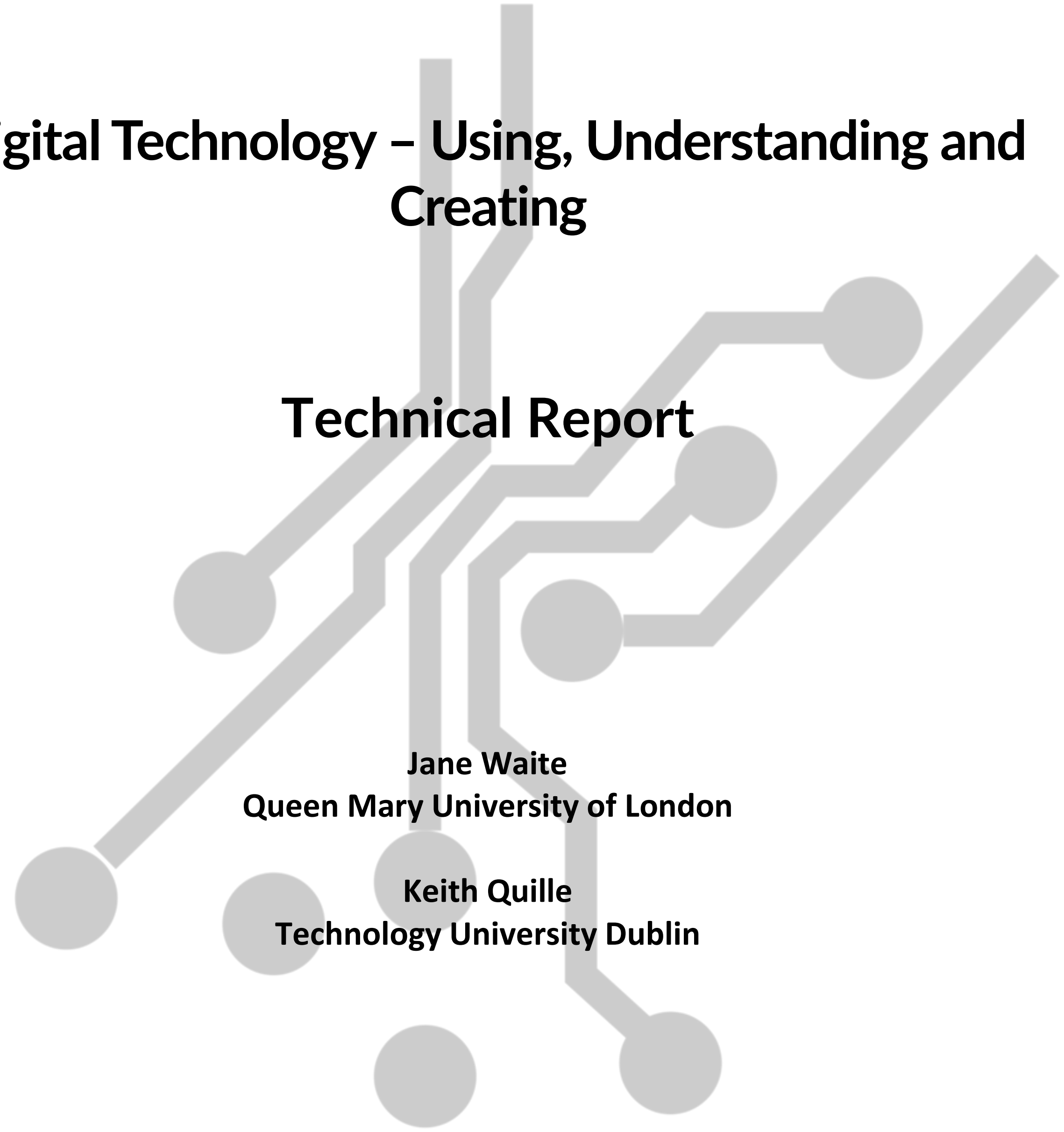
Research Paper in support of the introduction of Technology

in a redeveloped Primary School Curriculum



Digital Technology – Using, Understanding and Creating

Technical Report

Jane Waite
Queen Mary University of London

Keith Quille
Technology University Dublin

Contents

Jane Waite
Keith Quille

1. Introduction

This technical report focuses on the ‘Using, Understanding and Creating’ aspect of digital technology within the curriculum area of Science and Technology as proposed in the *Draft Primary Curriculum Framework* (NCCA, 2020). The report synthesises earlier NCCA studies and reports, as well as national and international research in computing education. Through the synthesis of research, this technical report presents a *Concept and Processes Model* (CP Model) for curriculum design to support learning in digital technology. It also provides examples of how to apply the model as well as pedagogical guidance for consideration in a redeveloped Primary School Curriculum.

The development of the CP Model proposed within this report is based on modern curriculum approaches which support the knowledge, skills, dispositions and values associated with the foundational underpinnings of learning about digital technology. As competencies are a feature of the proposed *Draft primary Curriculum Framework* (2020), the development of the CP Model is framed by considering the following question:

> "What are the competencies for primary-aged pupils in understanding digital technology?"

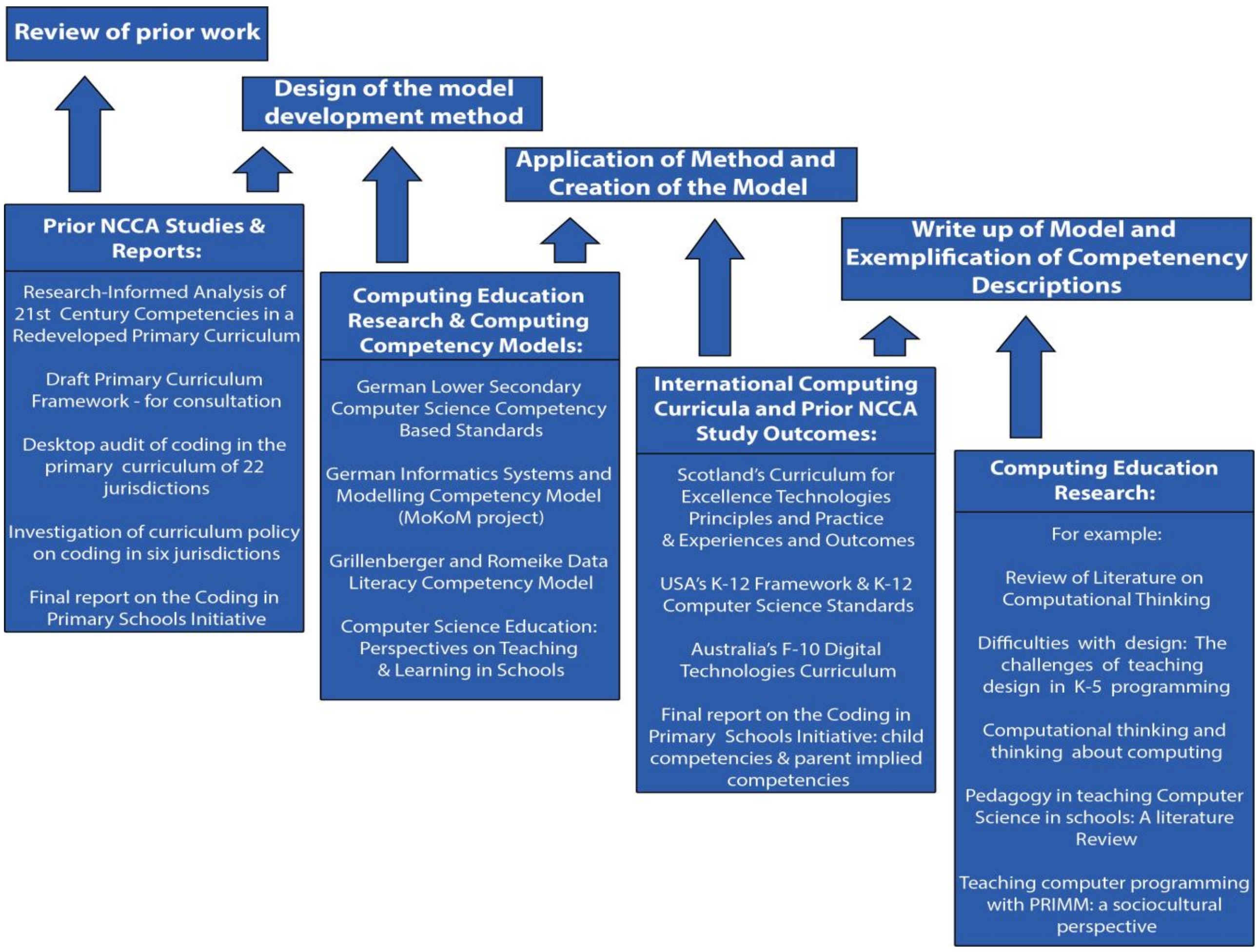
A process was designed to answer this question, which included synthesising earlier NCCA studies and reports, computing education research, and international digital and computing curricula, which resulted in the development of a model (CP Model) for primary-aged pupils in using, understanding and creating with digital technology. The process devised and followed is shown in Figure 1 and described in detail in the Methodology section below.

The following steps were used in the development process:

- Review of prior NCCA studies and reports, and computing education research

- design of model development method (including a suggested method for the elaboration of competencies)
- application of the method and creation of the model (including the selection of curricula for synthesis)
- write up of the model including exemplification of competency descriptions and example elaborations.

Figure 1: Development process of the Concept and Processes model


Review of prior work
Design of the model development method
Application of Method and Creation of the Model
Write up of Model and Exemplification of Competenency Descriptions
Prior NCCA Studies & Reports:
Research-Informed Analysis of 21st Century Competencies in a Redeveloped Primary Curriculum
Draft Primary Curriculum Framework - for consultation
Desktop audit of coding in the primary curriculum of 22 jurisdictions
Investigation of curriculum policy on coding in six jurisdictions
Final report on the Coding in Primary Schools Initiative
Computing Education Research & Computing Competency Models:
German Lower Secondary Computer Science Competency Based Standards
German Informatics Systems and Modelling Competency Model (MoKoM project)
Grillenberger and Romeike Data Literacy Competency Model
Computer Science Education: Perspectives on Teaching & Learning in Schools
International Computing Curricula and Prior NCCA Study Outcomes:
Scotland's Curriculum for Excellence Technologies Principles and Practice & Experiences and Outcomes
USA's K-12 Framework & K-12 Computer Science Standards
Australia's F-10 Digital Technologies Curriculum
Final report on the Coding in Primary Schools Initiative: child competencies & parent implied competencies
Computing Education Research:
For example:
Review of Literature on Computational Thinking
Difficulties with design: The challenges of teaching design in K-5 programming
Computational thinking and thinking about computing
Pedagogy in teaching Computer Science in schools: A literature Review
Teaching computer programming with PRIMM: a sociocultural perspective

The design of the development method was built upon computing education research on competency model development. The application of the method required the cross-referencing of a wide range of sources to find themes. Competencies from existing research-derived computing competency models, international curricula objectives and outputs from practical experience outcomes of prior NCCA studies, such as implied competencies from parent and children surveys, were also grouped.

