Proceedings of the Informing Science + Information Technology Education Conference

An Official Publication of the Informing Science Institute
InformingScience.org

InformingScience.org/Publications

June 30 – July 4, 2019, Jerusalem, Israel

VIEWS AND TENDENCIES OF INTRODUCING COMPUTATIONAL THINKING IN AUSTRALIAN [RESEARCH IN PROGRESS]

Dorian Stoilescu — Western Sydney University, Sydney, Australia — D.Stoilescu@westernsydney.edu.au

ABSTRACT

Aim/Purpose	This paper discusses theoretical and curricular aspects of computational thinking in curriculum and challenges noticed on introducing recent ICT perspectives in Australian Schools.
Background	It presents the way computational thinking is defined and understood in curriculum documents and a set of relatively new implementations that were designed nationally and in the New South Wales state.
Methodology	This paper uses qualitative research methods such as content analysis and text analysis methods.
Contribution	This research analyzes some recent trends in introducing computational thinking and explore the was these reforms are described in the official documents.
Findings	It was noticed that although the importance of computational thinking was highly emphasized, the documents cannot describe a consistent implementation of this set of educational policies, as at this time implementing computational thinking largely underperforming.
Recommendations for Practitioners	It is recommended a more systemic way of designing policies and curriculum content for the integration of computational thinking in Australian schools.
Future Research	Future research needs to explore reasons for delaying these reforms of introducing computational thinking.
Keywords	computational thinking, computer science education, ICT education, Australian curriculum reforms

Accepted by Executive Review by Editor Eli Cohen | Received: April 16, 2019 | Revised: April 22, May 6, 2019 | Accepted: May 7.
Cite as: Stoilescu, D. (2019). Views and tendencies of introducing computational thinking in Australian schools. *Proceedings of the Informing Science and Information Technology Education Conference, Jerusalem, Israel,* pp. 239-245. Santa Rosa, CA: Informing Science Institute. https://doi.org/10.28945/4348

INTRODUCTION

Computational thinking (CT) is a relatively new educational perspective of using computer science in curricula. Janette Wing introduced this concept first time in a brief conference paper in 2006, as a basic educational goal that all 21st educators should aspire. Computational thinking was defined as a core educational reference, similar to literacy and numeracy. More exactly, she introduces this term as an educational approach that "builds on the power and limits of computing processes, whether they are executed by a human or by machine" (Wing, 2006, p.33). As such, is it a way to model various projects and problems from a broad large of areas based on facilities that computer support offers:

Computational methods and models give us the courage to solve problems and design systems that no one of us would be capable of tackling alone. Computational thinking confronts the riddle of machine intelligence: What can humans do better than computers? and What can computers do better than humans? Most fundamentally it addresses the question: What is computable? (p. 33)

As such, computational thinking has had an important impact on educational curriculum and policies, as being a recent perspective introduced in the national curricula of numerous countries that need to be clearly understood and implemented (Aho, 2012; Hu, 2011). Its major question that underlines the expertize of computational thinking remains the previous inquiry initially formulated by Wing: What can be done by computers and what still cannot (Hu, 2011, Wing; 2006, 2008)? What happens in various areas of curricula, when we move from the areas that use computers and abstract algorithms to various software packages required in the school curricula? Do we need to change our school curriculum? If, yes, what needs to be updated? Do we only need change pedagogical approaches in all disciplines? If yes, how can we remodel our school curriculum content in order to effectively interact with computers? This is why computational thinking was introduced as a broader way of understanding interactions between computer and learning activities. Computational thinking involves understanding human interactions, patterns of problem-solving, designing systems, and implementing decisions (Grover & Pea, 2013`).

As a developed country, Australia has been attempting to introduce new policies and implement them across all areas of primary and secondary education. With this in mind, in this paper, we attempt to explore ways in which computational thinking is defined and implemented in Australian Curriculum.

The main research questions discussed here are:

1. How is computational thinking described and implemented in Australian curriculum?
2. What are the challenges in implementing computational thinking in Australia?

First, this research will study mostly the national curriculum and the way computational thinking was understood. As Australia has the education system designed and managed at state level, the discussions will focus mostly on the New South Wales (NSW) state curriculum, as being the most populous state in Australia and the other states, although they are not discussed in this current paper, have numerous similarities with NSW implementations.

