Computers in Human Behavior 80 (2018) 407–411

Contents lists available at ScienceDirect

Computers in Human Behavior

journal homepage: www.elsevier.com/locate/comphumbeh

# Exploring the computational thinking effects in pre-university education

*Keywords:*
Computational thinking
21st century competences
Coding
Pre-university education

A B S T R A C T

Several countries have usually adopted several priorities for developing ICT competences from kindergarten to secondary education. Most of them are focused on the development of key competences and/or coding skills. Although coding may be very attractive for young students and a very good practice or experience, it could be more interesting to develop students' logical thinking skills and problem-solving skills throughout programming approaches or computational thinking. This is a very exciting challenge with lots of possibilities regarding coding, robots, mobiles devices, Arduino-based application, game-based learning and so on. Thus, it is very important to explore the effect that these experiences have been taking into the pre-university students, both at primary and secondary education, with a special focus on the computational thinking as one of the components inside the toolbox to develop a reflexive and critical education in order to help children to solve problems using the technology with which they will live daily.



## 1. Introduction

We live in a software-driven world (Manovich, 2013) and current Society demands skilled professionals for ICT (Information and Communication Technologies) business sector. A very common situation in countries with a high rate of unemployment is they have unfilled positions for engineers and technicians for the industry and digital services. This has caused an increasing approach for introduce digital or information technology (IT) literacy from the early beginning of the individual development (Bers, Flannery, Kazakoff, & Sullivan, 2014; Cejka, Rogers, & Portsmore, 2006; Kazakoff & Bers, 2012) till the high school courses (Allan, Barr, Brylow, & Hambrusch, 2010), even in post-secondary institutions (Astrachan, Hambrusch, Peckham, & Settle, 2009), combining it with other key competences such as reading, writing and math skills.

New devices (Alonso de Castro, 2014; Ramírez-Montoya & García-Peñalvo, 2017; Sánchez-Prieto, Olmos-Migueláñez, & García-Peñalvo, 2014), from smartphones and tablets to electronic learning toys and robots, find new audiences with increasingly young children (Fonseca Escudero, Conde González, & García-Peñalvo, 2017). This causes new challenges for teachers (Sánchez-Prieto, Olmos-Migueláñez, & García-Peñalvo, 2017), for example how to define developmentally appropriate activities and content for children of different ages (Bers et al., 2014).

The most frequent approach to teaching digital literacy has been to gradually encourage the learning of programming, and the term code-literacy (diSessa, 2000; Hockly, 2012; Vee, 2013) has been coined to referrer the process of teaching children programming tasks, from the simplest and most entertaining to the most complex, this way the student's progress is centered on the difficulty of the tasks and in their motivating characteristic.

However, this approach has epistemological antecedents in Papert (1980) works with Logo programming language, which promotes a constructionism rooted in Piaget (1954) constructivism that conveys the idea that the child actively builds knowledge through experience and the related "learn-by-doing" approach to education.

Consequently, at the same time that children learn human languages, both for speaking and writing, natural languages, encompassing all matters related with the experimental sciences (physics, chemistry, biology, etc.), and humanity languages, involving social sciences and humanities, it is also necessary they learn digital languages, in which ones of the competences to be success in the digital world are included, using coding as the way to solve problems and computational thinking as working paradigm (Llorens-Largo, 2015).

With the awareness of the importance of digital skills and related information technology (eSkills), there are several proposals worldwide about the need to include coding from the curriculum of non-university levels, starting since primary education (or sooner) (Balanskat & Engelhardt, 2015; Brown et al., 2013; García-Peñalvo, Llorens Largo, Molero Prieto, & Vendrell Vidal, 2017; Llorens Largo, García-Peñalvo, Molero Prieto, & Vendrell Vidal, 2017), because of the code-literacy skills are becoming understood as a core element for STEM (Science, Technology, Engineering, & Mathematics) subjects (Gelman & Brenneman, 2004; Weintrop et al., 2016), computational thinking may play an important role in K-12 STEM education because computational modelling is an effective approach for learning challenging science and math

https://doi.org/10.1016/j.chb.2017.12.005
0747-5632/

concepts (Hambrusch, Hoffmann, Korb, Haugan, & Hosking, 2009) and imaginative programming is the most crucial element of computing because it closely aligns mathematics with computing and, in this way, brings mathematics to life (Felleisen & Krishnamurthi, 2009).

