scratch on K-12 students? And secondly, by using this TPACK Guided Scratch VEE and Scratch are students able to improve their computational thinking skills?

3.1 Theoretical Foundation

The design of this study drew on the TPACK framework (Koehler & Mishra, 2009) in the integration of the necessary knowledge and the development of a useful tool to transmit the concepts of computational thinking which Scratch supports (Brennan & Resnick, 2012). The TPACK model defines the area in which technology is consistently integrated in teaching and the transfer of knowledge to the student is enhanced. The intersection of three fields of knowledge is that of Content Knowledge (concepts of computational thinking); Pedagogical Knowledge (exhibition and serious games) and thirdly Technological Knowledge (programming with scratch and development of web pages.) At the intersections of the fields of knowledge, less significant areas of partial knowledge are generated, since they lack one of the areas.

According to Grover and Pea (2013), there is a consensus on the elements that should be included in a computational thinking curriculum, such as abstractions and generalizations of patterns, including models and simulations. The work carried out by the creators of Scratch was considered in the Creative Computing document (Brennan, Balch, & Chung, 2014) whose objective is to explore computational thinking by the Scratch programming language. This project is based on Brennan & Resnick (2012), which classifies computational thinking into three dimensions: Concepts, Practices and Computational Perspectives depicted in Table 1.

Table 1. Dimensions of Computational Thinking (Brennan & Resnick, 2012)

Computational Concepts (7): • Sequences • Loops • Parallelism • Events • Conditional • Operators • Data	Computational Practices (4): • Incremental and iterative development • Test and debug • Reuse and mix • Abstract and modularize Computational perspectives (3): • Express yourself • Connect • Question

The use of metaphors for educational purposes consists of transferring a known concept from one object to another to which it provides a new notion or intuition. In this study attempts have been made to make metaphors evident and, where possible, graphic representations have been used to facilitate assimilation. Below is a list of the main metaphors used (see Table 2). In the case of the "Operator" concept, it has not been considered necessary to evoke a metaphor, since the notion of mathematical operator is widely extended.

Table 2. Metaphors of Computational Concepts

Concept	Metaphor
Sequence	Cooking recipe
Variable	Container with label
Conditional	Detour on the road
Loop	How a clock works
Event	Traffic light operation
Synchronization	Set the same time on two watches
Computational thinking	NIM game

3.2 TPACK Scratch Visual Execution Environment

The TPACK Scratch Visual Execution Environment has pre-established programs, which include the theory and practice corresponding to each of the proposed lessons. This separation allows the teacher different sequencing from the one proposed, if necessary. Since some concepts are supported by prior learning it is necessary (e.g., to explain the operation of conditionals, it is necessary to previously understand logical operators). The order of topics proposed are presented in Table 3.

Table 3. Proposal for Sequencing Topics

Lesson 1. Sequences	Lesson 5. Loops
Lesson 2. Variables	Lesson 6. Events
Lesson 3. Operators	Lesson 7. Parallelism
Lesson 4. Conditionals	Lesson 8. Computational Thinking

The topics developed are closely related to the computational concepts implicit in Scratch, as a programming initiation language (Brennan & Resnick, 2012) and the first seven themes are valid tools for learning programming. The last concept includes the notion of computational thinking as an exercise of recapitulation and reinforcement of the previous points. Computational thinking is a complex competence that is related to the mental schemes of human beings, which allows to develop ideas and link abstraction (ideas-concepts) with pragmatism (action). It is not synonymous with programming, since it requires different degrees of abstraction and does not depend on computer equipment (unplugged). However, the use of computer equipment allows us to undertake tasks that without them would be unapproachable (Urbina Ramírez, 1999).

The TPACK Guided Scratch VEE has pre-established programs (Hernández Tijera & Perianes Rodriguez, 2018). It is a web application that allows accessing from any device apart from a PC (Smartphones, tablet, etc). It can be used with a MaKey-MaKey device or with the mouse and enables interaction at the students own pace. In order to access the Scratch applications embedded in the web pages, it is necessary to enable Flash Player and click on the green flag to start the different Scratch programs (http://scratch-tfm.000webhostapp.com/index.html).

In the development of each topic, the best way for the assimilation of the CT concept to be treated was considered and decided to first develop an exhibition and then carry out several practices. To highlight this separation, a visual key was used as a resource. The exhibition section has a classic slate background, and the practical part has a grid notebook background (see Figure 1). In the theoretical section, thanks to the interaction with the treated concept, students can interactively go at their own pace, allowing personalization to understand the concepts being exposed, developed around metaphors. In the practical section, thanks to the interaction with the treated concept, an instant feedback is showed, which allows both the assimilation and the accommodation of the new concepts (Figure 2).

