Coding and Computational Thinking: A time that came and went but will always be ours.

Elena Prieto- Rodriguez
University of Newcastle

In September 2015, the national Australian Curriculum was officially endorsed. This new curriculum included a Digital Technologies (DT) subject, within the Technologies learning area, which was to be focussed on the teaching of *"computational thinking and information systems to define, design and implement digital solutions"* (ACARA, 2018a, 2018b). The document stated that this subject was to be mandatory for all Australian students from Kindergarten to Year 8 and available as an elective for Year 9 and 10 students. All the different States and Territories in the country set to task and by 2018, education authorities had incorporated coding and computational thinking into their respective syllabi. Interestingly, neither of these syllabi incorporated computational thinking within mathematics, even though computational thinking is closely tied to mathematical thinking (Selby & Woollard, 2013). Also, at the time, there were serious concerns about the fact that very few teachers, particularly in the primary school sector, had any formal training in either of these knowledge areas, and there were also concerns that they possessed pedagogies to teach them authentically (Falkner et al., 2014).

Many Professional Development (PD) initiatives for teachers were thus developed by universities and private organisations to address concerns related to teacher preparation for the Australian DT subject (Commonwealth of Australia, 2016). These initiatives ranged from Massive Online Open Courses (MOOCs), such as those run by the University of Adelaide's Comuter Science Education Research (CSER) group (Vivian et al., 2014), and face-to-face workshops (Bower et al., 2017; Chalmers, 2018). A great effort was made to link these PD opportunities to the different syllabi, so that teachers would find the translation from PD to classroom use as seamless as possible.

In 2018, we wrote a paper where we quoted a colleague stating that by adding the terms 'coding' or 'computational thinking' to a research grant proposal in the field of education, you would pretty much guaranteed funding (Hickmott & Prieto-Rodriguez, 2018). While this was said in jest, there was a feeling at the time in Constructionist circles, that the teachings of Seymour Papert were now ripe for worldwide implementation. Indeed, Grover and Pea called computational thinking 'a competency whose time has come' (2018, p.1), and claimed:

> "In a world infused with computing, computational thinking is now being recognized as a foundational competency for being an informed citizen and being successful in all STEM work, and one that also bears the potential as a means for creative problem solving and innovating in all other disciplines" (2018, p. 34).

Presumably to facilitate incorporating knowledge of coding into all these other disciplines, the NSW Education Standards Authority (NESA) prepared the *Coding and computational thinking across the curriculum guide* for teachers. This guide aimed to develop algorithmic and computational thinking skills to better enable students and teachers to reach a coding goal. The guide highlighted the areas where computational thinking can be applied within the existing NSW K–8 syllabuses and contained activities and links to resources organised by stages of learning and learning areas.

Five years and a global pandemic have passed and, at least in Australia, the professional development funding climate seems to have lost interest in coding and computational thinking in favour of the old classics of basic literacy and numeracy. No longer we see an appetite for innovation that would see young people develop into the creative thinkers of the future, but we are forced into standardised lesson plans and strategies to improve our PISA rankings based on rote learning and memorisation. The *Coding and computational thinking across the curriculum* guide has been discontinued and can no longer be found in the NESA website. When I started writing this paper before the symposium in 2022 there were new priorities that teachers should focus on for integration across the curriculum: Aboriginal and Torres Strait Islander histories and cultures, Asia and Australia's engagement with Asia, and Sustainability.

The much-needed training for teachers to implement the digital technologies syllabus has also become harder to provide. Teachers have very busy schedules and having our PD initiatives accredited with authorities to contribute to the 50 hours of PD required of teachers, was always a way to ensure participation. This has also changed. It is practically impossible to have our initiatives accredited unless we have staff devoted to preparing endless applications every time we offer a workshop.

And yet, in this climate, my work continues to focus, as it did twenty years ago, on the integration of coding and computational thinking into the mathematics curriculum. Whether it is a funding priority, a curriculum priority; whether it is in the syllabus or not, I still believe in the power of computing to support the learning of mathematics and vice versa.

I am not the only one. From the 1960s, a limited yet highly influential group of educational researchers has delved into the integration of computer programming to enhance the understanding of mathematics. The year 2006 marked a turning point with the popularisation of the term 'computational thinking' by Jeannette Wing, triggering a notable upsurge in research activity within this field. The body of literature that connects mathematics education with computational thinking is not insignificant. In a systematic analysis we conducted in 2017, at the beginning of the explosion of funding mentioned above, we found that a substantial portion of studies originated from computer science academics rather than experts in the field of education. We also noted that although mathematics is somewhat a focus of the research, studies tend to revolve around teaching programming skills. Additionally, a predominant portion of these studies adopted small-scale research designs focused on self-reported attitudes and beliefs. As a result, we drew the conclusion that there are significant opportunities to pursue more robust research designs that explicitly target mathematics and report on tangible learning outcomes (Hickmott et al., 2018).