A code-literate person means that can read and write in programming languages (M. Román-González, 2014), computational thinking is referred to the underlying problem-solving cognitive process that allows it. Thus, coding is a key way to enable computational thinking (Lye & Koh, 2014) and computational thinking may be applied to various kinds of problems that do not directly involve coding tasks (Wing, 2008).

## 2. What computational thinking is

Computational thinking has an increasing presence in the discussions about how to teach technology to pre-university students. However, there is not a consensus about what computational thinking is among computer scientists' community.

Jeannette M. Wing (2006) defined computational thinking as: "involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science", with a very important message about this "computational thinking is a fundamental skill for everyone, not just for computer scientists". She 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).

Isbell et al. (2009) proposed a focus on providing services, interfaces, and behaviors that involve a more central role for modelling as a means of formulating relationships and identifying relevant agencies that are sources of change.

Moreover, Riley and Hunt (2014) asserted that the best way to characterize computational thinking is as the way that computer scientists think, the manner in which they reason.

Aho (2012) simplified this concept defining it as the thought processes involved in formulating problems so "their solutions can be represented as computational steps and algorithms".

García-Peñalvo (2016b) defined computational thinking as the application of high level of abstraction and an algorithmic approach to solve any kind of problems.

Barr and Stephenson (2011) provided an operational definition of computational thinking as a "problem-solving process that includes (but is not limited to) the following characteristics: formulating problems in a way that enables us to use a computer and other tools to help solve them; logically organizing and analyzing data; representing data through abstractions such as models and simulations; automating solutions through algorithmic thinking (a series of ordered steps); identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources; generalizing and transferring this problem solving process to a wide variety of problems".

Hemmendinger (2010) stated, the ultimate computational thinking should not be to teach everyone to think like a computer scientific nor to convert every child in a software engineer, but rather to teach them to apply these common elements to solve problems and discover new questions to explore within and across all disciplines. Close to this approach Sysło and Kwiatkowska (2013) also underlined that computational thinking is a set of thinking skills that may not result in computer programming, it should focus on the principles of computing rather than on computer programming skills.

And many more definitions and approaches to Computational Thinking (see more in (García-Peñalvo, Reimann, Tuul, Rees, and Jormanainen, 2016b)).

## 3. The core elements of computational thinking

Computational thinking really means an attempt to capture computing disciplinary ways of thinking and practicing. Thus, it is an active problem solving methodology where the students should use a set of concepts, such as abstraction, patterns matching, etc., to process and analyze data, and to create real or virtual artefacts. Computational thinking does not imply to use technology in a mandatory way to solve the problems, but it is oriented to student will be able to solve problems throughout the technology. For this reason, the goal of introducing ICT in pre-university curriculum is not the students become merely tool user but tool builders.

Different core computational thinking set of components are proposed to define specific computational thinking frameworks.

Barr and Stephenson (2011) present a structured model that emerged focused on identifying core computational thinking concepts and capabilities. The core concepts are data collection, data analysis, data representation, problem decomposition, abstraction, algorithms and procedures, automation, parallelization and simulation. The capabilities are computer science, math, science, social studies and language arts.

Brennan and Resnick (2012) propose a computational thinking where the components are classified into three dimensions:

1. Computational concepts, which are the concepts that students employ when they code: sequences, loops, events, parallelism, conditionals, operators, and data.
2. Computational practices, which are problem solving practices that occur in the process of coding: experimenting and iterating, testing and debugging, reusing and mixing, and abstracting and modularization.
3. Computational perspectives, which are the students' understandings of themselves, their relationships with others, and the digital world around them: expressing, connecting and questioning.

Gouws, Bradshaw, and Wentworth (2013) design a computational thinking framework to serve as foundation for creating computational thinking resources. This framework is a two-dimensional grid. One dimension gathers the skill sets that make up computational thinking: processes and transformations, models and transformation, patterns and algorithms, inference and logic, and evaluations and improvements. The other dimension means the different levels at which these skills may be practiced: recognize, understand, apply, and assimilate.