To situate the CP Model in the *Draft primary Curriculum Framework* (2020), a number of the CP Model themes have been mapped to the suggested key competencies found in the proposed framework document (NCCA, 2020, p. 7) (see The Concepts and Processes Model Section, Table 1). Each of the themes within the CP Model was then written up with an exemplification description (see The Concepts and Processes Model Section, Tables 2 and 3). These descriptions are narratives derived by the authors from the underlying detail of the contributing sources and research literature.

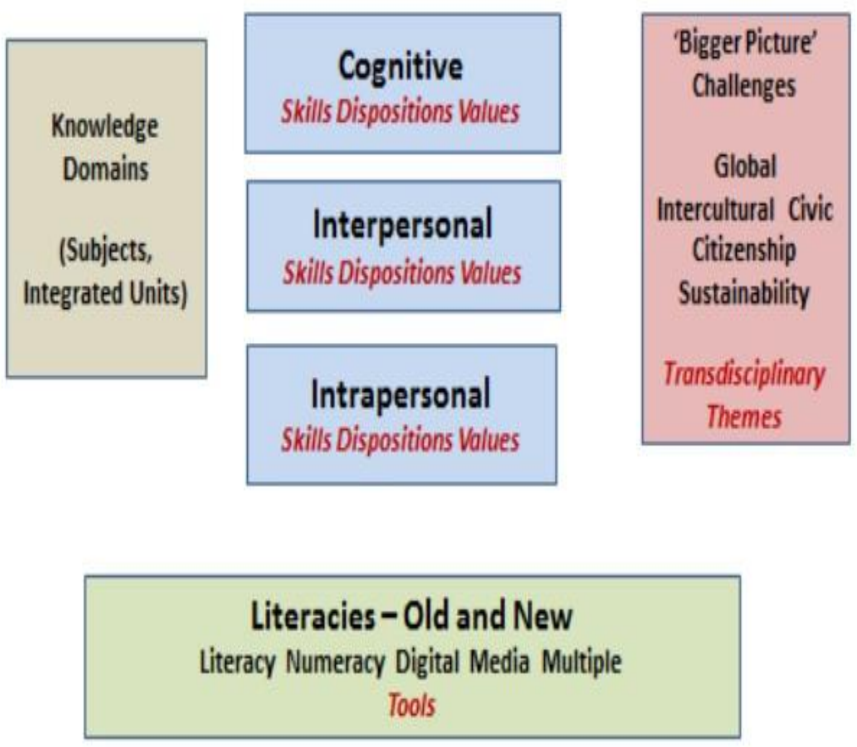
2. Methodology

Review of Prior Work

Three strands of investigation contributed to the author's consideration of the prior work. Firstly, a review of work that has been undertaken on competencies by the NCCA. Secondly, a review of NCCA work which contributed to the Coding in Primary Schools reports. Thirdly an investigation of competency models and competency-based curricula for primary digital technologies already used internationally.

In 2018, the NCCA commissioned Prof Carol McGuinness to review international trends in competency-based curricula and her resultant report proposed a classification model for key competencies as shown in Figure 2.

Figure 2. Classification model for key competencies, McGuinness (2018, p. 16)

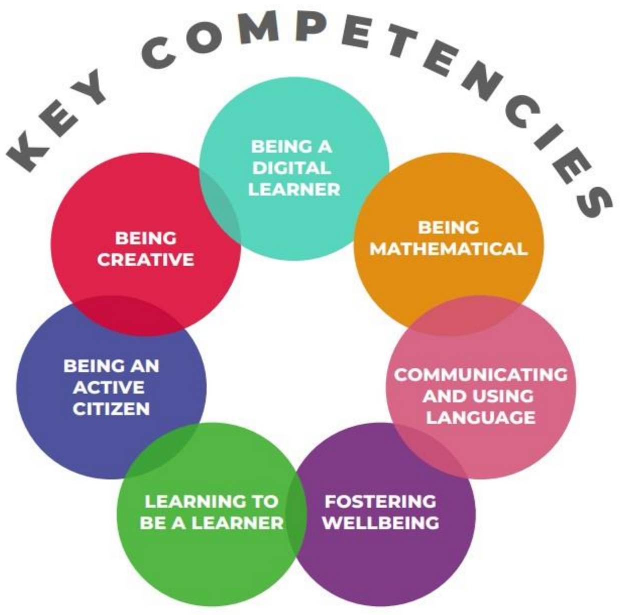


Knowledge Domains
(Subjects, Integrated Units)
Cognitive
Skills Dispositions Values
Interpersonal
Skills Dispositions Values
Intrapersonal
Skills Dispositions Values
'Bigger Picture' Challenges
Global Intercultural Civic Citizenship Sustainability
Transdisciplinary Themes
Literacies – Old and New
Literacy Numeracy Digital Media Multiple
Tools

The review highlights the idea of transferable or adaptive competencies, including the importance of carefully designing the contexts in which children learn. Rather than being “stuck” in a specific context, learners need to transfer conceptual understanding and process and production skills across settings. Being able to transfer competencies is not easy; pupils need experiences of working in a variety of contexts to learn how to draw on prior learning and apply it in new scenarios. McGuinness also emphasises the need for a range of culturally relevant contexts, including “Bigger Picture” Challenges incorporating Global, Intercultural, Civic, Citizenship and Sustainability aspects.

Following this, in 2020, the NCCA published the *Draft Primary Curriculum Framework* (NCCA 2020), including a proposal of seven key competencies which should underpin subject curricula, as shown in Figure 3. At the time of writing this report, the draft framework was still in the consultation phase.

Figure 3. NCCA *Draft Primary Curriculum Framework* (2020, p. 7) Key Competencies



For the Coding in the Primary Schools work, in 2016, the NCCA conducted a desktop audit of coding in the primary curriculum from 22 jurisdictions (NCCA 2016). Following this, in 2018, an NCCA report (NCCA 2018) reviewed six curricula of England, New Zealand, Finland, the USA' CSTA (Computer Science Teachers Association) standard, Northern Ireland and Scotland. Four of these curricula were present in the initial desktop audit. Two curricula were added (Finland and Northern Ireland). Selected curricula were based on an early introduction of coding, an integrated cross-curricular approach, a research-driven methodology and reputation for learning in technology and ICT integration.

A culminating coding initiative report was published in 2019 called the *Final Report on the Coding in Primary Schools Initiative* (NCCA 2019*)*. The Final Report included a set of learning outcomes that were used by schools in phase 2 of the initiative and results from teacher, pupil, and parent surveys.

The final component for the review of prior work was an investigation as to whether there are existing research-based digital technology competency models and existing digital technology curricula. This approach is in alignment with earlier NCCA reports and methods where theoretically based models, as well as in-practice curricula, were reviewed.

CP Model Design Method

The method designed to develop the CP Model, combined elements from the authors review of prior work and consideration of research on digital technology models (Sentence *et al*. 2018) and included the following steps:

1. selection of a set of research-based digital technology competency models,
2. selection of a set of competency-based primary level digital technology curricula,
3. list potential competency requirements from the pilot coding initiative report (NCCA 2019), including learning objectives and requirements implied from survey questions and the report discussion,

4. group the statements from Steps (1) to (3) in a competency model of concept and skills dimensions.

During Steps (1) to (3), as models, curricula and requirements were gathered, the statements, such as model dimensions, curricula standards, implied requirements, were added to a shared spreadsheet. In Step (4) the statements were mapped within McGuinness's competency classifications (McGuinness 2018). In addition, the NCCA *Draft Primary Curriculum Framework* Key Competencies (NCCA 2020) were also included as dimensions. A consensus through discussion was then reached as to the inclusion of the final CP Model or, indeed, those elements deemed to be outside the scope of the proposed model.

Application of the Method

Step 1: Selection of research-based digital technology competency models. According to Sentence *et al.* (2018), there has been limited research-informed competency-based computing curricula for schools and that which has been created still requires robust validation in schools. Despite this, in this first step, three competency-based models, all of which have been developed to teach an aspect of digital technologies, were selected:

- the 2009 German Lower Secondary Computer Science Competency Based Standards (Torsten *et al.* 2009)
- the 2010 German Informatics Systems and Modelling Competency Model (MoKoM Project) (Magenheim *et al.* 2010)
- and the 2012 Grillenberger and Romeike Data Literacy Competency Model (Grillenberger and Romeike 2018).

Whilst it is noted that all three models are developed for second-level school contexts, these research-based theoretical models report high-level competencies framed as content and processes which serve as a starting point for primary level curricula development.

The MoKoM project reports a list of competencies and curricula statements, whereas the other two present content and process standards models, which are shown in Figures 4 and 5.

