Computational Thinking in the Australian Curriculum

# Liliana Ructtinger

*The author would like to acknowledge the contributions of her colleagues at the NSW Department of Education and Communities, Dr Robert Stevens, Dr Susan Harriman, Sandra McKee, Katherin Cartwright and Tanya Coli, and those of Dr James Curran, University of Sydney, and Dr Matt Bower, Macquarie University. The views expressed in this paper are those of the author and are not necessarily the views of her employer, the NSW Department of Education.*

# Introduction

The Melbourne Declaration for the Educational Goals for Young Australians states that the Australian government commits to working in collaboration with all school sectors to support young Australians in becoming: successful learners; confident and creative individuals, and; active and informed citizens ([MCEECDYA, 2008](#_ENREF_19)).

The Melbourne Declaration recognises that education needs to support the new demands that are being placed on young Australians in the 21st Century to enable them to compete in a globalised economy. The nature of jobs is changing and the knowledge and skills required to participate in the workforce is placing new kinds of demands on learners.

Successful learners have essential skills in literacy and numeracy, are creative and productive users of technology, and are able to think deeply and logically. There are complex environmental, social and economic problems that represent significant challenges requiring the ability to engage with scientific concepts and principles and to approach problem-solving in new and creative ways. These new challenges are closely tied to the rapid advances in technology that have characterised our recent history. The Australian Curriculum and Reporting Authority has identified seven general capabilities that are considered to be essential skills for twenty-first century learners ([ACARA, 2013](#_ENREF_1)). Analogous to the way in which the development of the printing press is associated with the rise of the “Three Rs”, the development of digital technologies is giving rise to a new type of reasoning: computational thinking ([Jeannette Wing, 2006](#_ENREF_29)). Recent attention has been drawn to the need to make coding and computational thinking mandatory in all Australian schools ([Foo, 2014](#_ENREF_14)).

This literature review will answer the following research questions:

1. What is computational thinking and how does it differ to what we’re doing already?
2. What is the rationale for it?
3. How can it be cultivated?
4. How can it be demonstrated by students and measured?
5. How does computational thinking relate to the Australian Curriculum?
6. How does computational thinking relate to the General Capabilities?

# Literature Review

## What is computational thinking and how does it differ to what we’re doing already?

The term Computational Thinking has been accredited to Jeanette Wing, who proposed a definition and then the argument that it represented a universal attitude and skill set that everyone should have the opportunity to learn ([Jeannette Wing, 2006](#_ENREF_29)). According to Wing, computational thinking is characterised by recursive thinking, reduction, transformation, simplicity, elegance, aesthetics, decomposition, representation, modelling, modularization, and pattern recognition, all of which is underpinned by its defining feature: abstraction ([Jeanette Wing, 2011](#_ENREF_30)).

There is a growing body of literature suggesting that the most useful way to conceptualise of computational thinking is as a thought processes involved in formulating solutions to problems that can be represented in a form that can be carried out by an information processing agent. Denning maintains that computational thinking can be defined as the ability to interpret the world as algorithmically controlled conversions of inputs to outputs ([Denning, 2009](#_ENREF_13)).

Others consider that thinking in terms of arrays is one of the most important features of computational thinking, which is not an everyday occurrence that we can assume students will have experience with ([Sanford, 2013](#_ENREF_24)).

James Curran, from the University of Sydney views computational thinking as the ability to turn problems into information problems that can be digitally manipulated.

The International Society for Technology in Education (ITSE) and the Computer Science Teachers Association (CSTA) of the United States worked with educators, industry professionals and researchers to define computational thinking as a problem solving process with the following characteristics:

* Formulating problems in a way that enables us to use a computer and other tools to help solve them.
* Logically organising and analysing data
* Representing data through abstractions such as models and simulations
* Automating solutions through algorithmic thinking
* Identifying, analysing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
* Generalising and transferring this problem solving process to a wide variety of problems ([International Society for Technology in Education & Computer Science Teachers Association, 2011](#_ENREF_16)).

There are several terms that are related but arguably distinct from computational thinking. A distinction has been drawn between *computer literacy* as the ability to use computer applications (such as word processors), *computer fluency* to describe a high level understanding of the workings of a computer system, and *computational thinking* as the most sophisticated skill level reflecting reasoning skills and computational techniques required to solve problems ([Perkovi´c, Settle, Hwang, & Jones, 2010](#_ENREF_22)).