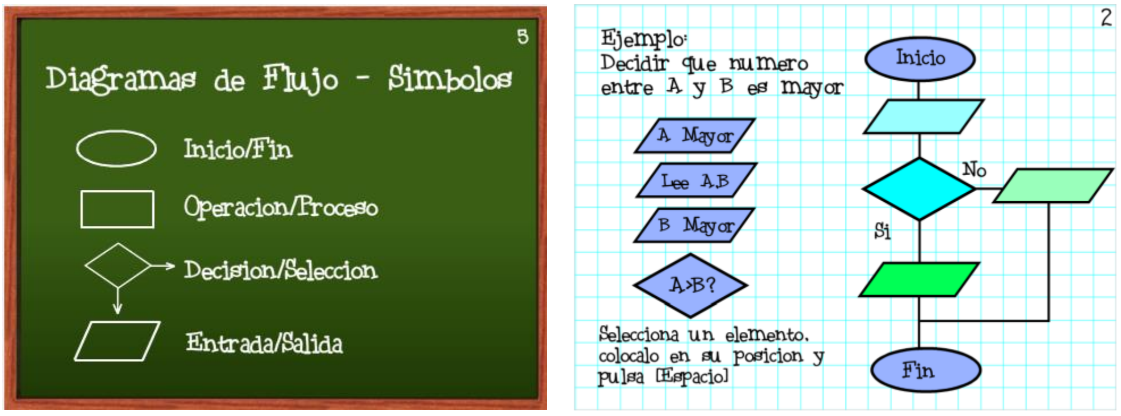
5
Diagramas de Flujo - Simbolos
Inicio/Fin
Operacion/Proceso
Decision/Seleccion
Entrada/Salida
2
Ejemplo:
Decidir que numero entre A y B es mayor
A Mayor
Lee A,B
B Mayor
A>B?
Selecciona un elemento, colocalo en su posicion y pulsa [Espacio]
Inicio
No
Si
Fin

Figure 1. Examples of exposure (left) and practice (right) in the TPACK Guided Scratch VEE

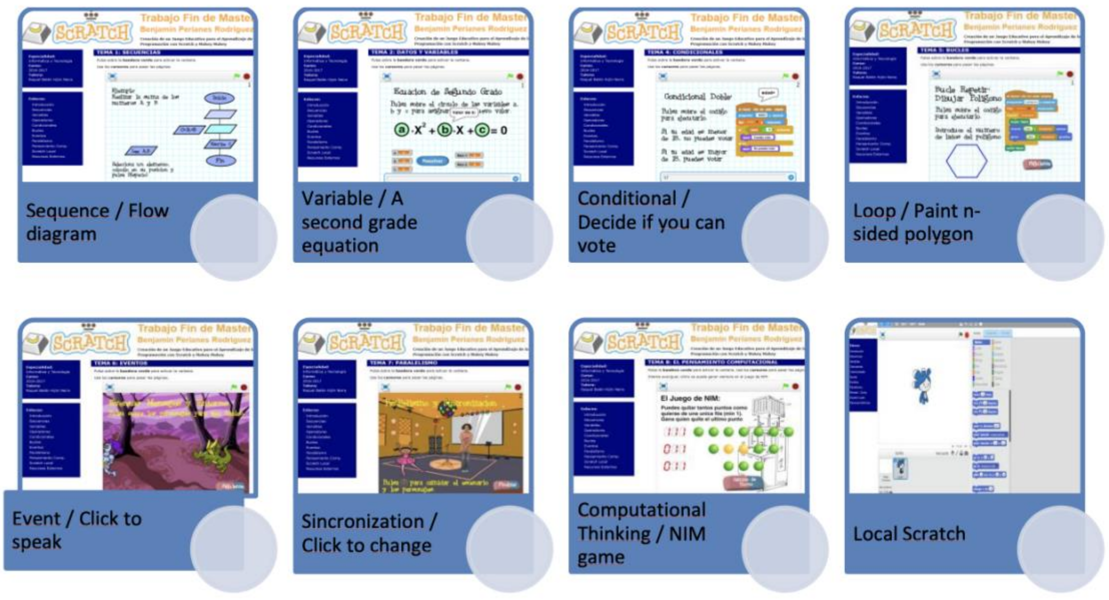
Sequence / Flow diagram
Variable / A second grade equation
Conditional / Decide if you can vote
Loop / Paint n-sided polygon
Event / Click to speak
Sincronization / Click to change
Computational Thinking / NIM game
Local Scratch

Figure 2. The TPACK Guided Scratch VEE with 7 computational Concepts and their serious games, last option is Local Scratch

3.3 Research Participants

The participants (N = 32) were K-12 students from two sites, some from an after-school programming camp (N = 6) and the others from a Madrid school (N = 26). The experience in both cases took place for 2 weeks and 6 hours per week, in total 12 hours. The distribution by gender is as follows: 54.8% girls and 45.2% boys. The experience took place in a computer room for 6 hours a week (2 hours a day) and no reward was given in grades or otherwise, the only reward was the students' own increasing motivation.

3.4 Data Collection

The experiment took place during the 2nd semester of two academic years, 2017-2018 and 2018-2019. A quasi-experimental procedure with pre- and post-test was followed. For the pre and post-tests, the same evaluation tests were used for concepts of computational thinking and programming on the one hand, and gains in computational thinking, on the other. Each test was completed individually by each student on their computer in class.

In each class, the students completed two pre-tests. The first was to verify what they knew about Brenan Resnick's 7 concepts of computational thinking (2012). The assessment consists of 12 free-text questions. These questions refer to the 7 computational Concepts of the first dimension of Computational Thinking (Sequence, variables, operators, conditionals, loops, events, parallelism, and the concept of computational thinking). The second test was to measure their computational thinking with a validated test (Román-González, Pérez-González, & Jiménez-Fernández, 2017). This test consisted of 28 multiple choice questions, with 4 possible options in each. Questions which cover the CT concepts of Basic directions & sequences, repeat times, repeat until, Simple conditional, complex conditional, while conditional and simple functions. The students then received 12 hours of class for 2 weeks, and they took the two post-tests to verify the improvements.