Figure 4. Content/Process standards for Computer Science in Lower Secondary Education in Torsten *et al.* (2008, p. 5)

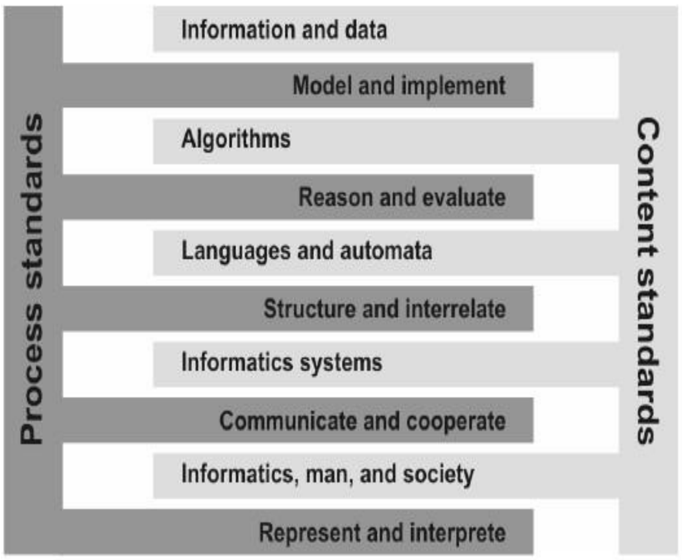
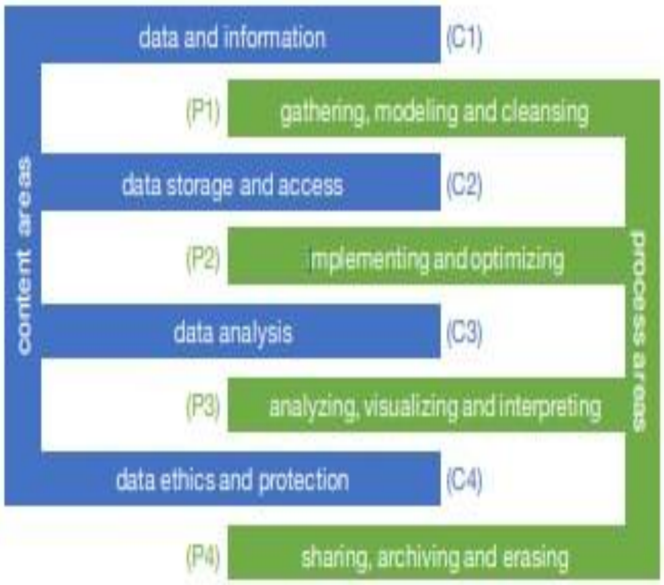


Figure 5. Data Literacy Competency Model (Grillenberger and Romeike 2018, p. 7)


content areas
process areas
data and information (C1)
(P1) gathering, modeling and cleansing
data storage and access (C2)
(P2) implementing and optimizing
data analysis (C3)
(P3) analyzing, visualizing and interpreting
data ethics and protection (C4)
(P4) sharing, archiving and erasing

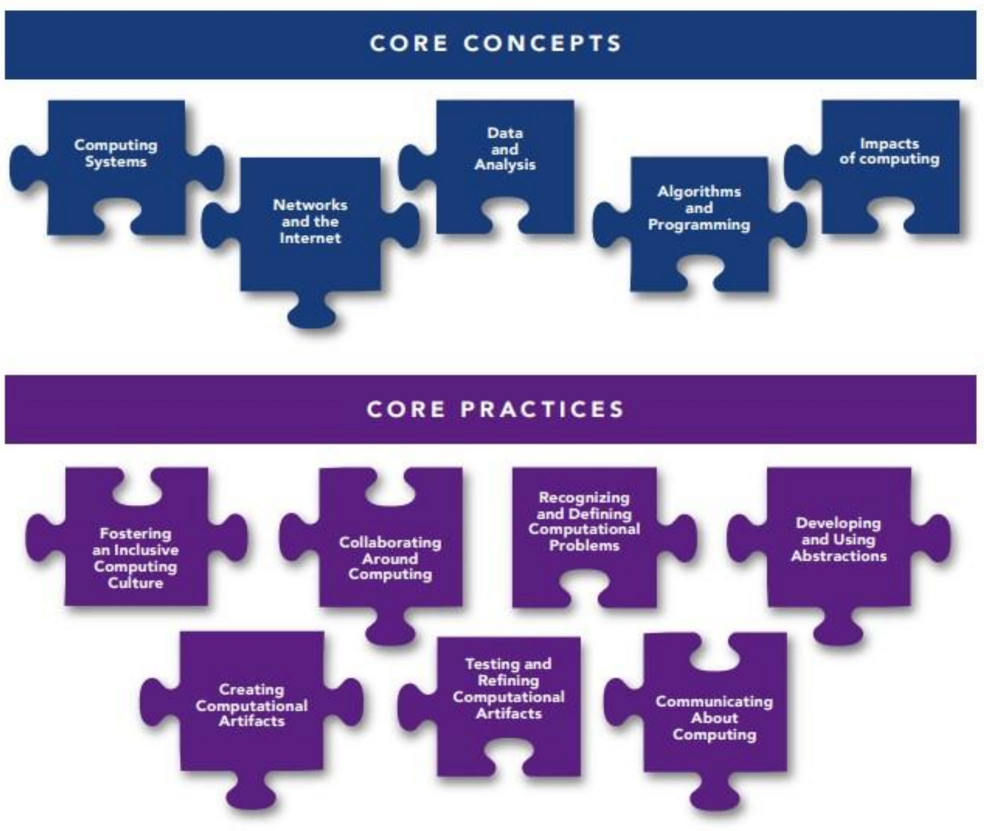
Step 2: Selection of primary-level competency-based digital technology curricula. The second step was to select a primary level competency-based digital curriculum. Three curricula selected were:

- Scotland's Curriculum for Excellence Technologies Principles and Practice & Experiences and Outcomes (Education Scotland 2021)
- USA's K-12 Framework and K-12 Computer Science Standards (Computer Science Teachers Association 2020)
- Australia's F-10 Digital Technologies Curriculum (Australian Curriculum, Assessment and Reporting Authority 2021a)

The Scottish Technologies Curriculum and USA K-12 Computer Science Framework and Standards were included in the NCCA's review of six curricula (NCCA 2018). The Australian Curriculum was also considered in the NCCA investigation to the approaches to teaching coding in 22 international jurisdictions (NCCA 2016).

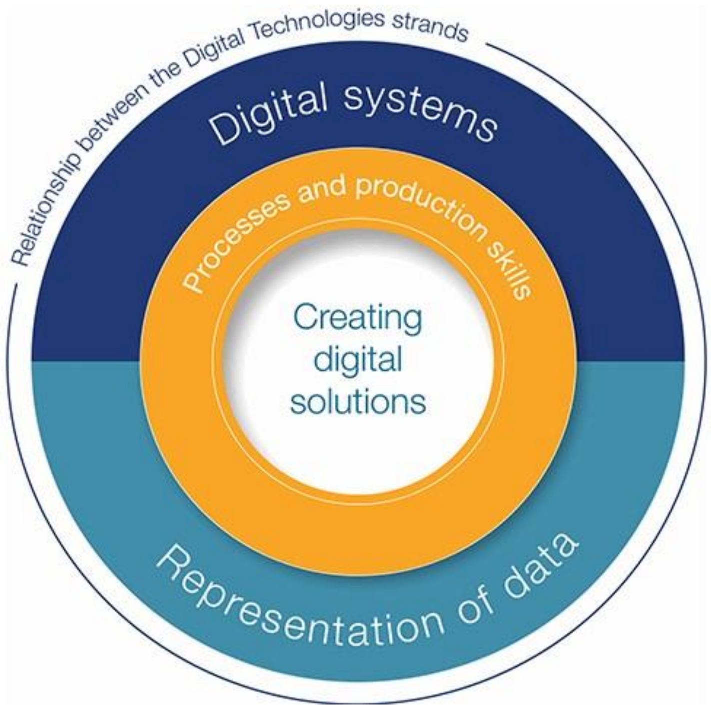
The Scottish Technologies Curriculum was selected for inclusion in this step as the Curriculum for Excellence (Education Scotland 2021) distinguishes between principles, practice, experiences, and outcomes. The USA's K-12 Framework (Computer Science Teachers Association 2020) was also considered as it sets out what learners are required to know and do through concepts and practices, see Figure 6. In developing the USA's K-12 Computer Science Framework consideration, was given to theoretically derived competency models.

Figure 6. The USA K-12 Computer Science Framework (2016, p. 2)


CORE CONCEPTS
Computing Systems
Networks and the Internet
Data and Analysis
Algorithms and Programming
Impacts of computing
CORE PRACTICES
Fostering an Inclusive Computing Culture
Collaborating Around Computing
Recognizing and Defining Computational Problems
Developing and Using Abstractions
Creating Computational Artifacts
Testing and Refining Computational Artifacts
Communicating About Computing

The Australian primary school curricula (Australian Curriculum, Assessment and Reporting Authority 2021b) are not specifically cited as being based on a competency model. However, the strands of general capabilities, cross-curriculum priorities, key ideas, knowledge and understanding content descriptions, and process and production skills are components that might be seen in a competency-based curriculum. In Figure 7, the relationship between these strands, where *Digital Systems* and *Representation of Data* relate to knowledge and understanding strands; and process and production skills include *investigating* and *defining*, *generating* and *designing*, *producing* and *implementing*, *evaluating* and *collaborating* and *managing*.

Figure 7. The Australian F-10 Digital Technologies Learning Area Relationship between stands (Australian Curriculum, Assessment and Reporting Authority 2021a).


Relationship between the Digital Technologies strands
Digital systems
Processes and production skills
Creating digital solutions
Representation of data

The Australian Digital Technologies curriculum was published in December 2015, and examples (elaborations) of the expected outcomes for year groups were added in 2016 (Australian Curriculum, Assessment and Reporting Authority 2021a).