Zapata-Ros (2015) tries to connect computational thinking with the learning theories conceptualizations and thinking models, proposing the following computational thinking components: bottom-up analysis, top-down analysis, heuristics, divergent thinking, creativity, problem solving, abstract thinking, recursion, iteration, Successive approximation methods (trial and error), collaborative methods, patterns, synectics and metacognition.

TACCLE 3 Coding European project (García-Peñalvo, 2016a; García-Peñalvo, Rees, Hughes, Jormanainen, Toivonen, and Vermeersch, 2016a; TACCLE 3 Consortium, 2017) organizes its guidelines and resources over three main dimensions: the ability to interpret phenomena as computations (coding/programming), the ability to harness computations for solving problems (logical thinking), and the ability to design and control automation tasks (control technology).

## 4. Computational thinking practices

Due to computational thinking has different interpretations, there are different approaches for introducing this approach into the classrooms, including those that consider computational thinking provides transparent advantages focusing on semantics rather the syntax of a specific language; those that prefer some kind of programming environment, based on blocks such as scratch or based on most traditional coding languages; those that control robots; or those that build physical kits to control things.

### 4.1. Developing mental models

The main idea behind this approach is computational thinking is not an alternative to learn coding; it is a way to reinforcing concepts and supplementing coding or programming education. The objective is that students may develop stronger mental models that ultimately make them better software engineers. Besides, the challenge is to use computational thinking approach to be useful and effective in a broader range of disciplines.

The CS Unplugged project (http://csunplugged.org) by CS Education Research Group at the University of Canterbury in New Zealand is a good example of pedagogical and creativity activities oriented to introduce computational thinking principles without using a computer (Bell, Witten, & Fellows, 2016).

Games have been used extensively for achieving this goal. Game design is a popular way to teach programming to students who have little or no prior programming experience (Chiazzese, Fulantelli, Pipitone, & Taibi, 2017; Peppler & Kafai, 2007; Repenning, 2006).

Basawapatna, Koh, and Repenning (2010; 2011) define Computational Thinking Patterns, which are abstracted programming patterns that are learned by students when they create games and can readily be used by students to model scientific phenomena.

### 4.2. Computational thinking through programming tools

Computational thinking is not coding, but computational thinking may be the outcome of a good planned programming practice. However, there are many experiences that use computational thinking as driver of programming skills (Tedre, 2017).

Introduce coding in the schools means to choose the programming language. There are approaches that prefer visual programming languages (Bau, Gray, Kelleher, Sheldon, & Turbak, 2017; Bennett, Koh, & Repenning, 2011) rather than traditional programming languages to facilitate the three dimensions of computational thinking (concepts, practices and perspectives) specially in K-12 contexts (Lye & Koh, 2014).

With these block-based commands languages students usually need only to drag and snap the blocks, reducing the cognitive load on the students and allowing them to focus on the logic and structures involved in programming rather than the mechanics of writing programs (Kelleher & Pausch, 2005).

Scratch (Resnick et al., 2009) is a very popular programming language to learn coding, languages (Burke, 2012; Lee, 2010), 3D models (Pinto-Llorente, Casillas-Martín, Cabezas-González, & García-Peñalvo, 2017; Pinto-Llorente, Casillas-Martín, Cabezas-Martín, & García-Peñalvo, 2016) or mathematics.

On the other hand, there exist opinions in favor of traditional languages (Vico, 2017). Logo is used to introduce coding (Lin & Liu, 2012), for example to help students with hearing disorders to learn English words (Miller, 2009) or to learn mathematics (Fessakis, Gouli, & Mavroudi, 2013). For order students, Python is increasingly used (Ahamed et al., 2010; Aiken et al., 2013).

### 4.3. Computational thinking for curriculum reform

The idea behind this approach is the claim of using a computational thinking approach in order to train future teachers' computational thinking ability.

Yu (2014) proposes that basic computer courses at the university level should embody many computational thinking methods, such as computer hardware components, various algorithms (sorting, recursion, etc.) in order to train students getting calculating thinking ability.