Other authors argue that earlier references to *computational literacy,* which take computer literacy beyond the material tool by considering social and cognitive aspects, are closest to our current use of the term computational thinking ([Grover & Pea, 2013](#_ENREF_15)). The authors also consider terms such as *information literacy*, *digital literacy* and *procedural literacy* to fall short of computational thinking.

Computational thinking can therefore be understood as solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science. Computational thinking includes a range of mental tools that reflect the breadth of the field of computer science. Computational thinking is not limited to the field of computer science. Rather it can be understood as the application of the strategies, skills and approaches that are used by computer scientists to address information problems in a diverse range of fields that are characterised by their complexity.

## What is the rationale for it?

### Pervasiveness of computing devices and digital artefacts

It has been argued that computational thinking is an integral part of our everyday life, as a result of the pervasiveness of computing devices ([Wu & Richards, 2011](#_ENREF_32)). As such, there has been growing interest in the notion that computational thinking should be considered essential to education ([Day, 2011](#_ENREF_11)). If we are to pre-empt a divide between those that “use” and those that “create” technology, we need to ensure that all students have the potential to understand the underlying functionality of these devices and digital artefacts. Such understanding has implications for the power that students will have to make decisions about how these devices control the nature of interactions and content that is becoming a dominant feature of their lives. These skills have the potential to diffuse potential sources of inequity that may parallel access to information, security and opportunity. The power structures that determine access to information are shifting; students have the right to be able to consider how the increasing corporatisation of our digital interactions, particularly through the advent of social media, can bear on our lived experiences. These skills can be developed through the cultivation of computational thinking.

### Allowing students to see more solution opportunities and the reformulation of problems

The ability to think about problems in an abstract manner is essential to students’ ability to see more solution opportunities; thinking in this abstract manner, or computationally, is a way of accessing solutions that are usually outside a student’s normal area of expertise ([Ioannidou, Bennett, Repenning, Koh, & Baswapatna, 2011](#_ENREF_17)). Other authors argue that computational thinking is a prerequisite for solving complex problems. Buckley ([2012](#_ENREF_8)) suggests that as we now live in a knowledge era where there is an ever-increasing progress in technology, our students need to learn how to solve problems thinking in a computer-like scientific manner.

### Application to a broad range of learning areas and disciplines

Computational thinking has been applied to solve problems in areas such as algorithmic medicine where it is used to develop anti‐inflammatory drugs, study chronic hepatitis, and interpret mammograms. Another areas is economics, where automated mechanisms are behind e-commerce functions such as advertisement placement, and on-line auctions ([Jeanette Wing, 2011](#_ENREF_30)).

In biology, phylogenetics use ‘consensus trees’ to represent evolutionary hypotheses. The production of consensus trees requires knowledge of several fundamental computational thinking concepts such as sorting branching patterns, hashing functions, and traversing trees ([Sul & Williams, 2011](#_ENREF_27)). The authors argue that computational thinking is not only required for biologists to be able to build the algorithms behind consensus trees, but that it provides an appreciation for the ability to use computational ideas to solve biological problems.

## 3. How can it be cultivated?

There is no consensus on whether the development of computational thinking necessitates the use of computers. Grover and Pea ([2013](#_ENREF_15)) argue that computational thinking without a computer prevents learners from accessing the common practice of computational experiences. They further specify that programming is also crucial for supporting the cognitive tasks involved in computational thinking and provide a way to demonstrate these competencies. The range of programming tools that are well-tailored for K-12 learners include graphical programming environments such as Scratch, Alice, Game Maker, Kodu and Greenfoot, simulation authoring tools such as Agentsheets and Agentcubes, and robotics and tangible creation media such as Arduino ([Grover & Pea, 2013](#_ENREF_15)).

The application of these tools include application development through the use of visual programming environments such as Scratch, gaming through environments such as FormulaT which is used to learn about kinematics, and agent-based modelling such as using NetLogo to program a simulation in order to collect data that will answer a science question. Informal opportunities include those provided by fab labs, makerspaces, and DIY movements, such as Maker Faire and Instructables that focus on tangible computational artefacts ([Grover & Pea, 2013](#_ENREF_15)).

The use of readily-available computer applications, such as spreadsheets, are proposed as possible although not essential ways to teach computational thinking ([Sanford, 2013](#_ENREF_24)). Other authors do not consider that programming is essential to developing computational thinking, since it is only considered a small part of computational thinking and should be considered an exemplar way to teach process and thinking skills, such as thinking in algorithms, avoiding ambiguity and being flexible and creative about finding solutions ([Buckley, 2012](#_ENREF_8)). Educational resources are available which specifically aim to foster computational thinking without computers (e.g., <http://csunplugged.org/>).

