NSW GOVERNMENT

CODING AND COMPUTATIONAL THINKING

What is the Evidence?

education.nsw.gov.au

About the research team

James Curran is an Associate Professor in the School of Information Technologies, University of Sydney. He is Director of the National Computer Science School (NCSS), the largest computer science school outreach program in Australia. Last year, over 10,000 students and teachers participated in the 5-week NCSS Challenge. James is a co-founder of Grok Learning, an Edtech startup that aims to teach children everywhere how to code. He was a writer on the Australian Curriculum: Digital Technologies, the new national computing curriculum. In 2014, James was named ICT Leader of the Year by the ICT Educators of NSW and the Australian Council for Computers in Education.

Karsten Schulz is an Associate Professor in the School of Information Technologies, University of Sydney. Karsten has a PhD in Computer Science and a Bachelor in Electrical Engineering with a focus on Software Engineering. For 10 years, Karsten led the research division of a large multi-national ICT company in Australia and the Asia-Pacific Region and between 2013 and 2016 he led the national Digital Careers Program. In 2010, Karsten created the Young ICT Explorers Competition and in 2014 he founded the Bebras Australia Computational Thinking Challenge which is part of the international Bebras Challenge.

Amanda Hogan is the current president of ICT Educators NSW which is the professional association for computing teachers in NSW. She is a secondary Computing Teacher at Tara Anglican School for Girls. She has come to teaching after working in the IT industry most recently at Microsoft Australia. Amanda is an active tutor and content creator in the Girls' Programming Network Sydney Chapter. She runs the Maths, Coding and Robotics clubs at school and is a keen programmable electronics hobbyist. In 2015, Amanda was named ICT Teacher of the Year by the ICT Educators of NSW.

About the Australian Computing Academy

The University of Sydney leads the Australian Computing Academy to provide the intellectual, technical, and practical leadership needed to fulfil the ambitious goals of the Australian Curriculum: Digital Technologies. This includes working directly with jurisdictions and systems as they implement the Digital Technologies curriculum. For more information about the Academy please visit aca.edu.au.

Australian Computing Academy

A/Prof James R. Curran
A/Prof Karsten A. Schulz
Amanda Hogan

Faculty of Engineering and Information Technologies
The University of Sydney, NSW 2006
A/Prof James R. Curran
Phone: (02) 9036 6037 / 0431 013 320
james.r.curran@sydney.edu.au
aca.edu.au

Table of Contents

Introduction

Overview

This report has been commissioned by the NSW Department of Education's Education for a Changing World initiative. It investigates the available evidence on the teaching of coding and computational thinking for student outcomes and explores and critiques what works in the implementation of coding and computational thinking within an educational context.

About this report

This report addresses the following areas, summarising the relevant research literature and evidence base as available:

- Definitions of coding, computational thinking and related computing terminology, especially with respect to the Australian Curriculum: Digital Technologies, the Australian Curriculum: Information and Communication Technology (ICT) Capability, and current and soon to be implemented NSW Education Standards Authority (NESA) syllabi
- Analysis of the history of coding and computational thinking in the classrooms dating back to the 1970s. Reflection on the momentum (or hype) surrounding coding and computational thinking with reference to Australian and international job projections, and the use of coding across careers and learning areas
- Analysis of evolving computing curricula internationally and the place of the Australian Curriculum and NSW syllabi within that
- Discussion of the learning outcomes (both learning-area specific and general capability) of teaching coding and computational thinking to all students across both digital technologies and integrated into other curriculum areas
- Summary of best-practice for teaching coding and computational thinking effectively to students including the use of learning progressions, technology tools and programs, unplugged activities and other resources. We also summarise the range of resources already available in Australia to support the digital technologies curriculum and
- Summary of state, national and international efforts in providing teachers with support and professional development for teaching coding and computational thinking, the small number of teachers with existing coding skills.

This report is intentionally broad, rather than narrow and technical, in order to provide a comprehensive overview of the aspects to be considered in formulating (i) a strategic vision for teaching coding and computational thinking in NSW public schools and (ii) in light of the report's target audience of policymakers, school leaders and non-specialists. The report concludes with proposed areas for future consideration.

Part 1: Defining computational thinking and coding in Australian education contexts

In schooling contexts, coding and computational thinking are taught in subjects such as digital technologies, computer science, informatics, computer programming, robotics, information and software technology, information processes and technology and software design. For the purpose of this report we will primarily use the contemporary Australian term digital technologies. When referring to an historical or international context, we will use the US term computer science.

Computational thinking

The Australian Curriculum, Assessment and Reporting Authority (ACARA) defines computational thinking as a **"problem-solving method that involves various techniques and strategies that can be implemented by digital systems. Techniques and strategies may include organising data logically, breaking down problems into parts, defining abstract concepts and designing and using algorithms, patterns and models"** (ACARA Glossary).

In defining computational thinking, the NSW Education Standards Authority (NESA) refers to the work of Jeannette Wing (2010, p. 1) and defines computational thinking as "the thought processes involved in **formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent."**

Contrary to the scientific/mathematical approach to problem-solving, in which a solution to a problem is generally expressed as an integrated formula, a computational-algorithmic solution to a problem involves a sequence of steps. Due to this step-by-step response to problem-solving, computer algorithms can be used to reflect on, and pose solutions to, complex real-world scenarios such as data sorting, without which databases would not work and therefore search engines such as Google could not exist.

At the core of the computational thinking approach sits the belief that a solution to a problem can be obtained by a repeated cycling over the problem. Each cycle results in incremental solutions that, fed back into the solution process produce a slow shift, eventually leading to a solution to the original problem. This is a fundamental difference between the computational thinking approach and all other thinking approaches presently taught at school.

Coding

The Australian Curriculum: Digital Technologies does not define coding and only refers to programming in the context of object-oriented, visual or general-purpose text programming (ACARA Glossary).

Generally, coding is an activity that converts a piece of information from one representation into another. In contrast, computer programming is the process of designing, writing, testing, debugging, and maintaining the source code of computer programs.

Historically, coding was simply seen as the last step to writing a computer program based on the specifications of a designer - usually a computer architect. This approach to coding has since been replaced by an iterative approach where a rapid succession of design-implement-debug-test phases leads to a solution. Increasingly, the designer of a computer program is also responsible for its implementation, debugging and testing. Coding has therefore become a synonym for programming.

Coding is done through the use of programming languages. A programming language is needed to translate a human representation such as a flowchart, text or pseudocode into a representation that a computer can understand and execute. Popular programming languages currently include Python, JavaScript and C/C++. Generations of computer scientists have developed abstraction layers, frameworks, libraries and programming languages to streamline programming. Historically, the early computer engineers had to move wires to program a loop in binary that would instruct the computer hardware to perform a multiplication through repeated addition. In contrast, today's programmers only have to press the '*' character on their keyboards to achieve the same outcome.

As will be explored throughout this report, there is no established consensus if and to what extent learning coding influences the development of higher-order computational thinking. According to Mohaghegh & McCauley (2016) and Grover (2013) learning with programming languages such as Scratch, Massachusetts Institute of Technology's (MIT) App Inventor, Kodu and Alice can help primary school students rapidly construct working programs and apps. Mohaghegh & McCauley argue that while this is "advantageous in learning programming skills, the degree of conceptual knowledge obtained through using these tools can be questioned, specifically to what extent students learn computational thinking skills" (2016, p. 1527). Rather than simply learning the syntax of a language, Grover (2013) advocates for a focus on solution construction and exploration of why some solutions are more effective than others. Grover suggests this approach allows young students' learning ability and creativity to be developed alongside computational thinking skills.

Digital Technologies Curricula in Australia and NSW

The Australian Curriculum: Digital Technologies

The Australian Curriculum: Digital Technologies is a new national subject within the Technologies learning area. The subject is mandatory from Foundation (Kindergarten in NSW) to Year 8, with elective offerings following for Year 9/10 students. Digital technologies was endorsed on 18 September 2015 by the Education Council, with jurisdictions committing to implementing the curriculum from 2016.

The digital technologies curriculum includes fundamental ideas from the academic disciplines of computer science, information systems and informatics. The curriculum states:

"In a world that is increasingly digitised and automated, it is critical to the wellbeing and sustainability of the economy, the environment

and society, that the benefits of information systems are exploited ethically. This requires deep knowledge and understanding of digital systems (a component of an information system) and how to manage risks. Ubiquitous digital systems such as mobile and desktop devices and networks are transforming learning, recreational activities, home life and work. Digital systems support new ways of collaborating and communicating, and require new skills such as computational and systems thinking. These technologies are an essential problem-solving toolset in our knowledge-based society" (ACARA, Digital Technologies subject rationale).

The developmental progression of learning in the subject is captured in the content descriptions for each band and held together by a collection of conceptual threads. This work is set out in the PDF document available on the Australian Curriculum website and includes:

- Abstraction (underpins all other key concepts)
- Digital technologies knowledge and understanding:
    - Digital systems
    - Data representation
- Digital technologies processes and production skills:
    - Collecting data and interpreting data
    - Specification (defining a problem)
    - Algorithms
    - Implementation (computer programming or coding)
    - Impact [1]
    - Interactions [2]

[1] Analysing and predicting how existing and created systems meet needs, affect people, and change society and the world.
[2] How users experience and interface with digital systems, and how we use them to communicate and collaborate.

While the digital technologies curriculum does not refer to computational thinking on a frequent basis, the stages of the design process correspond strongly to the main aspects of computational thinking (defined above).

Most Australian jurisdictions formally reported against this curriculum from 2018.

NSW Syllabus

The NSW Syllabi for Stages ES1-3 (Science and Technology) and Stage 4 (Technology (Mandatory)) were released in December of 2017 with implementation mandated for 2019. The Technology (Mandatory) Years 7-8 syllabus combines the Australian Curriculum: Design Technologies and Digital Technologies.

Digital technologies makes up at least 50 hours of the broader Technology (Mandatory) syllabus. It has a similar shape to the Australian Curriculum with the inclusion of knowledge and understanding and design and production skills student outcomes.

In primary education the digital technologies curriculum has been incorporated into the new science and technology syllabus, which has knowledge and understanding and skills as categories for student outcomes.

Part 2: The Australian and global contexts: The increasing importance of computational thinking and coding

Historical context

Pre-university computer science education started in the 1970s. The computer science teachers of the 1970s and 1980s were mostly mathematics and physics teachers who viewed computer science as extension of their respective fields. This is the reason why mathematics and physics dominated the early-use cases. In this section, we consider both Australian and global historical perspectives.

Australian perspective

In world terms, Australia moved into educational computing both at the higher education and school levels very early. University computing in Australia started in 1947 with the University of Sydney's course 'The Theory of Computation, Computing Practices, and Theory of Programming'. In 1949, the Radiophysics Laboratory at the CSIRO in Sydney released Australia's first and the world's fourth digital computer, the Council for Scientific and Industrial Research Automatic Computer (CSIRAC). The 1960s saw several key computing personnel moving from industry to academia to set up computing courses (Tatnall & Davey, 2004). "From 1965, computing courses became 'respectable' and were soon widely available" (Tatnall & Davey, 2004, p. 3). It was in the early 1970s that school computing began to emerge, usually as a result of teachers being exposed to computing during their university studies. "In 1972, for example, Burwood High School was loaned a PDP-8 computer by Digital Equipment [...]. In 1973 McKinnon High School received an Innovations Grant to enable the purchase of an 8k Wang computer costing over $10,000 (AUS) [...]. The biggest impact on schools, however, was [the] introduction of the Monash Educational Computer System (MONECS). Before the advent of personal computers (PCs) it was impossible for an average school to have hands-on access to a computer. In 1974 a group at Monash University produced a system [...] that allowed a class of 30 children to each get two runs [3] in a one-hour period (Monash Computing Museum 2003). The MONECS system was used to teach programming in FORTRAN or BASIC. At this stage, schools saw computing as a branch of mathematics concerned with algorithm design" (Tatnall & Davey, 2004, p. 4).

With the advent of PCs in the late 1970s and early 1980s, the foundations of most of the current computing curriculum in schools and universities were established. PCs lowered the barrier of entry into computing for schools and universities significantly as they were more affordable and easier to use than mainframes. In 1981 computer science was first offered as a Higher School Certificate (HSC) subject in Victoria (Tatnall & Davey, 2004, p. 5).

Global perspective

Resnick et al. (2009) describe the historical foundation of computer science teaching in the classroom. While they do not identify a particular country for their work, their descriptions resonate with the personal experiences of the authors of this paper in Australia and Germany. Personal computers were first introduced in the late 1970s and 1980s and generated initial enthusiasm for teaching children how to program. Thousands of schools taught millions of students programming in languages such as Logo, Basic and Pascal.

This initial excitement for computing in the classroom soon faded as schools shifted to using computers for other purposes. Resnick et al. (2009) argue that the reason for this was that (i) early

[3] Two runs likely refers to two attempts of running a program on the computer.

programming languages were too difficult to use and that many children couldn't master the syntax of programming, (ii) programming was often introduced alongside abstract activities, such as generating lists of prime numbers, that were not connected with purpose for the young learners, and (iii) a lack of guidance when things went wrong or encouragement of deeper exploration when things went right. This last point has significant implications for teacher competence and confidence.