It is interesting the strength the connection between the computational thinking and the computational values support such as open publication of educational materials and resources, the policy on open publication of academic research, and the open online instruction based on non-proprietary software platforms (Ableson, 2012).

### 4.4. Computational thinking assessment

Developing assessments of student learning is an urgent area of need for the relatively young computer science education community as it advances toward the ranks of more mature disciplines (Buffum et al., 2015).

Interesting contributions regarding the measure and the assessment of computational thinking are the Fairy Assessment (Werner, Denner, Campe, & Kawamoto, 2012), which tries to measure the understanding and use of different computational concepts that students utilize to solve problem.

Koh, Basawapatna, Bennett, and Repenning (2010) identify several computational thinking patterns that young students abstract and develop during the creation of video-games in a controlled environment; they create an automated tool that analyses the games programmed and represents graphically how far each game has involved the different patterns when compared with a model.

Basu, Kinnebrew, and Biswas (2014) propose a more systematic assessment of Computational Thinking based science learning, using CTSiM – a Computational Thinking based science learning environment.

Dr. Scratch (http://drscratch.org/) (Moreno-León & Robles, 2015b, 2015a) is a web application that analyses automatically Scratch projects and gives feedback to improve programming skills and to develop computational skills. The research group behind Dr. Scratch has developed a computational thinking test (M. Román-González, 2015) and has demonstrate its convergent validity with respect to other traditional software quality and complexity metrics (Moreno-León, Robles, & Román-González, 2016; Marcos; Román-González, Pérez-González, & Jiménez-Fernández, 2017).

## 5. Discussion

Perhaps there is not a consensus over what computational thinking is, but there exist a lot of proposals and experiences that are teaching or practicing computational thinking in the similar way computer scientifics have learned it, i.e., through modelling things, automating and mechanizing things, building and controlling machinery, creating information processing systems, designing system, promoting the creativity coding games, etc.

The challenge we have is to break the current classroom model to introduce transversal computational thinking collaborative projects that may involve different subjects, different teachers and so on.

By no means, computational thinking is a new silver bullet for 21st century education, nor is it intended to replace other

approaches, computational thinking is another entry in the teachers' toolbox with the aim to prepare the students for the future years instead using only the approaches of the past.

## 6. Special issue contents

With the above presented background, this special issue about computational thinking in effects in pre-university education, we have fostered interesting experiences that try to manage the open challenges proposed in the discussion section.

Basogain et al. present experiences of developing computational thinking in Latin America and USA in order to introduce core computational thinking core elements through the use of visual languages such as Scratch and Alice.

Martín-Ramos et al. develop a computational thinking in Portugal in order to engage students with STEM education through hands-on projects based on the low-cost Arduino platform. Their approach was based on a peer-to-peer coaching scheme for conducting the introductory Arduino programming course.

Molins-Ruano et al. have chosen LOGO ideas but updated with more powerful technologies such as Python, Arduino and 3D printing, combined altogether in Phogo, a low-cost robot that is easy to build and control. The open and maker philosophies behind Phogo makes it more interesting as students will be able to access and study the electronic components.

Román-González et al. state that computational thinking is still a poorly defined construct, given that its nomological network has not been established yet. In this work, the authors continue studying the correlations between computational thinking and some fundamental cognitive variables, such as primary mental abilities and problem-solving ability, specifically the with non-cognitive factors, through the study of the correlations between CT, self-efficacy and the several dimensions from the 'Big Five' model of human personality: Openness to Experience, Conscientiousness, Extraversion, Agreeableness, and Neuroticism.

Fernández-Llamas et al. have developed an experiment in which a robot and a human teacher were used for teaching computational concepts to a group of K-12 students. The main goal of this contribution is not to analyze the scores obtained by the students, but to focus on their attitude towards robots.

Marcelino et al. propose a teacher distance training program in computational thinking and Scratch.

## Acknowledgements

This special issue is related to EU Erasmus + Programme. KA2 project “TACCLE 3 – Coding” (2015-1-BE02-KA201-012307).