### Music

Efforts have been made to integrate computational thinking with creative arts disciplines such as music. The interdisciplinary Sound Thinking program developed by the University of Massachusetts uses the medium of music to teach education students computational thinking through the use of Scratch to code music ([Ruthmann, Heines, Greher, Laidler, & Saulters, 2010](#_ENREF_23)). The course is part of a series of Performatics courses that are designed to engage students in computer science through the use of performance and the arts with the aim of giving them experience in incorporating computational thinking in their non-computer science teaching domains. Making music through Scratch requires students to have an understanding of computational thinking concepts including looping, initialization, using variables, changing variables algorithmically, modularization and event processing.

Other educational resources developed for educators have students develop composition flowcharts to understand musical lyrics. This activity can be implemented through the use of existing graphing software, as well as offline through the use of low-tech solutions such as paper that can be cut out and shuffled around a board. The teacher can vary the degree of support by limiting the number and complexity of dimensions that the student is required to represent. Students may find that some music is less suitable for this exercise; exploring what properties of artefacts make them easier or more difficult to represent can be part of the exercise.

Applied in the context of an English classroom, this is an example of how computational thinking allows students to engage with a cultural artefact in a particular way, different from studying a song through thematic analysis or discourse analysis, for example. It does not detract from the artistic and subjective qualities of the text but has the ability to examine how meaning can be conveyed through non-linguistic properties. The benefits of such an analysis can extend to an appreciation and deeper engagement with an artefact to go beyond its content and the explicit message conveyed through the words of a song.

### Game design

The iDREAMS project aimed to develop middle school students’ interest in computer science through the use of the Scalable Game Design approach ([Ioannidou et al., 2011](#_ENREF_17)). Scalable Game Design is informed by the Fluency with Information Technology framework which involves three types of knowledge: intellectual capabilities that allow an individual to apply information technology in complex and sustained situations and to understand the consequences of doing so; fundamental concepts refer to some basic understandings about how information technology works; contemporary skills refer to the ability to use particular hardware and software to accomplish information processing tasks ([National Research Council, 1999](#_ENREF_20)). It is emphasised that “being FIT” is not a question of having these qualities or otherwise; rather FITness is personal, graduated and dynamic.

Another project that used game design required middle school students to use fundamental computational thinking skills in order to solve problems: decomposition, pattern recognition and generalisation, abstraction, algorithm design and data visualisation ([Wu & Richards, 2011](#_ENREF_32)). The context for this research was in the after-school setting and researchers did not make clear connections to particular learning areas.

### Agent-based modelling

Agent-based modelling is a way to simulate a phenomenon by programming the behaviours of entities that represent the agents involved in the system being simulated. This type of action-oriented activity focuses on the interactions between entities that combine to produce an outcome.

In Project GUTS (Growing Up Thinking Scientifically) middle-school students (aged 10 – 14) designed and implemented models that answered real-world concerns ([Lee et al., 2011](#_ENREF_18)). One example was the spread of disease through their school, which required students to take into account characteristics such as the layout of the school, the number of students, their movement, and the virulence of the disease. Students were required to be able to abstract the problem to be able to program the StarLogo agent modelling tool. This also required the understanding and use of automation as the program executed multiple “runs” of the simulation to derive probabilistic results that students analysed. Students were able to further their understanding by comparing simulated data with real data collected by the school on a recent epidemiological issue.

CTSiM is a computational thinking-based science learning environment where students construct a conceptual model to represent science phenomena and then construct and agent-based model to reflect their knowledge. During the conceptual stage, students are required to think in terms of *sense-acts* which refer to the properties sensed (such as “hunger”) and behaviours of agents in their system. Once they are able to describe these interactions, students move to the modelling stage to program how this behaviour is enacted ([Basu & Biswas, 2013](#_ENREF_3); [Basu, Kinnebrew, & Biswas, 2014](#_ENREF_4); [Sengupta, Kinnebrew, Basu, Biswas, & Clark, 2013](#_ENREF_25)). CTSiM comprises four modelling activities with increasingly complex conceptual and modelling requirements. Each activity has an expert solution, against which students are able to compare their own models.

### Robotics and mechanics

The iCODE project (Internet Community of Design Engineers) engaged middle- and high-school students in programming microcontroller-based projects culminating in the design of a robot ([Lee et al., 2011](#_ENREF_18)). The robot is to be able to react to real-world conditions, requiring students to abstract a set of conditions reflecting how to sense different stimuli into a set of numerical or true-false values. Students judge the extent to which their robot successfully interacts with their environment by understanding whether they performed as expected when encountering different environmental stimuli. Students may revisit their work by considering whether they correctly identified and implemented the required conditions.