In summary, from an historical viewpoint, the success of computer science education in the classroom depends on four factors:

1) Appropriate programming language tools and resources
2) Appropriate use of computers
3) Provision of a purposeful context that students can identify with
4) Teacher competence and confidence.

Relating historical success factors to the present

In this section we apply the four historical success factors from the 1970s and 1980s to the present situation in Australia.

Appropriate programming language tools and resources

The Australian Curriculum: Digital Technologies does not mandate the use of particular tools and resources other than a move from visual programming languages to general-purpose programming languages which occurs at the end of Year 6.

Visual programming languages, such as Blockly, Alice, GameMaker, Kodu, Lego Mindstorms, MIT App Inventor and Scratch address vocabulary and syntax, key concepts that all learners of languages need to learn. In programming language terminology the vocabulary refers to the instructions that a programmer needs to know to instruct the machine to perform a specific task. Examples include data types, loops, functions, assignments and operations. The syntax of a programming language specifies the rules of how the instructions are to be put together to form valid constructs that the computer can process. This is called grammar in human languages. Visual programming languages address vocabulary and syntax by providing predefined blocks from which students can choose. This is significant, as it frees students from needing to know much upfront about the programming language. Instead, students can choose from a limited set of blocks (vocabulary) which only click together in syntactically valid ways, thereby supporting accurate learning.

There are several disadvantages to visual programming languages, which the blog by Schulz and Fuda (2018) summarises comprehensively. The Australian Curriculum: Digital Technologies mandates the move to general-purpose (text) programming languages from Years 7/8 and up. Beyond programming languages themselves there is now a growing number of resources and tools that are useful (or claim to be useful) in teaching primary and secondary students computational thinking and/or programming. Education Services Australia maintains a curated list of over 700 of these resources in the Digital Technologies Hub and maps each resource against the content descriptors of the Australian Curriculum: Digital Technologies.

Lockwood & Mooney (2017, p. 30) conclude that a "huge number and range of tools have been developed to assist the teaching of computational thinking. These range from music tools to programming languages to games. Although several of these are in the early stages of development, it is encouraging to see so many efforts to make computational thinking fun and accessible to students of all ages, genders and abilities. The benefits for educators are many and include a variety of options of how to integrate computational thinking into their classrooms. Whether in a computer lab, a regular classroom or outside, in a one-on-one session or with a class of 30+ there is a tool out there which will suit educators' needs, and if there isn't then the evidence suggests that there

might well be soon!"

In summary, the evidence suggests that there are now age-appropriate programming languages to help primary-school students begin learning coding. This constitutes a significant change compared to the 1970s and 1980s when students had to work with text-based programming languages which provided less assistance around how to produce code.

Proper use of computers

In recent years, some schools have moved the student-related delivery of learning content and student-related administration onto computers. In Australia, one of the catalysts for this development was the Australian Government's 2008 Digital Education Revolution (DER) (Department of Education and Training), (ZDNet, 2013), (DEEWR, 2013). Although this program ended in 2013, it kicked off an investment in information technology (IT) infrastructure and support that was probably more significant in its long-term impact on schools than the financial support for the purchase of laptops themselves.

The DER program ended more than four years ago and most if not all of the laptops have now been retired. Today students use their own devices or school-provided computers for much if not most of their day-to-day learning.

Increasingly, tools such as OneNote are being used as a medium for homework, and paper textbooks are being replaced by eBooks or PDFs stored on the student's computer.

Whereas previously computers were primarily used for computer science education, they are now primarily a vehicle for learning content that is not related to computer science education.

This has resulted in **confusion between information and communications technology (ICT) general capabilities and coding and computational thinking**, which can lead to the mistaken belief that schools are already teaching digital technologies, when really they may only be requiring students have access to a computing device.

The DER mid-program review report (DEEWR, 2013) draws no specific correlation between the DER and enhanced learning outcomes. The authors point out the complexities of measuring educational outcomes, due to (i) concurrent implementation of reform activities that make it difficult to isolate and prove causality, (ii) the lack of control groups, (iii) the ramp-up time of DER and therefore the shortness of the analysis period and (iv) that the DER targets years 9-12 students. In contrast, the National Assessment Program Literacy and Numeracy (NAPLAN) is for years 3, 5, 7 and 9 and only tests "Reading, Writing, Language Conventions (Spelling, Grammar and Punctuation) and Numeracy"(DEEWR, 2013, p. 18). Instead, the report summarised non-DER related studies for which we refer the reader to (DEEWR, 2013) for a comprehensive overview. We revisit this point in the section titled "Implications for student learning outcomes."

Purposeful context that students can identify with

There is general agreement about the importance of teaching within a context that students can identify with. In the 1970s and 1980s computer science emerged as a spin-off from mathematics and physics and therefore its teaching drew many examples from those disciplines. Since then, computational thinking has been applied to almost all aspects of life, including astronomy, the environment, health, sports, and the economy. In parallel, embedded computing systems, such as the Arduino and the Raspberry Pi have emerged that enable deployments into the application domain that portable and ultra-portable laptops and tablets could not achieve. Therefore, identifying a context that students find interesting is now primarily limited by the imagination and domain-expertise of the teacher and not by the capability or the cost of technology.

Lockwood & Mooney (2017) summarise their findings concerning the application of coding and computational thinking to already existing

subjects, including biology, physics, mathematics and English. They conclude "that introducing computational thinking doesn't have to be done exclusively through new courses or even through computer science. Computational thinking is a skill that can be used in a possibly surprising range of disciplines and can benefit students studying in any area. The ability to break down a problem and develop a manageable solution is one that all students will find useful in both their academic and work lives... Computational thinking can be successfully taught in varying topics and subjects which can be especially helpful to educators dealing with already crammed curriculums" (Lockwood & Mooney, 2017, p. 21).

Teacher competence and confidence

Teacher competence and confidence emerges as one of the most significant factors in successfully teaching coding and computational thinking in the classroom. Yet many teachers expected to teach the digital technologies curriculum in classrooms today would have received limited relevant training in their initial teacher education.

While there is a shortfall in suitably trained digital technologies specialist teachers the task of delivering the new curriculum often falls to teachers, such as primary school teachers, who may have limited specialist knowledge in this area.

The process of using computational thinking and coding to solve a problem in a maths or science lesson is quite different compared to using these in an English or art lesson. As a consequence non-maths/physics subjects often use existing programs (apps) rather than making them, which provides limited opportunities for students to develop computational thinking skills.

Summary

Since the 1970s and 1980s, new programming languages have been specifically developed to support primary and secondary school students to learn coding. Conversely, the increasingly pervasive use of computers in schools can result in the subject being ignored due to the mistaken belief that using a computing device is the same. Embedding coding and computational thinking into purposeful scenarios that students can identify with is essential. Effective teaching of the subject is primarily limited by the imagination and domain-expertise of the teacher and not by the capability or cost of technology. Teacher competence and confidence emerges as one of the most significant factors in successfully teaching coding and computational thinking in the classroom.

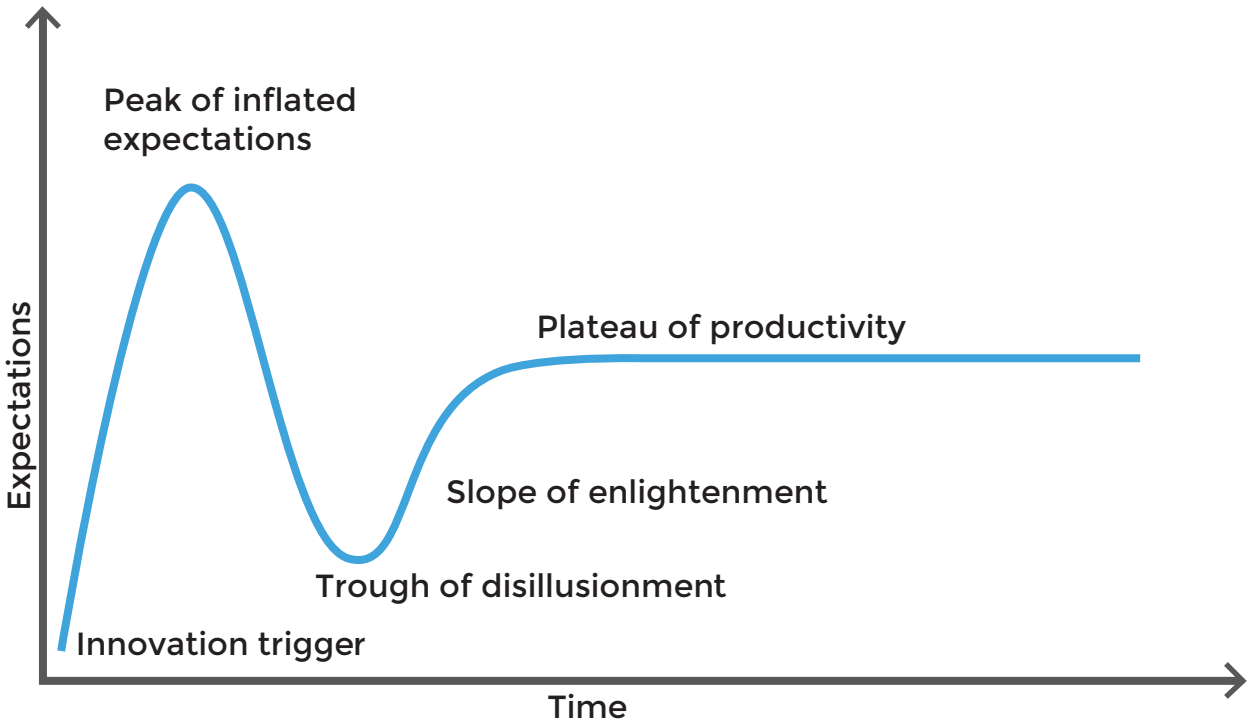
The tertiary sector and the IT industry

In this section, we investigate the link between schools, universities and the IT industry. We consider the impact of the dot-com boom and bust on university enrolment numbers and compare the open and high-paying jobs in the IT industry with other industries in Australia.

Dot-com boom (and bust)

The first wave of computing in the 1970s and 1980s resulted in what has been coined the dot-com boom. The early 1990s mark the beginning of the technology information age when a growing number of technology companies were founded and supplied with ample venture capital. It also marked the beginning of the industrialisation of company formation. Dot-com companies were highly valued on the stock exchanges, especially the NASDAQ. The dot-com boom follows the Gartner hype curve (Gartner) in which an innovation trigger leads to a peak of inflated expectations, a trough of disillusionment and a gradual slope of enlightenment until a plateau of productivity is reached.

The dot-com boom led to a bust that started in March 2001. By the end of the stock market downturn in 2002, stocks had lost US$5 trillion in market capitalisation. The stock market losses were felt by many mum and dad investors that had invested in a market that looked too good to fail. Hence, the effects of the tech-bubble was felt by ordinary people. In combination with the high-profile reporting by the media about failing companies and job losses in the tech industry, many people formed a view that (i) there is no professional future for and (ii) no jobs in IT.



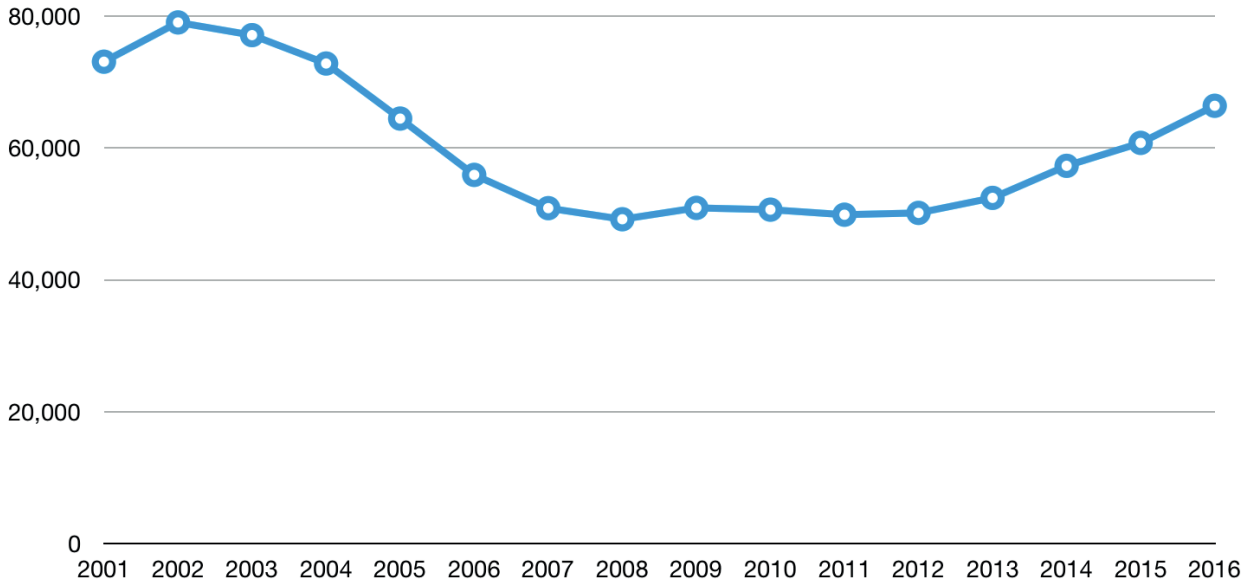
Gartner Hype curve, Graphics: Australian Computing Academy

University enrolment trends

Falkner (2017) at the University of Adelaide has researched the enrolment trends in Information Technology and Engineering courses at Australian universities. Her work is based on uCube data from the Australian Government Department of Education and Training. The analysis comprises the years 2002-2015 and depicts a decline in higher education enrolments from 2002 to 2008, followed by a plateau until 2012 and a gradual increase since 2013. The data shows a two-year lag between the bust of the dot-com bubble in 2000 and the decline in higher education enrolments in IT in 2002. A possible explanation is that the Year 10 students of the year 2000 had already made up their minds about their future tertiary fields of study.