But first we need to be clear on what these “tangible outcomes” are. If they are defined by achievement in formal examinations, we need to ask ourselves: will results in a standardised mathematics test taken by children who learned coding tell us whether they have indeed mastered any mathematics content that is worth knowing?

And as for the research designs, how do we know that the teachers who we train to facilitate this new way of knowledge construction, have indeed learned something? In terms of this second question, in 2018 we reported on our six year-long experience training teachers using Seymour Papert’s Constructionist principles (Papert, 1980). Our research concluded that testing teachers to ascertain whether they had learned the concepts and skills we were teaching them really did detract from the experience (Hickmott & Prieto, 2018).

As for the first, over the past few years I have been working with Celia Hoyles and Richard Noss on the implementation in Australia of ScratchMaths. ScratchMaths is a two-year computing and mathematics-based curriculum for Key Stage 2 in the UK (equivalent to Years 4 and 5 in Australia). For the Australian implementation, we conducted professional development with teachers for roughly 8 weeks, commencing with a 2-day professional development workshop and ending with a final showcase where teachers shared their experiences and samples of students' work. We also offered support during the interim classroom implementation period. The aim of the project was to explore participant teachers' perceptions of their ability to facilitate students' learning processes to develop mathematical ideas through coding, and how those perceptions varied after eight weeks of professional learning.

The project was one of the most successful ones I have run: teachers loved the materials, gained confidence in teaching coding (statistically significant!) and also learned coding. Furthermore, their students also showed significant improvement in their learning of coding and computational thinking but also in their enjoyment of mathematics.

We observed increases in participants' teaching self-efficacy across both mathematics and computing (see Table 1). A paired-samples t-test was conducted to check for differences before and after to attending the workshop and after attending showed a number of statistically significant results. There was a change in teachers' perceptions of their ability to teach mathematics with programming before (M = 3.3, SD = 0.47) and after the intervention (M = 4.5, SD = 0.08); $t = -5.09$, $p = 0.037$. There was also a statistically significant difference in their self-efficacy with regards to coding and computational thinking before (M = 3.2, SD = 0.78) and after the intervention (M = 4.3, SD = 0.23); $t = -5.05$, $p = 0.002$.

Table 1. Pre- and post-survey results (items in Mathematics scale preceded by an asterisk)

	Pre	Post	Gain
I feel confident using simple programs for the computer.	4.60	4.71	0.11
I know how to teach programming concepts effectively.	2.67	4.14	1.48
I can promote a positive attitude towards programming in my students.	4.13	4.57	0.44
I can guide students in using programming as a tool while we explore other topics.	2.93	4.43	1.50
*I can guide students in using mathematical thinking as a tool when programming.	3.13	4.43	1.30
I feel confident using programming as an instructional tool within my classroom.	2.67	4.17	1.50
I can adapt lesson plans to incorporate programming as an instructional tool.	2.93	4.29	1.35
I can create original lesson plans, which incorporate programming as an instructional tool.	2.87	4.14	1.28
I understand how mathematics concepts relate to programming concepts.	3.00	4.43	1.43
I appreciate the value of teaching mathematics and programming in an integrated manner.	3.87	4.57	0.70

*All items were presented in a 5-point scale from Strongly Disagree (coded as 1) to Strongly Agree (coded as 5).

It was to hear about the positive outcomes and insights from the ScratchMaths professional development workshops and the subsequent trial period. The teachers' feedback provided valuable information about the effectiveness of the program and areas for improvement. Here's a summary of the key points from the research project:

1. Positive Reception and Student Engagement:
   - Teachers expressed positive sentiments about ScratchMaths after the trial period.
   - Student work samples and reflective comments showcased students' engagement in learning.
   - Teachers reported that students looked forward to ScratchMaths sessions each week.
   - The resources were praised for being well scaffolded and promoting collaboration and social support for learning.
2. Increased Self-Efficacy:
   - Participant teachers reported an increase in their self-efficacy with mathematics and coding.
   - One teacher mentioned that ScratchMaths made complex concepts, like 2D shapes, practical and understandable.
   - Coding provided an opportunity for teachers to feel supported in an area where they might not have felt as confident (e.g., coding for a teacher strong in mathematics).
3. Integration of Mathematics and Coding:
   - While ScratchMaths led to sustained student engagement, not all students engaged equally with the mathematical concepts within the activities.
   - Teachers acknowledged that the mathematical aspects needed to be more explicitly directed to ensure all students engaged with them.
   - Some students used a trial-and-error approach to complete activities, rather than engaging with the mathematical concepts.
   - Teachers acknowledged that the activities were effective in reinforcing concepts already taught in a practical manner.
4. Differing Perspectives on Learning Outcomes:
   - In one regional focus group, teachers viewed coding as a more significant learning outcome than mathematics.
   - In the metropolitan area, teachers observed that some students focused more on trial-and-error approaches rather than mathematical problem-solving.
   - All teachers agreed that the activities were useful for reinforcing concepts.

Based on this feedback, we concluded that ScratchMaths was successful in engaging students and improving teachers' self-efficacy. However, there's still a need to address the varying levels of engagement with the mathematical content among students. Future iterations of the program could focus on providing more explicit guidance for students to engage with the mathematical aspects of the activities. Additionally, it might be valuable to continue emphasising the integration of mathematics and coding, highlighting how coding can serve as a tool for enhancing mathematical understanding. This could involve refining the instructional approach to strike a better balance between coding exploration and mathematical engagement.

Overall, the feedback from the teachers who participated provided important insights for refining ScratchMaths to align with the desired learning outcomes and ensuring that both coding and mathematics are effectively integrated. These results were presented in the annual STEM conference (Prieto-Rodriguez at al., 2019).

We have conducted another four years of (heavily interrupted by a worldwide pandemic) workshops for teachers since the results above. Our amalgamated analysis of survey responses collected during these 4 years of professional development shows results entirely consistent with the ones presented in the 2019 publications.

As a final reflection I would like to remark that this research shows that both teachers and students benefit from the integration of mathematics and computer science, and will continue to do so, whether funding agencies consider it worthwhile or not.

References

ACARA. (2018a). Digital Technologies curriculum rationale. Retrieved from https://www.australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies/rationale/

ACARA. (2018b). F-10 Curriculum - General Capabilities. Retrieved from https://www.australiancurriculum.edu.au/f-10-curriculum/general-capabilities/

Bower, M., Wood, L. N., Lai, J. W., Highfield, K., Veal, J., Howe, C., ... & Mason, R. (2017). Improving the computational thinking pedagogical capabilities of school teachers. *Australian Journal of Teacher Education*, *42*(3), 53-72.

Chalmers, C. (2018). Robotics and computational thinking in primary school. *International Journal of Child-Computer Interaction*, *17*, 93-100.

Commonwealth of Australia. (2016). *STEM Programme Index*. Retrieved from http://www.chiefscientist.gov.au/2016/01/spi-2016-stem-programme-index-2016-2/

Grover, S., & Pea, R. (2018). Computational thinking: A competency whose time has come. *Computer science education: Perspectives on teaching and learning in school*, *19*(1), 19-38.

Hickmott, D., Prieto-Rodriguez, E., & Holmes, K. (2018). A scoping review of studies on computational thinking in K–12 mathematics classrooms. *Digital Experiences in Mathematics Education*, *4*, 48-69.

Hickmott, D., & Prieto-Rodriguez, E. (2018). To assess or not to assess: Tensions negotiated in six years of teaching teachers about computational thinking. *Informatics in Education*, *17*(2), 229-244.

Falkner, K., Vivian, R., & Falkner, N. (2014). *The Australian Digital Technologies Curriculum: Challenge and Opportunity*. Paper presented at the Australasian Computing Education, Auckland, New Zealand.

Papert, S. A. (1980). *Mindstorms: Children, Computers, And Powerful Ideas*. Hachette UK.

Prieto-Rodriguez E, Holmes K, Hickmott D, Berger N, (2019) 'Using Coding to Teach Mathematics: Results of a Pilot Project', Using Coding to Teach Mathematics: Results of a Pilot Project, Queensland University of Technology, Brisbane, Australia.

Selby, C., & Woollard, J. (2013). Computational thinking: the developing definition. Paper presented at the 18th annual conference on innovation and technology in computer science education, Canterbury, United Kingdom.

Vivian, R., Falkner, K., & Falkner, N. (2014). Addressing the challenges of a new digital technologies curriculum: MOOCs as a scalable solution for teacher professional development. *Research in Learning Technology, 22*.