The tasks carried out by the students consisted of first an introduction to the theory and practice of each concept in the TPACK Guided Scratch VEE and then continuing working on it in Scratch. The tasks carried out were based on: algorithms, flow diagrams, operators, variables (Scratch's "ask" and "answer"), conditionals, loops, and then they worked with everything learned with a project we called "*Working Geometry with Scratch* ", students work on this project allowed them to incorporate the 2 dimensions of Brennan & Resnick (2012) left: Computational Practices (Incremental and iterative development, test and debug, reuse and mix, and abstract and modularize) and Computational perspectives (express yourself, connect and question). In this project the students had to make little programs that painted polygons, for example:

1. Draw an equilateral triangle of side 100
2. Draw a square of side 80
3. Draw a pentagon from side 50
4. Create a program that does the following:
    - Set a suitable scenario and character (as if he were a school teacher)
    - The character should say, “Hello, we are working with equilateral triangles. Equilateral triangles are those that have their three equal sides and their three angles measure 60º. I am going to draw the triangle that you want. How much do you want me to measure on your side?
    - Then the program must draw a triangle aside the "answer" that the user enters on the keyboard.
5. We are going to do the same for a square.
    - Modify the triangle program, it is very simple. Just change the number of laps and degrees.
6. The same for a pentagon
7. Same for n hexagon
8. Could you do it for a polygon with n sides?

4. Results and Discussion

4.1 Computational Thinking Concepts and Programming

A descriptive analysis of the results obtained is shown in Table 4, showing the minimum, maximum, mean and standard deviation of each test (pre and post). The results show that there are significant results in the results in the post-test. The minimum, maximum and mean values increase remarkably in the post-test results, although the dispersion increases minimally.

Table 4. Mean and Typical Deviation in The Test of Computational Thinking Concepts

	(n=32)			
	Min	**Max**	**Media**	**SD**
Pre	0,688	4,063	2,160	0,660
Post	6,125	9,750	8,867	0,931

The box-plots of the results in the evaluation of the Basic Computational Thinking Concepts, demonstrate the pre- and post-test, where each is delimited by the values Q1 (first quartile) and Q3 (third quartile). Each box groups 50% of the cases, highlighting the median. The lowest and highest value at the end of each diagram correspond to the values that are not less than Q1-1.5 · (Q3-Q1) and are not greater than Q3 + 1.5 · (Q3-Q1). The Analysis of variance of a factor (Anova) has been carried out to study the pre-test and the post-test and a value for F = 1105,31 and a p-value << 0,005 have been obtained, therefore it can be concluded that the data are significantly different. Comparing the pre-test with the post-test, after analysing the data, normality can be concluded for the study group (obtaining p> 0.05 significance using the Shapiro-Wilk test), allowing us to use the t-Student test for paired samples (p> 0.05 using bivariate correlation tests). In this test, it has been assumed that the null hypothesis can be established, since there are no differences between the means. Therefore, a p-value greater than 0.05 will reveal homogeneity in the samples. As a result, the difference between the pre-test and the post-test in the study of improvement in basic programming knowledge (t test analysis -6.707 and p-value 0.0001) and therefore, it is deduced that the students had a significant improvement in the test scores when following the course planning (p <0.0001). In order to collect additional information on the magnitude of the change produced in the students following the methodology explained in previous subsection, the size of the effect in the study group was calculated by means of the variation (Cohen, 1988), obtaining a g value = 8.4, corresponding to a very large effect (since it is> 0.5). According to these results, the students achieved a significant improvement in their global learning in the 7 concepts of Computational Thinking, the size of the effect is very large.

4.2 Computational Thinking Results for Each Concept

To verify which CT concepts, demonstrate the greater or less improvement achieved, we proceed to analyse which were more complicated or more affordable. The results show that there are significant results in the results in the post-test. The minimum, maximum and mean values increase remarkably in the post-test results, although the dispersion increases slightly in all concepts except for memory and sequence. Figure 3 shows the box-plots of the detailed results of each of the Computational Thinking Dimensions / concepts worked with Guided Scratch VEE in the pre- and post-test broken down by the 8 Dimensions studied (sequence, variables, operators, conditionals, loops, events, parallelism and the Computational Thinking Nim Game). Each box is delimited by the values Q1 (first quartile) and Q3 (third quartile). Each box groups 50% of the cases, highlighting the median, lowest and highest value at the end of each diagram corresponds to the values that are not less than Q1-1.5 · (Q3-Q1) and are not greater than Q3 + 1.5·(Q3-Q1). The Analysis of variance of a factor (Anova) has been carried out to study the pre-test and the post-test and a value for F = 1105,31 and a p-value of << 0,005 have been obtained, therefore it can be concluded that the data are significantly different. Comparing the pre-test with the post-test, after analysing the data, normality can be concluded for the study group (obtaining p> 0.05 significance using the Shapiro-Wilk test), allowing us to use the t-Student test for paired samples (p> 0.05 using bivariate correlation tests). In this test, it has been assumed that the null hypothesis can be established, since there are no differences between the means. Therefore, a p-value greater than 0.05 will reveal homogeneity in the samples.