IT enrolments in Australia have not yet reached the levels of their peak in 2002. As of 2016, the field of information technology appears to be in the slope of enlightenment in relation to the Gartner hype curve. The plateau of productivity appears not to have been reached yet when comparing the IT higher education enrolment chart with the Gartner hype graph.

The Australian Government Department of Education and Training annual report of the Undergraduate Applications and Offers listed 8,719 offers being made in 2017, which constituted an 8.1% increase on 2016 levels.



IT Higher Education Enrolments, based on uCube data (2001-2016). Graphics: Australian Computing Academy

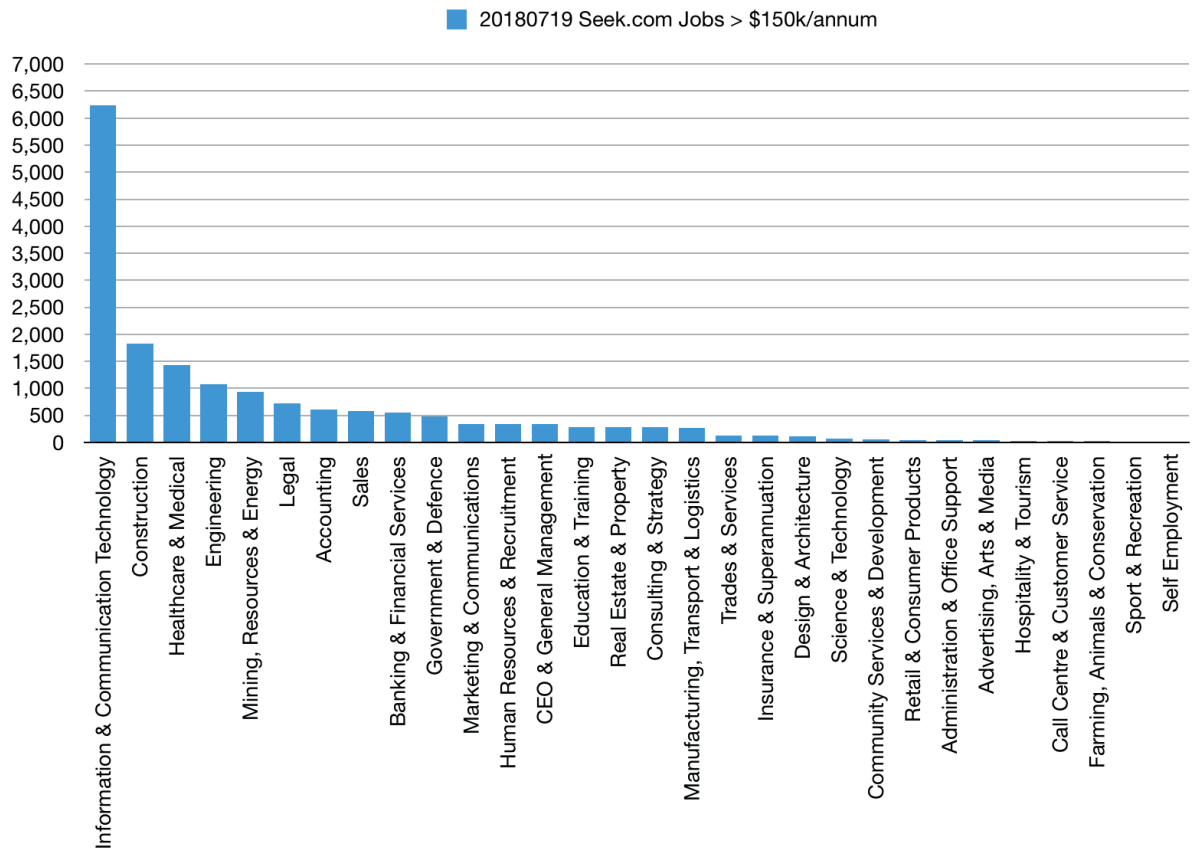
Field of education	Offers			Offer rates		
	2016	2017	% Change	2016	2017	Change
Natural and Physical Sciences	29,982	31,326	4.5%	97.5%	97.3%	-0.3
Information Technology	8,062	8,719	08.1%	84.9%	83.8%	-1.0
Engineering and Related	16,251	16,628	2.3%	86.2%	84.8%	-1.4
Architecture and Building	6,932	7,278	5.0%	80.0%	75.7%	-4.2
Agriculture, Environmental &	4,194	4,064	-3.1%	90.7%	89.7%	-1.0
Health	64,451	65,153	1.1%	73.7%	72.4%	-1.2
Medical Studies	2,130	2,226	4.5%	25.2%	23.9%	-1.2
Dental Studies	867	785	-9.5%	41.3%	38.8%	-2.5
Veterinary	1,142	1,051	-8.0%	56.1%	52.6%	-3.5
Nursing	26,788	26,247	-2.0%	77.2%	73.2%	-4.0
Health Other	33,524	34,844	3.9%	83.5%	85.5%	2.0
Education	23,718	24,113	1.7%	83.5%	82.9%	-0.6
Teacher Education	22,215	22,382	0.8%	81.7%	80.7%	-1.0
Education Other	1,503	1,731	15.2%	122.5%	127.7%	5.2
Management and Commerce	40,316	38,728	-3.9%	90.0%	88.7%	-1.3
Society and Culture	66,958	66,470	-0.7%	87.0%	85.7%	-1.3
Law	9,008	8,641	-4.1%	68.1%	66.3%	-1.7
Creative Arts	23,634	22,499	-4.8%	79.0%	78.2%	-0.9
Mixed field programs	1,338	1,234	-7.8%	112.2%	106.7%	-5.5
Total	**285,846**	**286,216**	**0.1%**	**83.7%**	**82.5%**	**-1.2**

Offers and offer rates by field of education, 2016-2017. Australian Government, Department of Education and Training (2017), Licensed under: Creative Commons Attribution 3.0 Australia

Jobs

In August 2014, the Australian job portal Seek listed 12,659 open positions in ICT out of which 3,453 were remunerated over $150,000 per annum.

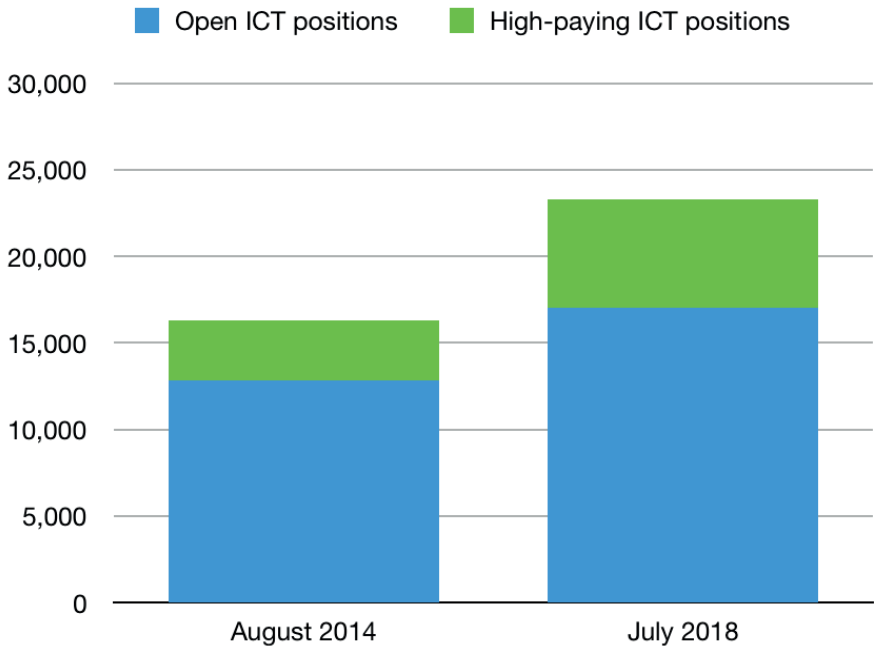
As of 19 July 2018 Seek listed a total of 172,525 open positions. 17,064 (9.9%) fell under the general category of ICT. Seek also lists high-paying jobs with an annual remuneration package of over $150,000. Here, ICT has 6,234 (36%) out of 17,318 open positions.



High-paying open positions by industry (19.7.2018). Seek.com data. Graphics: Australian Computing Academy

The development of open ICT positions between 2014 and 2018 follows an upward trend with an average annual increase of 8.3%. The number of high-paying ICT job adverts increased by 20.1% on average per annum. The ratio of high-paying to open positions increased from 26.9% (2014) to 36.5% (2018).

	Open ICT positions	High-paying ICT positions	High paying / open
August 2014	12,825	3,453	26.9%
July 2018	17,064	6,234	36.5%
Annual increase	8.3%	20.1%	



Comparison of open and high-paying ICT industry positions as of August 2014 and July 2018. Seek.com data. Graphics: Australian Computing Academy

Summary

The bursting of the dot-com bubble has shaped a negative perception by market participants concerning the job prospects in the IT industry. This has led to a significant downturn in IT course enrolments at tertiary institutes, which has resulted in a shortage in IT specialists leading to a large number of open and high-paying positions. The IT industry is presently in a growth phase in relation to the Gartner hype curve (Gartner) and job prospects and therefore university enrolments will likely continue to increase in the short to mid-term.

European Union policy analysis

On behalf of the European Commission Bocconi et al. (2016) conducted a comprehensive study analysing various European Union (EU) and non-EU initiatives concerning computational thinking, coding and related concepts such as programming, algorithmic thinking in compulsory education. The purpose of the study was to show the implications for policy and practice.

According to Bocconi et al. (2016) the most significant contributions to the relationship between

digital competence and computational thinking come from the following four policy documents:

- Running on Empty: The Failure to Teach K–12 Computer Science in the Digital Age (Wilson, Sudol, Stephenson & Stehlik, 2010)
- Shut down or restart? The way forward for computing in UK schools (The Royal Society, 2012)
- L'enseignement de l'informatique en France - Il est urgent de ne plus attendre (Académie des Sciences, 2013)
- Informatics education: Europe cannot afford to miss the boat (Gander et al., 2013)

Bocconi et al. (2016) criticises pedagogy and practice which focuses only on students having access to technology as opposed to learning about the ideas and science which underpin that. The Australian Curriculum: Digital Technologies follows this approach by focusing on key concepts and not mandating use of any specific technology. The detailed exploration of these four papers is beyond the scope of this report, save for recommendations from the Royal Society paper (2012), which we discuss in Part 2.

Student assessment

Bocconi et al. (2016, p. 7) state that experts and practitioners are emphasising the importance of assessing students' ICT skills. However, they argue that "only a limited amount of research has been carried out and currently there are only a few actual experiences of assessing students' grasp of computational thinking concepts and of transferring of computational thinking skills to other knowledge domains."

Implementation in national curricula

The countries investigated by Bocconi et al. (2016) approach the teaching of computational thinking in different ways. Some countries integrate computational thinking across subject areas, particularly at the primary level. Others, especially at the secondary level, include computational thinking as a separate subject. Some countries, such as Wales and Austria include computational thinking and related concepts as part of the digital competence curriculum. Digital competence is a term used by the European Commission and Norwegian scholars to describe ICT general capabilities.