BACKGROUND

Similar to computational thinking, there are already introduced in ICT education research terms such as digital literacy, coding literacy, computational modelling, IT literacy, IT fluency (García-Peñalvo, J., Reimann, Tuul, Rees, & Jormanainen, 2016). While an exhaustive discussion of terms used in the research literature is not the purpose of this paper, we will briefly discuss some differences between previous terms connected to computational thinking. Computational thinking is often seen as becoming familiar with various digital technologies. However, computational thinking is more than just learning how to access information through various digital devices and software packages, which is the definition of digital literacy. As well, it is different from digital fluency which explores the skil-

fulness of computational thinking is logically related to programming as people see it as a way to connect with learning programming. However, it is not narrowly focused on creating a software that is solving that problem.

Computational thinking attempts to change the way students learn. For instance, when solving a problem, students using computational thinking paradigms might ask: "What is the most practical way to solve a problem and how difficult is?" or "Is any software able to solve that problem? If not, is the computer helping to ease the solving of the problem? How?" The learning paths are changed in other ways as well: "Can we approximate the main stages of solving a problem with an algorithmic path?" In other words, computational thinking attempts to rephrase the initial problem into something less difficult, through different reductionist paths by using reducing complexity, creating different scenarios, using random data, and simulation.

Computational thinking is using various strategies to achieve its impact on learning. For instance, by using abstraction and decomposition, some characteristics are generalized or emphasized. As such, the content becomes less complex and easier to get digitally processed. Selecting specific criteria, the problems are reduced to some general type or class of problems and algorithmically approached.

Computational thinking was recently connected to teaching broad skills such as literacy and numeracy. They are similar in the way that students need to master both in order to succeed in today's society (Setle et al. 2012). Computational think as such needs to be delivered in a more broad path of understanding so that technological tools and algorithms need to be deployed in STEM disciplines, social studies, languages, and arts. As well, concepts, tools and the language used in manipulating these are requested to be more flexible, so that when learners decompose the problems, they need to be easy to work with by various types of learners trying to solve complex types of problems.

Computational thinking was recently introduced in many countries such as US, China, Australia, Israel, and several European countries such as Netherland, Ireland, UK, and Finland. While computational thinking implementations in national curricula are still in early stages, some trends emerged. For instance, the researchers and educators attempt to make computational thinking distinct from programming. For many researchers (Garcia Pelvano et al. 2016; Voogt, Fisser, Good, Mishra & Yadav, 2015) one of the major difficulties remains using computational thinking in other areas different from the traditional computer science discipline. An interesting approach is introducing computational thinking in other areas different from STEM such as English, Latin, history, graphic arts, and ethics (Barr & Stephenson, 2011; Seoane-Pardo, 2016; Settle et al, 2012). Another major debate is which type of coding should be chosen. More exactly, in teaching computing, there are two different paths. First of them is teaching traditional languages such as Python, C/C++/C#, Java, Perl, Visual Basic, HTML, SQL. A major difficulty encountered by people promoting this oath is that these languages require a considerable level of expertise for teachers willing to try for their classrooms. As such, teachers would need more formal classes and training in programming courses, things difficult to support in the developing or developed countries. The second major path was the use of non-traditional programming languages, and visual programming platforms such as Logo, Scratch, Alice, AgentCubes, Flowgorithm, GameSalad, Kodu Games Lab, LARP, Raptor, Toon Talk, Visual Logic. Etc. Some non-traditional programming languages such as Logo and Scratch consider learning programming these languages as a way of playing, designing, and interacting with different objects and actors. These programming languages put playing and user interactions in the center of learning programming. As such, these are not related to a rigid writing of a specific syntax. Recently, new trends in learning programming that emphasize interactions and simulations of robotics, actor-model programming languages, programming microcontrollers, and programming Internet of Things technologies have been emerging. Some products already used in schools and universities are Arduino, Circuit Wizard, GENIE, PICAXE, Raspberry PI, Micromite, Intellecta, Bee-Bots, Lego Mindstorms, WeDo (Lego-based) and Intel Edison.

METHOD

This study uses quality research methods, mainly document analysis (Bowen, 2009). "Documents that may be used for systematic evaluation as part of a study take a variety of forms. Researchers typically review prior literature as part of their studies and incorporate that information in their reports. However, where a list of analysed documents is provided, it often does not include previous studies. Surely, previous studies are a source of data, requiring that the researcher rely on the description and interpretation of data rather than having the raw data as a basis for analysis. The analytic procedure entails finding, selecting, appraising (making sense of), and synthesising data contained in documents" (p 27).

In the following we will summarise some main tendencies noticed in the official websites of the Australian Curriculum and The NSW Education Standards Authority (NESA). We attempted to include some previous studies such as previous documents about computational thinking used in Australia and other countries such as US and UK. By using document analysis, content is structured into major themes, categories, and case examples specifically through content analysis (Labuschagne, 2003).

