Revista Interuniversitaria de Investigación en Tecnología Educativa (RIITE)
Nº 7 Diciembre 2019 pp. 26-35 ISSN: 2529-9638 DOI: http://dx.doi.org/10.6018/riite.397151

Not the same: a text network analysis on computational thinking definitions to study its relationship with computer programming

No es lo mismo: un análisis de red de texto sobre definiciones de pensamiento computacional para estudiar su relación con la programación informática

Jesús Moreno-León
Programamos (España)
jesus.moreno@programamos.es

Gregorio Robles
Universidad Rey Juan Carlos (España)
grex@gsyc.urjc.es

Marcos Román-González
Universidad Nacional de Educación a Distancia (UNED) (España)
mroman@edu.uned.es

Juan David Rodríguez García
Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado (España)
juanda.rodriguez@educacion.gob.es

Recibido: 26/09/2019
Aceptado: 9/12/2019
Publicado: 26/12/2019

ABSTRACT

Even though countries from all over the world are modifying their national educational curriculum in order to include computational thinking skills, there is not an agreement in the definition of this ability. This is partly caused by the myriad of definitions that has been proposed by the scholar community. In fact, there are multiple examples in educational scenarios in which coding and even robotics are considered as synonymous of computational thinking. This paper presents a text network analysis of the main definitions of this skill that have been found in the literature, aiming to offer insights on the common characteristics they share and on their relationship with computer programming. As a result, a new definition of computational thinking is proposed, which emerge from the analysed data.

KEYWORDS

Computer Science Education; Programming; Text Structure

RESUMEN

A pesar de que países de todo el mundo están modificando su plan de estudios nacional para incluir habilidades de pensamiento computacional, no hay un acuerdo en la definición de esta capacidad. Esto se debe en parte a la gran cantidad de definiciones propuestas por la comunidad académica. De hecho, hay múltiples ejemplos en escenarios educativos en los que la programación e incluso la robótica se consideran sinónimos del pensamiento computacional. Este artículo presenta un análisis de la red de texto de las principales definiciones de esta habilidad que se han encontrado en la literatura, con el objetivo de ofrecer información sobre las características comunes que comparten y sobre su relación con la programación informática. Como resultado, se propone una nueva definición de pensamiento computacional que emerge de los datos analizados.

PALABRAS CLAVE

Educación informática; Programación; Estructura de Texto

CITA RECOMENDADA

Moreno-León, J., Robles, G., Román-González, M. y Rodríguez, J.D. (2019). Not the same: a text network analysis on computational thinking definitions to study its relationship with computer programming. *RIITE. Revista Interuniversitaria de Investigación en Tecnología Educativa*, 7, 26-35. Doi: http://dx.doi.org/10.6018/riite.397151

Principales aportaciones del artículo y futuras líneas de investigación:

- Las diferentes definiciones del concepto de pensamiento computacional coinciden en los elementos principales.
- Es posible aunar nodos comunes que evidencian que la dispersión conceptual es solo aparente.

1. INTRODUCTION

All over the world, governments have started to modify their national curriculum at both primary and secondary educational levels to incorporate Computational Thinking (CT), since this ability is considered a key set of problem-solving skills that must be developed by all learners (Bocconi et al., 2016). Still, there seems to be a lack of consensus on a formal definition of CT (Grover, 2015; Kalelioglu, Gülbahar, & Kukul, 2016; Román-González, Moreno-Leon & Robles, 2017) and, consequently, a myriad of CT definitions has been proposed in the last few years.

This diversity of theoretical approaches to CT, and the resulting lack of standardization, is problematic from the educational point of view. This is evidenced by the fact that in many educational contexts CT and programming (or coding) are used almost as synonymous (Balanskat & Engelhardt, 2015). However, what is the relationship between programming and CT, based on the definitions of the latter? Does programming arise as a fundamental core of CT? And what about the relationship between CT and robotics?

In addition, how different are the definitions of CT proposed during the last years? Do they share some common characteristics? Or are they focused on distinct dimensions of this competence?