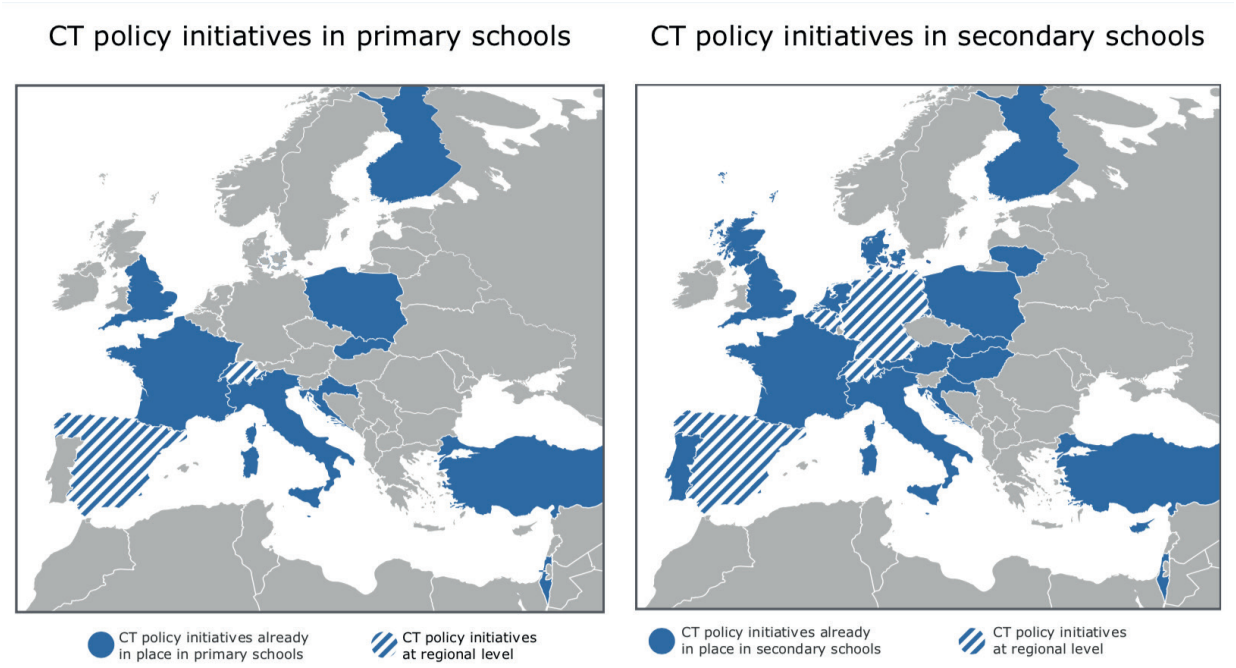
Bocconi et al. (2016) remark that a lack of a common understanding of computational thinking inhibits its implementation in school curricula:

"An upsurge in the integration of computational thinking and, more broadly, of Computer Science in compulsory education is evident, as indicated by the recent wave of curricula reforms. Eleven countries in Europe (Denmark, France, Finland, Croatia, Italy, Malta, Poland, Turkey, UK-England, UK-Scotland) have recently concluded a reform process that includes computational thinking and related concepts. Seven others (Czech Republic, Greece, Ireland, Netherlands, Norway, Sweden, Wales) are currently planning to introduce computational thinking into compulsory education. Moreover, seven other countries (Austria, Portugal, Cyprus, Israel, Lithuania, Hungary, Slovakia) are integrating computational thinking by building on their long-standing tradition in computer science education, mainly in upper secondary schools. Some of these are expanding computer science education to include the lower secondary and primary levels. For those countries (namely: Spain, Germany, Belgium and Switzerland), where curricula development is managed at regional level, the integration of computational thinking in school varies from region to region." (Bocconi et al., 2016, p. 9)

The study analysed how computational thinking is positioned in the curriculum along two axes: educational levels and subjects. Most countries integrate computational thinking in secondary school. However, there is now a growing trend towards primary school integration.

Implementation framework

The study puts forward an integration framework comprising consolidated understanding,



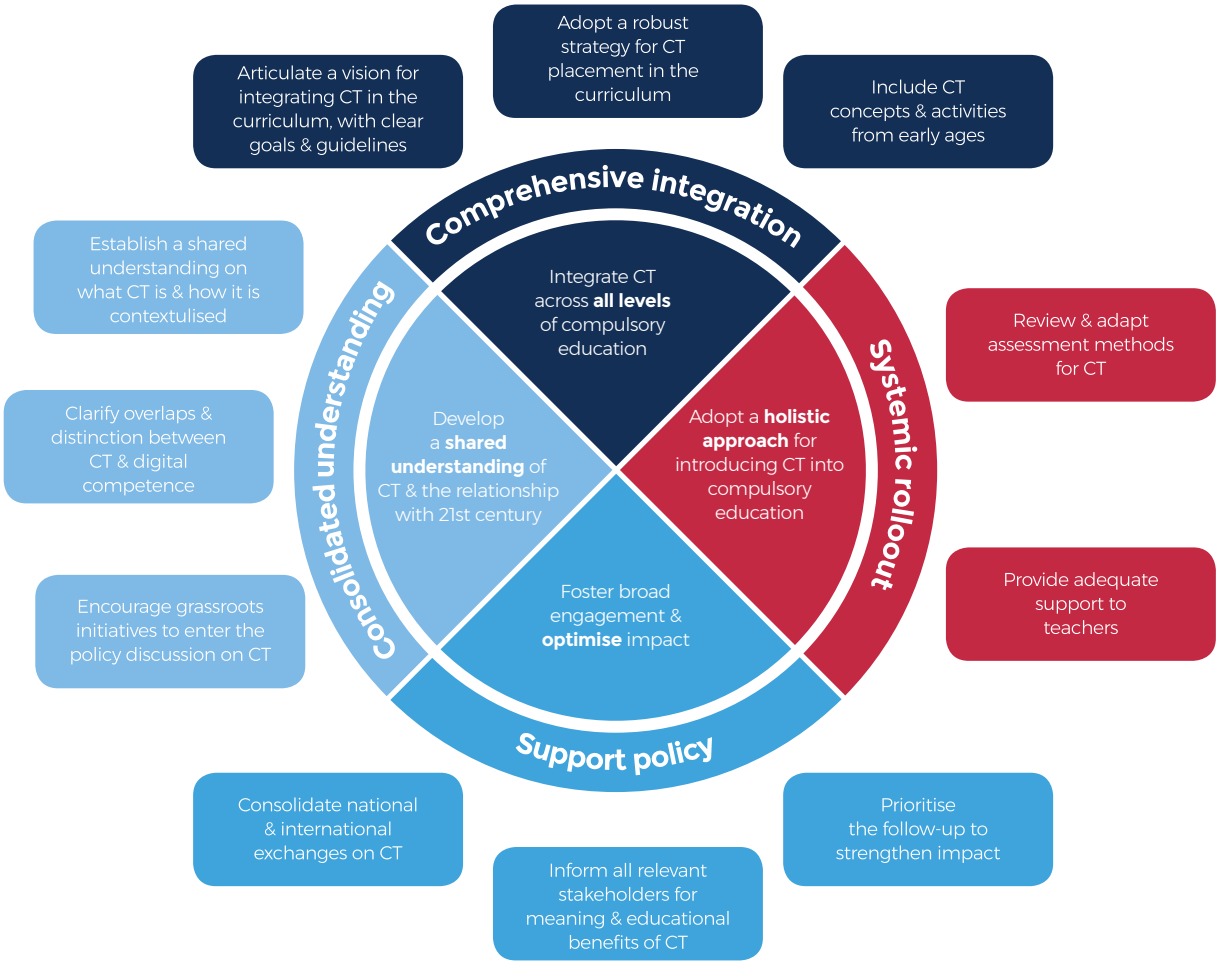
Computational thinking integration by education level in Europe and beyond (Turkey and Israel) Reproduced with permission by the European Commission – DG JRC – Directorate Innovation and Growth

comprehensive integration, systemic rollout and support policy:

"In order for computational thinking to be integrated comprehensively across all levels of compulsory education, it is necessary to **define a clear vision and set specific goals. As computational thinking involves far more than offering a few hours of coding,** placing it in the curriculum calls for a robust strategy that takes into account the wide range of factors involved. A key consideration is the extent to which computational thinking is allocated across the full spectrum of subject area studies and, also, in multi-disciplinary and interdisciplinary contexts. **Introducing computational thinking concepts to children early on in school is commonly held to be desirable**. These considerations call for a holistic approach to computational thinking integration in compulsory education, which embraces essential aspects such as suitable assessment strategies and adequate teacher training" (Bocconi et al., 2016, p.7).

ACARA, in line with the implementation framework proposed by Bocconi et al. (2016), has made the study of digital technologies compulsory for all students from Foundation through to year 8. Further study through to year 10 is optional.

Bocconi et al. (2016) encourage close links between stakeholders including policymakers, grassroots initiatives, research centres, and teachers to allow them to learn from each other and history to avoid the repetition of mistakes and to promote good practice. Other stakeholder groups not directly involved in curricula development should also be adequately informed about what computational thinking is and how it is relevant to compulsory education. Bocconi et al. (2016) recommend a wide-angle monitoring and analysis strategy to measure the impact and sustainability of implemented actions.



Introducing computational thinking into compulsory education: implications for policy and practice. Reprinted from (Bocconi et al., 2016).

Reproduced with permission by the European Commission – DG JRC – Directorate Innovation and Growth

International case studies

In most countries the main rationale for introducing computational thinking and coding is to foster 21st-century skills. These are seen as essential for active and fruitful participation in the knowledge economy and for employment in a digitally-oriented jobs market. The degree, however, varies. Where Austria, Denmark and Hungary take a logical thinking and problem-solving approach, Finland and Turkey take a more holistic approach that includes logical thinking, problem-solving, fostering key competencies (ICT skills) and coding, aimed at attracting more students into computer science and fostering employability in the ICT sector.

In this section we present examples of countries that have decided to move forward with computing in the classroom. We use this general term intentionally since the terminology, foci and scope of the initiatives ranges vastly. The constituting elements of the national approaches are as follows:

1. Achieving student proficiency in the use of a range of technologies (laptops, 3D printers, software)
2. Coding
3. Computational thinking and algorithmic thinking

	Austria	Czech Republic	Denmark	Finland	France	Greece	Hungary	Italy	Lithuania	Poland	Portugal	Switzerland	Turkey
Fostering logical thinking skills	■	■	■	■	■	■	■	■	■	■	■	■	■
Fostering problem solving skills	■	■	■	■	■	■	■	■	■	■	■	■	■
Fostering other key competences		■		■		■		■	■	■	■	■	■
Attracting more students into Computer Science				■	■				■	■			■
Fostering coding and programming skills				■	■				■	■	■	■	■
Fostering employability in the ICT sector				■	■								■

Rationale for integrating computational thinking in the curriculum by country, based on (Bocconi et al., 2016). Ongoing or planned activities.

4. Teacher training
5. The development of teaching resources for non-computing subjects that use digital technologies.

A rigorous evaluation of the initiatives cannot yet be expected given their relative novelty. Even the Estonian ProgeTiger program (discussed below), which can be considered as a testbed and frontrunner, has not yet published a program analysis.

Estonia

In primary education Estonia has a national cross-curricular theme called 'Technology and Innovation' which requires all teachers to implement technology in their teaching. That means that teachers have to integrate technology into different subjects, for example using Scratch in mathematics or music programs in music lessons. The curriculum does not specify how to, or exactly what technology teachers must use, leaving them with a level of personal discretion.

In 2012, when the idea of teaching programming and robotics was introduced to Estonian schools, Estonia launched the ProgeTiger program. **The goal of ProgeTiger is to enhance learners' technological literacy and digital competence.** From an Australian curriculum perspective it involves a mix of ICT general capabilities, design and technologies, and digital technologies and their integration into the teaching and learning of different subjects and extracurricular activities. The program financially supports kindergartens and schools in acquiring different programmable devices. ProgeTiger is supported and funded by the Estonian government through the Ministry of Education and Research.

In March 2018 the Estonian Information Technology Foundation for Education reported on the scale of ProgeTiger in Information Technology Foundation for Education (2018): "Over the past five years, equipment in the amount of more than 830,000 euros has been purchased to 446 schools and kindergartens for teaching robotics, programming, 3D modelling as well as multimedia. Attention has also been paid to increasing the teachers' technological literacy. More than 4,100 teachers have participated in trainings of ProgeTiger." The ProgeTiger program yet to publish an evaluation.

Finland

Finland introduced a national computer science school curriculum in the 2016-17 school year. The curriculum does not use the word 'code', but instead discusses computational thinking (Finnish: algoritminen ajattelu) and programming (Learning Environments research group, 2015). The focus is less about programming and more about computational thinking. The underlying motivation has been expressed by Guzdial (2017): "We want students to understand what a computer can do, what a human can do, and why that's different. To understand computing is to have a robust mental model of a notional machine". According to Deruy (2017), "Samuel Abrams, a professor at Columbia University [...] compared Finland's high marks on international education tests to those produced by other, similarly sized Nordic countries that are also relatively more homogenous and egalitarian than the United States. Those countries—Sweden, Denmark, and Norway—**score lower than Finland** and more in line with America" where the focus is on coding.

Because of its status as an early adopter and its focus on computational thinking, the results from Finland both in terms of its Programme for International Student Assessment (PISA) and other reports should be closely watched.

United Kingdom

The inspiration for the digital technologies curricula in the United States of America, Australia and New Zealand is the UK national computing program (UK Government, 2013). The program focuses on coding and computational thinking but has removed ICT general capabilities and digital literacy:

"It explores the deep links with mathematics, science and design and technology, and provides insights into both natural and artificial systems. The core of computing is computer science, in which pupils are taught the principles of information and computation, how digital systems work and how to put this knowledge to use through programming. Building on this knowledge and understanding, pupils are equipped to use information technology to create programs, systems and a range of content. Computing also ensures that pupils become digitally literate – able to use, and express themselves and develop their ideas through, information and communication technology – at a level suitable for the future workplace and as active participants in a digital world" (UK Government, 2013).