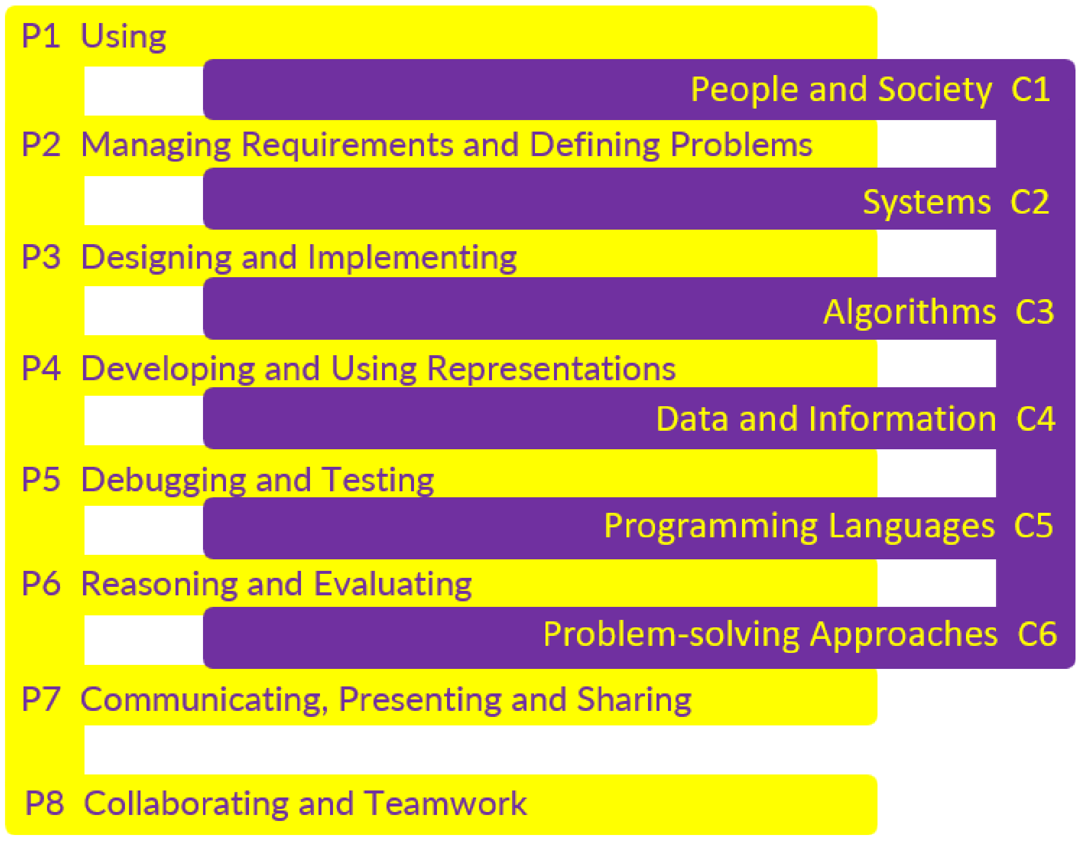
Step 3: List potential competency requirements from the pilot coding initiative report. The third step included a compilation of the potential primary-aged digital competencies developed in phases 1 and 2 of the *NCCA Report on the Coding Initiative Final Report* (NCCA, 2019). There are several implied competency requirements within the teacher, parent and pupil surveys and pilot learning outcomes of the report. For example, 86% of parents interviewed reported they expected children should “Be aware of and demonstrate safe and ethical behaviour in the use of information and technology” (NCCA 2019, p. 65). In pupil interviews, 88% of pupils reported yes to the question, “Do you think it is important to understand how digital devices work?” (NCCA 2019, p. 71). A pilot learning outcome was

"Follow the problem-solving process, design and implement digital solutions using algorithms that involve decision-making and user input" (NCCA 2019, p. 24).

Step 4: Group the statements from Steps 1 to 3 in a competency model. The fourth and final step of the process was to map each of the statements from Steps 1 to 3 to McGuinness's competency classification (McGuinness 2018, p. 8). This gave a very high-level broad categorisation. Next, the German Lower Secondary Computer Science Competency-Based Standards content and skill competencies (Grillenberger and Romeike 2018, p. 7), as shown in Figure 5, were selected as initial candidate dimensions. Added to this was the seven extra candidate dimensions of the NCCA *Draft Primary Curriculum Framework* Key Competencies (NCCA 2020, p.7) as shown in Figure 3.

Each author then took a set of statements and mapped them to the candidate dimensions. If a statement could not be mapped, the statement was recorded as unmapped. Suggestions for changes to candidate dimensions were made, and the two authors discussed reaching a consensus. This approach was conducted over multiple iterations with the two authors checking each other's allocations in detail. It was initially planned that each author would check at least 50% of the other author's mappings, but this estimated review rate was far exceeded. The final suggested CP Model is presented in Figure 8.

Figure 8. Concepts and Processes Model (C - Concepts, P - Processes and Production Skills)


P1 Using
People and Society C1
P2 Managing Requirements and Defining Problems
Systems C2
P3 Designing and Implementing
Algorithms C3
P4 Developing and Using Representations
Data and Information C4
P5 Debugging and Testing
Programming Languages C5
P6 Reasoning and Evaluating
Problem-solving Approaches C6
P7 Communicating, Presenting and Sharing
P8 Collaborating and Teamwork

3. The Concepts and Processes Model

Overview of CP Model

The CP Model has two dimensions: a concept and a process or production skill dimension. Each dimension has themes; there are six concept themes and eight process or production skill themes (see Figure 8 above). Each theme has a summary. The summaries are derived from a synthesis of the contributing curricula statements, which have been mapped to each theme and relevant literature**.** The summary views are not exhaustive; they do not detail progression, when themes should be introduced, the weighting of themes, misconceptions to be addressed, the impact of contexts, consideration of pupil specific needs or assessment requirements.

As well as the theme summaries, guidance is also provided on contexts. Contexts correlate to the Transdisciplinary theme of McGuinness's (2018, p. 16) classification model (see Figure 2). Using McGuinness' classification model for key competencies, the broad classification of each theme and contexts are as follows:

- Knowledge domains/concept building: *C2 Systems*, *C3 Algorithms*, *C4 Data and information*, *C5 Programming languages*
- Literacy Competency: P1 *Using*
- Cognitive Competency: *P2 Managing requirements and defining problems*, *P3 Designing and implementing*, *P4 Developing and using representations*, *P5 Debugging and testing*
- Intra-personal: *C6 Problem-solving approaches*, *P6 Reasoning and evaluating*
- Inter-personal: *P7 Communicating, presenting and sharing*, *P8 Collaborating and teamwork*
- Transdisciplinary Themes: *C1 People and society*, *Contexts*

Digital Technology activities can often be set in one of a range of contexts, giving teachers the opportunity to choose from different settings. In mapping statements to the concepts and processes model, several statements draw attention to the importance of contexts. In the development of the model, some statements also directly relate to the draft NCCA primary Key Competencies (NCCA 2020, p. 7) see Table 1

Table 1. Mapping of Key Competencies to Digital Technologies Themes and examples statements

Key Competency	**Digital Technologies Themes**
Being a digital learner	*P1 Using*
Being mathematical	*C4 Data and information*
Communicating and using language	*P7 Communicating, presenting, and sharing*
Fostering wellbeing	*C1 People and society*
Learning to be a Learner	*C6 Problem solving approaches*
Being an active citizen	*C1 People and society*
Being creative	*P3 Designing and implementing*

Concept Dimension

There are six themes within the Concept Dimension as shown in Figure 9 below. The concepts incorporate literacy and cognitive competencies, intra-personal and knowledge domains and are described in Table 2.

Figure 9: Concept Dimension – Concepts orKnowledge and Understanding

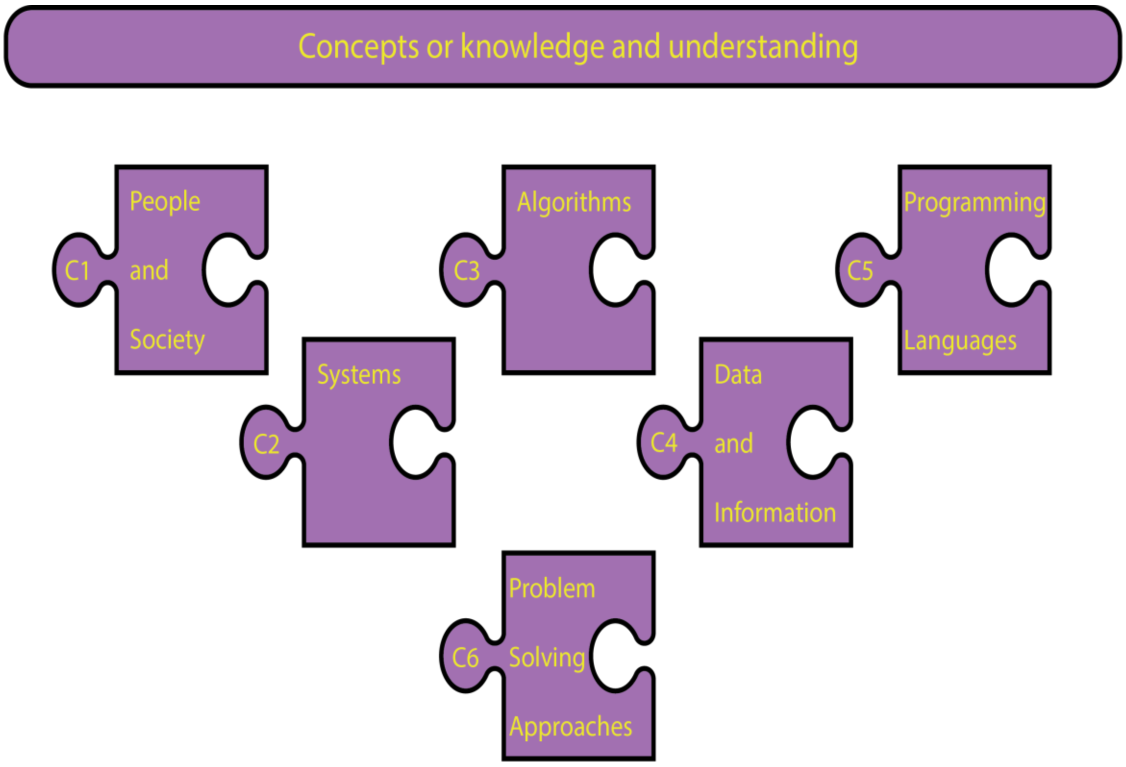


Table 2. Concepts of Knowledge and Understanding

C1 **People and Society**	Digital technologies are not created in a vacuum; they are developed for people within a social context. This concept-based theme covers strands such as historical, current and future trends, impacts and uses, ethics, cyber-resilience and internet safety, related careers, high expectations of digital technology and the importance of an inclusive digital landscape. Learners are likely to investigate, reason, evaluate, communicate, present and share about these strands, often collaborating to develop their understanding. Within this concept, children show a developing knowledge of the historical, current and future uses of digital technology. Children start to understand the impact this has on their own lives, familiar and broader

	groups and society in general. Included in this is an appreciation of the sustainability of digital solutions and the impact on the environment and economy. Through this, pupils start to gain an understanding of the world in general and the role of technology to meet the needs of people and society. Children begin to understand ethical considerations of digital technologies ranging from a personal to a broader social viewpoint. Children will start to understand how and why they should act respectfully and responsibly online, they will also start to consider ethics when designing and evaluating digital technologies. Similarly, by knowing about cyber-resilience and internet safety, children will start to understand how and why they should stay safe when using digital technology, as well as considering e-safety when reviewing and developing digital solutions. Learners will start to appreciate the pervasive nature of technology, including how and why it is used in a wide range of circumstances, from the arts to the sciences. They will start to learn about the digital technology knowledge and associated skills required for familiar and a broader range of jobs and careers. Learners will also develop an emerging understanding that technology is a profession in its own right and that there are many different careers involved. In doing this, they will become aware that being a user is different from being a developer of technological systems and will start to appreciate the tensions between the two roles.