First-year introductory mechanics students solved physics problems using the VPython programming environment. Activities consisted of a test case, consisting of a fully-worked solution and a grading case, where students were provided with a partially-completed program and required to debug problems with the program. The authors argue that the instructional value of computational thinking in mechanics is in developing students’ qualitative habits of mind; this can be understood as the creativity required to be able to debug ([Caballero, Kohlmyer, & Schatz, 2011](#_ENREF_9)).

### Interactive journalism

According to Ursula Wolz, there are concepts of computational thinking that are common to journalism ([National Research Council, 2011](#_ENREF_21)). The primary connection is through language; languages can be natural, as found in journalism or formal, as found in computer science. Languages involve access to information, aggregation of data, and synthesis of information. Language is also concerned with the concepts of reliability, privacy, accuracy, and logical consistency. Both fields rely on knowledge representation and abstraction from cases.

The connection between computational thinking and expository writing has been explored in a project where students created an online news magazine ([Wolz, Stone, Pearson, Monisha, & Switzer, 2011](#_ENREF_31)). Students used a collaborative age-appropriate system for online editorial and production to publish stories. Stories included text, video recorded and edited by students, still images, and animations developed in Scratch.

In this example expository writing was a vehicle to exposing students to computational thinking through the creation of digital artefacts for the communication of information. Of relevance here is the distinction between the use of computational thinking *for* expository writing and computational thinking *to program* expository writing artefacts. As such, this study presents work which relates to the absorption, synthesis and representation of points of view through multimedia in a manner that is deductively and quantifiably sound. Computational thinking was not a central component although it does provide a good example of how to introduce students to computational thinking as a means to achieve an outcome of a product of their thinking.

### Theoretical frameworks

Despite the availability of a range of materials, there is concern that research has not engaged sufficiently with theories of the learning sciences, socio-cultural and situated learning, distributed and embodied cognition, and activity, interaction and discourse analyses in the development of computational thinking resources and assessment practices ([Grover & Pea, 2013](#_ENREF_15)).

The “use-modify-create” progression is proposed as a way to help the learner start as a user and eventually create computational artefacts ([Lee et al., 2011](#_ENREF_18)). This framework is based on the premise that scaffolding increasingly deep interactions with digital environments will promote the acquisition and development of computational thinking. This begins with the use of an already-developed computer program which has the potential to be modified in increasingly sophisticated ways (for example, beginning with surface-level changes such as the appearance of the interface and later developing new code). Iterative refinements to the work are integral to this framework.

Research suggests that accurate notional machines are the basis of successful computational thinking ([Bower & Falkner, in press](#_ENREF_6)). Notional machines are discipline-specific mental models that allow users to make predictions about how a machine will perform. They are characterised by functional (minimal instructions required for specification), logical (problems are contained in scale) and syntactic (rules for writing instructions are accessible and uniform) simplicity. In computer science, the notional machine represents an abstract version of a computer. When applied to teaching, the notional machine provides a theoretical framework for making decisions about the required approach to learning, provide accurate or ideal mental models of the target concepts, and help elucidate non-viable mental models that can inform how students might think about computing and where misconceptions may arise ([Bower & Falkner, in press](#_ENREF_6)).

Developing learning activities which are underpinned by theoretical frameworks such as the “use-modify-create” progression and the notional machine may be ways to address the concern raised by Grover and Pea.

## 4. How can it be demonstrated by students and measured?

Assessment efforts have focused on the use of student-created artefacts to evaluate their understanding and application of computational thinking concepts to solve problems. Debugging is another popular assessment technique, such as providing students with a prebuilt erroneous solution for them to solve. The degree to which students engage in the vocabulary and language of computational thinking is also suggested to be a potential indicator of computational thinking competency ([Grover & Pea, 2013](#_ENREF_15)).

### Object-oriented programming

In a project involving design-based learning activities through the use of interactive media assessment was based on the three key dimensions of computational concepts (concepts employed during programming), computational practices (practices developed during programming), and computational perspectives (designers’/programmers’ perspectives about the world around them and themselves) ([Brennan & Resnick, 2012](#_ENREF_7)). Each dimension comprises a specific set of criteria that informs assessment criteria. Three options for assessing computational thinking were employed: a portfolio analysis, artefact-based interviews, and design scenarios.