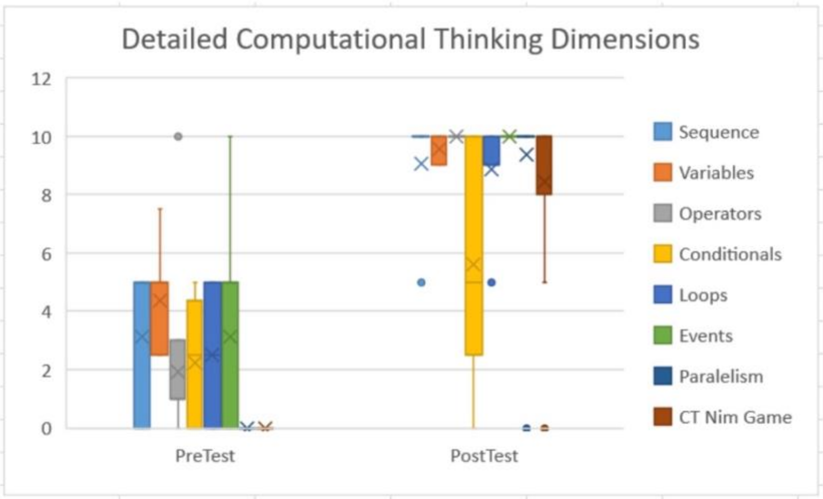

Figure 3. Box-plots for the group of students in pre- and post-tests in each of the CT dimensions worked with the TPACK Guided Scratch VEE

Table 5 shows the difference between the pre-test and the post-test in the study of improvement in each concept worked on following the procedure studied. Therefore, it is deduced that the students had a significant improvement in the knowledge of these dimensions of Computational Thinking at the end of the interaction (p <0.0001). In order to collect additional information on the magnitude of the change produced in the students following the TPACK Guided Scratch VEE methodology explained in previous subsection; the effect size in the study group was calculated by variation (Cohen, 1988), obtaining a value *variables* of g = 5,4 corresponding to a very large effect (since it is> 0.5), for *operators* of g = 6,9, corresponding to a very large effect (since it is> 0.5), for *conditionals* of g = 1,2, corresponding to a large effect (since it is> 0.5), for *loops* of g = 3,6, corresponding to a very large effect (since it is> 0.5), for *events* of g = 4 corresponding to a very large effect (since it is> 0.5), for *parallelism* of g = 7,6 corresponding to a very large effect (since it is> 0.5), for CT of g = 6,5 corresponding to a very large effect (since it is> 0.5).

Table 5. Study using t-Student and P-Value Analysis

	t test analysis	**p-value**
Sequence	-5,938	0,0001
Variables	-5,188	0,0001
Operators	-8,063	0,0001
Conditionals	-3,406	0,0001
Loops	-6,375	0,0001
Events	-6,875	0,0001
Paralelism	-9,375	0,0001
CT Nim Game	-8,438	0,0001

According to these results, learning is significant for all CT dimensions worked with the TPACK Guided Scratch VEE. At the beginning (pre-test) the newest or most unknown concepts for students are: *parallelism* and the concept of *computational thinking*. On the other hand, the most familiar concepts are that of *sequence, loops* and *events*, although they do not yet dominate. At the end of the intervention, all the dimensions have achieved a significant improvement, we can say that the concepts of *sequence, operators, events* and *parallelism* dominate; The rest of the concepts (*variables, loops, computational thinking*) are dominated by more than 80% of the group of participating students, and in *conditionals* it is where there is more dispersion, achieving more than 50% of the class to overcome it as well. On the other hand, the concepts with the greatest effect on learning are (in that order): *parallelism, operators, computational thinking, variables, loops, events, sequence* and *conditionals*.

4.3 Assessment of Computational Thinking

In regard to assessment of computational thinking we firstly provide a descriptive analysis of the results obtained. Table 6 shows the minimum, maximum value, mean and standard deviation of each test (pre and post). The results show that there are significant results in the results in the post-test. The minimum, maximum and mean values increase remarkably in the post-test results, although the dispersion increases minimally.

Table 6. Mean and Typical deviation in the assessment of Computational Thinking

		(n=32)		
	Min	**Max**	**Media**	**SD**
Pre	3,929	7,500	4,576	0,987
Post	4,643	7,500	6,295	0,653

The box-plots of the results in the evaluation of the Computational Thinking Test in the pre- and post-test and the analysis of variance of a factor (Anova) has been carried out to study the preTest and the postTest and a value for F = 67,49 and a p-value < 0,005 have been obtained, therefore it can be concluded that the data are significantly different. Comparing the pre-test with the post-test, after analyzing the data, normality can be concluded for the study group (obtaining p> 0.05 significance using the Shapiro-Wilk test), allowing us to use the t-Student test for paired samples (p> 0.05 using bivariate correlation tests). In this test, it has been assumed that the null hypothesis can be established, since there are no differences between the means. Therefore, a p-value greater than 0.05 will reveal homogeneity in the samples. The t-test analysis is -1.719 and therefore the difference between the pre-test and the post-test in the study of improvement in basic programming knowledge. Therefore, it is deduced that the students had a significant improvement in the test scores when following the course planning (p <0.0001). In order to collect additional information on the magnitude of the change produced in the students following the methodology explained in previous subsection, the size of the effect in the study group was calculated by means of the variation in Cohen's D (Cohen, 1988), obtaining a g value = 2,1 corresponding to a very large effect (since it is> 0.5). According to these results, the students achieved a significant improvement in their Computational Thinking, the size of the effect is very large.

4.4 Assessment Computational Thinking Test by Concepts

In order to determine if by the intervention students improved differently and how on their Computational Thinking, we proceed to study the concepts assessed to determine which had the most improvement and which ones had the least. The results show that there are significant results in the results in the post-test. The minimum, maximum and mean values increase remarkably in the post-test results, although the dispersion increases slightly in all concepts except memory and sequence. Figure 4 shows the box-plots of the detailed results on the Computational Thinking Test exploited by concepts (*basic directions & sequences, repeat times, repeat until, simple conditional, complex conditional, while conditional, simple functions*). The Analysis of variance of a factor (Anova) has been carried out to study the preTest and the postTest and a value for F = 67,49 and a p-value of << 0,005 have been obtained, therefore it can be concluded that the data are significantly different.