In order to address these questions, we have collected the main CT definitions published in the literature, which are presented in Section 2. We have studied these definitions using a text network analysis (Paranyushkin, 2011), which is described in Section 3. Section 4 summarizes the main results, with a special focus on the most influential elements of the CT definitions, the main themes or topics of words, and the structure of the discourse. Finally, in Section 6 we discuss these findings and their implications, and conclude the paper with a new “data-driven” definition of CT.

2. BACKGROUND

In order to review the main definitions of CT that can be found in both academic and grey literature, we have reused the literature review performed in the recently published doctoral dissertation by one of the authors of this work (Moreno-León, 2018).

The first appearance of the term CT, although without elaboration, was in Seymour Papert's *Mindstorms* when discussing about the idea of creating samba schools for mathematics:

> There have already been attempts in this direction by people engaged in computer hobbyist clubs and in running computer drop-in centers. In most cases, although the experiments have been interesting and exciting, they have failed to make it because they were too primitive. Their computers simply did not have the power needed for the most engaging and shareable kinds of activities. Their visions of how to integrate computational thinking into everyday life was insufficiently developed. But there will be more tries, and more and more. And eventually, somewhere, all the pieces will come together and it will catch. (Papert, 1980, p. 182).

But the term CT did not become popular until 2006, when Wing published her seminal paper on CT with the following definition:

> CT involves solving problems, designing systems, and understanding human behavior, by drawing on the concepts fundamental to computer science. CT includes a range of mental tools that reflect the breadth of the field of computer science [...]. It represents a universally applicable attitude and skill set everyone, not just computer scientists, would be eager to learn and use. (Wing, 2006, p. 33).

The timing was more opportune at that moment, and the term quickly gained popularity and raised the interest of both the scholar and educational communities. Since then, other influential authors and organizations have proposed new definitions for CT from different perspectives.

One of these new, alternative definitions is provided by Lu and Fletcher, who defend that being proficient in CT "helps us to systematically and efficiently process information and tasks" (Lu & Fletcher, 2009, p. 261).

Aiming to support educators in the introduction of CT in K-12, the Computer Science Teachers Association and the International Society for Technology in Education developed the following operational definition of CT:

> CT is 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; and generalizing and transferring this problem-solving process to a wide variety of problems. (ISTE & CSTA, 2011, p. 1).

The vision of CT proposed by Wing has also received criticism, though. Hence, Denning argues that CT is equivalent to algorithmic thinking, a concept well known since the 1950s that could be defined as "a mental orientation to formulating problems as conversions of some input to an output and looking for algorithms to perform the conversions." (Denning, 2009, p. 28).

Most of the complaints that the term received were in terms of ambiguity and vagueness. As a result, in 2011 Wing proposed a new definition of CT aiming to clarify certain aspects of her initial proposal: "CT 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. 1).

A similar definition is introduced by Aho, who defines CT as the "thought processes involved in formulating problems so their solutions can be represented as computational steps and algorithms." (Aho, 2011, p. 2).

Placing the focus on the educational community, Barr & Stephenson define CT as:

> […] an approach to solving problems in a way that can be implemented with a computer. Students become not merely tool users but tool builders. They use a set of concepts, such as abstraction, recursion, and iteration, to process and analyze data, and to create real and virtual artifacts. CT is a problem-solving methodology that can be automated and transferred and applied across subjects. (Barr & Stephenson, 2011, p. 49).

In a report advocating for computing education in UK schools, the British Royal Society goes one step forward by highlighting the presence of computation in nature as another reason to teach CT, which was defined as "the process of recognizing aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes." (Furber, 2012, p. 29).

Following this approach, in the context of an intervention to integrate CT with K-12 science education, Sengupta et al. propose a theoretical framework where authors state that

> CT draws on concepts and practices that are fundamental to computing and computer science. It includes epistemic and representational practices, such as problem representation, abstraction, decomposition, simulation, verification, and prediction. However, these practices are also central to the development of expertise in scientific and mathematical disciplines. (Sengupta et al., 2013, p. 351).