The portfolio analysis used a Scratch user analysis tool called Scrape to create a visual representation of the projects uploaded by a student to get an overall snapshot of the computational concepts employed by a user by showing the blocks they used in each project. Being a product-oriented analysis, the process of development is obscured; there is no understanding of the practices involved. It is not possible to know whether a student understands how a block they have used works.

The artefact-based interviews ask students to reflect on projects they have developed. They are asked about history and motivation for the project, process of development, how it evolved, what they needed to know to make the project, what problems were encountered and how they were dealt with. This assessment approach elucidates students’ fluency with practices through the use of indicators such as their ability to articulate practices and strategies, the range of strategies in their repertoire and how they assisted in them reaching their goals. One drawback from this approach, however, is that students are asked to recall their processes, rather than work through them in real-time; memory and communicative ability therefore limits students’ capacity to demonstrate competence. Authors also noted that this was a time-intensive approach.

Design scenarios are project sets of increasing degrees of complexity that have been purposely-developed with particular types of concepts to assess students. Students are required to explain what a project does, how it could be extended, fix a bug, and remix by adding a feature. These sets offer systematicity in the concepts and practices that are being assessed, are able to be used in a formative fashion since they are designed to be implemented at three different time points, and emphasise process by assessing students while they are working, rather than asking them to recall. Disadvantages include being time-intensive, ambiguity about whether assistance should be offered, and whether the projects reflect the students’ personal interest which relate to intrinsic motivation.

The authors provide 6 suggestions that should guide assessment developed for computational thinking:

* Supporting further learning by using assessment for learning by engaging students in assessment activities requiring self-reflection, learning how to seek resources, and intellectual engagement
* Incorporating artifacts; creating and critically examining projects; rich, concrete and contextualised examples,
* Illuminating processes which can act to develop meta-cognitive ability; an important capacity for self-regulated learning,
* Employing formative assessment practices by assessing students at multiple waypoints,
* Including multiple viewpoints such as the online community, peer feedback, parents, teachers and researchers’ assessments,
* Valuing multiple ways of knowing by addressing the importance of understanding concepts as well as practices in computational thinking.

The last suggestion can be informed by theories of domain-general types of reasoning, such as declarative (knowing *what*), procedural (knowing *how*), schematic (knowing *why*), and strategic ([knowing when to apply knowledge; Shavelson, Ruiz-Primo, Li, & Ayala, 2003](#_ENREF_26)).

### Semiotic analysis

One of the abilities required for computational thinking is to be able to express a narratives from a natural language, or the language in which human interact in the real world, into tightly constrained artificial languages, such as programming languages, in order to be able to write a computer program. Assessing students’ ability to make this transfer can be achieved by semiotic or linguistic analysis which compares meaning representations in a student’s natural (written narrative descriptions of a game) and artificial texts (programming code for the game). Researchers presenting this approach contrast it to a focus on student’s ability to use programming elements such as variables, loops and conditionals ([de Souza, Bicharra Garcia, Slaviero, Pinto, & Repenning, 2011](#_ENREF_12)). They consider that a focus on signs, representations and meanings, is a way to understand the connection from the students’ psychological and cultural world into the world of computing. Externalising and formalising this discourse has the ability to refine imprecise instances of computational thinking. The ability to conduct this analysis requires the teacher to have a sophisticated understanding of semiotic theory and to adopt an “expert” understanding of the ability to transfer meaning between natural and artificial language.

### Expert comparison

Other work has used the premise of comparing students’ work to an expert solution. In a science modelling environment, students were assessed for their science and computational thinking knowledge according to a ‘vector-distance model accuracy metric’ which used expert models of each activity as reference points to determine students’ understanding ([Basu et al., 2014](#_ENREF_4)). The expert comparison was one of a suite of assessment activities which also included pre-post testing for science knowledge and computational thinking knowledge, as well as the evolution of their computational model across science units. It was found that students who had more effective edits to their model had higher accuracy

## 5. How does computational thinking relate to the Australian Curriculum?

Computational thinking is presented in the Australian Curriculum in the context of the Technologies curriculum, particularly the Digital Technologies learning area. Digital Technologies focuses on the use of digital systems, information and computational thinking to create solutions for identified needs and opportunities and articulate human knowledge. The curriculum also requires students to develop an understanding of the interaction between real and virtual worlds, or the relationship and interconnectedness between the components of digital systems and authentic situations.

Computational thinking is described as “a problem-solving method that is applied to create solutions that can be implemented using digital technologies. It involves integrating strategies, such as organising data logically, breaking down problems into parts, interpreting patterns and models and designing and implementing algorithms”. ([ACARA, 2014](#_ENREF_2)).