This project has been funded with support from the European Commission. This communication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

## References

diSessa, A. A. (2000). *Changing minds: Computers, learning, and literacy*. Cambridge: MIT Press.

Ableson, H. (2012). From computational thinking to computational values. In *Proceedings of the 43rd ACM technical symposium on computer science education - SIGCSE '12* (p. 239). New York, NY, USA: ACM.

Ahamed, S. I., Brylow, D., Ge, R., Madiraju, P., Merrill, S. J., Struble, C. A., et al. (2010). Computational thinking for the sciences: A three day workshop for high school science teachers. In *Proceedings of the 41st ACM technical symposium on computer science education* (pp. 42–46). Milwaukee, Wisconsin, USA: ACM.

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

Aiken, J. M., Caballero, M. D., Douglas, S. S., Burk, J. B., Scanlon, E. M., Thoms, B. D., et al. (2013). Understanding student computational thinking with computational modeling. In P. V. Engelhardt, A. D. Churukian, & N. S. Rebello (Eds.), *2012 Physics education research conference* (Vol. 1513, pp. 46–49).

Allan, V., Barr, V., Brylow, D., & Hambrusch, S. (2010). *Computational thinking in high school courses*.

Alonso de Castro, M. G. (2014). Educational projects based on mobile learning. *Education in the Knowledge Society, 15*(1), 10–19.

Astrachan, O., Hambrusch, S., Peckham, J., & Settle, A. (2009). The present and future of computational thinking. *SIGCSE Bulletin Inroads, 41*(1), 549–550. https://doi.org/10.1145/1539024.1509053.

Balanskat, A., & Engelhardt, K. (2015). *Computing our future. computer programming and coding Priorities, school curricula and initiatives across Europe*. Retrieved from Brussels, Belgium: https://goo.gl/i5aQiv.

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12: What is Involved and What is the role of the computer science education community? *ACM Inroads, 2*(1), 48–54. https://doi.org/10.1145/1929887.1929905.

Basawapatna, A. R., Koh, K. H., & Repenning, A. (2010). Using scalable game design to teach computer science from middle school to graduate school. In *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education, ITiCSE '10* (pp. 224–228). New York, NY, USA: ACM.

Basawapatna, A. R., Koh, K. H., Repenning, A., Webb, D. C., & Marshall, K. S. (2011). Recognizing computational thinking patterns. In *SIGCSE'11-Proceedings of the 42nd ACM technical symposium on computer science education* (pp. 245–250). New York, NY, USA: ACM.

Basu, S., Kinnebrew, J. S., & Biswas, G. (2014). Assessing student performance in a computational-thinking based science learning environment. In S. Trausan-Matu, K. E. Boyer, M. Crosby, & K. Panourgia (Eds.), *Intelligent tutoring systems, its 2014* (Vol. 8474, pp. 476–481).

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.

Bell, T., Witten, I. H., & Fellows, M. (2016). *CS Unplugged. An enrichment and extension programme for primary-aged students*. New Zealand: University of Canterbury. CS Education Research Group Version 3.2.2. .

Bennett, V., Koh, K., & Repenning, A. (2011). Computing learning acquisition?. In *Paper presented at the proceedings - 2011 IEEE symposium on visual languages and human centric computing, VL/HCC 2011*.

Bers, M. U., Flannery, L., Kazakoff, E. R., & Sullivan, A. (2014). Computational thinking and tinkering: Exploration of an early childhood robotics curriculum. *Computers and Education, 72*, 145–157. https://doi.org/10.1016/j.compedu.2013.10.020.

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

Brown, N. C. C., Kölling, M., Crick, T., Peyton Jones, S., Humphreys, S., & Sentance, S. (2013). Bringing computer science back into schools: Lessons from the UK. In *Proceeding of the 44th ACM technical symposium on computer science education, SIGCSE '13* (pp. 269–274). New York, USA: ACM.

Buffum, P. S., Lobene, E. V., Frankosky, M. H., Boyer, K. E., Wiebe, E. N., & Lester, J. C. (2015). A practical guide to developing and validating computer science knowledge assessments with application to middle school. In *Proceedings of the 46th ACM technical symposium on computer science education. SIGCSE '15 (Kansas city, Missouri, USA, March 4th-7th)* (pp. 622–627). New York, NY, USA: ACM.

