Exploring the Introduction of Computational Thinking in STEM Education in Australian Schools

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

Abstract

This paper discusses theoretical and curricular aspects of computational thinking in curriculum and detects recent perspectives and challenges noticed in introducing computational thinking in STEM in Australian Schools. 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 state of New South Wales. This paper uses qualitative research methods such as content analysis and text analysis. The research analyses some recent trends in introducing computational thinking and explores these reforms that are described in the official documents. 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 is largely underperforming. It is recommended a more systemic way of designing policies and curriculum content for the integration of computational thinking in Australian schools is needed. Future research needs to explore reasons for delaying these reforms of introducing computational thinking.

Keywords: computational thinking, STEM education, Australian curriculum reforms.

Introduction

Computational thinking (CT) is a relatively new educational perspective for using computer science in the curricula. This concept was introduced initially by Janette Wing in a brief conference paper in 2006, as a basic educational goal to which all 21st century educators should aspire. In that paper, computational thinking was defined as a core educational reference, similar to literacy and numeracy. More exactly, she introduced this term as an educational approach that “builds on the power and limits of computing processes, and on the opportunities and the potential that is capable of offering, whether they are executed by a human or by machine” (Wing, 2006, p.33). As such, it is a way to model various projects and problems from a broad area 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 and needs to be clearly understood and implemented (Aho, 2012; Hu, 2011). The major question that underlines the expertise 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 to 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). Especially in STEM (Science Technology Engineering Mathematics) education, the impact of computation thinking is major as it includes and uses some parts of the content in these disciplines (Jona et al., 2014).

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, it is attempted 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 the 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's education system is designed and managed at state level, the discussions will focus mostly on the New South Wales state curriculum, the most populous state in Australia, and the other states although not discussed in this current paper, have numerous similarities with the New South Wales implementations.

Literature Review

Similar to computational thinking, there are already components existing in ICT education with research terms such as digital literacy, coding literacy, computational modelling, IT literacy, IT fluency (García-Peñalvo et al., 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 skilfulness of computational thinking related to programming and people see it as a way to connect with learning programming. However, it is not narrowly focused on creating 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 this problem and how difficult is it?" or "Is any software available to solve this problem? If not, can the computer help to ease the solving of this problem? How?" The learning paths are changed in other ways as well: "Can we approximate the main stages of solving this 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 reducing complexity, creating different scenarios, using random data, and simulation.

Computational thinking uses various strategies to achieve its impact upon 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. By selecting specific criteria, the problems are reduced to some general types or classes of problems and are then algorithmically approached.

Computational thinking was recently connected to teaching broader 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 broader path of understanding so that technological tools and algorithms are deployed in not only STEM disciplines, but also social studies, languages, and the arts. As well, the concepts, the tools, and the language used in manipulating these are required to be more flexible, so that when learners decompose the problems, they will be easy to work with by various types of learners when trying to solve complex types of problems.

Computational thinking was recently introduced in many countries such as USA, China, Australia, Israel, and several European countries such as Netherland, Ireland, UK, and Finland. While computational thinking implementations into national curricula are still in the early stages, some trends have emerged. For instance, the researchers and educators attempt to separate computational thinking as distinct from programming. For many researchers (García-Peñalvoet al. 2016; Voogt, Fisser, Good, Mishra & Yadav, 2015) this is a major difficulty when using computational thinking in other areas different from the traditional computer science discipline. However, there are attempts to introduce computational thinking in other areas different from STEM such as English, Latin, history, graphic arts, ethics (Barr and Stephenson, 2011; Seoane-Pardo, 2016; Setle et al., 2012).

Another major debate is which type of coding language should be chosen. More exactly, in teaching computing, there are two different paths. The 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 path is that these languages require a considerable level of expertise for teachers willing to try them in 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 is 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 are considered a way of playing, designing, and interacting with different objects and actors. These programming languages put playing and user interactions in the centre of learning programming. As such, these are not related to a rigid writing of a specific syntax.

Recently, new trends have been emerging in learning programming through emphasizing interactions and simulations of robotics, actor-model programming languages, programming microcontrollers, and programming Internet of Things technologies. 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. These tools are receiving increasing attention as they are hands-on and require less computer programming skills, if any.

Method

This research paper explores tendencies of introducing computational thinking revealed in the curriculum documents. It uses qualitative 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 section, the main tendencies will be summarised as they are displayed in the official websites of the Australian Curriculum and The New South Wales Education Standards Authority. Some attempt was made to include some previous documents on computational thinking used in Australia and other countries such as the USA and the UK. By using document analysis, the content is structured into major themes, categories, and case examples specifically through content analysis (Labuschagne, 2003).

Findings

Attempts of Introducing Computational Thinking in Australian Schools

Computational thinking was introduced in all Australian states. In New South Wales, the current state-level educational organization that establishes and monitors teaching preparation and school standard is called the New South Wales Education Standards Authority (NESA). They establish the criteria for designing and updating the state curriculum, for assessments and examinations, teaching certifications and professional development, and assessments. Australian curriculum documents clearly state that information processing is not the same as computing (Piccinini & Scarantino, 2010). However, it is hard to segregate them and create separate distinct curricular disciplines.