Below is a representation of how computational thinking relates to the other components of the technologies curriculum.



Figure 1. Relationship between key ideas and technologies subjects in the Australian Curriculum. Source: <http://www.australiancurriculum.edu.au/technologies/content-structure>

### Computational and systems thinking

Computational thinking is presented figuratively as an application of systems thinking. At a conceptual level, the relationship to systems thinking is also evident in the learning progression underlying the development of computational thinking. The knowledge and skills required for students to progress through the digital technologies curriculum are well aligned to a systems thinking learning progression ([Cheng, Ructtinger, Fujii, & Mislevy, 2010](#_ENREF_10)). While computational thinking tends to be limited to reasoning about closed systems in which reduction, abstraction and algorithmic thinking leads to predictive outcomes, this is only one part of systems thinking. Systems thinking extends to complexity, emergence (where an outcome cannot be predicted by the initial state of the system) and openness (where it is impossible to account for all variables and the impacts of their interactions); these are beyond the scope of the curriculum. The curriculum focuses on learning about *digital systems* in earlier years, later extending student learning to the connections with other technical, environmental, economic and social systems.

### Digital technologies curriculum

The development of computational thinking across the digital technologies curriculum is presented in a scope and sequence document which provides guidance according to two strands: knowledge and understanding, and processes and production skills. Each year level adds to the degree of complexity of the previous by specifying activities that require higher-order thinking skills, require students to take additional factors into consideration when designing or making decisions, and expose students to situations with multiple representation types.

The curriculum develops computational thinking in several ways across year levels. *Context* starts with a focus on the personal and local experiences of the student, moving to regional, national and global contexts. *Outcomes,* in the form of objects, products and interactions begins with use of maps and recording of information, game-based play, moving on to modelling and simple simulations of real-world phenomena, the creation of interactive objects towards the incorporation of online databases and artificial intelligence solutions. *Data* is a consistent theme throughout the curriculum; beginning with an exploration of data types and use, moving on to the collection, analysis, representation and manipulation of data, through to the validation and understanding of modes of transmission of data and how it shapes and is shaped by related systems. Computational *concepts*, abstraction in particular, begins with basic abstraction of linear steps in a procedure, the incorporation of branching and repetition, and then multilevel abstractions. *Programming* begins with visual programming, general-purpose programming and object-oriented programming. Aspect-oriented programming, a further extension of this continuum, is not a feature of the digital technologies curriculum.

At years F – 2, students are expected to use existing devices, functions and commands in response to personal needs. At years 3 – 4, students are expected to understand the nature of a problem, to describe a process and to construct and test a solution. By years 5 – 6 the focus is on the ability to compare fundamental design features to make decisions about solutions. At years 7 – 8 students develop research and modelling skills in addition to others developed in earlier years. Years 9 – 10 have students understanding information problems through reverse engineering and using data to evaluate designed solutions. Senior secondary, covered in earlier drafts of the curriculum, focuses on software engineering through the development of self-generated solutions and the application of specialised systems.

### Beyond digital technologies

The benefits of computational thinking are not limited to this learning area. A number of dispositions or attitudes that are considered to be cultivated through the development of computational thinking include: confidence in dealing with complexity, persistence in working with difficult problems, tolerance for ambiguity, the ability to deal with open-ended problems, and the ability to communicate and work with others to achieve a common goal or solution ([International Society for Technology in Education & Computer Science Teachers Association, 2011](#_ENREF_16)). ACARA also encourages the integration of computational thinking in other learning areas, particularly in the primary school setting.

There are concerns that given the difficulty of explaining computational thinking concepts as well as the difficulty for experts to interpret these concepts with consensus will present ongoing difficulties for teachers working with the curriculum ([The Australian Council for Computers in Education, 2013](#_ENREF_28)). The success of implementation depends on the quality of learning and teaching ([Bower & Falkner, in press](#_ENREF_6)).

## How does computational thinking relate to the general capabilities?

The general capabilities are a set of seven skills, abilities, dispositions that are considered to be the "new basics". Forming one dimension of the curriculum, they are not discipline-specific; they are intended to be developed across all learning areas of the curriculum. They are:

* Literacy
* Numeracy
* Information and communication technology capability
* Critical and creative thinking
* Personal and social capability
* Ethical understanding
* Intercultural understanding

ACARA presents guidance on how the technologies curriculum as a whole relates to each of the seven general capabilities. Computational thinking has strong links to the general capabilities of critical and creative thinking, and information and communication technologies.

