Developing Computational Thinking: Approaches and Orientations in K-12 Education

Conference Paper · June 2016

7 authors, including:


Stefania Bocconi
Italian National Research Council
77 PUBLICATIONS **761** CITATIONS
SEE PROFILE

Augusto Chioccariello
Italian National Research Council
29 PUBLICATIONS **357** CITATIONS
SEE PROFILE


Giuliana Dettori
Italian National Research Council
253 PUBLICATIONS **1,105** CITATIONS
SEE PROFILE


Anusca Ferrari
European Schoolnet
22 PUBLICATIONS **1,559** CITATIONS
SEE PROFILE

Proceedings EdMedia 2016, Vancouver , June 28-30, 2016

Developing Computational Thinking: Approaches and Orientations in K-12 Education

Stefania Bocconi
Augusto Chioccariello
Giuliana Dettori
Institute for Educational Technology, National Research Council (CNR) of Italy
Italy
bocconi@itd.cnr.it
augusto@itd.cnr.it
dettori@itd.cnr.it

Anusca Ferrari
Katja Engelhardt
European Schoolnet
Belgium
anusca.ferrari@eun.org
katja.engelhardt@eun.org

Panagiotis Kampylis
Yves Punie
European Commission, DG JRC Institute for Prospective Technological Studies
Spain
panagiotis.kampylis@ec.europa.eu
yves.punie@ec.europa.eu

This paper reports on initial findings of a study on developing computational thinking (CT) as a 21st Century skill. Extensive desktop research collecting evidences from the (academic and grey) literature has been complemented with a survey on policy documents and several semi-structured interviews with policy makers, researchers and practitioners involved in the implementation of relevant policy and grassroots initiatives to further understand the uptake of CT approaches in K-12 educational contexts. Preliminary findings from the literature review indicate that the debate on definitional issues remains open. Despite an increasing number of CT implementations in both formal and informal education settings, research still appears necessary on how CT skills develop in K-12 students, what pedagogical approaches can facilitate the effective introduction of CT concepts, and how the acquisition of CT skills should be assessed in practice.

Introduction

Digital technology has radically changed the way people think and work. Informatics has contributed to the scientific and technological development of our society and to the digital revolution. Computational thinking (CT) is the term in use to refer to the key ideas of the disciplinary areas of informatics and computer science. This topic has been increasingly gaining attention in the educational field in the past decade, by researchers, practitioners and policy makers, giving rise to an increasingly large amount of academic and grey literature, as well as being mentioned, explicitly or implicitly, also in several policy-related documents. An element that emerges from the debate is the importance of the topic not only for its crucial content, but also for the positive influence its study can have on the development of general thinking skills.

The complexity of the CT field, however, and the lack of a unique definition and orientation on its development, highlight the need to carry out some investigation in order to understand more deeply its nature and how it could be fruitfully introduced into K-12 curricula. Contributing in this respect is the aim of this study. This article presents the preliminary outcomes of a wide-angle literature analysis. In the next section we give a short account of the initial definitions and development attempts, then we present the aims and methodology of our study and sketch the landscape that is emerging from our compound analysis.

Setting the Context

"Computational Thinking" is the title of a 'viewpoint' published in the Communications of the ACM in March 2006 by Jeanette Wing, where it is used as a shorthand for "thinking as a computer scientist", i.e., the ability to use computer science concepts to solve problems. Moreover, CT is presented as "a fundamental skill for everyone, not just for computer scientists. To reading, writing, and arithmetic, we should add computational thinking to every child's analytical ability" (Wing, 2006). This article has stimulated a lively international debate from academia, education, industry, and policy makers that is still active. Consensus on CT definition, however, has not yet been reached, as one might expect on an issue related to the foundations and epistemology of computer science as a discipline. The epistemological discussion includes, just to mention a few:

- *the distinction between computer science and mathematics*: "Computer science is … the study of algorithms" (Knuth, 1974);
- *computing as natural vs. artificial science*: "Computation is present in nature even when scientists are not observing it ... Computation is more fundamental than computational thinking" (Denning, 2009).