	Having high expectations of their digital behaviours and the design of digital systems should be established, including the importance of ensuring that digital systems are accessible to all. Empathy for different perspectives and diverse needs when evaluating and designing digital technologies should be encouraged to ensure young learners understand the importance of an inclusive digital culture.
C2 **Systems**	The concept of a system as a set of things that work together is central to learning about digital technologies. This includes knowledge of the components of digital systems and how they interconnect as well as understanding, on a basic level, of hardware and software systems. At the primary phase, children should learn about the purpose of everyday digital system components, including understanding simple inputs and outputs, such as a keyboard and a screen. When learning about networks, they will start to understand, on a simple level, how networks transmit data. Learners are likely to use, recognise, reason and evaluate, communicate and present about systems. They will use representations of systems, such as when engaging with unplugged activities which represent networks. When learning about the systems and particularly networks, there are strong links to ethics and internet safety. There are particular opportunities to teach about components of systems when working with physical computing.
C3 **Algorithms**	Children's understanding of algorithms begins with them following or physically enacting, simple instructions including, making choices at decision points. Through this, they start to understand the need for accuracy in terms of putting steps in the right order and including the

	right level of detail so that when following the instructions there will always be the same outcome. This work helps learners understand what an algorithm is - a precise and unambiguous set of instructions for completing a task or goal. The algorithm provides a structure that controls what will happen. As learners begin to solve problems, rather than following another person's solution, they will create their own solution, included in the solution will be their algorithm. Algorithms may be implemented as a set of commands, from a restricted set, for a person to follow or as a program for a computer to execute. In primary classrooms, it is likely that block-based programming languages, such as Scratch (Massachusetts Institute of Technology 2021), or simple text languages such as LOGO (Massachusetts Institute of Technology 2015) will be used to implement designs (including algorithms). However, it is essential to remember that algorithms can also be implemented through menu-driven software, such as making games, animations, quizzes or presentations. The complexity of algorithms which children use, design, implement, test, debug and evaluate will possibly increase over time. This is likely to start with simple sequences of steps that result in some visible output, next, introducing repetition of steps, adding decision-making, incorporating user input, catering for more than one thing happening at the same time and storing and using simple data values as children make progress. Similarly, how algorithms are represented is likely to vary, both from task to task and across contexts. Young children may enact an algorithm

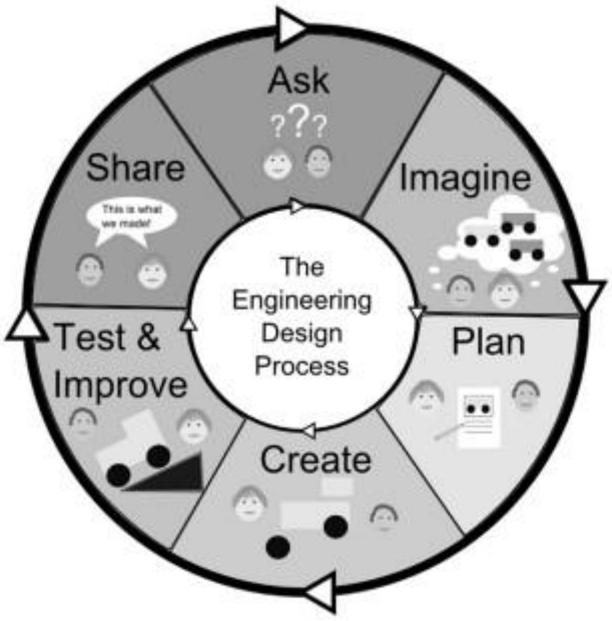
	through movement and role-play, older learners may write more formal representations with lists and labelled diagrams, even with flowcharts. Algorithms or algorithmic thinking are sometimes incorporated as concepts in computational thinking definitions and as such could be included in the C6 Problem-solving approaches dimension of our Primary Digital Technologies Competency Model. However, algorithms have been singled out as being a "big idea" in the development of an early understanding of programming, warranting it having its own dimension (Denning 2003).
C4 **Data and Information**	As a counterpoint to algorithms, data, or, more correctly, data structures, sits as the second "big idea" of programming (Denning 2003). However, when supporting primary-aged children, it is unlikely that the term data structure will be introduced, rather data. The vocabulary around data and information has not yet been well defined for the teaching and learning of digital technologies at the primary phase. Different data and information concepts are likely to be associated with different contexts. For example, in programming activities, variables will be the most common data concept that primary children will encounter. When retrieving and evaluating information from the internet, concepts such as keywords and fake news will be necessary. In sharing information, concepts such as data representation and visualisation may be encountered. If answering questions using data terms such as data collecting, data processing, managing, and analysing, data values, attributes, and data types might be introduced.

	It could also be considered particularly important that primary-aged children start to understand AI (Artificial Intelligence) and machine-learning systems' potential bias and trustworthiness, especially as the area of AI and data in education is becoming more pertinent. Trustworthiness in AI and data includes human agency, safety, privacy, transparency, diversity, non-discrimination and fairness, societal and environmental wellbeing and accountability. However, teaching about data, particularly AI and machine learning, to primary-aged children has not been well researched; therefore, classroom vocabulary, concepts, progression, pedagogy, and assessment will require careful design. Grillenberger and Romeike (2018) recently suggested a data literacy competency model, including the concepts and practices for gathering, storing, processing and visualising data for school (particularly for high school) students, as shown in Figure 5. While the CP Model incorporates dimensions that accommodate this data literacy model, if a data literacy competency model is used to develop next-level curriculum detail for primary classrooms, work will be needed to select and adapt competency statements for younger learners.
C5 Programming Languages	Fundamental to digital technologies is the concept of Programming Languages. Programming languages are used to implement designs or solutions which solve problems. There are different types of programming languages that can be categorised under programming paradigms such as procedural, functional, object-oriented, database and scripting. However, primary-aged children should not be required to understand this depth of detail. It is more important for children of this

	age group to develop a high degree of familiarity with and a deepening understanding of the three fundamental programming constructs: sequence, repetition and selection. Tangible and block-based programming languages feature in many primary curricula. However, these are not the only contexts in which primary children may learn about and start to write programs. Opportunities to learn to read and write programs may include using: • tangible programmable languages, e.g. the buttons pressed on Bee-Bot (TTS 2018) • block-based visual programming languages, e.g. Scratch (Massachusetts Institute of Technology 2021) • simple text-based programming languages, e.g. LOGO (Massachusetts Institute of Technology 2015) • functional programming languages such as used in spreadsheets, e.g. in Excel (Microsoft 2021) • developing simple web pages using menu-driven software or scripting languages, e.g. school website software, gsites (Google 2021) and HTML • context-specific programming, e.g., computer-aided design software such as Sketchup (Trimble 2021), training and programming simple machine learning activities such as Machine Learning for Kids (Machine Learning for Kids 2021), game-making software such as Construct 3 (Construct 2021), microworld simulation building software such as Minecraft (Minecraft 2021).

	Each of these programming languages may "do" different things for what may seem like the same "command". This idea that each command in each programming language, when executed, results in a specific outcome require an understanding of the idea of "notional machines". The mental model of notional machines is based on three assumptions; One, the machine is always running, two, the machine does exactly what it is told to do, and three, the programmer knows exactly how to tell the machine what to do. This term "notional machine" was introduced to computing education research in 1986 by du Boulay and is seen as a fundamental concept and one that learners must become aware of (Du Boulay 1986). The idea of the "notional machine" also links closely to the idea of code comprehension and the importance of being able to predict what a program will do when executed. Through learning about and using programming languages, pupils should develop a high degree of familiarity with and a deepening understanding of the three fundamental programming constructs. These constructs are sequence, repetition (BBC 2021a) and selection (BBC 2021b). These three constructs are sometimes called control structures, or that they provide the structure of control of the program. Linked to the *C4 Data and information* concept, children should start to understand variables, including how to set variables to a start value (initialisation) and about common patterns (design patterns) of use, such as maintaining a score. They should also learn about program events (a program in action), that more than one thing can happen at the same

	time, in parallel (concurrently), and that different types of comparisons can be made between values (operators/commands). As they learn about output devices, they will know that programs control the outputs shown to the user on these devices, such as switches (e.g. light or sound sensors) and simple outputs such as motors and LEDs. All of which can be controlled through a simple block or text-based programming languages, e.g., Crumble Controller (Redfern 2021) or the more complex Mircobit: Makecode (Microsoft 2021b). An essential aspect of learning about programming languages is the difference between the syntax of the language and the semantics. Like learning about spoken languages, there are a set of pre-defined words, commands, which must be spelt correctly and have grammatical rules about how to assemble them. Simply put, these rules are the syntax of a programming language. Conversely, just as in a spoken language, words sentence, paragraphs, and texts have a meaning, as do commands; groups of commands and programs also have meaning when programming. This meaning is called the semantics of a programming language. Being able to read programs, sometimes called code comprehension, is like reading a written text; the reader needs to understand not only the grammar but also the meaning. So, in reading programs, children need to understand the syntax and the semantics. Just as in learning a written language, children need to be able to speak it and read it before they can write it. There is strong research evidence that learners need to be able to read a program before they can write a program using the same kinds of commands (Lister 2011).