### Critical and creative thinking

Authors highlight that engaging in computational thinking engages students in performing qualitative analyses associated with debugging practices in programming to solve analytical problems ([Caballero et al., 2011](#_ENREF_9)). Developing these skills allows students to solve open-ended problems, such as those they will likely face in their future work and lives. James Curran also emphasises that the challenge of debugging is the systematicity required to find out why a program didn’t work; critical thinking and logic is crucial for this process, much more than during the process of writing the program itself. This activity also requires resilience to deal with a challenging problem and perseverance to find a solution.

The development of critical thinking is required in ill-structured domains, such as those in which computational thinking may be applied. The theory of cognitive flexibility, which requires the ability to restructure one’s knowledge as an adaptive response to changing situational demands, provides a useful framework for designing instructional tasks to develop this ability ([Bower, 2004](#_ENREF_5)). The basic principles of this theory hold that instruction should reflect real-world complexities faced by practitioners, should involve multiple cases to allow learners to abstract between examples, and should be context-dependent to allow learners to acquire deeper appreciation of the subtleties that affect decision making and potentially address misconceptions.

Buckley suggests that critical thinking and knowledge application alone is insufficient when it comes to real technological problems; rather these problems require an additional skill – computational thinking ([Buckley, 2012](#_ENREF_8)). Technological problems require computational thinking and it is an increasingly essential tool for doing scientific research.

Disagreements exist about the distinctness of computational thinking and its primacy as a problem solving skill. It has been considered to be just one of many thinking skills, none of which need to be prerequisites and that it may in fact be a part of critical and creative thinking.

### Information Communication Technologies

The ability to use digital tools to solve problems and carry out tasks connects computational thinking with the ICT general capability. In order to be successful in solving problems in this manner students needs to develop an understanding of what the device is capable of and appreciate its limitations; it is argued that this requires instructional tasks that provide authentic contexts for learning ([ACARA, 2013](#_ENREF_1)). Computational thinking is required for students to learn explicitly how digital technologies work in order to develop information solutions.

# Conclusion

The development of computational thinking as a recognised thinking skill is a recent but growing phenomenon in the education sector. The ability for students to think computationally has implications for their problem solving abilities, employment opportunities, and equity. While there are a range of resources available for educators to use in cultivating and assessing this skill, many of them require the use of sophisticated software and a deep understanding of complex computational thinking concepts. These characteristics may make it difficult for many teachers to incorporate computational thinking into their teaching practice, especially if they do not have a background in computing nor the time to learn to use specialised software packages.

The development of critical and creative thinking as well as information and communication technology capability are closely related to computational thinking; as such these could be used both as guides to learning activities that might be well suited to the development of computational thinking as well as provide teachers with pathways to approach computational thinking from different learning areas.

The Australian Curriculum views computational thinking in the context of the technologies curriculum where it has been identified as one of three types of thinking. ACARA encourages teachers to use the information in the technologies curriculum to create integrated teaching programs that combine the technologies curriculum with other learning areas. While the literature presented in this paper provides suggestions for such opportunities, stronger support in the form of explicit connections in other learning areas and professional development opportunities may be required for successful implementation.

# References

ACARA. (2013). General Capabilities in the Australian Curriculum. *Australian Curriculum.* 7.2. Retrieved 20/5/2014, from <http://www.australiancurriculum.edu.au/GeneralCapabilities/Pdf/Overview>

ACARA. (2014). Technologies content structure. *Australian Curriculum.* 7.2. Retrieved 10/10/2014, from <http://www.australiancurriculum.edu.au/technologies/content-structure>

Basu, S. & Biswas, G. (2013). *A Computational Thinking Approach to Learning Middle School Science.* Paper presented at the 16th International Congerence, AIED 2013, Memphis, TN.

Basu, S., Kinnebrew, J. S. & Biswas, G. (2014). *Assessing Student Performance in a Computational-Thinking Based Science Learning Environment.* Paper presented at the 12th International Conference, ITS 2014, Honolulu, HI.

Bower, M. (2004). *Teaching and Learning Computing*. Macquarie University. Sydney. Retrieved from <http://hswiki.ltc.mq.edu.au/Department_of_Education/Research/Computational_Thinking>

Bower, M. & Falkner, K. (in press). *Computational Thinking, the Notional Machine, Pre-service Teachers, and Research Opportunities*. Paper presented at the 17th Australiasian Computing Education Conference, Sydney, Australia.

Brennan, K. & Resnick, M. (2012). *New frameworks for studying and assessing the development of computational thinking*. Paper presented at the Annual Meeting of the American Educational Research Association, Vancouver, BC.