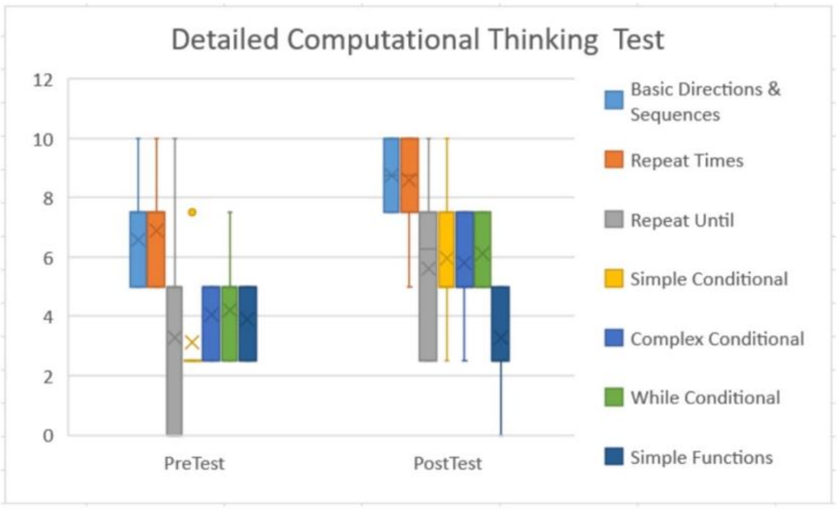

Figure 4. Box-plots for the group of students in pre- and post-tests in each CT concept assessed

Comparing the pre-test with the post-test, after analyzing the data, normality can be concluded for the study group (obtaining p> 0.05 significance using the Shapiro-Wilk test), allowing us to use the t-Student test for paired samples (p> 0.05 using bivariate correlation tests). In this test, it has been assumed that the null hypothesis can be established, since there are no differences between the means. Therefore, a p-value greater than 0.05 will reveal homogeneity in the samples. It is deduced that the students had a significant improvement on these concepts of Computational Thinking at the end of the interaction (p <0.0001) but for Simple Functions (p<0,018).

In order to collect additional information on the magnitude of the change produced in the students on each concept, the effect size in the study group was calculated by variation in Cohen's D (Cohen, 1988), obtaining a value for *basic directions & sequence* of g = 1,5, corresponding to a very large effect (since it is> 0.5), for *repeat times* of g = 1.06, corresponding to a very large effect (since it is> 0.5), for *repeat until* of g = 0.79, corresponding to a large effect (since it is> 0.5), for *simple conditional* of g = 1.54, corresponding to a very large effect (since it is> 0.5), for *complex conditional* of g = 1.16 corresponding to a very large effect (since it is> 0.5), for *while conditional* of g = 1.37 corresponding to a very large effect (since it is> 0.5).

5. Conclusion

There is worldwide interest on finding ways to improve students Computational Thinking skills and ways to assess it. This project uses a guided TPACK framework, incorporating a Scratch VEE for secondary school students as a method to teach and assess computational thinking. The objective was to investigate if computational thinking and programming concepts could be improved upon in applying this approach and if the K-12 children are able to improve their computational thinking skills.

The use of the serious games in the Guided Scratch VEE enables creativity in looking for solutions, similar to previous studies (Granic, Lobel, & Engels, 2014). The possibility of using a Makey-Makey can promote class interactivity (Álvarez Martínez & Llosa Espuny, 2010) and learning opportunities (Lee et al., 2014). Moreover, metaphors have been used to explain concepts (*Pérez-Marín, et al., 2020*; Pérez-Marín, Hijón-Neira, Martín-Lope, 2018), as well as abstractions and generalization of patterns (*Grover & Pea 2013)* presenting them as pre-established programs to guide the student in their learning. A balance of theory and practice, combined with the ordering of topics, which follows the sequence proposed in studies to teach CT (Brennan & Resnick, 2012). Furthermore, adopting user centred design (UX) the exhibition section and the practical section have different backgrounds, offering instant feedback. The experiment also took part over two academic years aligning to the validated test (Román-González, Pérez-González & Jiménez-Fernández, 2017). As mentioned previously, prior to the tasks being carried out by students, they first developed understanding of each concept in the TPACK Guided Scratch VEE and the continued their learning with Scratch, enabling them incorporate the 2 dimensions of computational practices and computational perspectives (Brennan & Resnick, 2012)

The first Research Question, RQ1, asked: Can Computational Thinking and programming concepts be improved with a Visual Execution Environment and Scratch on K-12 students? In concluding the study, it has been observed that a TPACK Guided Scratch VEE, and an iterative incremental project based on polygons, that the students created in Scratch at their own pace, incorporating computational practices (*incremental and iterative*, *test and debug*, *reuse and mix*, and *abstract and modularize*) and computational perspectives (*express yourself, connect and question*). The students achieved significant improvement in their learning of the concepts of Computational Thinking. The students achieved a significant improvement in their global learning in the seven concepts of Computational Thinking, the size of the effect is very large. According to these results, learning is significant for all CT dimensions worked with the TPACK Guided Scratch VEE. At the beginning (pre-test) the newest or most unknown concepts for students are: *parallelism* and the concept of *computational thinking*. On the other hand, the most familiar concepts are that of *sequence, loops,* and *events,* although they do not yet dominate. At the end of the intervention, all the dimensions have achieved a significant improvement, we can say that the concepts of *sequence, operators, events,* and *parallelism* dominate; The rest of the concepts (*variables, loops, computational thinking*) are dominated by more than 80% of the group of participating students, and in *conditionals* it is where there is more dispersion, achieving more than 50% of the class to overcome it as well. On the other hand, the concepts with the greatest effect on learning are (in that order): *parallelism, operators, computational thinking, variables, loops, events, sequence,* and *conditionals*.