Aiming to gather the elements that are accepted as comprising CT in most CT definitions in educational environments, Grover & Pea review aforementioned definitions and propose the following elements as the basis of curricula that aim to support CT learning and assessment:

> […] abstractions and pattern generalizations (including models and simulations); systematic processing of information; symbol systems and representations; algorithmic notions of flow of control; structured problem decomposition (modularizing); iterative, recursive, and parallel thinking; conditional logic; efficiency and performance constraints; and debugging and systematic error detection. (Grover & Pea, 2013, p. 39).

Also with the aim of supporting educators, Computing at School proposed a framework that states that, when working in the classroom, CT involves both concepts (logic, algorithms, decomposition, patterns, abstraction, and evaluation) and approaches (tinkering, creating, debugging, persevering, and collaborating), thus pointing to some non-cognitive skills being part of CT (Csizmadia et al., 2015).

As we can see, the computer science educational community has had difficulties in finding a definition of CT that everyone agrees upon. This is a view shared by Mannila et al., who wrote a report on the current status of the coverage of computer science in K-9 education in several countries (Mannila et al., 2014). In this report, the authors define CT as a set of concepts and thinking processes from computer science that help in formulating problems and their solutions in different disciplines.

Besides multiple contributions to define CT, we also find authors and organizations that modify their initial proposals over time. Hence, in the [Interim] CSTA K-12 Computer Science Standards1 we find yet another definition of CT by the CSTA Standards Task Force:

> We believe that CT is a problem-solving methodology that expands the realm of computer science into all disciplines, providing a distinct means of analyzing and developing solutions to problems that can be solved computationally. With its focus on abstraction, automation, and analysis, CT is a core element of the broader discipline of computer science. (CSTA, 2016, p. 6)

More recently, two new definitions for CT have been published. Tedre & Denning describe CT as "a popular phrase that refers to a collection of computational ideas and habits of mind that people in computing disciplines acquire through their work in designing programs, software, simulations, and computations performed by machinery." (Tedre & Denning, 2016, p. 120). Lastly, in a more informal approach, Wolfram states that CT:

[1] https://www.csteachers.org/Page/standards

> "intellectual core is about formulating things with enough clarity, and in a systematic enough way, that one can tell a computer how to do them [...] CT is a broad story, because there are just a lot more things that can be handled computationally [...] But how does one tell a computer anything? One has to have a language" (Wolfram, 2016).

As a summary, Table 1 shows the reviewed publications that include a definition of CT ordered by date of publication, and also shows if each contribution was published in a book, journal, magazine, conference proceedings or official report, among others. As can be seen, a majority of the proposals following Wings' definition were published in computer science environments, while since 2011 most of the definitions were proposed in scenarios closer to the educational community.

Table 1. Reviewed publications that propose a CT definition

Publication	Type
(Papert, 1980)	Book - Mindstorms
(Wing, 2006)	Magazine - Communications of the ACM
(Lu & Fletcher, 2009)	Newsletter – SIGCSE Bulletin
(Denning, 2009)	Magazine - Communications of the ACM
(Wing, 2011)	Magazine – The LINK
(Aho, 2011)	Symposioum – Ubiquity
(Barr & Stephenson, 2011)	Journal – Inroads
(ISTE & CSTA, 2011)	Report - ISTE & CSTA
(Furber, 2012)	Report - Royal Society
(Sengupta et al., 2013)	Journal – Education and information technologies
(Grover & Pea, 2013)	Journal - Educational researcher
(Mannila et al., 2014)	Report – Working group ITiCSE
(Csizmadia et al., 2015)	Report - CAS
CSTA K-12 CS Standards, 2016	Report - CSTA
(Tedre & Denning, 2016)	Proceedings - Koli calling
(Wolfram, 2016)	Opinion column

3. METHODS

In order to detect the central concepts of CT that emerge from the myriad of CT definitions that have been reviewed, a text network analysis (Paranyushkin, 2011) was performed on a document containing all these definitions. Especifically, we used InfraNodus, which is an open-source tool used by the academic community to perform text-related studies and to make sense of pieces of disjointed textual data (Paranyushkin, 2019). The solution automates the visualization of a text as a network; shows the most relevant topics, their relations, and the structural gaps between them; and enables the analysis of the discourse structure and the assessment of its diversity based on the community structure of the graph (Paranyushkin, 2019).