The Report on "Workshop on the Scope and Nature of Computational Thinking", which was organised by the USA National Research Council (NRC) in 2010 and involved key international researchers, among whom J. Wing, documents the lack of consensus on the basic definitions and a number of open questions. Throughout the workshop, "participants expressed different views about the scope and nature of computational thinking; almost every participant held his or her own perspective on computational thinking that placed greater emphasis on particular aspects or characteristics, such as, for instance, how and to what extent, if any, the ability to program is an essential aspect of computational thinking, as well as how to define programming in the context of CT" (NRC, 2010, p. 59). The relationship between programming and computational thinking is still an open issue. In the NRC report, it is related with the connection of CT to technology, claiming that "the computer -and notions of computer programming—can make the concepts, principles, methods, models, and tools of CT tangible, in much the same spirit that LOGO was first inspired" (NRC, 2010, p. 61).

As a further contribution to this debate, J. Wing, with input from Al Aho, Jean Cuny and Larry Snyder (2011), revised the original definition as follows: "CT is the thought processes involved in formulating a problem and expressing its solution(s) in such a way that a computer—human or machine—can effectively carry out." This definition has become a reference point for subsequent discussions on CT in the literature. Wing (2014) elucidates it suggesting that "informally, computational thinking describes the mental activity in *formulating a problem* to admit a computational solution. The solution can be carried out by a human or machine. Also, computational thinking is not just about problem solving, but also about *problem formulation*".

It is important to notice, however, that in several articles on CT, the invention of the term is attributed to Papert (1980). In his perspective, knowledge emerges as a result of an active engagement with the world through the creation and manipulation of artifacts that are seen as objects to think with. Hence, the Logo research programme and related studies might be still relevant to CT at school (Grover & Pea, 2013).

In relation to the introduction of CT skills in K-12 education, a number of prominent European institutions have intervened in the debate. In 2012, the Royal Society published the report "Shut down or restart? The way forward for computing in UK". The Académie des Sciences (France) intervened on this subject with the report "L'enseignement de l'informatique en France – Il est urgent de ne plus attendre". Moreover, Informatics Europe and the Association of Computing Machinery (ACM) Europe, Working Group on Informatics Education, urged Europe "not to miss the boat" on this subject. All those reports call for a change in the curricula to make room for informatics as a discipline. However, the academic debate alone is not enough to influence the policy level. For instance, a request to include Computing and CT in the recent US K-12 STEM curriculum was made by the computer science community, arguing that computing and CT are "an integral part of science and therefore constitute essential knowledge and practices for students" (NRC, 2012, p. 334). By that time, however, such request was rejected.

A parallel UK request that was supported by industry statements like the Next Generation Report and Eric Schmidt's speech on UK education managed to convince the government that there was a problem to be solved. As a result, still in 2012 the UK Department for Education declared the introduction of computer science teaching into UK schools an official goal (Brown et al., 2013). Attempts to add a new discipline to an already crowded K-12 curriculum, however, presents some problems. Thus, introducing CT into schools requires special attention to contents. Subjects like science, technology, engineering and math (STEM) seem to offer a direct match with CT key ideas (Weintrop et al., 2015). However, other subjects can be seen as linked to

CT as well. For example, interactive storytelling and video games construction (Repenning et al., 2015) are activities related to media art that can offer interesting connections with CT. In response to a gap between perceived social needs for skills related to computing, CT and the current offering of the educational settings, a number of grass-roots initiatives are emerging as relevant actors in the CT debate. Some of those initiatives have quickly reached an international status like Fab Foundation lunched in 2009 in the USA, CoderDojo started in Ireland in 2011, or Code.org in the USA in 2013. Some are backed by industry (e.g. Code.org), whereas others are based on volunteer activity (e.g. CoderDojo). The latter grassroots initiatives have a relevance that goes beyond their lobbing power. They are generally developed in informal settings, not necessarily tied to curricular constraints, and foster a participatory technological culture based on community of practices. This trend might help the inclusion of CT in the school curriculum or enter in conflict with school education in case of lack of evolution on the school side.