The second Research Question, RQ2 investigated if by using this TPACK Guided Scratch VEE are students able to improve their computational thinking skills? The students achieved a significant improvement in their Computational Thinking and the size of the effect is very large. According to these results, learning is significant for all CT concepts the test measured but for *simple functions*. At the beginning (pre-test) the newest or most unknown concepts for students are: *repeat until, simple* and *complex conditionals, while conditional* and *simple functions*. As in previous analysis, the most familiar concepts are that of *basic directions & sequence* in the first place, followed by *repeat times,* they are all about to pass. At the end of the intervention, all the concepts but *functions* have achieved a significant improvement, we can say that the concepts of *basic directions & sequence* and *repeat times* students got it excellent; the rest of the concepts (*repeat until, simple* and *complex conditional, while conditional* are also well understood. On the other hand, the concepts with the greatest effect on learning are (in that order): *simple conditions, basic directions, while conditional, complex conditionals, complex conditional, repeat times, repeat until* and *simple functions*.

The results obtained on K-12 students for a total of 12 hours of classes using the TPACK Guided Scratch VEE and developing guided projects with Scratch, the results of this experiment demonstrate that students gained a significant improvement in the test scores when following the course planning. According to these results, the students achieved a significant improvement in their global learning in the seven concepts of computational thinking, the size of the effect is very large. Therefore, the use of such VEE provides an aid to the teacher in introducing the CT concepts, and provides guidance specific to metaphors and serious games. It provides a better introduction to just starting with Scratch from Scratch, offering well established steps for explaining abstract concepts to young students.

When studying the results for each specific dimension, it is deduced that students had a significant improvement in the knowledge of all dimensions of computational thinking at the end of the interaction, and that is a good way for explaining such new concepts to students as parallelism or computational thinking. Since at the end of the intervention, all dimensions have achieved a significant improvement, we can say that using the Guided TPACK VEE and Scratch we can help teachers teach such complex and novel concepts such as: Sequence, Variables, operators, conditionals, loops, events, parallelism and the CT concept itself.

It has been proven that there is a significant improvement in students Computational Thinking. The magnitude of change produced by students on each concept is a very large for all and ordered form more to less are: basic directions & sequence, repeat times, repeat until, simple conditional, complex conditional, while conditional. Therefore, by using the Guided TPACK VEE and Scratch teachers can teach CT concepts in the classroom more smoothly and it will reduce preparation time providing very significantly outcomes for their students.

While the results demonstrate a definite statement on improvement of CT skills when a TPACK Guided Scratch VEE was used, the work is specific to one geographic location and pedagogic approach adopted. The work, being a quasi-experimental research case study with one school and after school context, has the primary limitation of a narrow focus. While such an approach does not facilitate the development of generalisations, it can effectively point out possible results, which require further investigation and validation.

This paper presented a study carried out with K-12 secondary students. A procedure was followed where the computational concepts, practices, and perspectives, according to a TPACK Guided Scratch VEE were adopted. The class built a project on Scratch where the concepts only worked combining computational thinking perspectives and practices. To assess the computational thinking two tests were carried out to answer to our research questions and the results demonstrate knowledge gained on computational and programming concepts. The findings demonstrate that students managed to make complex programs combining what they learned in an incremental way, which provides insight to CS educators in pedagogical approaches.

References

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832-835. https://doi.org/10.1093/comjnl/bxs074

Álvarez Martínez, C., & Llosa Espuny, J. (2010). *Formative evaluation with quick feedback using interactive controls*. Paper presented at the XVI Conference on University Teaching of Computing, University of Santiago de Compostela. Escola Técnica Superior d'Enxeñaría.

Armoni, M., Meerbaum-Salant, O., & Ben-Ari, M. (2015). From scratch to “real” programming. *ACM Transactions on Computing Education (TOCE), 14*(4), 1-15. https://doi.org/10.1145/2677087

Baron, G. L., Drot-Delange, B., Grandbastien, M., & Tort, F. (2014). Computer science education in French secondary schools: Historical and didactical perspectives. *ACM Transactions on Computing Education (TOCE), 14*(2), 1-27. https://doi.org/10.1145/2602486

Basu, S., Kinnebrew, J. S., & Biswas, G. (2014). Assessing student performance in a computational-thinking based science learning environment. In *International conference on intelligent tutoring systems* (pp. 476-481): Springer.

Bell, T., Andreae, P., & Lambert, L. (2010). *Computer science in New Zealand high schools.* Paper presented at the 12th Australasian Conference on Computing Education, Brisbane, Australia.

Bell, T., Andreae, P., & Robins, A. (2012). *Computer science in NZ high schools: the first year of the new standards.* . Paper presented at the 43rd ACM technical symposium on Computer Science Education.