As a first step, the tool removes the syntax information (such as commas and dots) and converts the words into their morphemes to reduce redundancy (Paranyushkin, 2019). For instance, "computers" becomes "computer" or "programmed" becomes "program". In addition, the tool removes articles, conjunctions, auxiliary verbs and some other frequently used words, such as 'is' or 'the'. Thus, a sentence that reads "the process of recognizing aspects of computation in the world that surrounds us" is turned into "process recognize aspect computation world surround".

The resulting sequence is then converted by Infranodus into a directed network graph, where the nodes are the different words while the edges represent their co-occurrences. The tool

identifies the nodes that appear most often on the shortest paths between any two randomly chosen nodes in the network -i.e., betweenness centrality- and detects the groups of nodes that tend to appear more often together -topical groups-. The result is a visual network representation of the text that, based on colors and sizes, enables a clear vision of its structure and topics. Finally, the tool also identifies the structure of the discourse, which can be categorized as dispersed, diversified, focused or biased (Paranyushkin, 2019).

4. RESULTS

The text network analysis generates 148 nodes (words) and 658 edges (co-occurrences). The average degree, which represents the number of nodes every node is connected to, is 4.45.

Figure 1 is a graph image that can be used to get a clear visual representation of the main topics and influential keywords of the reviewed CT definitions. Colors in Figure 3.1 indicate the distinct contextual clusters, or themes, which are communities of words that are closely related. On the contrary, words that appear in different contexts are shown far away from each other. The size of the nodes reflects their betweenness centrality, which is the number of different themes or contexts each node connects.