ATTEMPTS OF INTRODUCING COMPUTATIONAL THINKING IN AUSTRALIAN SCHOOLS

In New South Wales, the current state-level educational organization that establishes and monitors teaching preparation and school standard is called the NSW Education Standards Authority (NESA). They establish the criteria for designing and updating the state curriculum, for assessments and exams, teaching certifications and professional development, and assessments. Australian curriculum documents document well that information processing is not the same as computing (Piccinni & Scarantini, 2010). However, it is hard to delimitate them and create separate distinct curricular disciplines. For instance, Australian curriculum attempts a clear delimitation between these two curriculum areas as in the last two years. In New South Wales, for example, the state curriculum has two different computing disciplines, one related to information processing (Information Processes and Technology or IPT) and the other related with computation (Software Design and Development or SDD). While Information Processes and Technology in taught in many schools and remains widely spread in various areas of curricula and in informal activities, Software Design and Development is at the beginning stage and it is not well integrated with other curriculum areas. The NSW primary education is from kindergarten to year 6 and its curriculum is structured in Learning stages from Early Stage 1 for kindergarten and three stages for years 1 to 6. Technology is part of the Key Learning Area (KLAs). Secondary education in from year 7 to year 12 and has Stages 4,5 and 6. From kindergarten to year 10, national curriculum has developed Digital Technologies. In primary education, ICT technology is part of Science and Technology curriculum.

An important document about computational thinking appeared recently (New South Wales Education Standards Authority [NESA], 2017) about introducing computational thinking in NSW curriculum. Computational technologies are part of the Digital Technologies curriculum for the F to year 10 schooling. Computational thinking is defined as "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" (NESA, 2017). Digital technologies strands are structured on two main related strands:

- knowledge and understanding – Describes information and digital systems (hardware, network, and software)
- processes and production skills – using digital systems to create ideas and information, and to define, design and implement digital solutions, and evaluate these solutions and existing information systems against specified criteria.

Informally, computational thinking is not described as a programming activity. Rather, it is described as a mental activity in modelling and formulating a problem, that finally relates to a computational solution. The solution can be carried out by a human or machine. This latter point is important as it shows that humans can compute and learn computational thinking without having a computer. Also, it emphasizes that computational thinking is not just about problem-solving, but also about problem formulation and modeling. As well, the document emphasizes the importance of critical thinking in modelling and establishing a hierarchy of abstractions.

An important aspect of the document is that it encourages programming without pressuring the students to learn a specific programming language. There are many voices encouraging promoting more coding in Australian curriculum. For instance, the Digital Careers consider that computer programming is a requirement for successful future careers. The present guide in computational thinking draws not only in technology areas but almost in every learning area where computational thinking can be applied. As it easy to understand, usually, these multidisciplinary areas of curriculum do not require the use of coding, but they do aim to develop algorithmic and computational thinking skills to better enable students and teachers to reach a coding goal.

Challenges of Introducing Computational Thinking in Australia

Nowadays, computational thinking is still stirring important debates. One of them is whether computational thinking is producing new ideas. And if yes, how do we evaluate the novelty and the importance of these new ideas. Another aspect is that the terms and actions of the Digital Technologies outcomes are often too prescriptive and too narrowly related to programming. As well, the terms in use are often very abstract and difficult to follow. Often, these terms and ideas look from university textbooks. As such, it is important to use these terms in more non-sophisticated ways, as the curriculum is for a large number of teenagers.

An important discussion refers to the ethical aspect of teaching and learning produced be computational learning. Several researchers emphasize that the abstract tendency of processing knowledge in order to make it "computable" has as an impact on the disembodiment and embodiment of the content involved in computational thinking. As well, the areas of use computational thinking are extended. While computational thinking was traditionally linked with STEM disciplines, now it is more than that. Computational thinking touches almost all learning areas, not only the STEM disciplines. As well, there are important language, emotional, social, cultural, and ethical aspects that computational thinking needs to keep into consideration when educators and students attempt to use it in broad disciplines. Another important aspect related to computational thinking links to the critical thinking. Is computational thinking overlooking critical thinking aspects? As computational thinking simplifies the discourse and the strategies requested to solve the problem, critical thinking aspects come as a very delicate topic as computational thinking might overlook some of these social issues. This is why one of the major reasons to improve is by considering critical thinking strategies for using computational thinking, as the soft aspect of problems always needs to be considered first before simplifying and modelling with digital tools.