Burke, B. (2012). *Gamification 2020: What is the future of Gamification?* Gartner, Inc.. Nov, 5. Retrieved from.

Cejka, E., Rogers, C., & Portsmore, M. (2006). Kindergarten robotics: Using robotics to motivate math, science, and engineering literacy in elementary school. *International Journal of Engineering Education, 22*(4), 711–722.

Chiazzese, G., Fulantelli, G., Pipitone, V., & Taibi, D. (2017). Promoting computational thinking and creativeness in primary school children. In J. M. Dodero, M. S. Ibarra Sáiz, & I. Ruiz Rube (Eds.), *Fifth international conference on technological ecosystems for enhancing multiculturality (TEEM'17) (cádiz, Spain, October 18-20, 2017)*. New York, NY, USA: ACM (Article 6).

Felleisen, M., & Krishnamurthi, S. (2009). Why computer science doesn't matter. *Communications of the ACM, 52*(7), 37–40. https://doi.org/10.1145/1538788.1538803.

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5–6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87–97. https://doi.org/10.1016/j.compedu.2012.11.016.

Fonseca Escudero, D., Conde González, M.Á., & García-Peñalvo, F. J. (2017). Improving the information society skills: Is knowledge accessible for all? *Universal Access in the Information Society*. https://doi.org/10.1007/s10209-017-0548-6 (In press).

García-Peñalvo, F. J. (2016a). A brief introduction to TACCLE 3 – coding European project. In F. J. García-Peñalvo, & J. A. Mendes (Eds.), *2016 international symposium on computers in education (SIIE 16)*. USA: IEEE.

García-Peñalvo, F. J. (2016b). What computational thinking is. *Journal of Information Technology Research, 9*(3), v–viii.

García-Peñalvo, F. J., Llorens Largo, F., Molero Prieto, X., & Vendrell Vidal, E. (2017). Educación en Informática sub 18 (EI<18). *ReVisión, 10*(2), 13–18.

García-Peñalvo, F. J., Rees, A. M., Hughes, J., Jormanainen, I., Toivonen, T., & Vermeersch, J. (2016a). A survey of resources for introducing coding into schools. In F. J. García-Peñalvo (Ed.), *Proceedings of the fourth international conference on technological ecosystems for enhancing multiculturality (TEEM'16) (Salamanca, Spain, November 2-4, 2016)* (pp. 19–26). New York, NY, USA: ACM.

García-Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A., & Jormanainen, I. (2016b). *An*

*overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers. Belgium*. TACCLE 3 Consortium. https://doi.org/10.5281/zenodo.165123.
Gelman, R., & Brenneman, K. (2004). Science learning pathways for young children. *Early Childhood Research Quarterly, 19*(1), 150–158.
Gouws, L. A., Bradshaw, K., & Wentworth, P. (2013). Computational thinking in educational activities: An evaluation of the educational game light-bot. In *Proceedings of the 18th ACM conference on innovation and technology in computer science education, ITiCSE '13* (pp. 10–15). New York, NY, USA: ACM.
Hambrusch, S., Hoffmann, C., Korb, J. T., Haugan, M., & Hosking, A. L. (2009). A multidisciplinary approach towards computational thinking for science majors. In *Proceedings of the 40th ACM technical symposium on computer science education, SIGCSE '09, March 4-7, 2009, Chattanooga, TN USA* (pp. 183–187). New York, NY, USA: ACM.
Hemmendinger, D. (2010). A plea for modesty. *ACM Inroads, 1*(2), 4–7. https://doi.org/10.1145/1805724.1805725.
Hockly, N. (2012). Digital literacies. *ELT Journal, 66*(1), 108–112. https://doi.org/10.1093/elt/ccr077.
Isbell, C., Stein, A., Cutler, R., Forber, J., Fraser, L., Impagliazzo, J., … Xu, Y. (2009). (Re) Defining computing curricula by (re)defining computing. *ACM SIGCSE Bulletin, 41*(4), 195–207. https://doi.org/10.1145/1709424.1709462.
Kazakoff, E., & Bers, M. (2012). Programming in a robotics context in the kindergarten classroom: The impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia, 21*(4), 371–391.
Kelleher, C., & Pausch, R. (2005). Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers. *ACM Computing Surveys, 37*(2), 83–137. https://doi.org/10.1145/1089733.1089734.
Koh, K. H., Basawapatna, A., Bennett, V., & Repenning, A. (2010). Towards the automatic recognition of computational thinking for adaptive visual language learning. In *2010 IEEE symposium on visual languages and human-centric computing, VL/HCC 2010 (Leganés, Madrid, Spain, 21-25 Sept. 2010)* (pp. 59–66). USA: IEEE.
Lee, Y.-J. (2010). Developing computer programming concepts and skills via technology-enriched language-art projects: A case study. *Journal of Educational Multimedia and Hypermedia, 19*(3), 307–326.
Lin, J. M., & Liu, S. F. (2012). An investigation into parent-child collaboration in learning computer programming. *Educational Technology and Society, 15*(1), 162–173.
Llorens Largo, F., García-Peñalvo, F. J., Molero Prieto, X., & Vendrell Vidal, E. (2017). La enseñanza de la informática, la programación y el pensamiento computacional en los estudios preuniversitarios. *Education in the Knowledge Society, 18*(2), 7–17. https://doi.org/10.14201/eks2017182717.
Llorens-Largo, F. (2015). Dicen por ahí…que la nueva alfabetización pasa por la programación. *ReVisión, 8*(2), 11–14.
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, 41*, 51–61. https://doi.org/10.1016/j.chb.2014.09.012.
Manovich, L. (2013). *Software takes command*. New York, USA: Bloomsbury.
Miller, P. (2009). Learning with a missing sense: What can we learn from the interaction of a deaf child with a turtle? *American Annals of the Deaf, 154*(1), 71–82.
Moreno-León, J., & Robles, G. (2015a). Analyze your Scratch projects with Dr. Scratch and assess your computational thinking skills. In *Paper presented at the scratch conference, Amsterdam, The Netherlands*. http://jemole.me/replication/2015scratch/InferCT.pdf.
Moreno-León, J., & Robles, G. (2015b). Dr. Scratch: A web tool to automatically evaluate scratch projects. In *Proceedings of the workshop in primary and secondary computing education, WiPSCE '15 (London, United Kingdom, November 9-11, 2015)* (pp. 132–133). New York, NY, USA: ACM.
Moreno-León, J., Robles, G., & Román-González, M. (2016). Code to learn: Where does it belong in the K-12 curriculum? *Journal of Information Technology Education: Research, 15*, 283–303.
Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. New York, NY, USA: Basic Books.
Peppler, K. A., & Kafai, Y. B. (2007). Collaboration, computation, and creativity: Media arts practices in urban youth culture. In *Proceedings of the 8th iternational conference on computer supported collaborative learning* (pp. 590–592). New Brunswick, New Jersey, USA: International Society of the Learning Sciences.
Piaget, J. (1954). *The construction of reality in the child*. New York, USA: Basic Books.
Pinto-Llorente, A. M., Casillas-Martín, S., Cabezas-González, M., & García-Peñalvo, F. J. (2017). Building, coding and programming 3D models via a visual programming environment. *Quality & Quantity*. https://doi.org/10.1007/s11135-017-0509-4 (In press).