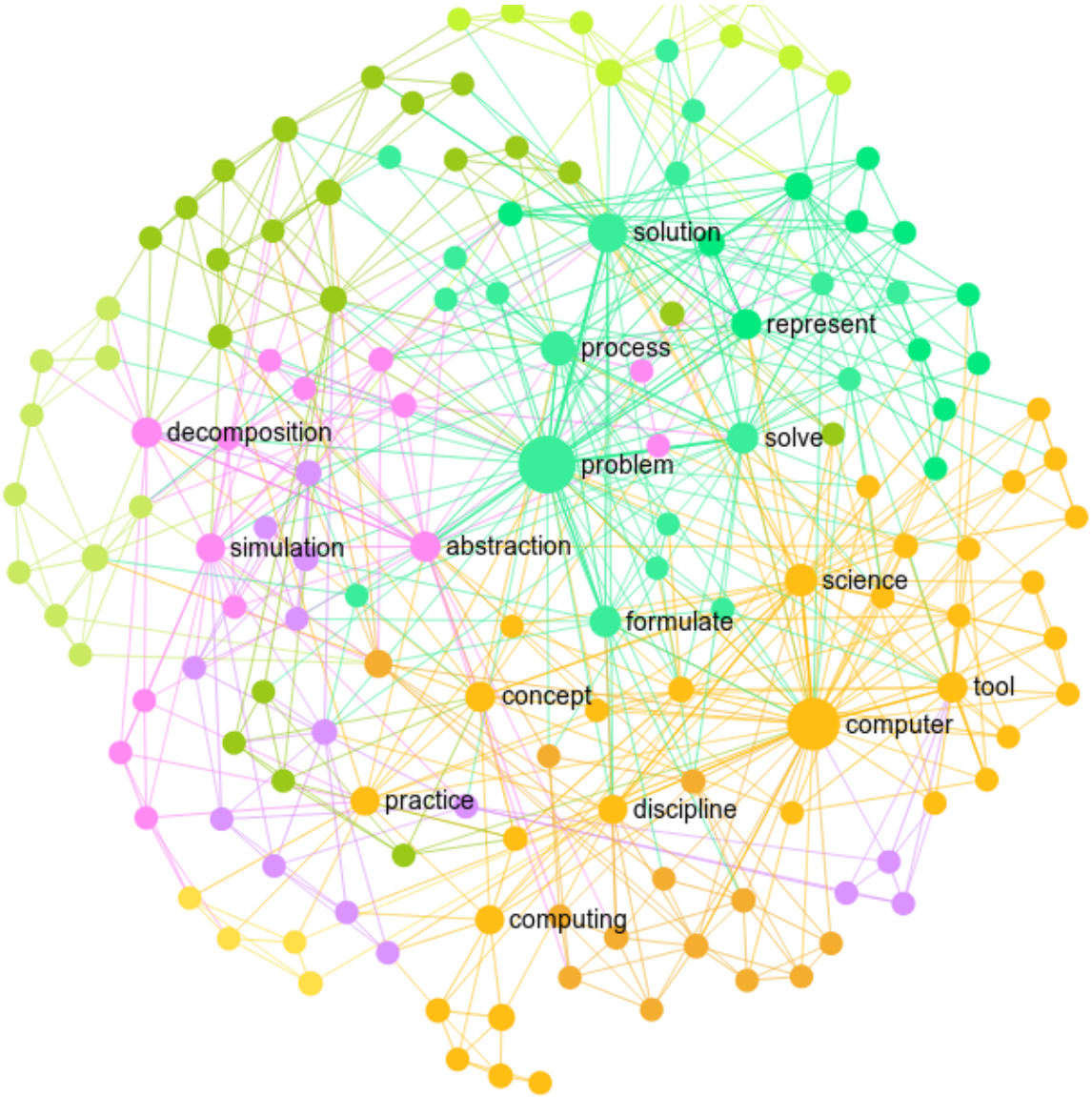

Figure 1. Visual representation of the main topics and influential keywords in CT definitions.

As can be seen in Figure 1, the most influential elements of the network, since they link different topics together, are “problem”, “computer”, “solution” and “process”. These nodes are shown bigger on the graph.

Table 2 presents the main words within the most influential contextual clusters. These words are the nodes that have more connections within each group, being in consequence the most influential words of the themes. However, connections to the other clusters in the network are not considered in this case. Color column in Table 2 refers to the colors in Figure 1.

Table 2. Most influential communities of words in CT definitions.

Cluster	Words in the context	Color[1]
1	computer, science, tool	Orange
2	problem, solve, solution	SpringGreen
3	abstraction, simulation, decomposition	Fuchsia
4	system, information, algorithmic	Olive
5	logic, debug, performance	Purple

[1] Refers to the colors used in Figure 1

In terms of network structure, the analysis indicates that it is “focused” (modularity -which measures how pronounced is the community structure- is 0.49, 18% of words are in the top topic and its influence dispersal is 40%). This means that the most influential words are concentrated around one topic and the discourse is focused on a certain perspective.

4. DISCUSSION AND CONCLUSIONS

The results of the text network analysis show that neither programming nor coding emerge among the most influential words of the main CT definitions. Why are, then, CT and programming considered almost synonymous in many contexts?

As discussed by Voogt et al.:

> the concepts of CT and the practice of programming are difficult to delineate in the literature because many CT studies or discussions of theory use programming as their context [...]. This can be confusing to the reader and often lead to the impression that CT is the same as programming or at the very least that CT requires the use of programming. (Voogt et al., 2015, p. 716)

So even though, as stated by our text network analysis, scholars do not claim that programming must be the required context to develop CT skills, a vast majority of interventions in which these skills are trained make use of different types of programming tasks (Kalelioglu et al., 2016; Lye and Koh, 2014).

However, although programming makes CT concepts concrete and nowadays is therefore a *de facto* method for the learning and teaching of these skills (Bocconi et al., 2016) this situation might change in the near future due to several factors. On the one hand, educators and researchers may find other strategies to develop CT skills, as it is already the case with the use of unplugged activities (Brackmann et al., 2017). On the other hand, the intense development of artificial intelligence solutions, especially those based on machine learning, may alter dramatically the way computer programming is performed (Rodríguez-García, Moreno-León, Román-González & Robles, 2019).