The Study

The CompuThink study (https://ec.europa.eu/jrc/en/computational-thinking) is a forward looking exploration aiming to contribute to the current debate on CT, coding and transversal skills at European and international level. Its ultimate goal is to further understand the core concepts and attributes of CT as a key skill in K-12 education, in particular on the following aspects*: What characterizes CT differently from other thinking skills? What is the relationship to programming and to Computer Science? Should CT be addressed within a specific subject (e.g., CS), integrated in STEM, or as a cross-curriculum topic? What pedagogical approaches could facilitate CT introduction in K-12 education? How should CT be assessed? What is needed to further the CT agenda in K-12 education?* The overall methodology of the study is based on a qualitative approach. An extensive review of the literature using a wide range of academic and grey literature has been complemented with a survey of policy documents with Ministries of Education in Europe in order to identify countries that are including CT in their curricula, as well as to spot relevant documents (e.g. curricula, guidelines) for further analysis. A review matrix outlining main research studies, findings and implications is used to structure the in-depth and comparative analysis of the sources and to identify key issues. Informed insights on crucial factors for a successful development of CT skills in K-12 is being collected via semi-structured interviews with policy makers, researchers and practitioners so as to validate and complement the literature review.

Initial Findings: General Trends on Computational Thinking in K-12 Education

Two main trends emerged on the rationale for including CT in K-12 education: (1) fostering CT to boost economic growth, fill vacancies in ICT, and prepare for future jobs; (2) develop CT skills in children and young people to enable them to think in a different way, to express themselves through a variety of media, to solve problems and analyze everyday issues with a different perspective. Many articles mention both trends when explaining rationales for teaching CT: the general benefits of CT as a thinking skill and the need to develop new skills for the employment market. The emphasis on one or the other depends on the stakeholder's point of view: some criticize a too narrow focus on employability aspects (e.g. CoderDojo, Computing at School), others put a strong emphasis on the job opportunities that coding can provide (e.g. the recent initiative of President Obama "Computer Science for all").

CT Definitions and Characterizations

Regarding the definition of CT and its main characterizations, the majority of the literature refers to J. Wing's definitions (2006, 2011). One voice against this widely adopted definition comes from Jones (2011) who argues that it remains abstract and fails to differentiate CT from other forms of thinking. In his blog article, Guzdial (April 2012) suggests to include risks and cyber-security in CT definition. Aaron Sloman (London Knowledge Lab, 2012) even defines CT as "one of the required approaches to understanding the universe, which was not available to the deepest thinkers in most of the history of science, mathematics, engineering and philosophy". Some papers also refer to the creative aspects of CT: "computational thinking is, indeed, a key to developing the capacity to discover, create and innovate" (Allan et al., 2010). Many authors refer to CT in relation to problem solving, or to modeling and solving real world problems (e.g. Lee, 2016). Regarding what differentiates CT from other thinking skills, some authors argue that a deeper analysis is needed on the typology of thoughts that are key to CT. Grover & Pea (2013) extract from the literature several CT characterizations:

- 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
- Debugging and systematic error detection.

CT Relation to Programming and Computer Science

CT is recognized not to necessarily need programming of computers, but rather to be an approach to problem solving that uses strategies such as algorithms, abstraction and debugging (Dede et al., 2013). However, programming is creative and engaging; it illustrates in concrete terms otherwise-abstract concepts (Peyton-Jones, 2015). There is, therefore, a mutual influence between CT and coding/programming:

- To some extent, the acquisition of computational thinking skills has been a side effect of learning to program: while learning programming languages, computer science students also to pick up higher level skills which are applicable outside of computer science (Howland et al., 2009).
- Design-based learning activities – in particular, programming interactive media – support the development of CT in young people (Brennan & Resnick, 2012).

Due to this mutual influence, the borders between CT and coding/programming are seen as not really clear-cut: the concepts of CT and the practice of programming are difficult to separate in the literature because many CT studies use programming as their reference context. This can be confusing and often leads to the impression that CT is the same as programming or, at the very least, that CT requires the use of programming to be developed (Voogt et al., 2015).

CT Assessment in K-12 Education

A number of methodologies and tools for assessing the acquisition of CT skills in compulsory education emerged from the literature. Along more traditional multiple choice tests and open ended questions, a project-based approach emerges as essential element of assessments systems. Two approaches to project-based CT assessment are present in the literature: (automated) analysis of projects portfolio and artifact-based interviews. Brennan & Resnick (2012), e.g., describe three approaches to assessing the development of CT:

- analyzing the portfolio of projects uploaded by a particular community member and generating a visual representation of the blocks used (or not used) in every project;
- artifact-based interviews, based on two projects selected by the interviewee;
- design scenarios; given three sets of projects with increasing complexity the interviewee is asked to select one and "(1) explain what the selected project does, (2) describe how it could be extended, (3) fix a bug, and (4) remix the project by adding a feature".