Pinto-Llorente, A. M., Casillas-Martín, S., Cabezas-Martín, M., & García-Peñalvo, F. J. (2016). Developing computational thinking via the visual programming Tool: Lego education WeDo. In F. J. García-Peñalvo (Ed.), *Proceedings of the fourth international conference on technological ecosystems for enhancing Multiculturality (TEEM'16) (Salamanca, Spain, November 2-4, 2016)* (pp. 45–50). New York, NY, USA: ACM.
TACCLE 3 Consortium. (2017). *TACCLE 3: Coding Erasmus + project website*. Retrieved from http://www.taccle3.eu/.
Ramírez-Montoya, M. S., & García-Peñalvo, F. J. (2017). La integración efectiva del dispositivo móvil en la educación y en el aprendizaje. *Revista Iberoamericana de Educación a Distancia, 20*(2), 29–47. https://doi.org/10.5944/ried.20.2.18884.
Repenning, A. (2006). Excuse me, I need better AI!: Employing collaborative diffusion to make game AI child's play. In *Proceedings of the 2006 ACM SIGGRAPH symposium on videogames* (pp. 169–178). Boston, Massachusetts: ACM.
Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K., … Kafai, Y. (2009). Scratch: Programming for all. *Commuication of the ACM, 52*(11), 60–67. https://doi.org/10.1145/1592761.1592779.
Riley, D. D., & Hunt, K. A. (2014). *Computational thinking for the modern problem solver*. Boca Raton, FL, USA: CRC Press.
Román-González, M. (2014). Aprender a programar 'apps' como enriquecimiento curricular en alumnado de alta capacidad. *Bordón. Revista de Pedagogía, 66*(4), 135–155. https://doi.org/10.13042/Bordon.2014.66401.
Román-González, M. (2015). Computational thinking Test: Design guidelines and content validation. In *Proceedings of EDULEARN15 conference, 6th-8th July 2015, Barcelona, Spain* (pp. 2436–2444).
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.
Sánchez-Prieto, J. C., Olmos-Migueláñez, S., & García-Peñalvo, F. J. (2014). Understanding mobile learning: Devices, pedagogical implications and research lines. *Education in the Knowledge Society, 15*(1), 20–42.
Sánchez-Prieto, J. C., Olmos-Migueláñez, S., & García-Peñalvo, F. J. (2017). MLearning and pre-service teachers: An assessment of the behavioral intention using an expanded TAM model. *Computers in Human Behavior, 72*, 644–654. https://doi.org/10.1016/j.chb.2016.09.061.
Sysło, M. M., & Kwiatkowska, A. B. (2013). Informatics for all high school students : A computational thinking approach. In I. Diethelm, & R. T. Mittermeir (Eds.), *Informatics in schools. Sustainable informatics education for pupils of all ages. 6th international conference on informatics in Schools: Situation, evolution, and perspectives, ISSEP 2013, Oldenburg, Germany, February 26–March 2, 2013. Proceedings* (pp. 43–56). Heidelberg: Springer.
Tedre, M. (2017). Many paths to computational thinking. In *Paper presented at the TACCLE 3 final conference, Brussels, Belgium*.
Vee, A. (2013). Understanding computer programming as a literacy. *LiCS Literacy in Composition Studies, 1*(2), 42–64.
Vico, F. (2017). ToolboX: Una estrategia transversal para la enseñanza de la programación en entornos educativos. *ReVisión, 10*(2), 53–68.
Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., et al. (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.
Werner, L., Denner, J., Campe, S., & Kawamoto, D. C. (2012). The fairy performance assessment: Measuring computational thinking in middle school. In *Proceedings of the 43rd ACM technical symposium on computer science education, SIGCSE '12* (pp. 215–220). New York, NY, USA: ACM.
Wing, J. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33–35. https://doi.org/10.1145/1118178.1118215.
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. In G. Costagliola, A. Ko, A. Cypher, J. Nichols, C. Scaffidi, C. Kelleher, et al. (Eds.), *2011 IEEE symposium on visual languages and human-centric computing* (p. 3).
Yu, D. (2014). The research on teaching reform of the "university computer basic" course. In *Police active colleges proceedings of the 2nd international conference on education technology and information system (ICETIS 2014)* (pp. 583–586). Amsterdam: Atlantis Press.
Zapata-Ros, M. (2015). Pensamiento computacional: Una nueva alfabetización digital. *RED, Revista de Educación a distancia, 46*.

Francisco José García-Peñalvo*
*Computer Science Department, Research Institute for Educational Sciences, GRIAL Research Group, University of Salamanca, Spain*

Antònio José Mendes
*Centre for Informatics and Systems, University of Coimbra, Portugal*
*E-mail address:* toze@dei.uc.pt.

* Corresponding author.
*E-mail address:* fgarcia@usal.es (F.J. García-Peñalvo).

Available online 5 December 2017