The program has a strong focus on computer science, which appears to have drawn inspiration from an influential paper published by the Royal Society (2012). The paper laments the vicious cycle of:

1. Shortage of teachers with sufficient subject knowledge
2. ICT lessons delivered by non-specialists

3. ICT curriculum delivered as digital literacy
4. ICT perceived as being low level skills
5. Decisions are made based on negative impressions
6. Few people study degrees which result in rigorous Computer Science qualifications, which again leads to a shortage of teachers with sufficient subject knowledge

The Royal Society (2012, pp. 8-11) makes 11 recommendations, most notably:

A. Clarity of terminology: the term ICT as a brand should be reviewed and the possibility considered of disaggregating this into clearly defined concepts.
B. Teacher recruitment: the government should set targets for the number of computer science and information technology specialist teachers, and monitor recruitment against these targets in order to allow all schools to deliver a rigorous curriculum.
C. Industry support: government departments with responsibility for education in the UK should seek industry support to extend existing funding in this area
D. Resources: suitable technical resources should be available in all schools to support the teaching of computer science and information technology.
E. Education of teachers: in order to redress the imbalance between academic and vocational qualifications in this area – and to ensure that all qualifications are of value to those who take them – the departments for education across the UK should encourage awarding organisations to review their current provision and develop Key Stage 4 qualifications in computer science in consultation with the UK Forum, universities and employers.
F. Professional standards: The UK Forum should advise awarding organisations on appropriate assessment methods for qualifications in digital literacy, information technology and computer science.
G. Teacher professional development: the UK Forum should put in place a framework to support non-formal learning in computer science and to support teachers.

In Australia the focus so far has been on the development of a National Digital Technologies Curriculum, led by ACARA, the curation of resources (Digital Technologies Hub, led by Education Services Australia), some development of local resources by various players and the development of training courses to upskill in-service teachers. There are some industry-supported programs, mainly developed by industry itself. However it appears that government has not yet set targets for the number of computer science and information technology specialist teachers and Australian universities do not yet graduate computer science specialist teachers in large scale to meet the demands of the schools.

The UK national computing program has been criticised for being too abstract and trying to produce mainly programmers (Everett, 2018) whilst neglecting soft skills and lacking a recognition that most students will not ever work as programmers. This is not necessarily a fair criticism. Most students will not work as mathematicians or scientists or linguists, yet mathematics, science, and languages are compulsory at school. The purpose of school curricula is the development of cognitive skills and not only the production of work-ready graduates.

South Korea

The South Korea Software Education program is focused on developing computational thinking, coding skills and creative expression through software and is due to be rolled out at all levels of education. Primary and lower secondary face the most dramatic change because the new program is mandatory at these levels as of 2018. Training for primary teachers is especially critical to the success of this policy since they teach all subjects and there are no separate IT/computer teachers. By 2018 60,000 primary school teachers (30% of the total) will receive specialized training in software education and 6,000 of that trainee population will receive in-depth training. In addition 1,800 middle school teachers who are certified to teach IT/computing will receive additional training on software education.

Summary

Many Organisation for Economic Co-operation and Development (OECD) countries have decided to move ahead with the implementation of digital technologies at primary and/or secondary level. The underlying intents vary and so do the approaches. The Australian Curriculum: Digital Technologies has been inspired by the UK national computing program which commenced in 2013. Many of the recommendations from the Royal Society (2012) are applicable to Australia and go beyond what has been implemented in Australia to-date, in particular with regards to the above-listed recommendations B, C, E, F, and G. It will be interesting to study the results from the different approaches chosen by the UK, Estonia and Finland:

- UK: coding
- Finland: computational thinking
- Estonia: a combination of ICT general capabilities, design and technologies, and digital technologies

In terms of the teacher professional development we note the interesting large-scale approaches of the UK Computing at School (CAS) program and the related efforts in South Korea.

Part 3: Implications for teaching and learning

Implications for teaching and teacher professional development

Teacher training and professional development are crucial to the success of any new curriculum including digital technologies. Digital technologies is very different from most other Science, Technology, Engineering and Mathematics (STEM) subjects due to the relative newness of its knowledge base. Most science and mathematics content taught at school is over 50 years old, and much of it is several centuries old. Modern calculus, probably the most advanced form of mathematics taught at Australian high schools, dates back to the late 17th century. Curriculum changes in these subjects have a significant lag time and teachers in these fields acquire most of the content knowledge required for a successful forty-year teaching career at university.

In contrast, digital technologies is an emerging field. Modern computing has its origins in the 1930s, and the concept of general-purpose information processing machines and their application is still in development.

Most teachers remain unaware that the fundamental computing concepts used today have changed little since the 1930s. The focus on these original concepts, rather than on the application of changeable technologies, tools or operating systems is a significant credit to the Australian Curriculum: Digital Technologies. It is therefore beneficial for teachers to have a firm grasp of computational thinking concepts.

The professional development needs of digital technologies teachers are very different from the requirements of mathematics teachers, especially as many digital technologies teachers are self-taught. So while the former require significant professional development in terms of core concepts, the latter require hardly any. However, comparable best practice ways of delivering content can change and are reviewed regularly.

According to Bocconi et al. (2016, p. 7), "there is broad consensus among experts and practitioners that the introduction of computational thinking in school curricula at all levels is creating demand for large-scale in-service continuous professional development. Training activities are often designed specifically to be hands-on so that teachers can more easily transfer their new skills to their classrooms. Grassroots efforts are also contributing to teachers' professional development." This observation is confirmed by the Australian Computing Academy's own anecdotal evidence from the field.

In this section, we provide an overview of teacher coding and computational thinking training activities in Australia, the US and the UK, with a particular focus on England. The US was chosen due to the drive of their private-sector in supporting the professional development of teachers. We selected the UK because of the vanguard nature of the UK curriculum and the strong government involvement. Between the US and the UK, Australia appears to be taking a position closer to that of the UK with creation of a national curriculum, publicly-funded projects, and some private sector investment.

According to Bocconi et al., (2016, p. 13), the "[...] teacher training opportunities discussed in the literature largely focus on pedagogical aspects rather than technological skills. Most training seems to be designed for all subject teachers, sometimes with a particular focus on STEM teachers. Pedagogical approaches addressed include storytelling, problem solving, deductive and inductive pedagogies with a

focus on computational models and simulation."

All the experts interviewed by Bocconi et al. (2016) discuss or at least mention teacher training. They propose:

1. A multi-perspective approach in preparing teachers,
    A. Specialisation of teachers for upper secondary level;
    B. An array of competencies of teachers at primary school level.
3. Professionalisation of teachers who are asked to impart coding and computational thinking lessons.
4. Making room in teacher education programs for computer science specialists who can teach at least basic notions of computational thinking, related to STEM subjects.

Professional development of teachers in computational thinking in Australia

In Australia, the introduction of the Australian Curriculum: Digital Technologies has led to a surge in demand for in-service teacher training in this area.

Only a relatively small number of digital technologies teachers in Australia have a formal university-level background in digital technologies, or, more generally, computer science as it is called at university. According to estimates, Australian universities graduate in the order of 100 digital technologies specialist teachers annually. This means that a vast majority of Australian digital technologies teachers are teaching out of field, though some of them have undergone some teacher professional development. A typical two-day teacher professional development course consists of twelve hours of content. Whilst better than nothing, it is small compared to a cumulative total of months (of years) of in-depth training at university. Training efforts in Australia, therefore, focus on five aspects that we elaborate in the following section:

1. Unpacking the Australian Curriculum: Digital Technologies
2. Resources
3. Train the trainer
4. Online courses
5. Supporting disadvantaged schools

Unpacking the Australian Curriculum: Digital Technologies

The Australian Curriculum: Digital Technologies was endorsed in September 2015, but it is an entirely new field of study for many schools and teachers. To counter this problem, a number of organisations have developed training resources or run training sessions to assist teachers with unpacking the new curriculum.

Lockwood and Mooney (2017) summarise their findings about ways in which "[...] teacher's enthusiasm for, knowledge of and ability to teach computational thinking and computer science in their classrooms can be improved/increased. Most popular seem to be day-long workshops and workshops that are heavily practical in nature. The ideas, tools and lessons that are given during these workshops seem to give teachers a greater understanding of what computational thinking is and how it can be useful for their students whilst also giving them very practical ways to implement this in a variety of contexts. [...] Interestingly it seems that one significant barrier to computer science and computational thinking in education is teachers and educator's **misconceptions** about what these are. One advantage of having teachers attend these training days and workshops is that these misconceptions and misunderstandings can be corrected, which is successfully done in most of the described studies. It can also be seen from these papers that teacher's willingness and interest in teaching computer science/computational thinking is vital in its implementation in both primary, secondary and tertiary education."

Some of the typical misconceptions that the authors have encountered in the field are that digital technologies is only about coding, photo editing, stop-motion animations or about using 'educational' apps, usually in a form of edutainment.

The Australian Computing Academy (ACA) delivers federally-funded professional learning workshops to show teachers how to interpret the Australian Curriculum: Digital Technologies. The Australian Computing Academy also sponsors training delivered by the state Computer Education Groups (CEGs), several of which run workshops unpacking the curriculum.

The workshops contain both coding and computational thinking activities. This includes computational thinking concepts such as representation and abstraction that are derived from human communication and then applied and developed further in the context of computing. The estimated balance between computational thinking and coding contents is approximately 60:40. The ACA found that deriving digital technologies concepts from everyday life makes it easier for teachers to make connections with their students, and apply their prior knowledge to teaching the subject. Examples include a human sandwich making robot (played by the teacher) that is programmed by the students in a clear and unambiguous way to make a sandwich.

Resources

Teachers are in need of high-quality resources to support the implementation of the Australian Curriculum: Digital Technologies. While the internet provides easy access to an array of resources it can be difficult for teachers to find those most relevant to, and linked with the curriculum.

To address this in part, the Digital Technologies Hub was created under the Australian Government Department of Education funding to support the Australian Curriculum: Digital Technologies. It is a repository of resources drawn from across the internet. Each resource is tagged to relevant areas of the Australian Curriculum: Digital Technologies.

The Computer Science Education Research group (CSER) based at the University of Adelaide maintains a lending library of digital technologies resources. Teachers can borrow a selection of digital technologies educational equipment that are accompanied by lesson plans, based on the CSER Massive Open Online Courses (MOOCs) (see below), designed for different age groups and mapped to relevant content descriptors in the Australian Curriculum.

The ACA develops online digital technologies challenges for coding and computational thinking (Australian Computing Academy Website, 2018). The focus of the digital technologies challenges is computational thinking, with coding being the vehicle to deliver computational thinking concepts and to keep the activities engaging and practical.

Train the trainer

An alternative approach to training the teachers directly is to train a group of lead teachers who then train other teachers. In 2018, the Western Australian Department of Education established Teachers Can Code (TCC) in partnership with the ACA to develop, design and deliver a lead teacher training program, consisting of multimodal, face-to-face and online professional learning and resources covering curriculum insights, as well as training in computational thinking and coding.

The objectives of the TCC program are to build the capacity of up to 100 lead teachers, in either Years 3-6 or Years 7-10. The program aims to build capacity in the lead teachers to deliver TCC modules to schools and networks and build expertise in the most technically challenging parts of the digital technologies curriculum, especially coding.

The TCC program develops eight primary and secondary professional learning modules, aimed at supporting teachers to improve their own coding skills (Department of Education: Western Australian Curriculum Support, 2018).

TCC has established a reporting regime in which teachers are regularly asked about their confidence, capability, engagement and other aspects of their teaching. This data informs the development of the project resources and delivery methods.

Train the trainer models have also been implemented by some of the state CEGs. A particular case is the ICT Educators NSW (ICTENSW) Regional Champions Scholarship program that funded regional teachers to travel to Sydney for training in content and extension training in how to run workshops, ICTENSW also runs regional workshops.

Online courses

CSER has developed online courses designed to support Australian teachers with implementing the Australian Curriculum: Digital Technologies. These are free online courses that provide teachers with background knowledge about concepts and topics in the curriculum, as well as practical examples that can be tried in the classroom. CSER brings together a collection of existing, and purpose-built resources. At present, CSER offers four MOOCs: F-6 Foundations, F-6 Extended, 7-8 Next Steps and 9-10: Explore (University of Adelaide, MOOCs). The MOOCs cover both coding and computational thinking.

Supporting disadvantaged schools

The Digital Technologies in Focus project operates at the intersection of teacher professional development and equity. The project is a part of the Australian Government's National Innovation and Science Agenda. ACARA has been funded to support the implementation of the Australian Curriculum: Digital Technologies in some of Australia's most disadvantaged schools across all states and territories. One hundred and sixty schools with a low Index of Community Socio-Educational Advantage (ICSEA) participate in the project. From July 2017 leaders and teachers from these schools take part in workshops to support the implementation of digital technologies subjects within their schools. Nine digital technologies specialists (also known as curriculum officers) support clusters of schools, providing digital technologies expertise to primary and secondary school teachers. The project develops tailored plans for each school and stakeholder networks comprising schools, industry, universities, teacher associations and others.

Professional development of teachers in coding and computational thinking in the US

The US has a highly decentralised K-12 education system. Public schools belong to school districts, which are governed by school boards. Each district is set up as a legally separate body corporate.

Hence, the US K-12 education system's fragmentation is considerably higher relative to Australia where ACARA is tasked with the development of national curricula that are subsequently implemented (sometimes with adaptations) by all schools. In contrast, there is no national US curriculum and the state governments only set standards and mandate standardised tests.

Teacher professional development for ICT and Digital Technologies is driven mainly by the US National Science Foundation (NSF) and the private sector.

The NSF-financed CSK10 program (2010-2016) trained 10,000 upper secondary computer science teachers. Private-sector grassroots efforts are also contributing to teachers' professional development. For example, Code.org trained about 30,000 teachers in US compulsory education over the last three years, organizing professional workshops and holding conferences for teachers and teacher trainers.