While computational thinking is pervasive, we need to explain what computational thinking is not. As we mentioned already, computational thinking is not coding. Yet, many people still automatically associate them and at times it is requested unreasonable level of knowledge in coding. In addition, there are requirements to use more computational thinking approaches across more areas of the curriculum. Another area to improve is referenced to technological design as computational design often has the purpose of obtaining a technological artifact. As well, references to thinking skills need to be emphasized. More emphasis on Learning based Project pedagogy needs to be pursued. More emphasis on developing problem-solving skills and modelling. Alternative ideas and solutions for digital approaches need to be designed.

Another concept developed is CS + X which means "computing science plus whatever it is that you are passionate about or engaged with" (Australian National Curriculum 1, 2017). IT systems are becoming more commonplace and all-pervasive, and the development of the Internet of Things and machine-to-machine communication standards will further our reliance on them.

Critical pedagogy is important in discussing the output of computational thinking. It was noticed that sensitive teaching is an important request so that the problems of computational thinking are not becoming irrelevant or unethical. As well, issues of safe use of technology need to be widely discussed. In addition, ethics and social equity are broadly targeted as the use of technology needs to be accessible for people with various backgrounds. Another aspect was the teaching and learning of computational thinking for various minorities such as aboriginals or people with disabilities.

Introduction of literacy around coding and ICT remains a difficult task as there are few educators involved in computational thinking that connect them with broad areas of curricula. As a result, due to the reduced number of educators involved in computational thinking makes it implemented in few disciplines. concepts and language used in technologies. As such, the amount of work involved in computational thinking is very different and still in an incipient stage in Australian schools.

CONCLUSION AND DISCUSSIONS

Implementing computational thinking is an exciting opportunity for every country. As such, although Australia is considered an advanced knowledge economy, computational thinking remains still known by few educators. As such, implementations of computation thinking are still at the incipient stage and are considered relatively a challenging task. While it was noticed that the main dimensions for computational thinking, such as a flexible way to encourage modelling and interactions between human and computer devices, was clearly understood, in order to be well integrated into the national and state curriculum, more efforts are required to disseminate the recent policies and interpretations on computational thinking and persistent efforts to implement them across all curricula.

REFERENCES

Aho, A. V. (2012). Computation and computational thinking. *The Computer Journal, 55*(7), 832-835.

Australian National Curriculum 1 (2017) http://educationstandards.nsw.edu.au/wps/portal/nesa/k-10/learning-areas/technologies/coding-across-the-curriculum

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.

Bowen, G. A. (2009). Document analysis as a qualitative research method. *Qualitative Research Journal, 9*(2), 27-40.

García-Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A., & Jormanainen, I. (2016). An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers. Belgium: TACCLE3 Consortium. doi:10.5281/zenodo.165123..

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

Hu, C. (2011, June). Computational thinking: what it might mean and what we might do about it. In Proceedings of the 16th *Annual Joint Conference on Innovation and Technology in Computer Science Education* (pp. 223-227). ACM.

Labuschagne, A. (2003). Qualitative research: Airy fairy or fundamental? *The Qualitative Report, 8*(1), Article 7. Retrieved 17 April 2019, from https://nsuworks.nova.edu/tqr/vol8/iss1/7/

Piccinini, G., & Scarantino, A. (2010). Computation vs. information processing: why their difference matters to cognitive science. *Studies in History and Philosophy of Science*, 41(3), 237-246. 0-264.

New South Wales Education Standards Authority [NESA] (2017). Digital Technologies and ICT Resources http://educationstandards.nsw.edu.au/wps/portal/nesa/k-10/learning-areas/technologies/coding-across-the-curriculum

Seoane-Pardo, A. M. (2016, November). Computational thinking beyond STEM: an introduction to moral machines and programming decision making in ethics classroom. In Proceedings of the *Fourth International Conference on Technological Ecosystems for Enhancing Multiculturality* (pp. 37-44). ACM.

Setle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., & Wildeman, B. (2012, July). Infusing computational thinking into the middle-and high-school curriculum. In Proceedings of the 17th *ACM annual conference on Innovation and technology in computer science education* (pp. 22-27). 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. M. (2006). Computational thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. M. (2008). Computational thinking and thinking about computing. In *Philosophical Transactions of the Royal Society of London A: mathematical, physical and engineering sciences, 366*(1881), 3717-3725.Roussev, B. (2003b). Teaching introduction to programming as part of the IS component of the business curriculum.

BIOGRAPHY

Dorian Stoilescu is lecturer in ICT and mathematics education at the School of Education, Western Sydney University. Some of his research topics are: ICT curriculum and policies, integrating ICT in mathematics curriculum and policies, integrating ICT in mathematics education, equity in ICT and mathematics education.