Skill transfer to other contexts is another form of assessment being investigated, as for example the capability of transferring computational understanding built in a visual programming environment to a textual one (Grover, Pea & Cooper, 2015).

Implementation of CT Initiatives

Most of the papers analyzed on CT implementation represent serious and effective attempts to develop practical learning activities apt to foster some of the skills and competences that characterize CT. Some authors (e.g., Taub et al.,2014) give emphasis to the development of CT characterizing skills, such as abstraction, yet referrring to CS instead of to CT. As concerns the kind of activity proposed to develop CT, most author refer to programming tasks, yet often carried out with learning environments/tools (e.g., Scratch) that do not require coding in a formal language. The objective of the proposed programming activities is mostly the development of games, that are considered excellent situations in which abstraction (of moves and actions) can be understood and meaningfully used. There are some notable exceptions, however. For instance:

- Weintrop et al. (2015) propose the development of CT within STEM course;
- Yevseyeva and Towhidnejad (2012) aim to disclose the advantages of CT to non-computing students, by organizing collaborative activities on world issues, thanks to the fact that CT helps decompose

problems into manageable steps, employ abstraction to deal with complexity, recognize patterns and create scalable algorithms to solve real problems;

- Taub et al. (2014) see abstraction as a common element between CT and Physic, and plan a physics problem solving activity apt to highlight different levels of abstraction in physics.

Some attention is also given to introduce CT to girls (e.g., Weintrop et al., 2015), by proposing activities closer to girls' tastes and hence more motivating to them. As concerns the school levels considered, most implementations concern high school, but also middle school (e.g., Cher 2015) and primary school (e.g., Quinlan, 2015) are considered.

Conclusion and Future Work

This brief paper has presented initial findings emerging from literature analysis in the context of the CompuThink study. Although the debate on definitional issues remains open and ongoing, much of the recent work in the field increasingly focuses on CT specific characterizations and related skills. We would argue that this point is a cornerstone for the successful and meaningful implementation of CT in education, especially in curricula: clear definitions and conceptualizations lead to well-structured and effective curricula and learning objectives. As more and more CT implementations emerge (both in formal and informal education settings), major attention is given to the disciplinary contexts in which CT should be addressed in K-12 (i.e., within a specific subject such as Computer Science, integrated in STEM, or as a cross-curriculum topic). Large gaps, however, still exist that call out for extensive research on (among other issues) how CT skills develop in K-12 students and what pedagogical approaches can facilitate the effective introduction of CT concepts throughout K-12 education. CT assessments practices are also underinvestigated, and, in particular, what kind of assessment can elicit students' problem solving and CT skills in authentic contexts. Grassroots initiatives that are emerging are currently not focusing their action on assessing CT, possibly as a consequence of their inception status in several cases and for their extra curricular application in others. To help fill these gaps, in the context of CompuThink, curricula and policy documents, together with grassroots initiatives will be analyzed more in depth, so as to highlight the red thread that goes from conceptual constructions (i.e. research and studies, as exemplified in the review of the academic and grey literature) to practices (i.e. grassroots initiatives), passing through policy-making (as emerging from the analysis of curricula and policy-documents surveyed with EU Ministries of Education). Informed insights collected from stakeholders and practitioners involved in CT initiatives will also contribute to building an overall picture of this field, shedding lights on the crucial factors for a successful development of CT skills in K-12 students.

References

Allan, W., Coulter, B., Denner, J., Erickson, J., Lee, I., Malyn-Smith, J., & Martin, F. (2010). Computational Thinking for Youth. *ITEST Small Working Group on Computational Thinking*.

Brennan, K., & Resnick, M. (2012). New frameworks for studying and assessing the development of computational thinking. In *2012 annual meeting of the American Educational Research Association (AERA)*. Vancouver, 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. *Proceeding of the 44th ACM Technical Symposium on Computer Science Education* New York, USA: ACM. 269–274.