Professional development of teachers in coding and computational thinking in England

According to Jeannette Wing (Loble, Creenaune, & Hayes, 2017, p. 131), the US takes its lead from the UK: "The UK, through their Computing at School initiative, has introduced computing at all levels. It is a very courageous effort. The UK is my exemplar.

I hope countries around the globe look to the UK as a leader and learn from them as they push the frontiers of education in computer science."

According to Bocconi et al. (2016), in England, there are approximately 24,000 schools, including 16,800 primary schools, 3,400 secondary schools and 2,400 independent schools (primary and secondary). There are more than 500,000 teachers in compulsory education. In secondary schools the new 'computing' subject is assigned to the 14,000 existing ICT teachers who have to be up-skilled. In primary school there are around 200,000 teachers who currently teach all subjects, a load which will now also include computing.

The teacher training task in England is being addressed by CAS, a grassroots initiative with financial support from the Department for Education. CAS has established the Network of Teaching Excellence (NoE) in Computer Science for supporting, training, and equipping teachers as they implement the computing programs of study in their classroom. The NoE's plan is to recruit 600 master teachers each supporting forty local schools by designing and running not-for-profit continuing professional development (CPD) activities for those schools.

"The NoE has selected and trained Master Teachers, i.e. experienced classroom teachers with a passion for the subject; enthusiasm, energy, and a desire to support others. Master Teachers, with the support of their Head Teacher, are expected to dedicate one afternoon a week to train other teachers in their area. We develop some learning material centrally (such as QuickStart Computing), but is up to the master teacher how to run the training. Now, we have 350 Master Teachers active in the NoE. Starting from September 2015 we have introduced 10 Regional Centres based in Universities, involving either the Computer Science department or the School of Education and in some cases both, working with the CAS Master Teachers in their area to promote and support relevant teacher engagement and CPD activities" (Bocconi et al., 2016, p. 43). NoE's master teacher model resembles the Western Australian Government's approach for TCC (see above).

"While several MOOCs have been developed, a face-to-face component of teacher training is still relevant. A recent survey of over 900 in-service teachers in England concluded that face-to-face events and training, paired with an online community, are considered to be particularly effective in addressing their needs in content knowledge and pedagogical content knowledge related to computational thinking" (Bocconi et al., 2016, p. 42).

Lessons for Australia

Teacher professional development for in-service teachers in Australia would benefit from embracing a focused, accredited and large-scale initiative similar to the UK's CAS program with specific focus on teaching excellence in coding and computational thinking.

This could be enhanced by a public-private partnership between federal and state governments and the private sector. Currently this relationship is weak relative to the US. WA's TCC train the trainer initiative is also shaping up as a potential blueprint that warrants closer inspection by the other states and territories.

In parallel, Australian universities could support the training of more specialist digital technologies teachers, in part by treating coding and computational thinking as a core subject area, just like literacy and numeracy. This approach would ensure that graduate teachers are well versed in coding and computational thinking and can make links to other subject areas. It is likely this approach would reduce the need for teacher professional development intervention for in-service teachers.

Implications for student learning outcomes

Pedagogy

The development of pedagogy in computer science education lags behind that of other subjects. In contrast to computer science, mathematics has been taught at schools for centuries and there is broad consensus about teaching key concepts at different year levels, taking into account the changing cognitive capabilities as students age. With respect to computing and computational thinking, this consensus has not yet been developed. As a result, it is possible to either underwhelm or overwhelm students with content that they are either too old or too young for, leading to disenchantment with the subject. Jeannette Wing states that she strongly believes that it is important to do more research in this area (Loble, Creenaune & Hayes, 2017).

The experts who were interviewed by Bocconi et al. (2016, p. 36) "agreed that multiple pathways to computational thinking should be used throughout compulsory education. In particular, Mitchel Resnick and Joke Voogt pointed out the benefits of providing students with the opportunity to design, create and experiment in real-life situations, and other areas they care about. A number of interviewees questioned the current uptake of coding. Judith Gal-Ezer, for instance, points to the fact that over-reliance on coding might give pupils a false impression of what computational thinking is. Simon Peyton Jones discussed the need to undertake research to evaluate the effectiveness of different approaches."

If a student were to always code as directed and were never allowed to experiment with algorithmic development, and thus a range of computational thinking skills, then this would be similar to someone only pushing the buttons of a machine in a repetitive fashion without providing any direction as to the function of the machine or to the design of its output. Instead, the objective of digital technologies teaching is to develop higher-order thinking skills and not to produce a droid army of programmers.

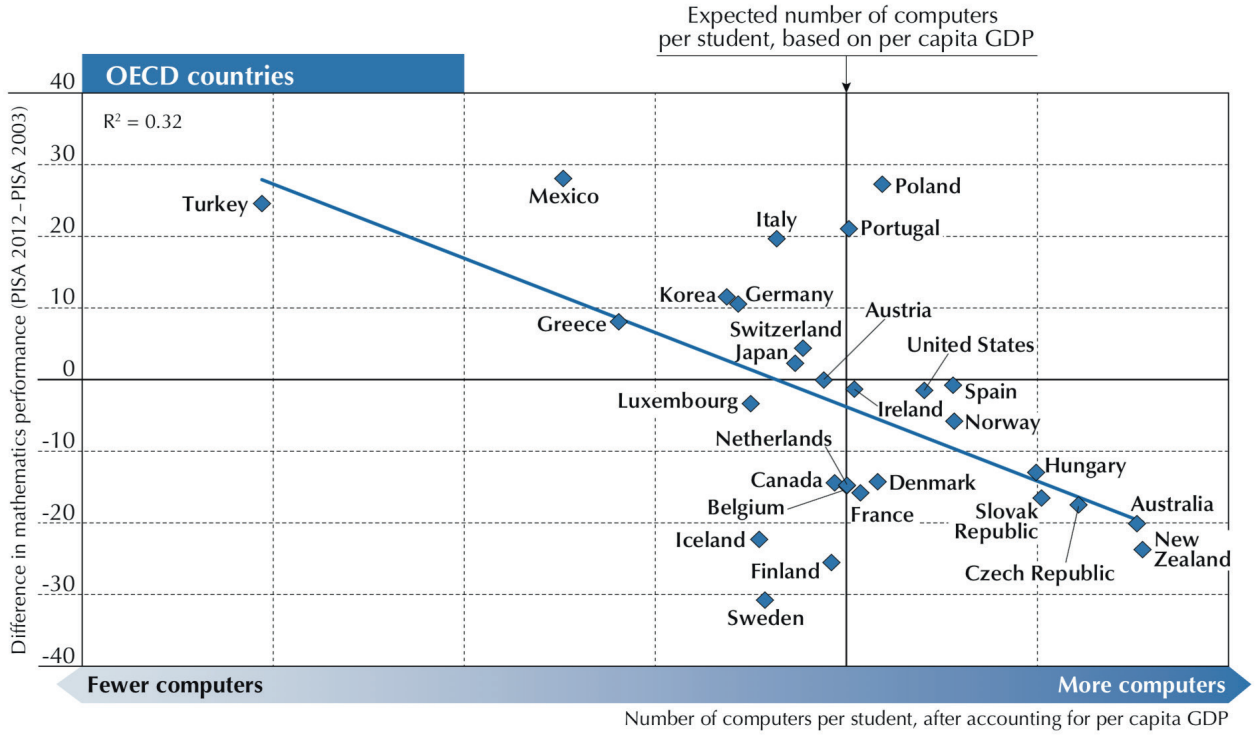
To appreciate the importance of this point one has to consider the creativity and skill-sets necessary for turning an abstract idea into a tangible, useful product. Steve Jobs is attributed the quote "real artists ship!" in which he refers both to the artistry and creativity of the software engineering process as well as to an outcome that does something.

The role of computers and laptops

Computers play an increasing role in the classroom. In addition to their historic role as instruments to do coding, they have assumed a broader role as a vehicle for student administration and as a delivery channel for learning content that is not related to computer science education. This leads Jeannette Wing to ask the question: "[...] how best and when should we use 'the computer' in the classroom to teach and reinforce computational thinking concepts? Here **my concern is throwing technology into the classroom and thinking the students are going to learn anything,** let alone computing. We need further research on how computing technology can be used effectively for learning and not hinder the learning process. We also need research on how such technology can help reinforce the learning of computational thinking specifically" (Loble, Creenaune & Hayes, 2017, p. 131).

The OECD released the results of a landmark study in 2015 about students, computers and learning (OECD, 2015). The authors conclude that **"where computers are used in the classroom, their impact on student performance is mixed at best.** Students who use computers moderately at school tend to have somewhat better learning outcomes than students who use computers rarely. **But students who use computers very frequently at school do a lot worse in most learning outcomes,** even after accounting for social background and student demographics" (OECD, 2015, p. 15).

"The countries with the greatest integration of ICT in schools are Australia, Denmark, the Netherlands and Norway. Rapid increases in the share of students doing school work on computers can often be related to large-scale laptop-acquisition programs, such as those observed in Australia, Chile, Greece, New Zealand, Sweden and Uruguay" (OECD, 2015, p. 50). Despite the rapid increase of computers in Australian classrooms between 2003 and 2012, student mathematics performance in PISA decreased during the same period (OECD, 2015).

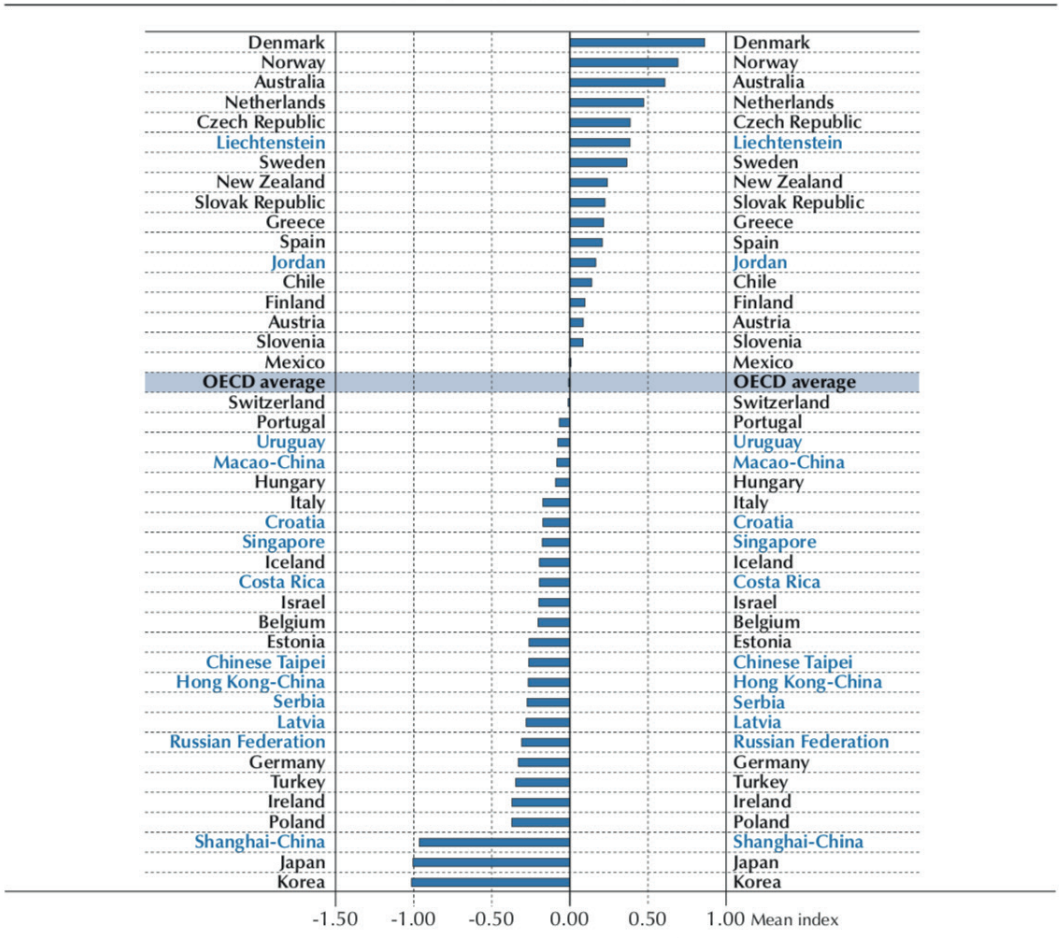


Mathematics performance and number of computers in school (OECD, 2015).

Licensed under: Creative Commons Attribution 3.0

Correlation does not constitute causation, and isolating factors that contribute to an effect is difficult in education where curricula, syllabi, teacher training and confidence, pedagogy and cultural norms intersect. If anything is definite, it is that the effects - both positive and negative - of computers and their usage for education inside and outside of the classrooms require further investigation.

It is generally believed that more technology in the classroom is beneficial to the learning outcomes of students and that schools and students from areas with a high socio-educational advantage (ICSEA) value would perform better academically because of the high-ICSEA students' higher access to technologies. Based on the OECD report (2015), high-ICSEA students that have high access to computers might be at a disadvantage relative to lower-ICSEA students with less computer access.