Bell, T., Andreae, P., & Robins, A. (2014). A case study of the introduction of computer science in NZ schools. *ACM Transactions on Computing Education (TOCE), 14*(2), 1-31. https://doi.org/10.1145/2602485

Bell, T., Duncan, C., & Atlas, J. (2016). *Teacher feedback on delivering computational thinking in primary school.* Paper presented at the 11th Workshop in Primary and Secondary Computing Education. https://doi.org/10.1145/2978249.2978266

Bienkowski, M., Snow, E., Rutstein, D., & Grover, S. (2015). *Assessment design patterns for computational thinking practices in secondary computer science: A first look.* Retrieved from https://pact.sri.com/downloads/Assessment-Design-Patterns-for-Computational%20Thinking-Practices-Secondary-Computer-Science.pdf

BOE. (2015). *Orden ECD/65/2015 por la que se describen las relaciones entre las competencias, los contenidos y los criterios de evaluación de la Educación Primaria, la Educación Secundaria Obligatoria y el Bachillerat*.

Bower, M., Wood, L. N., Lai, J. W., Highfield, K., Veal, J., Howe, C., ... & Mason, R. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education*, *42*(3), 53-72.

Brennan, K., Balch, C., & Chung, M. (2014). *Creative computing: Scratch curriculum guide*. Retrieved from https://creativecomputing.gse.harvard.edu/guide/

Brennan, K., & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking.* Paper presented at the American educational research association, Vancouver, Canada.

Brown, N. C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., & Sentance, S. (2013). *Bringing computer science back into schools: Lessons from the UK.* Paper presented at the 44th ACM Technical Symposium on Computer Science Education (SIGCSE'13). New York. https://doi.org/10.1145/2445196.2445277

Brown, N. C., Sentance, S., Crick, T., & Humphreys, S. (2014). Restart: The resurgence of computer science in UK schools. *ACM Transactions on Computing Education (TOCE), 14*(2), 9. https://doi.org/10.1145/2602484

Cohen, J. (1988). *Statistical Power Analysis for the Behavioural Sciences*. Hillsdale, NJ, USA: Erlbaum.

Collective, B. M., & Shaw, D. (2012). *Makey Makey: improvising tangible and nature-based user interfaces.* Paper presented at the Sixth International Conference on Tangible, Embedded and Embodied Iinteraction.

Deek, F. P., & Kimmel, H. (1999). Status of computer science education in secondary schools: One state's perspective. *Computer Science Education, 9*(2), 89-113. https://doi.org/10.1076/csed.9.2.89.3808

Duncan, C., Bell, T., & Atlas, J. (2017). *What do the teachers think? Introducing computational thinking in the primary school curriculum.* Paper presented at the Nineteenth Australasian Computing Education Conference.

Fields, D. A., Searle, K. A., Kafai, Y. B., & Min, H. S. (2012). *Debuggems to assess student learning in e-textiles.* Paper presented at the 43rd ACM technical symposium on Computer Science Education. https://doi.org/10.1145/2157136.2157367

Fields, D. A., Lui, D., & Kafai, Y. B. (2019). Teaching computational thinking with electronic textiles: Modeling iterative practices and supporting personal projects in exploring computer science. In *Computational thinking education* (pp. 279-294). Springer, Singapore.

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

García-Valcárcel-Muñoz-Repiso, A., & Caballero-González, Y. A. (2019). Robotics to develop computational thinking in early Childhood Education. *Comunicar. Media Education Research Journal*, *27*(1).

Granic, I., Lobel, A., & Engels, R. C. (2014). The benefits of playing video games. *American psychologist, 69*(1), 66. https://doi.org/10.1037/a0034857

Grover, S. (2011). *Robotics and engineering for middle and high school students to develop computational thinking.* Paper presented at the American Educational Research Association, New Orleans, LA.

Grover, S., & Pea, R. (2013). Computational Thinking in K–12: A Review of the State of the Field. *Educational Researcher, 42*(2), 38-43. https://doi.org/10.3102/0013189X12463051

Grover, S. (2017). Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom. In *Emerging research, practice, and policy on computational thinking* (pp. 269-288). Springer, Cham.

Hargreaves, A., Earl, L. M., & Ryan, J. (1996). *Schooling for change: Reinventing education for early adolescents*: Routledge.

Hemmendinger, D. (2010). A Plea for Modesty. *ACM Inroads, 1*(2), 4-7. https://doi.org/10.1145/1805724.1805725

Hernández Tijera, I., & Perianes Rodriguez, B. (2018). *Camino hacia la excelencia. Premios Trabajo Fin de Máster en Formación del profesorado de Educación Secundaria.* Retrieved from Colegio Oficial de Docentes. Sial Pigmalión.

Hubwieser, P. (2012). Computer science education in secondary schools: The introduction of a new compulsory subject. *Transactions in Computing Education, 12*(4), 161-164. http://dx.doi.org/10.1145/2382564.2382568.

INTEF. (2019). *Programación, robótica y pensamiento computacional en el aula: Situación en España*.

Koehler, M. J., & Mishra, P. (2009). What is technological pedagogical content knowledge? *Contemporary Issues in Technology & Teacher Education, 9*(1), 60-70.

Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Towards the automatic recognition of computational thinking for adaptive visual language learning. *IEEE Symposium on Visual Languages and Human-Centric Computing*, 59-66. https://doi.org/10.1109/VLHCC.2010.17

Lee, E., Kafai, Y. B., Vasudevan, V., & Davis, R. L. (2014). Playing in the arcade: Designing tangible interfaces with MaKey MaKey for Scratch games. In *Playful user interfaces* (pp. 277-292). Singapore: Springer.