In other words, just like we distinguish between verbal aptitude -which is in the order of human cognitive abilities, with an important innate base- and literacy skill -which is an instrumental competence that requires a relatively formal teaching and learning process- we could similarly establish a distinction between CT -human cognitive ability- and programming skills -instrumental competence- (Román-González, Pérez-González & Jiménez-Fernández, 2017). However, if CT is, first and foremost, a human cognitive ability, it is very striking that "cognition" or "cognitive ability" do not appear as key terms of the analysis. Perhaps this is a result of the fact that there

are more definitions of CT proposed by computer scientists than by psychologists or pedagogues, as shown in Table 1?

It is also worth noting that robotics, which is sometimes used in school scenarios as a context to develop CT skills (Balanskat & Engelhardt, 2015) does not even appear in the 148 nodes of the analysis.

As mentioned earlier, the analysis indicates that the network structure is "focused". Such discourse structure is characteristic "for newspaper articles, essays, reports, which are designed to provide a clear and concise representation of a certain idea" (Paranyushkin, 2019). This result is quite interesting, since one might expect that a list of definitions could have a more dispersed or diversified structure. Consequently, this result shows that, even with some differences -since the structure is not biased-, the definitions have lots of elements in common.

Finally, taking into account both the most influential elements and the communities of words highlighted by the text network analysis, we could almost propose (yet) a new definition of CT. Based on this data, CT would be the ability to formulate and represent problems to solve them by making use of tools, concepts and practices from the computer science discipline, such as abstraction, decomposition or the use of simulations. Such data-driven definition could be of interest for the educational community, since it clarifies the relationship between CT and programming (or robotics, for that matter), being the former a cognitive ability of the subject, and the latter just one of the means to develop it.

5. REFERENCES

Aho, A. V. (2011). Ubiquity symposium: Computation and computational thinking. *Ubiquity*, *2011*(January), 1.

Balanskat, A., & Engelhardt, K. (2015). *Computing our future: Computer programming and coding-Priorities, school curricula and initiatives across Europe*. European Schoolnet.

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.

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education-Implications for policy and practice* (No. JRC104188). Joint Research Centre (Seville site).

Brackmann, C. P., Román-González, M., Robles, G., Moreno-León, J., Casali, A., & Barone, D. (2017). Development of computational thinking skills through unplugged activities in primary school. In *Proceedings of the 12th Workshop on Primary and Secondary Computing Education* (pp. 65-72). ACM.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C., & Woollard, J. (2015). Computational thinking-A guide for teachers.

CSTA Standards Task Force. (2016). *[Interim] CSTA K-12 Computer Science Standards*. New York: CSTA.

Denning, P. J. (2009). The profession of IT beyond computational thinking. *Communications of the ACM, 52*(6):28–30.

Furber, S. (2012). *Shut down or restart? The way forward for computing in UK Schools*. London: The Royal Society.

Grover, S. (2015, April). "Systems of Assessments" for deeper learning of computational thinking in K-12. In *Proceedings of the 2015 annual meeting of the American educational research association* (pp. 15-20).

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational researcher*, *42*(1), 38-43.

ISTE and CSTA (2011). *Operational definition of computational thinking for K-12 education*. Available at: http://www.iste.org/docs/ct-documents/computational-thinking-operational-definition-flyer.pdf

Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A Framework for Computational Thinking Based on a Systematic Research Review. *Baltic Journal of Modern Computing, 4*(3), 583.

Lu, J. J., & Fletcher, G. H. (2009, March). Thinking about computational thinking. In *ACM SIGCSE Bulletin* (Vol. 41, No. 1, pp. 260-264). ACM.

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.

Mannila, L., Dagiene, V., Demo, B., Grgurina, N., Mirolo, C., Rolandsson, L., & Settle, A. (2014, June). Computational thinking in K-9 education. In *Proceedings of the working group reports of the 2014 on innovation & technology in computer science education conference* (pp. 1-29). ACM.