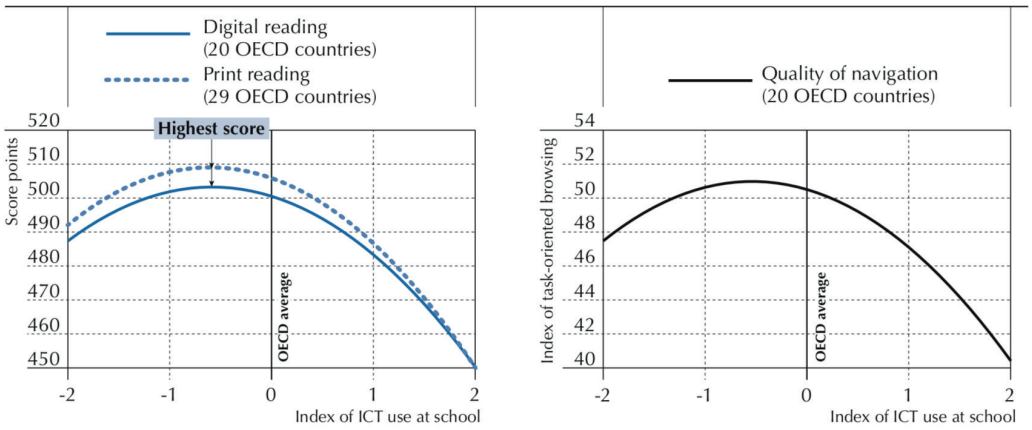


Countries and economies are ranked in descending order of the mean index of ICT use at school.
Source: OECD, PISA 2012 Database, Table 2.2.
StatLink http://dx.doi.org/10.1787/888933252700

Index of ICT use at school (OECD, 2015). Licensed under: Creative Commons Attribution 3.0

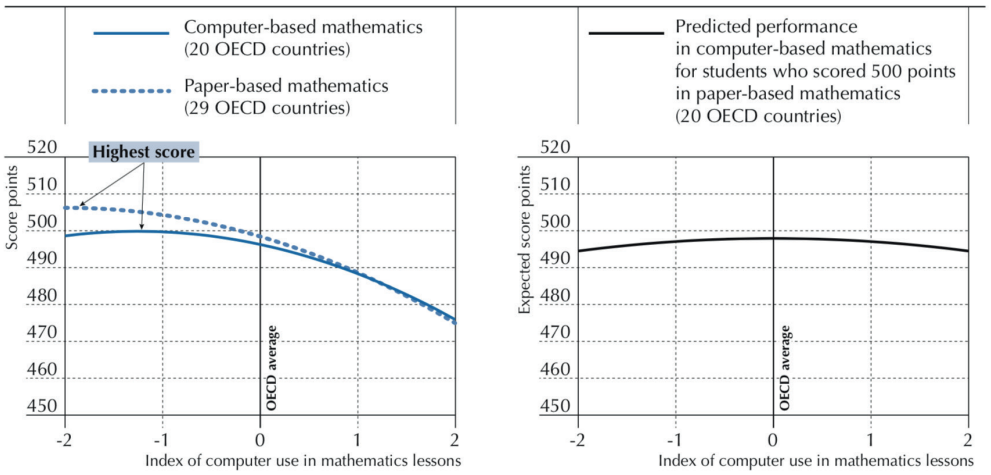
Australia's mean index of ICT use at school is 0.6 points higher than the OECD average. Australia therefore finds itself on the high side of computer use in the classroom in relation to most other OECD countries.

"Overall, the most frequent pattern that emerges in PISA data when computer use is related to students' skills is a weak or sometimes negative association between investment in ICT use and performance. While the correlational nature of this finding makes it difficult to draw guidance for policy from it, the finding is remarkably similar to the emerging consensus in the research literature, based on studies that use more rigorously designed evaluations. [...] Overall, the evidence from PISA, as well as from more rigorously designed evaluations, suggests that solely increasing access to computers for students, at home or at school, is unlikely to result in significant improvements in education outcomes"(OECD, 2015, p. 164).



Notes: The lines represent the predicted values of the respective outcome variable, at varying levels of the *index of ICT use at school,* for students with a value of zero on the *PISA index of economic, social and cultural status* (ESCS), in schools where the average value of ESCS is zero.
Quality of navigation refers to students' ability to plan and regulate their navigation behaviour on line; this is measured by the *index of task-oriented browsing* (see Chapter 4).
Source: OECD, PISA 2012 Database, Table 6.2.
StatLink http://dx.doi.org/10.1787/888933253280

Students' skills in reading, by index of ICT use at school (OECD, 2015). Licensed under: Creative Commons Attribution 3.0 OECD average relationship, after accounting for the socio-economic status of students and schools.



Notes: The lines represent the predicted values of the respective outcome variable, at varying levels of the *index of computer use in mathematics lessons*, for students with a value of zero on the *PISA index of economic, social and cultural status* (ESCS), in schools where the average value of ESCS is zero.
Source: OECD, PISA 2012 Database, Table 6.4.
StatLink http://dx.doi.org/10.1787/888933253302

OECD average relationship, after accounting for the socio-economic status of students and schools

Performance in computer-based and paper-based mathematics, by index of computer use in mathematics lessons (OECD, 2015). Licensed under: Creative Commons Attribution 3.0

Measuring computational thinking

Lockwood & Mooney (2017) investigated the methods and tools that exist to test students' computational thinking abilities and improvement. In their paper, they list several tools, quizzes, and concepts and discuss the issue of assessing computational thinking without a dependency on coding. Lockwood and Mooney come to the conclusion that "overall work in testing for computational thinking is in its infancy. Most of the examples...are in the early stages of development. Tools do exist such as Dr. Scratch and the tools developed by the Scalable Design Group but there is a need for more research into this area. Other forms of test are based on problem-solving and analytical thinking tests. Whilst these are potentially beneficial, if computational thinking is to become a common skill taught in schools and universities then built-for-purpose tools and assessments might be required" (Lockwood & Mooney, 2017, p. 15).

Case Study: Measuring computational thinking skills

The National Foundation for Education Research (NFER) in the UK has published the results (Straw, Bamford, & Styles, 2017) of a randomized controlled trial and process evaluation of Code Clubs – a UK network of after-school clubs where children aged 9-11 learn to program by making games, animations, websites, and applications. Code Club UK produces material and projects that support the teaching of Scratch, HTML/CSS, and Python. The clubs, which are extracurricular in their nature, are supported by volunteers, and usually run for one hour a week after school during term time.

The evaluation assessed the impact of Code Clubs on Year 5 students' computational thinking, programming skills, and attitudes toward computers and coding. Twenty-one schools in the UK took part in the trial which used a student-randomized design to compare student outcomes in the intervention and control groups. Intervention group students attended Code Club during the 2015/16 academic year, while control group students continued as they would do normally. The primary outcome measure was the Bebras Computational Thinking Challenge, which has been running in the UK since 2013. Bebras UK is a 40 min online quiz of 15 questions, grouped in 3 categories (easy/medium/hard).

The results of the evaluation showed that attending Code Club for a year did not impact students' computational thinking any more than might have occurred anyway, but did significantly improve their coding skills in Scratch, HTML/CSS, and Python. This was true even when control children learned Scratch as part of the computing curriculum in school.

Code Club students reported increased usage of all three programming languages – and of computers generally. However, the evaluation data suggests that attending Code Club for a year does not affect how students view their abilities in a range of transferable skills, such as following instructions, problem-solving, learning about new things and working with others.

Reflection on Methodology

This study leads to two important questions:

Is Bebras the right metric to measure computational thinking?

Is coding at UK Code Clubs taught in a way that is mechanical, non-creative and therefore removed from core concepts that can be found in computational thinking?

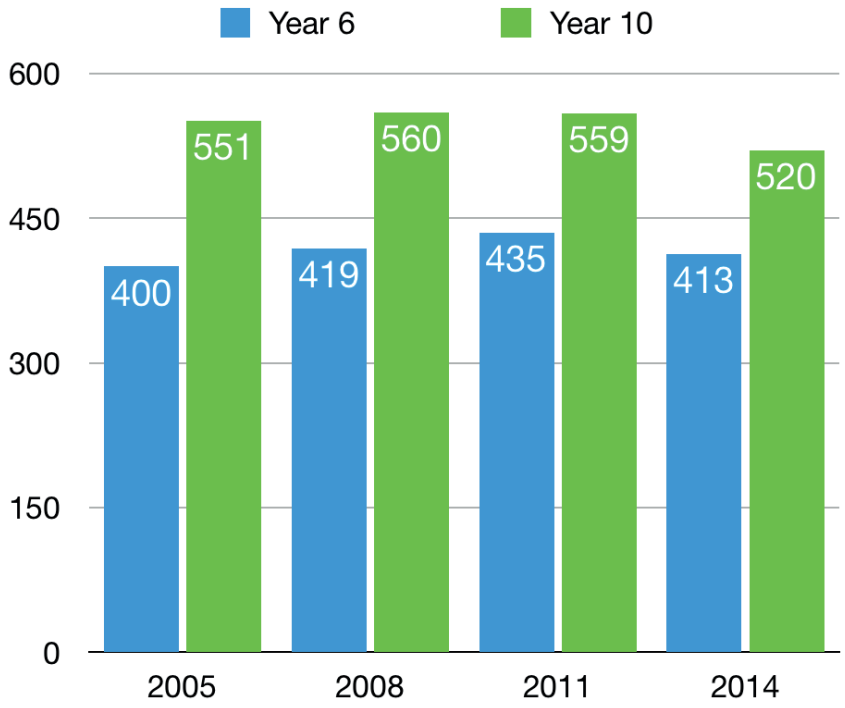
Bebras started in 2003 in Lithuania and was founded by Prof. Valentina Dagienne from the University of Vilnius. Nowadays, representatives from over 35 participating countries annually produce a set of non-coding computational thinking questions that undergo a quality management process and a one week workshop before they are implemented for deployment mostly through Bebras online servers. Whilst every effort is made to map the question to computational thinking concepts, there is not a single canonical framework at the foundation of the question development. The earlier mentioned lack of an agreed definition of computational thinking is an issue. Every country is at liberty to choose from the annual question catalogue for their national Bebras week (or fortnight). Since question sets

vary annually and between countries and national Bebras organisers are free to (1) choose how many points are allocated per question (2) set the age groups (3) set the duration and time allocations for a national Bebras contest. It is therefore extremely difficult to draw conclusions as to the (1) relevance of computational thinking concepts, (2) country by country comparisons and (3) long-term trends. From this perspective, we argue that whilst Bebras is a useful tool in the classroom, it should, at this point in time, not be considered as a metric against which other computational thinking activities or resources be measured and therefore no policy decisions should be taken that are based on a Bebras metric alone.

In Australia, ACARA has published the National Assessment Program – ICT (NAP-ICT) Literacy report on the ICT proficiency of Australian year 6 and 10 students in 2005, 2008, 2011, 2014 and most recently in late 2018.

The 2014 report identified a decline in the mean scale scores for years 6 and 10 from 2011 to 2014. The Year 6 cohort's mean score has dropped below 2008 levels and the Year 10 cohort's mean score is below 2005 levels.

The NAP-ICT of 2014 did not measure coding and computational thinking skills. Its focus is on "The ability of individuals to use ICT appropriately to access, manage and evaluate information, develop new understandings, and communicate with others in order to participate effectively in society" (ACARA, 2015). In contrast to OECD (2015), NAP-ICT (2014) did not investigate the impact of ICT on students' literacy and numeracy skills.



NAP-ICT mean performance score (ACARA, 2015, p XXII). Graphics: Australian Computing Academy

Interpretation

A firm understanding of the correlation of the increasing computer use and a decline in academic outcomes has not yet been scientifically established. We can presently only put forward potential factors that require further investigation:

1. Distraction: Students use their computers not only for focused academic work, but instead for other things, such as browsing the web, playing games, or social media.
2. Task suitability: computers may not always be the best instrument for a task. Using a computer can consume significant cognitive effort on the side of the student, which would otherwise have been available to solve the task. Example: Writing a formula in a formula editor is much harder and slower than writing it by hand on paper. Here, the student could be focussing on mastering the tool, rather than thinking about the mathematics.
3. Unavailability: computers are not always functioning reliably. In a class of 20-30 students, some computers are bound to malfunction, experience network issues, block content, prevent login, etc. In our experience, a teacher can lose 5-10 minutes from each period just because of computer issues. Assuming 5 lost minutes per lesson x 4 lessons/day x 200 school days = 4,000 minutes or 67 hours of lost time per school year.
4. Varied, but incorrect information: whilst traditional textbooks are extensively reviewed, computer-use encourages the use of a wide range of resources from the internet that have not been rigorously reviewed.
5. Superficiality: reading online is different from reading a book. Online reading is more about skimming information. As a result, the learning may be less permanent (Alexander & Trakhman, 2017).
6. Inability to connect: Continual access to the internet may lead teachers and students to believe knowing where to find information is more important than, or the same as, knowing essential facts. The latter is necessary of students are to make connections, draw conclusions, and therefore develop new insights.
7. Rise of mobile technology: the authors of the NAP-ICT report (ACARA, 2015) suggest the testing regime, which focusses on laptop use, has been superceded by tablets that have a drag and swipe touch user interface.
8. ACARA (2015, p. 114) further suggest other potential influences:
    A. Changes in teaching and learning with ICT have resulted in less emphasis being placed on the teaching of skills associated with ICT literacy.
    B. “Development of ICT literacy competencies has been taken for granted in Australia where the level of access to ICT in schooling is extremely high.”
    C. “The emergence of mobile computing technology devices has led to increased emphases in teaching and learning on different skills (such as those associated with online communication).”