C6 Problem-solving Approaches	To solve problems, children will need approaches and techniques to support them. Simple problem-solving life cycles, such as shown in Figure 10 suggested by Marina Bers of Tufts University's Early Childhood Dev Tech Research Group (Bers *et al.* 2014), provide a structured approach. Alternatively, pupils may solve problems through exploration. **Figure 10. Computational thinking and tinkering: Exploration of an early childhood robotics curriculum (Bers *et al.* 2014, p. 155).**  Ask Imagine Plan Create Test & Improve Share This is what we made! The Engineering Design Process When solving problems in computing contexts, a popular phrase that is often coined is Computational Thinking (CT). In the NCCA review of literature on Computational Thinking "CT has been defined as combining problem solving and design to create useful solutions, informed by the possibilities that Computing offers" (Millwood *et al.* 2018, p.4). The term, CT, is often attributed to Seymour Papert (Papert 1980), who mentioned the term when describing the thinking that children exercised as they used programming to solve problems. However, algorithmic thinking, a subset of CT, has a long history in computing back to the 1950s and

	before that to the origin of the term algorithm in mathematics in the 17th Century (Denning 2009). Championed by Jeanette Wing (2006), CT is a common term in curriculum across the world (Hubwieser *et al*. 2014). At present, there is much contention as to what CT includes, how and when it should be taught and what the impact is on children's learning (Curzon *et al.* 2019; Denning, 2009). Despite the lack of consensus of exactly what CT is, there may be some pressure to call concept *C6* Computational Thinking rather than *C6 Problem-solving approaches*. As outlined in the NCCA report on the literature on Computational Thinking (Millwood *et al.* 2018), there are a number of common concepts and approaches which might be expected to be seen in a primary digital technologies' curriculum at any point in time. The Australian curriculum states: "Computational thinking is 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" (Digital Technologies Hub 2021). However, these are likely to change in line with research as the topic unfolds. For example, in recent research by García *et al.* (2019), an additional five CT concepts of classification, prediction, generation, training/validating/testing and evaluation have been suggested to be added to Brennan and Resnick's (2012) popular CT framework to cater for Machine Learning.

	A second example, in a 2020 study (Nouri *et al*. 2020) which included teacher views of digital competence and 21st-century skills when learning programming in primary and early secondary education, CT skills were added to include themes of: • cognitive skills and attitudes • language skills • collaborative skills and attitudes • creative problem-solving skills and attitudes. As a third example, in a 2020 editorial commentary on CT (Li *et al*. 2020), CT was framed such that it applies in STEM disciplines beyond computer science and mathematics. The commentary suggested that "CT is about searching for ways of processing information that are always incrementally improvable in their efficiency, correctness and elegance" (Li *et al*. 2020, p. 156), recommending that: • practice and • skill acquisition and • improvement, be added to CT. Despite this changing landscape, a curriculum needs to be devised at a point in time; therefore, programming constructs and concepts, sometimes classified as CT is added to *C5 Programming languages*. In *C6 Problem-solving approaches*, more generic problem-solving CT concepts such as abstraction, decomposition, generalisation (or spotting patterns) and logical reasoning should be included.

Processes Dimension

There are eight themes in the process dimension as shown in Figure 11 and are described in Table 3.

Figure 11. Processes Dimension – Process and Production Skills

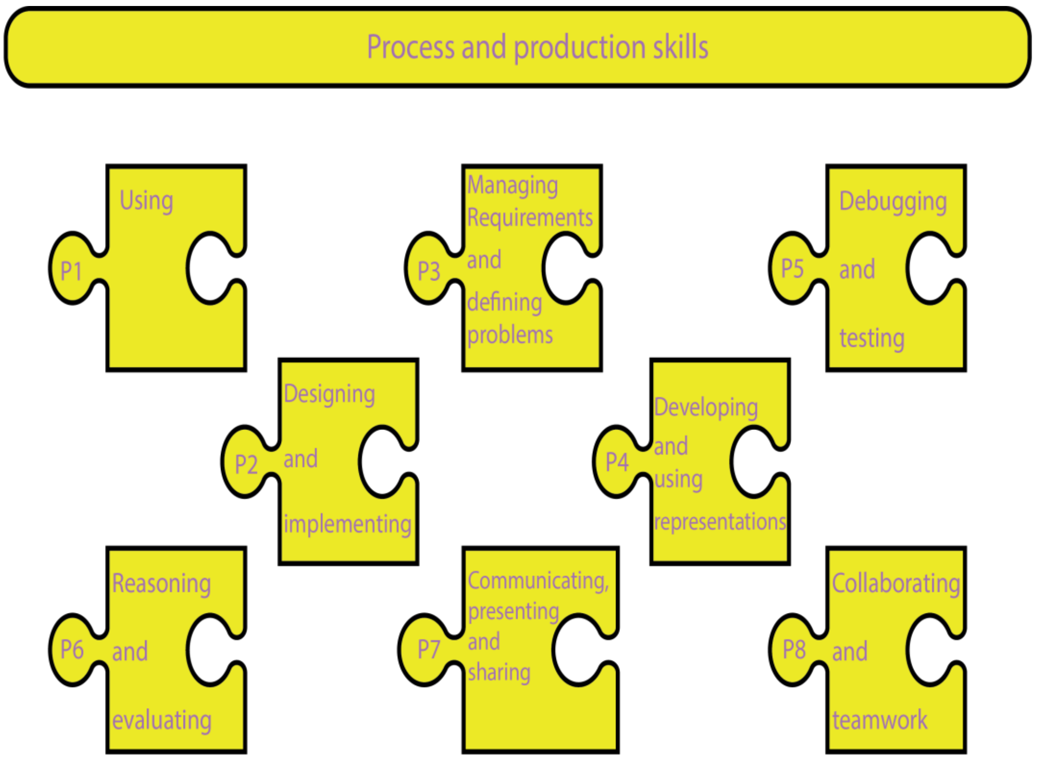

Process and production skills
P1
Using
P3
Managing Requirements and defining problems
P5
Debugging and testing
P2
Designing and implementing
P4
Developing and using representations
P6
Reasoning and evaluating
P7
Communicating, presenting and sharing
P8
Collaborating and teamwork

Table 3. Process and Production Skills

P1 **Using**	Children should use digital technologies to achieve purposeful outcomes in various relevant contexts. "Using" can manifest in many ways, for example, when children are investigating, following, or exploring. Using applies across all the primary digital technologies concepts. When children are "using" digital technologies, they should investigate how people safely use common information systems to engage with information, communication, and recreation needs - at the same time, learning about ethics, cybersecurity and internet safety. By a child

	exploring as a "user", their experience, perceptions, and expectations of technology can be highlighted to help them think more carefully about their design and implementation decisions. For example, children should consider how data is entered or information is presented so that it is accessible for all people. When children search for and retrieve data and information, they are "Using" digital technologies. By creating search criteria, children start to develop evaluation competencies as they think about suitable, reliable, and ethical data sources. Children should also begin to notice patterns in data and develop their reasoning skills. Digital technologies are always evolving and changing; therefore, children should be given early opportunities to use new and emerging types of technologies and systems, for example: mobile devices, desktop computers, robotics and other physical computing devices. Machine learning and AI (Artificial Intelligence) (tracking searches online) also provide opportunities for children to engage with technologies that have a direct impact on their lives.
P2 **Managing Requirements and Defining Problems**	Defining requirements and identifying problems includes thinking about the people (the users), what systems are needed, what data will be used or created, what the requirements (the goal) are and possible constraints. Defining problems and managing requirements are closely related skills. These skills in primary teaching and learning may be called other things such as tasks, activities, expressing ideas, plans and even exploring. Defining problems includes stating what the area of interest is (sometimes called the domain or scope), who is involved (the users), if

	there are any constraints that may affect the planned activity and what is required of the final solution (the requirements). The level of detail of any problem definition is likely to vary, but helping children evaluate the feasibility of a task is important to introduce. As they define their problem, they will start to outline the requirements. Requirements can also be called other things, such as goals, task definitions, problem outlines and aims. This may include using requirements developed by others, working out what is meant by their requirement, adding detail to, changing, adapting and creating new requirements. Requirements can include user and system specifications (at a user, software or hardware level), functional views, and Human-Computer interaction or accessibility requirements. Requirements may also be applicable to a single user or group of users. Managing requirements independently and with others will be fundamental for projects and the creation of technological solutions/artefacts that meet the needs of a person, school or the community. The recognition, definition, evaluation, management and communication of requirements apply to any digital technology problem. For example, there are requirements to be considered when making a digital presentation, developing a computer game, finding things out by searching online for information or gathering data to answer a question. Modelling how to think about the different aspects of requirements and how to define problems may be needed for children in this age group. Young learners are likely to work in contexts where they, their peers or their family will be the user. In doing so, their view of requirements may

	be narrow. Experiences where children can think more widely are essential so that they can develop skills in making digital technologies accessible to all, for example, including requirements that speech recognition/dictation, as well as typing on a keyboard, are methods for capturing data.
P3 Designing and Implementing	Designing and implementing could have been divided into two separate process and production skills. However, the line between these two process and production skills can be blurred, dependent on the approach that is being used by children to develop products. Turkle and Papert (1990) highlighted the difference between *Bricolage* and *Planning* as approaches for developing programs, suggesting that children preferred one approach or the other. Bricolage has been defined as "a practical process of learning through tinkering with materials. It involves continual transformation, with earlier products or materials that are ready to hand becoming resources for new constructions" (Sharples *et al*. 2014, p.5). More recent research has suggested a different view, that learners move between exploration and more purposeful design-based development as they learn how to program (Schulte *et al.* 2017). The process and production skill of designing and implementing apply across all primary digital technology concepts. Designing and implementing can be done as steps, for example, one activity after another for an entire project. A task can be decomposed into parts, and each part is designed and implemented in turn. Debugging and testing are processes that occur during implementation this is detailed further in *P5 Debugging and testing*.