Moreno-León, J (2018). *On the development of computational thinking skills in schools through computer programming with Scratch* (Doctoral dissertation).

Paranyushkin, D. (2011). Identifying the pathways for meaning circulation using text network analysis. *Nodus Labs*, *26*.

Paranyushkin, D (2019). InfraNodus: Generating Insight Using Text Network Analysis, *Proceedings of WWW '19 The World Wide Web Conference*, Pages 3584-3589, San Francisco, CA, USA — May 13 – 17, 2019

Papert, S. (1980). *Mindstorms: Children, computers, and powerful ideas*. Basic Books, Inc..

Rodríguez-García, J., Moreno-León, J., Román-González, M., & Robles, G. (2019). Developing computational thinking at school with machine learning: an exploration. In *2019 International Symposium on Computers in Education (SIIE)* (pp. 49-54). IEEE.

Román-González, M., Moreno-León, J., & Robles, G. (2017). Complementary tools for computational thinking assessment. In *Proceedings of International Conference on Computational Thinking Education (CTE 2017), S. C Kong, J Sheldon, and K. Y Li (Eds.). The Education University of Hong Kong* (pp. 154-159).

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.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*(2), 351-380.

Tedre, M., & Denning, P. J. (2016, November). The long quest for computational thinking. In *Proceedings of the 16th Koli Calling International Conference on Computing Education Research* (pp. 120-129). ACM.

Voogt, J., Fisser, P., Good, J., Mishra, P., & Yadav, A. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*(4), 715-728.

Wing, J. (2006). Computational thinking. *Communications of the ACM*, *49*(3), 33-35.

Wing, J. (2011). Research notebook: Computational thinking—What and why. *The Link Magazine*, 20-23.

Wolfram, S. (2016). How to teach computational thinking. *Stephen Wolfram Blog*. Available at https://blog.stephenwolfram.com/2016/09/how-to-teach-computational-thinking/

INFORMACIÓN SOBRE LOS AUTORES

Jesús Moreno-León
Programamos

Jesús Moreno-León is a researcher at Programamos, a non-profit organization promoting computational thinking skills in education. He participates as an advisor in multiple international committees and expert groups regarding the use of computer programming in education. As an example, since 2013 he has collaborated with different roles -at this moment as a national ambassador- with EU Code Week, an initiative promoted by the European Commission that reached over 2.7 million people during the last edition. His main lines of research are related to the inclusion of computational thinking in schools, the assessment of the development of this ability, and the evaluation of its educational impact.
Web: http://jemole.me/

Gregorio Robles
Universidad Rey Juan Carlos

Gregorio Robles is Associate Professor at the Universidad Rey Juan Carlos, in Madrid, Spain. He mainly does research in following two fields: a) Software engineering: he is specialized in software analytics of Free/Libre/Open Source Software systems. His primary focus is on mining software repositories, socio-technical issues such as community metrics, software evolution, and development effort estimation. And b) Computational thinking (CT): he investigates the effect of using coding as a way to help students learn beyond coding. I also work on how the development of CT skills can be assessed.
Web: http://gsyc.urjc.es/grex

Marcos Román-González
Universidad Nacional de Educación a Distancia (UNED)

Marcos Román-González is Associate Professor at the Department of Methods of Research and Diagnosis in Education I (Faculty of Education, UNED). His research lines are related to code-literacy (teaching-learning processes with/through computer programming languages) and computational thinking (cognitive problem-solving ability that underlies computer programming tasks, among others). He is the author of the Computational Thinking Test (CTt), which has been endorsed by the research community through several publications.
Web: http://goo.gl/oox5Qn

Juan David Rodríguez García
Instituto Nacional de Tecnologías Educativas y de Formación del Profesorado

Juan David Rodríguez García is Teaching Technical Advisor at INTEF, the unit of the Spanish Ministry of Education and Vocational Training responsible for the integration of ICT and Teacher Training in the non-university educational stages. He is currently working toward his PhD Thesis in the field of Computational Thinking (CT) development and the use of machine learning contents as a means to develop CT.
Web: http://juandarodriguez.es