Case Study: Teaching coding and computational thinking without the need for computer hardware

Computer Science Unplugged (CS Unplugged) is a collection of resources that teaches Computer Science without the need for computing hardware. The content instead favours games and puzzles that use cards, string, crayons and lots of running around. CS Unplugged, therefore, constitutes a low barrier of entry resource that can be used by all schools independent of their budget.

Bell and Vahrenhold conducted an analysis of the effectiveness of CS Unplugged in their paper CS Unplugged—How Is It Used, and Does It Work? (Böckenhauer, Komm, & Unger, 2018). They found that "surprisingly few empirical studies about the use of CS Unplugged activities in a regular classroom setting have been conducted" (Böckenhauer, Komm, & Unger, 2018, p. 504). As part of the review of the studies, it was identified that:

- CS Unplugged activities changed middle-school students' view of computer science toward mathematical thinking. However, these students still considered computers essential to computer science and, were found to become less attracted to the field. CS Unplugged activities are only loosely related to "central concepts" in computer science and not explicitly linked to students' prior knowledge. This "is a direct consequence of the activities having been developed for very young children who—by definition—cannot be expected to have much prior knowledge in computer science or mathematics to build upon" (Böckenhauer, Komm & Unger, 2018, p. 504)."The union of the derived learning objectives did not fully cover all dimensions of Bloom's revised taxonomy. However, [...] CS Unplugged activities indeed address objectives well suited for outreach and for introducing new topics in class, thus showing the applicability of CS Unplugged beyond reasons of playfulness or creating intrigue" (Böckenhauer, Komm, & Unger, 2018, p. 504).
- Unplugged activities are a useful approach to teach programming where "an Unplugged explanation or walk-through is used before getting on the computer to help students understand the design of a computer program or language elements" (Böckenhauer, Komm, & Unger, 2018, p. 507). Interestingly, this is the way computer science was taught in the 1970s and 80s when the student to computer ratio was higher. Teachers had no other option but to teach CS in an unplugged way, often on a blackboard, with only the final step of the training conducted at the computer.

Bell and Vahrenhold conclude that, despite mixed evidence as to the impact of unplugged activities on student learning, the approach appears to be popular with teachers as a pedagogical approach. More research is needed to determine how best to use it effectively.

Key themes for future consideration

This research report has considered the evidence base for the teaching of computational thinking, including its relationship with coding, and reflected on some of the implications of this for schooling and school systems.

Our research raises a number of important themes that we believe the NSW Department of Education could consider:

1. To what extent do the Australian Curriculum and NSW Syllabi appropriately cover the teaching of computational thinking and coding?
2. Does the terminology related to coding and computational thinking concepts need to be more clearly defined?
3. What resources need to be in place to support the teaching of computational thinking and coding in NSW and to what extent do we already have them?
4. Would initial teacher training benefit from a greater focus on computational thinking and/or coding?
5. Do we have sufficient understanding of the extent to which students are achieving proficiency in computational thinking, algorithmic thinking and familiarity with a range of contemporary technologies?
6. Is more work needed to reach agreement on appropriate pedagogies and age-appropriate teaching and assessment approaches?
7. Are current learning continua/progressions of computational thinking or coding concepts adequate and informed by current research and theory?
8. Is coding an effective instrument to teach computational thinking? Are some approaches more successful and what do these involve?
9. What is the capacity for computational thinking to be developed in and integrated with other key learning areas such as maths and biology?
10. What do effective assessment and success metrics look like in judging the extent to which students are attaining proficiency in computational thinking?

Abbreviations

ACA – Australian Computing Academy

ACARA - Australian Curriculum, Assessment and Reporting Authority

CAS – Computing at School

CEG – Computer Education Group

CPD – Continuing Professional Development

CS Unplugged – Computer Science Unplugged

CSER – Computer Science Education Research

CSIRAC - Council for Scientific and Industrial Research Automatic Computer

DER - Digital Education Revolution

EU – European Union

ICT – Information and Communications Technology

ICTENSW – ICT Educators NSW

ICSEA – Index of Community Socio-Educational Advantage

IT – Information Technology

MONECS - Monash Educational Computer System

MOOC – Massive Open Online Course

NAPLAN – National Assessment Program Literacy and Numeracy

NAP-ICT - National Assessment Program-Information and Communication Technology

NESA – NSW Education Standards Authority

NoE – Network of Teaching Excellence

NSF – National Science Federation (US)

OECD - Organisation for Economic Co-operation and Development

PC – personal computer

PISA – Programme for International Student Assessment

STEM – Science, Technology, Engineering and Mathematics

TCC – Teachers Can Code

Bibliography

Académie des Sciences. (2013). L'enseignement de l'informatique en France - Il est urgent de ne plus attendre.

ACARA. Digital Technologies subject rationale. Retrieved from https://www.australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies/rationale/. Last accessed 16.7.2018

ACARA Glossary. Computational Thinking, Retrieved from https://www.australiancurriculum.edu.au/f-10-curriculum/technologies//Glossary/?term=Computational+thinking. Last accessed 16.7.2018

ACARA (2015). National Assessment Program – ICT Literacy, Years 6 & 10, Report 2014. Retrieved from https://www.nap.edu.au/_resources/D15_8761__NAP-ICT_2014_Public_Report_Final.pdf

Alexander, P., Trakhman, L. (2017). The enduring power of print for learning in a digital world. Retrieved from http://theconversation.com/the-enduring-power-of-print-for-learning-in-a-digital-world-84352

Australian Computing Academy Website (2018). Retrieved from https://aca.edu.au

Australian Government, Department of Education and Training (2017). Undergraduate Applications, Offers and Acceptances 2017 Report. Retrieved from https://docs.education.gov.au/system/files/doc/other/undergraduate_applications_offers_and_acceptances_2017.pdf

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., Engelhardt, K. (2016). Developing computational thinking in compulsory education – Implications for policy and practice; EUR 28295 EN; doi:10.2791/792158, Retrieved from http://publications.jrc.ec.europa.eu/repository/bitstream/JRC104188/jrc104188_computhinkreport.pdf

Böckenhauer, Komm & Unger (2018) Adventures Between Lower Bounds and Higher Altitudes. Springer, ISBN 978-3-319-98355-4

Computer Science Unplugged. Retrieved from https://csunplugged.org/en/. Last accessed 16.7.2018

DEEWR (2013). Digital Education Revolution Program Review. Retrieved from https://docs.education.gov.au/documents/digital-education-revolution-program-review

Department of Education – Procurement Solutions Directorate (2018). Request for Quote, Coding and computational thinking research report, Part A Department of Education and Training. Digital Education Revolution. Retrieved from https://docs.education.gov.au/category/deewr-program-group/digital-education-revolution

Department of Education: Western Australian Curriculum Support (2018). Teachers Can Code. Retrieved from http://det.wa.edu.au/curriculumsupport/detcms/school-support-programs/curriculum-support/news-items/expression-of-interest-lead-teachers-for-teachers-can-code-professional-learning-program.en

Deruy (2017). In Finland, Kids Learn Computer Science Without Computers. Retrieved from https://www.theatlantic.com/education/archive/2017/02/teaching-computer-science-without-computers/517548/

Digital Technologies Hub. Resources. Retrieved from http://www.digitaltechnologieshub.edu.au/search#/site-search?cnttype=resource

Digital Technologies in focus. Retrieved from http://www.acara.edu.au/curriculum/learning-areas-subjects/technologies/digital-technologies-in-focus-dtif-project

Everett, C. (2018). Is the UK's computing curriculum too focused on coding? Retrieved from https://www.computerweekly.com/feature/Is-the-UKs-computing-curriculum-too-focused-on-coding

Falkner, K. (2017). A Look At It And Engineering Enrolments In Australia – Updated! Retrieved from https://blogs.adelaide.edu.au/cser/2017/02/15/a-look-at-it-and-engineering-enrolments-in-australia-updated/

Gander, W., Petit, A., Berry, G., Demo, B., Vahrenhold, J., McGettrick, A., ... Drechsler, M. (2013). Informatics education: Europe cannot afford to miss the boat (p. 21). Joint Informatics Europe & ACM Europe Working Group on Informatics Education.

Gartner, Gartner Hype Cycle. Retrieved from https://www.gartner.com/technology/research/methodologies/hype-cycle.jsp Last accessed 18.7.2018

Guzdial, M. (2017). In Finland, Teaching Computer Science across the Curriculum, Retrieved from https://computinged.wordpress.com/2017/03/06/in-finland-teaching-computer-science-across-the-curriculum/

Grover, S. (2013). OPINION: Learning to Code Isn't Enough. Retrieved from https://www.edsurge.com/news/2013-05-28-opinion-learning-to-code-isn-t-enough

Information Technology Foundation for Education (2018). Technology education has reached majority of Estonian schools by support of the ProgeTiger program. Retrieved from https://www.hitsa.ee/about-us/news/technology-education-has-reached-majority-of-estonian-schools-by-support-of-the-progetiger-program

Lockwood, J., Mooney, A. (2017). Computational Thinking in Education: Where does it fit? A systematic literary review. Department of Computer Science, Maynooth University, Maynooth, Co. Kildare, Ireland, Retrieved from https://arxiv.org/pdf/1703.07659.pdf

Learning Environments research group (2015). Coding in school: Finland takes lead in Europe. Retrieved from https://legroup.aalto.fi/2015/11/coding-in-school-finland-takes-lead-in-europe/

Loble, L., Creenaune, T., Hayes, J. (2017) Future Frontiers. Education for an AI World. Melbourne University Press. A Conversation About Computational Thinking With Jeannette M Wing, pp 127-140

Mohaghegh, M, McCauley, M. (2016). Computational Thinking: The Skill Set of the 21st Century. International Journal of Computer Science and Information Technologies, Vol. 7 (3), 1524-1530

NSW Education Standards Authority, Digital Technologies And ICT Resources. Retrieved from http://educationstandards.nsw.edu.au/wps/portal/nesa/k-10/learning-areas/technologies/coding-across-the-curriculum/!ut/p/z1/pZCxDoIwEIYfqXfXUnDsILTQpIMi2MUwmSaKDsbnt-ngoIgY_-2S77vcf8yznvlxuIfjcAuXcTjFee_lQRgNwIEslCRA8R1UK-PqTCLrEkAKJWqBDeQFgtoK0WQlIFjJ_D--4T_6rkq-Q0l1Tq7NlvnwlQqW-TOAn1_fMZ-QZ4OC1rEBb7TFjSXr-Csw8eJZwLxtmHjStzOv5zamh2CCegAdpnQ4/#fn1. Last accessed 16.7.2018

OECD (2015). Students, Computers and Learning: Making the Connection, PISA, OECD Publishing, Paris, Retrieved from https://doi.org/10.1787/9789264239555-en

ProgeTiger Program 2015–2017 Report. Retrieved from https://www.hitsa.ee/it-education/educational-programmes/progetiger

Resnick et al. (2009) Scratch: Programming for All, Communications of the ACM, Vol. 52, No. 11. Retrieved from https://dl.acm.org/citation.cfm?id=1592779

Royal Society (2012). Shut down or restart? The way forward for computing in UK schools. Retrieved from https://royalsociety.org/~/media/education/computing-in-schools/2012-01-12-computing-in-schools.pdf

Sentance, S., Csizmadia, A. (2017). Computing in the curriculum: Challenges and strategies from a teacher's perspective. Education and Information Technologies, 22(2), 469-495, Retrieved from https://kclpure.kcl.ac.uk/portal/files/51589206/ComputingInTheCurriculum2016.pdf

Straw, S., Bamford, S., Styles, B. (2017). Randomised Controlled Trial and Process https://www.nfer.ac.uk/publications/CODE01/CODE01.pdf

Schulz, K., Fuda, B. (2018). Australian Computing Academy. Visual or Text Programming? Retrieved from https://blog.aca.edu.au/visual-or-text-programming-c75046312ff7

Tatnall, A., Davey, B. (2004). Streams in the History of Computer Education in Australia. History of Computing in Education. J. Impagliazzo and Lee, J. A. N. Assinippi Park, Massachusetts, Kluwer Academic Publishers / IFIP: 83-90. Retrieved from goo.gl/ae7jQq

uCube. Department of Education and Training, Retrieved from http://highereducationstatistics.education.gov.au, Last accessed 18.7.2018

University of Adelaide. Lending Library. Retrieved from https://csermoocs.adelaide.edu.au/library/ last accessed 16.7.2018

University of Adelaide. MOOCs (Massively Open Online Courses). Retrieved from https://csermoocs.adelaide.edu.au/moocs/ Last accessed 16.7.2018

Uk Government (2013). National curriculum in England: computing programs of study. Retrieved from https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study

Wilson, C., Sudol, L. A., Stephenson, C., & Stehlik, M. (2010). Running On Empty: The Failure to Teach K–12 Computer Science in the Digital Age. ACM&CSTA.

Wing, J. (2010) Computational Thinking: What and Why? Retrieved from https://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

ZDNet (2013) Australian government quietly ends laptops in schools program. Retrieved from https://www.zdnet.com/article/australian-government-quietly-ends-laptops-in-schools-program/

Report commissioned by the NSW Department of Education (futurefrontiers@det.nsw.edu.au).