	Designing requires children to develop and use representations (*P4*) and naturally follows or depends on managing requirements and defining problems (*P2*). Children should create designs that include thinking about: • Algorithms (for example, defining what needs to happen in terms of steps and rules) • data (for example, working out what "objects" are needed such as characters, backgrounds and scores, both in terms of the names of the "objects" and what they will look like) • systems (for example, thinking about what input and output might be needed for a user to enter data and choices, such as through a keyboard or microphone) • people and society (for example, evaluating their design to make sure it meets the needs of all users). How designs are implemented varies according to the tool or programming language used. For example, if children are implementing an animation in Scratch (Scratch 2021), they will write code which implements their algorithm, set up sprites and variables, convert artwork designs into backgrounds and sprite costumes. If they are implementing the design for a problem which involved handling data, for example using a spreadsheet to answer a question, the implementation steps will be different. Children will need to set up the spreadsheet, add any formulae they have designed and gather and add data. Whatever the context, when implementing a solution, children should think about the quality of data that is to be used. Simply put, checking

	data quality is called validation. Whether primary children need to know the term validation is not clear, but children should have the opportunity to gain related skills. When working in industry and with older learners, the term deployment is used to describe the process used to share a final product with others. Whether the term deployment should be used with younger learners is questionable; however, the concept of publishing and making work available to others is essential (linking to *C1 People* and society and safe and ethical development). For example, when children share a presentation on a school website or change the permission of a project to "public" on the Scratch community (Scratch 2021). Or when they load code to a physical computing device (Future Learn 2021) or add a link to a HTML project, they have made on social media (Code.org 2021).
P4 Developing and Using Representations	Developing and using representations could be part of *P2 Managing requirements and defining problems*, *P3 Designing and implementing* and *P5 Debugging and testing*. As a skill developing and using representations incorporates the following examples (but it is not limited to): abstracting, modelling, simulating, and representing data. Some curricula have abstraction, models, and representations as a concept, but in situations where they are listed as a concept, confusion exists as to what the concept consists of. By listing these as some of the examples of skills under the *P4 Developing and using representations*, combined with the six concept dimensions, the application generates standards that are clear.

	The process and production skill *P4 Developing, and Using Representations* applies across all primary digital technologies concepts. *P4* has three themes that underpin the skill of developing and using representations: abstracting, data representations and, modelling and simulating. Abstracting as a theme includes developing and/or using abstractions, from sequence, selection and iteration to more advanced algorithms that may include sorting and searching algorithms and perhaps may expand to measures of complexity, but the level of depth should be considered carefully for primary children. Developing and using representations includes the following examples (but it is not limited to): abstracting, modelling, simulating and representing these are fundamental for systems, algorithms and data. An algorithm is an abstraction; it is a representation of the precise steps or rules needed to solve a task. For example, children might have a set of verbal instructions of an unplugged activity of how to walk a square shape. They might say their representation of the algorithm is "take one step, turn a right angle to your left, take another same size step, turn a right angle to your left, repeat these two more times". Their representation has not included much detail about how far each "same size step" is nor where they started to take their first step from. This detail was not important for this representation. However, as they implement this representation of their algorithm as a program to draw a square, say in Scratch, they need to think about the detail that was not decided upon in their unplugged verbal representation. At this point, they

	may need to think about where to start the Sprite and the length of each side drawn. Knowing that working out the right level of abstraction for a task is important for older learners, but whether primary children need to use this vocabulary is an open question. However, the underlying skill its-self is essential to develop. Therefore, learning outcomes that focus on this concept and skill will be important, and teachers are likely to need to overtly model this concept, thinking about the right level of detail. Data representations for children refers to the skill of storing, processing, and transmitting data. Storing may include examining how numbers are used to represent data in digital systems, acquiring data from user inputs using different types of data and validating the data. Processing the data may include data cleaning and sorting and transmitting data may be between files and systems or between systems themselves. Finally, transforming data into information (numerically or visually/graphically) to solve problems or provide insights is a capstone to data representations. The final theme of modelling and simulating will combine pupils programming skills and other skills such as visualising, to model systems, from software, physical or perhaps other systems such as biological systems. These models and simulations will help children form and test expectations of a systems behaviour. Primary children are likely to work with simple ways to organise data, such as variables, lists/arrays, and tables. When working with data representations, children will need to decide what level of detail they need to capture and what kind of data they are going to be using, such

	as text, numbers, a yes or no type of data item, an image, a sound, or a video clip. When sharing data, children need to think again about how they will represent the information they are communicating and what level of detail is needed and in what format. Skills related to the use and creation of models and simulations of real-world and imaginary contexts require children to work out what are the parts of the contexts they are representing (abstracting). The contexts that are to be used in primary classrooms are likely to be from and across other subjects. For example, in science, children could model the life cycle of a butterfly and share this using a presentation. In geography, they could model what happens when a volcano explodes using an animation created in Scratch (Scratch 2021). In maths, children could model the costs of running a party by creating a spreadsheet and by exploring how much we can spend on different items. They could also simulate traffic lights by wiring up sensors, coloured lights and a micro-controller (Redfern 2021) and program the action using a simple block-based programming language (Weintrop 2019). In all these examples, children need to work out what objects they need to represent, what are the important attributes of these objects, what they need to include, and what they can ignore. Once they have decided on this and created a design with this information, say as a labelled diagram, they can implement their design. Implementation can be through named images and text blocks in presentation software; sprites, costumes, variables and lists in Scratch; and columns in spreadsheets.

P5 Debugging and Testing	Debugging and testing could be included in the process and production skill of *P6 Reasoning and evaluating* but developing this expertise is crucial in the context of digital technologies, therefore these skills merit a distinct theme. Simply put, a bug is when something is not quite right with an algorithm or program. Children can identify bugs when they follow an algorithm or run a program when it does not do what they expect. However, this simplistic view hides a wide range of reasons that might have caused the bug. Reasons might be that they did not understand how the programming language commands work, or they have used the wrong command to implement a step in our design, or there may be a problem with their algorithm or another aspect of their design. To debug an algorithm or program, learners will have to find and fix the bug. Children should be encouraged to expect bugs and to be patient and resilient in finding and fixing them. There is a difference between the terms debugging and testing. Debugging is used to describe a reaction to something which has gone wrong, whereas testing is planned. Testing is the process of designing specific tests to ensure that our solution will meet the requirements. Testing is the skill to find potential bugs before they happen. In primary classrooms, testing is likely to be quite simple. For example, in a Scratch quiz (Scratch 2021), children might plan to test entering the right and the wrong answer. Children might test what happens if they enter a string of characters in response to a question which they designed to take a number. Or investigate what happens if they make a very loud noise and a very quiet

	noise in a project they made that is sound activated. During testing, children might ask their friends to “break” their solution, encouraging them to think of all the unexpected things they could do. As children begin to become more familiar with the different types of bugs that they encounter, they can start to be more systematic about testing. Class checklists of typical bugs and good tests can be developed. Through the use of high-quality program examples as models, children may start to recognise how they can design and implement solutions that make testing and debugging easier. For example, good naming of things is a key skill, such as names of characters and backgrounds in designs, sprites and variables in programs, functions and sheets in spreadsheets, templates in presentations, heading types in word processing documents.
P6 **Reasoning and Evaluating**	Reasoning and evaluating are distinct but similar skills, both of which are sometimes included in Computational Thinking descriptions (Millwood *et al*. 2018). Reasoning, sometimes called logical reasoning, is the process by which learners use existing knowledge to predict an outcome and then analyse what occurs to change or confirm their ideas. By using reasoning, children can justify explanations and interpretations. Learners may start with a specific case to develop their reasoning (inductive) or may begin with a general view (deductive). How significant these two forms of logical reasoning are to the development of digital technologies skills in primary classrooms is not yet clear; however, learning how to learn from personal experiences and general theories seem essential in developing a deep understanding of complex concepts.

	Evaluation, on the other hand, requires learners to compare something against a set of criteria. After having made a comparison, learners are likely to use reasoning to make judgements about the aspect that is being evaluated. In the teaching of primary level programming, care should be taken in teaching children to evaluate one particular concept: the efficiency of programs. While it may be unlikely that children will be able to have an impact on the efficiency of the programs, it is more appropriate that children learn to evaluate readability, reuse and elegance of the programs they encounter. The term analysing is within the scope of this process and production skill, as learners break things down into parts that they can then study in more detail to reason and evaluate.
P7 Communicating, Presenting and Sharing	Communicating, presenting, and sharing are general problem-solving skills needed by children when they work with digital technology. Children will present their ideas and information and learn about different forms of representations to suit different needs (see *P4 Developing and using representations)*. In doing this, children will learn about and start to act ethically, safely and responsibly. When learning about ethics, children will reflect on digital technologies with regard to diversity, discrimination and fairness. With respect to safety, learners will start to think about potential environmental, societal, and personal harm that digital systems can cause. In the context of responsible use, Children should also consider where data has originated from, who has developed systems and be introduced to the notion of ownership and accountability.

P8 Collaborating and Teamwork	Sometimes included as a Computational Thinking (CT) approach (Millwood *et al*. 2018), collaborating and teamwork are process and production skills that should be fostered not only to support general problem solving but also to gain specific digital technologies skills. For example, pair programming is a standard industry technique and working in teams to complete projects is common. There is some evidence that this approach may help to make learning to program in educational settings easier; however, there is also research that counters this view (Waite 2017a). Further research is needed on the best way to support children to learn about and whether to use this technique.

4. Summary

Applying the Concept and Processes Model during Curriculum Development

The next steps for using the proposed CP Model would be to elaborate the Concept and Process themes as more detailed statements. These statements will inform the development of the digital technology specifications and learning outcomes for primary-aged children. As an example of how the USA K-12 framework practices and concepts are converted to standards, a form of learning outcome is shown in Figure 12. In the same way, the CP Model could be used to inform learning outcome development as shown in Figure 13 and Figure 14, where the CP Model themes are mapped to the Australian curriculum statements.

Figure 12. Example of integrating a practice and a concept to create a standard from Figure 7.9 in K-12 Framework, (2016, p. 139).