Buckley, S. (2012). *The Role of Computational Thinking and Critical Thinking in Problem Solving in a Learning Environment* Paper presented at the 11th European Confernece on e-Learning, The Netherlands.

Caballero, M. D., Kohlmyer, M. A. & Schatz, M. F. (2011). *Fostering Computational Thinking in Introductory Mechanics.* Paper presented at the Physics Education Research Conference 2011, Omaha, NE.

Cheng, B. H., Ructtinger, L., Fujii, R. & Mislevy, R. (2010). Assessing Systems Thinking and Complexity in Science. In A. M. Colker, G. Haertel, R. Mislevy & R. Fried (Eds.), *Application of Evidence-Centred Design for Large-Scale State Science Assessment* (Vol. Technical Report 7). Menlo Park, CA: SRI International.

Day, C. (2011). Computational Thinking is Becoming One of the Three Rs. *Computing in Science & Engineering, 11*, 88.

de Souza, C. S., Bicharra Garcia, A. C., Slaviero, C., Pinto, H. & Repenning, A. (2011). *Semiotic Traces of Computational Thinking Acquisition.* Paper presented at the Third International Symposium, IS-EUD 2011, Torre Canne, Italy.

Denning, P. J. (2009). The profession of IT: Beyond computational thinking. *Communications of the ACM, 52*(6), 28-30.

Foo, F. (2014, 9/12/2014). Christopher Pyne urged to act on coding in classrooms, *The Australian*. Retrieved from <http://www.theaustralian.com.au/technology/christopher-pyne-urged-to-act-on-coding-in-classrooms/story-e6frgakx-1227149001034>

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

International Society for Technology in Education & Computer Science Teachers Association. (2011). Operational Definition of Computational Thinking for K-12 Education. Retrieved 15/8/2014, from https://csta.acm.org/Curriculum/sub/CurrFiles/CompThinkingFlyer.pdf

Ioannidou, A., Bennett, V., Repenning, A., Koh, K. H. & Baswapatna, A. (2011). *Computational Thinking Patterns*. Paper presented at the Annual Meeting of the American Educational Research Association, New Orleans.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., . . . Werner, L. (2011). Computational Thinking for Youth in Practice. *ACM Inroads, 2*(1), 32-37.

MCEECDYA. (2008). Melbourne Declaration on Educational Goals for Young Australians. Victoria: Ministerial Council on Education, Employment Training and Youth Affairs

National Research Council. (1999). Being Fluent with Information Technology. Washington, DC: The National Academies Press.

National Research Council. (2011). Report of a Workshop on the Pedagogical Aspects of Computational Thinking. Washington, DC: The National Academies Press.

Perkovi´c, L., Settle, A., Hwang, S. & Jones, J. (2010). *A Framework for Computational Thinking across the Curriculum.* Paper presented at the ITiCSE ’10, 15th Annual Conference on Innovation and Technology in Computer Science Education, Ankara, Turkey.

Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P. & Saulters, C. (2010). *Teaching Computational Thinking through Musical Live Coding in Scratch*. Paper presented at the SIGCSE’10, Milwaukee, WI.

Sanford, J. F. (2013). Core concepts of computational thinking. *International Journal of Teaching and Case Studies, 4*(1), 1-12.

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G. & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies, 18*(2), 351-380.

Shavelson, R., Ruiz-Primo, M. A., Li, M. & Ayala, C. C. (2003). Evaluating New Approaches to Assessing Learning *CSE Report 604*. University of California, Los Angeles: National Center for Research on Evaluation, Standards, and Student Testing.

Sul, S. J. & Williams, T. L. (2011). Big cat phylogenies, consensus trees, and computational thinking. *J Comput Biol, 18*(7), 895-906.

The Australian Council for Computers in Education (2013). [Draft Curriculum Technologies Response Summary].

Wing, J. (2006). Computational Thinking. *Communications of the ACM, 49*(3), 33-35.

Wing, J. (2011). Computational Thinking: What and Why? Retrieved from <http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why>

Wolz, U., Stone, M., Pearson, K., Monisha, P. S. & Switzer, M. (2011). Computational Thinking and Expository Writing in the Middle School. *ACM Transactions on Computing Education, 11*(2).

Wu, M. L. & Richards, K. (2011). Facilitating Computational Thinking through Game Design. In M. Chang, W.-Y. Hwang, M.-P. Chen & W. Müller (Eds.), *Edutainment Technologies. Educational Games and Virtual Reality/Augmented Reality Applications* (pp. 220-227): Springer Berlin Heidelberg.