Levy, R. B. B., Ben-Ari, M., & Uronen, P. A. (2003). The Jeliot 2000 program animation system. *Computers & Education, 40*(1), 1-15. https://doi.org/10.1016/S0360-1315(02)00076-3

Lui, D., Walker, J. T., Hanna, S., Kafai, Y. B., Fields, D., & Jayathirtha, G. (2020). Communicating computational concepts and practices within high school students' portfolios of making electronic textiles. *Interactive Learning Environments*, *28*(3), 284-301.

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*. https://doi.org/10.1016/j.chb.2014.09.012

Malone, R. (2011). Curriculum Studies. In B. Walsh (Ed.), *Education Studies in Ireland: the Key Disciplines*. Dublin: Gill & Macmillan Ltd.

Martinez, S. L., & Stager, G. (2013). *Invent to learn. Making, Tinkering, and Engineering in the Classroom.* Paper presented at the Construting Modern Knowledge., Torrance, Canada.

Moreno-León, J., & Robles, G. (2015). *Dr. Scratch: A web tool to automatically evaluate Scratch projects.* Paper presented at the Workshop in primary and secondary computing education.

Moreno-León, J., Robles, G., & Román-González, M. (2015). Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking. *Revista de Educación a Distancia, 46*, 1-23.

NCCA. (2017). *Leaving Certificate Computer Science Specification*. Retrieved from http://ncca.ie/en/Curriculum_and_Assessment/Post-Primary_Education/Senior_Cycle/Consultation/LC-Computer-Science.pdf

NRC. (2010). *Report of a Workshop on the Scope and Nature of Computational Thinking* National Research Council, Washington DC: National Academies Press.

Pérez-Marín, D., Hijón-Neira, R., & Martín-Lope, M. (2018). A methodology proposal based on metaphors to teach programming to children. *IEEE Revista Iberoamericana de tecnologias del aprendizaje*, *13*(1), 46-53.

Pérez-Marín, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children?. *Computers in Human Behavior*, *105*, 105849.

Piaget, J., & Inhelder, B. (1999). *The Child's Conception of Space*. UK: Routledge.

Resnick, M., & Robinson, K. (2017). *Lifelong kindergarten: Cultivating creativity through projects, passion, peers, and play*: MIT Press.

Rich, K. M., Spaepen, E., Strickland, C., & Moran, C. (2020). Synergies and differences in mathematical and computational thinking: Implications for integrated instruction. *Interactive Learning Environments*, *28*(3), 272-283.

Román-González, M., Pérez-González, J. C., & Jiménez-Fernández, C. (2017). Which cognitive abilities underlie computational thinking? Criterion validity of the Computational Thinking Test. *Computers in Human Behavior, 72*, 678-691. https://doi.org/10.1016/j.chb.2016.08.047

Sáez-López, J. M., Román-González, M., & Vázquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools. *Computers & Education*, *97*, 128-141. https://doi.org/10.1016/j.compedu.2016.03.003

Snow, E., Tate, C., Rutstein, D., & Bienkowski, M. (2017). *Assessment design patterns for computational thinking practices in exploring computer science*. Retrieved from https://pact.sri.com/downloads/AssessmentDesignPatternsforComputationalThinking%20PracticesinECS.pdf

Sung, E. (2019). Fostering computational thinking in technology and engineering education: an unplugged hands-on engineering design approach. *Technology and Engineering Teacher*, *78*(5), 8-13.

Urbina Ramírez, S. (1999). Informática y teorías del aprendizaje. Píxel-Bit. *Revista de medios y educación, 12*, 87-100.

Velázquez-Iturbide, J. Á. (2018). *Report of the Spanish Computing Scientific Society on Computing Education in Pre-University Stages*. Paper presented at the TEEM'18: Proceedings of the Sixth International Conference on Technological Ecosystems for Enhancing Multiculturality, Salamanca, 2018.

Walsh, B., & Dolan, R. (2009). *A guide to teaching practice in Ireland*: Gill & Macmillan Ltd.

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2016). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*(1), 127-147. https://doi.org/10.1007/s10956-015-9581-5

Weintrop, D., & Wilensky, U. (2017). Comparing block-based and text-based programming in high school computer science classrooms. *ACM Transactions on Computing Education (TOCE), 18*(1), 1-25. https://doi.org/10.1145/3089799

Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). *The fairy performance assessment: measuring computational thinking in middle school.* Paper presented at the 43rd ACM technical symposium on Computer Science Education.

Wing, J. M. (2006). Computational Thinking. *Communications of the ACM, 49*(3).

Wing, J. M. (2008). Computational thinking and thinking about computing. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences, 366*(1881), 3717-3725. https://doi.org/10.1098/rsta.2008.0118

Wing, J. M. (2011). *Computational thinking.* Paper presented at the 2011 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC).

Yadav, A., Gretter, S., Good, J., & McLean, T. (2017). Emerging Research, Practice, and Policy on Computational Thinking. In R. P. & H. C. (Eds.), *Emerging Research, Practice, and Policy on Computational Thinking. Educational Communications and Technology: Issues and Innovations* (pp. 205-220): Springer, Cham.

Yadav, A., Gretter, S., Hambrusch, S., & Sands, P. (2016). Expanding computer science education in schools: understanding teacher experiences and challenges. *Computer Science Education*, *26*(4), 235-254. https://doi.org/10.1080/08993408.2016.1257418