For instance, Australian curriculum attempts a clear demarcation 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 to computation (Software Design and Development or SDD). While Information Processes and Technology is taught in many schools and remains widely spread in various areas of the curricula and in informal activities, Software Design and Development is at the beginning stage and is not well integrated with other curriculum areas. The New South Wales primary education 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 (KLA). Secondary education in from year 7 to year 12 and has Stages 4, 5 and 6. From kindergarten to year 10, the national curriculum has included Digital Technologies. In primary education, ICT technology is part of the Science and Technology curriculum.

An important document appeared recently (NESA, 2017) about introducing computational thinking into the New South Wales curriculum. Computational technologies are part of the Digital Technologies curriculum for the years K 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 along two main related strands:

1. knowledge and understanding, that describes information and digital systems (hardware, network, and software); and
2. processes and production skills, that uses 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 modelling. 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 and 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 upon technology areas but also in almost every learning area where computational thinking can be applied. In contrast, 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

We noticed that in the documents it is mentioned that nowadays, computational thinking is still stirring important debates. One of them is whether the terms and actions of computational thinking outcomes are often too prescriptive and too narrowly related to programming. In addition, the terms in use are often very abstract and difficult to follow. Often, these terms and ideas appeared to be taken from university textbooks. Alternatively, it is important to use these terms in more non-sophisticated ways, as the curriculum is for a large number of teenagers.

Another aspect is whether computational thinking is producing new ideas? And if yes, how do we evaluate the novelty and the importance of these new ideas? It was noticed that computational thinking is often related to robots and sensors. Therefore, most of the suggestions in computational thinking activities centres upon getting robots to move in different directions. While this is producing an initial excitement for students, the content of the curriculum needs to have more depth in order to have an educative value. Although it is important to use the potential of robotics, computational thinking is offering much more for a larger variety of fields, these robots are often expensive, thus there are schools that might not be able to afford purchasing them.

Another concept in the process of being developed is CS + X which means computing science plus whatever it is that you are passionate about or engaged with (National Curriculum 1, 2017). As IT systems are becoming more commonplace and all-pervasive, and with the development of the Internet of Things and machine-to-machine, communication standards will result in our greater reliance upon them. Thus, the areas of use for computational thinking are extended. While computational thinking was traditionally linked with STEM disciplines, now it touches almost all learning areas, not only the STEM disciplines. There are also important aspects involving language, emotions, social issues, cultural sensitivities, and ethical considerations that computational thinking needs to take into consideration when educators and students attempt to use it in the broader disciplines.

Critical pedagogy is important in discussing the output of computational thinking. It was noted that thoughtful teaching is important so that the problems of computational thinking do not become irrelevant or unethical. How are computational thinking and critical thinking related? Is computational thinking overlooking critical thinking aspects? As computational thinking simplifies the discourse and the strategies used to solve the problem, critical thinking aspects come as a very delicate topic where computational thinking might overlook some of the social issues. This is why one of the major reasons for considering critical thinking strategies when using computational thinking is the softer aspects of problems that always need to be considered first before simplifying and modelling with digital tools. It involves issues around the safe use of technology that need to be widely discussed. Other issues that are broadly targeted includes ethics and social equity where the use of technology needs to be accessible for people with various backgrounds as well having the teaching and learning of computational thinking provided for various minorities such as aboriginals or people with disabilities. 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. Critical pedagogy also needs to place greater emphasis: on thinking skills; on learning based project pedagogy; on developing problem-solving skills and modelling. It involves encouraging and designing alternative ideas and solutions for digital approaches.

While computational thinking is pervasive, we need to explain what computational thinking is not. As mentioned already, computational thinking is not coding. It has some content from algorithms, yet many people still automatically associate the two as one and demand unreasonable levels of knowledge in coding. Another area is incorrectly identifying technological design as computational design. Technological design often has the purpose of obtaining a technological artefact.

The introduction of literacy around coding and ICT remains a difficult task as there are few educators involved in computational thinking that connect them with the broad areas of curricula. As a result, due to the shortage of educators involved in computational thinking the implementation is restricted to a few disciplines. As such, the amount of work involved in computational thinking is very different and still in an early stage in Australian schools.

Conclusion and Discussions

Implementing computational thinking is an exciting opportunity for every country (Lu & Fletcher, 2009). Although Australia is considered an advanced knowledge economy, there are numerous barriers in encouraging enough students into STEM education and, in particular, computational thinking remains a major challenge for educators as well as a major opportunity (Swaid, 2015). Therefore, implementations of computation thinking are still at an emerging stage and are considered a relatively 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, and 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 as well as 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.

Caspersen, M. E., & Nowack, P. (2013, January). Computational thinking and practice: A generic approach to computing in Danish high schools. In *Proceedings of the Fifteenth Australasian Computing Education Conference-Volume 136* (pp. 137-143). Australian Computer Society,

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.

Jona, K., Wilensky, U., Trouille, L., Horn, M. S., Orton, K., Weintrop, D., & Beheshti, E. (2014). Embedding computational thinking in science, technology, engineering, and math (CT-STEM). Proceedings in *Future directions in computer science education summit meeting*, Orlando, FL.

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/

Lu, J. J., & Fletcher, G. H. (2009). *Thinking about computational thinking.* In ACM SIGCSE Bulletin, 41(1), 26.

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.

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 Retrieved from: 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.

Swaid, S. I. (2015). Bringing computational thinking to STEM education. *Procedia Manufacturing*, 3, 3657-3662.

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.

Acknowledgement

Part of this article was presented the Informing Science Institute Conference in July 2019 - InSITE 2019: Informing Science + IT Education Conferences: Jerusalem.