Charlton, P., & Luckin, R. (2012). Time to re-load? Computational Thinking and Computer Science in Schools. *What research says*. Briefing 2, 27 April. London Knowledge Lab, University of London.

Cher, B. (2015). Singapore: From Smart Nation to Code Nation. *Digital News Asia*.

Daily, S., Leonard, A., Jörg, S., Babu, S., Gundersen, K., & Parmar, D. (2014). Embodying Computational Thinking: Initial Design of an Emerging Technological Learning Tool. *Technology, Knowledge and Learning*, 20(1), 79–84.

Dede, C., Mishra, P., & Voogt, J. (2013). Advancing computational thinking in 21st century learning. *EDUsummIT 2013,* International summit on ICT in education, Washington, DC.

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

Grover, S., & Pea, R. (2013). Computational Thinking in K–12 A Review of the State of the Field. *Educational Researcher*, 42(1), 38–43.

Grover, S., Pea, R., & Cooper, S. (2015). Designing for deeper learning in a blended computer science course for middle school students. *Computer Science Education*, *25*(2), 199–237.

Guzdial, M. (April, 2012.). *Computing Education Blog*.

Howland, K., Good, J., & Nicholson, K. (2009). Language-based support for computational thinking. *IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC 2009)*. IEEE. 147–150.

Jones, E. (2011). *The Trouble with Computational Thinking*. Retrieved from http://www.csta.acm.org/Curriculum/sub/CurrFiles/JonesCTOnePager.pdf

Knuth, D. E. (1974). Computer science and its relation to mathematics. *American Mathematical Monthly*, 81(4), 323–343.

Lee, I. (2016). Reclaiming the Roots of CT. *CSTA Voice - Special Issue on Computational Thinking*, 12(1), 3-5.

National Research Council. (2010). *Report of a Workshop on The Scope and Nature of Computational Thinking*. Washington, D.C.: National Academies Press.

National Research Council, (2012). *A Framework for K-12 Science Education: Practices, Crosscutting Concepts, and Core Ideas*. (H. Quinn, H. Schweingruber, & T. Keller, Eds.). Washington, D.C.: National Academies Press.

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

Peyton Jones, S. (2015). Decoding the new computing programmes of study. *Computing at School*.

Quinlan, O. (2015). *Young Digital Makers - Surveying attitudes and opportunities for digital creativity across the UK*. London, UK: NESTA.

Repenning, A., Webb, D. C., Koh, K. H., Nickerson, H., Miller, S. B., Brand, C., … Repenning, N. (2015). Scalable Game Design: A Strategy to Bring Systemic Computer Science Education to Schools Through Game Design and Simulation Creation. *ACM Transactions on Computing in Education*, 15(2), 1– 31

Taub, R., Armoni, M., & Ben-Ari, M. (Moti). (2014). Abstraction as a bridging concept between computer science and physics. *Proceedings of the 9th Workshop in Primary and Secondary Computing Education*. ACM Press. 16–19.

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.

Weintrop, D., Orton, K., Horn, M., Beheshti, E., Trouille, L., Jona, K., & Wilensky, U. (2015). Computational Thinking in the Science Classroom: Preliminary Findings from a Blended Curriculum. *Annual meeting of the National Association for Research in Science Teaching* (*NARST*).

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

Wing, J. M. (2011). Research Notebook: Computational Thinking - What and Why? *The Link*. Pittsburg: Carnegie Mellon.

Wing, J. M. (2014). Computational Thinking Benefits Society. *Social Issues in Computing Blog*. New York: Academic Press.

Yevseyeva, K., & Towhidnejad, M. (2012). Work in progress: Teaching computational thinking in middle and high school. *2012 Frontiers in Education Conference (FIE)*. 1–2.

Acknowledgements

CompuThink is funded and designed by the Joint Research Centre (JRC), Institute for Prospective Technological Studies of the European Commission (EC) (Contract No. 199551-2015A08IT). The JRC is an EC-run service which involves scientists to provide independent scientific advice and support to EU policy. The study, which runs from December 2015 to September 2016, is carried out by the Institute for Educational Technology of the Italian National Research Council and European Schoolnet, which represents a network of Ministries of Education in Europe. The data presented, the statements made and the views expressed are purely those of the authors and should not be regarded as the official position of the European Commission.