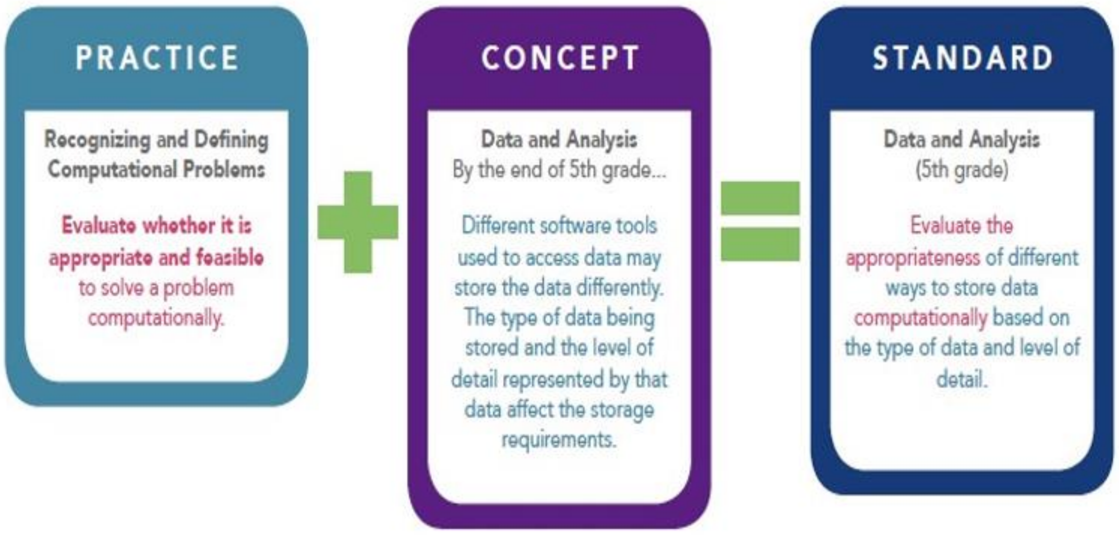

PRACTICE
Recognizing and Defining Computational Problems
Evaluate whether it is appropriate and feasible to solve a problem computationally.
CONCEPT
Data and Analysis
By the end of 5th grade...
Different software tools used to access data may store the data differently. The type of data being stored and the level of detail represented by that data affect the storage requirements.
STANDARD
Data and Analysis
(5th grade)
Evaluate the appropriateness of different ways to store data computationally based on the type of data and level of detail.

Figure 13. Using the CP Model themes to represent an Australian curriculum learning outcome/statement (Australian Curriculum Assessment and Reporting Authority 2021a).

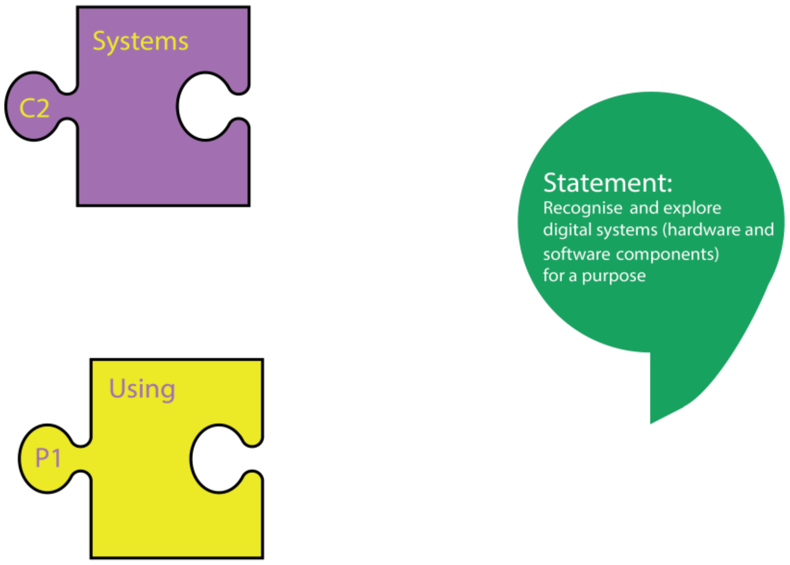
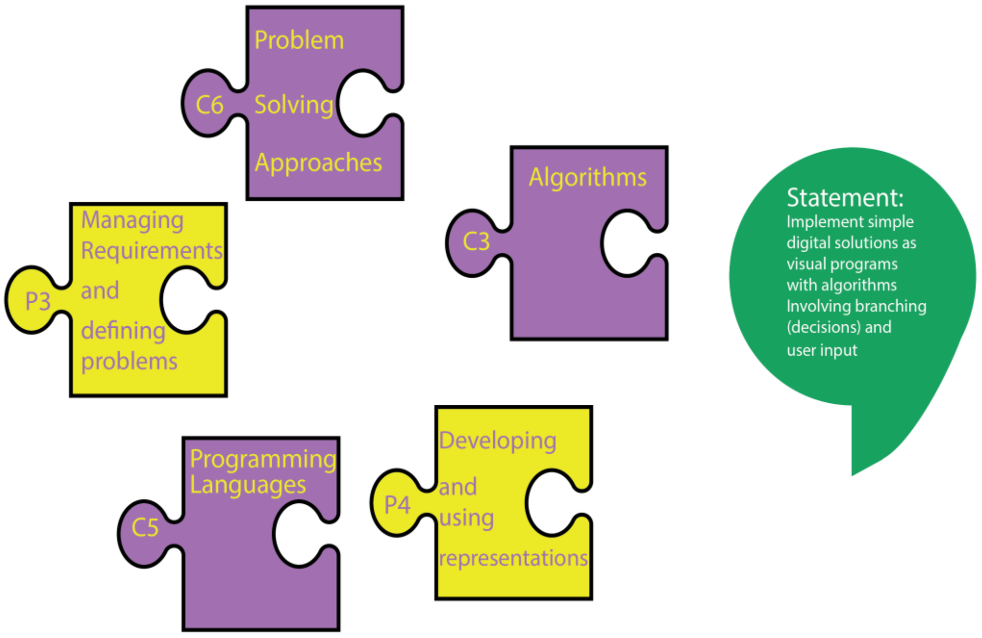

Systems
C2
Using
P1
Statement:
Recognise and explore digital systems (hardware and software components) for a purpose

Figure 14. Using the CP Model themes to represent an Australian curriculum learning outcome/statement where multiple concepts and processes are involved (Australian Curriculum Assessment and Reporting Authority 2021a).

.


Problem Solving Approaches
C6
Algorithms
C3
Managing Requirements and defining problems
P3
Statement:
Implement simple digital solutions as visual programs with algorithms Involving branching (decisions) and user input
Programming Languages
C5
Developing and using representations
P4

As guidance for curriculum designers, each theme in the CP Model does not hold the same weight. Children at different points in progression may attend more or less to certain themes than others; for example, young children may spend more time on *P1 Using* and *C2 Systems* as they find out what is possible when using a particular software application. However, curriculum designers may decide that it is important for children to quickly start making their own artefacts and balance *P1 Using* with *P3 Designing and implementing*. Other curriculum designers may think that young children should be thinking more deeply and introduce activities which promote *P6 Reasoning and evaluating*, for example, asking children to predict what programs do, and introducing code comprehension early. This will depend on the pedagogy that is being adopted. It is important to ensure that the knowledge and understanding of concepts are included as learning outcomes as well as the process and productions skills that put them into practice otherwise, there is a risk that practical activities are not linked to the underlying concepts that are being introduced.

The CP Model serves only as broad headings to support elaboration rather than acting as a constraint. For example, there is no requirement to say that for every child in any year that they must encounter a learning outcome for each and every combination of the CP Model. Learners should follow a path of learning outcomes as a progression for each theme. What this progression looks like will depend on many things, including theories related to how children learn, including what misconceptions are likely and must be overcome, pedagogy and teacher knowledge and confidence.

Theories related to knowledge building such as Maton's (2013) Semantic Waves which has been applied to learning about algorithms (Waite *et al*. 2019), and Van Merriënboer et al's (2002) complex learning model may be particularly useful to support the development of both the concepts and process and production skills, needed to solve programming problems.

Considerations for Teaching and Learning

This document does not provide a review of pedagogy for teaching digital technologies in primary classrooms, but when mapping statements to themes, certain approaches became evident and could be easily incorporated. The *P1 Using* concept theme neatly fits with the Use, Modify, Create (Lee *et al*. 2011) approach of starting sequences of learning activities by using artefacts created by others and gradually increasing pupil ownership as they modify and then create their own products. Building on this, approaches such as PRIMM (Predict, Run, Investigate, Modify, Make stages) (Sentance *et al.* 2019) incorporate *P6 Reasoning and Evaluating* through code reading in the predict and investigate stages of a sequence of activities. In this approach there are links to the use of a simple idea of levels of abstraction (Waite *et al.* 2017b), design including the algorithm, programming and running the program may be useful to help children understand how *P2 Managing Requirements and Defining Problems*, *C3 Algorithms and P3 Designing and Implementing*. Also, the importance of talk, in the development of deep understanding (Schulte *et al.* 2017). Including both design and implementation in *P3 Designing and Implementing* process and production skill provides curriculum designers with a choice of a Turkle & Papert's "Bricolage" or "Planning" (Turkle and Papert 1990) approaches or combination of these (Schulte *et al.* 2017) to *C6 Problem-solving Approaches*. For learning and assessment activities, a balance of open-ended culturally inclusive creative projects, as well as narrower targeted tasks, is recommended by some authors (Waite 2017a). Sequences of targeted tasks such as code comprehension activities (Izu *et al*. 2019), including code tracing activities, debugging activities and Parson's problems, (Raspberry Pi 2021) should be considered to focus on specific concepts or process and production skill development. The importance of creativity and the inclusion of all children should not be forgotten through contexts, pedagogy and in understanding how children learn.

How Children Learn Digital Technologies in School

There has been very limited research on how children learn digital technologies in primary settings, including what misunderstandings may arise (Waite 2017). Misconceptions are likely to be affected by both the pedagogy used and the progression of conceptual learning. In computing, concepts are often abstract and complex. Therefore, simplifying complex abstract ideas for young children is needed to make them more accessible. In simplifying concepts and skills, misconceptions may be introduced on purpose with the intention that these will be addressed later. The misconception is used to bridge a complex progression in learning, developing an intermediary mental model that is a "staging post" to reduce complexity and cognitive load.

For example, very young children may think of a variable as being like a box. This simile is a popular way to introduce an abstract concept, and there has been evidence that it helps develop understanding; however, it introduces misconceptions (Hermans *et al.* 2018). The misconceptions inherently introduced by the box analogy, are likely at some point, will need to be tackled. Resolving the misconception may occur gradually, as children learn about computer architecture.

Other early digital technologies misconceptions, which have been identified, include children thinking that computers are like people. This is a serious misconception, as computers are deterministic machines, which only do what they are programmed to do or what the data they have been trained on reveals. Computers do not have free will; they cannot think. However, very young children personify inanimate objects as a matter of course. There has been limited research on the impact of personification of computers has on the teaching of digital technologies. One impact has been suggested is the "intentionality bug" (Sentance *et al.* 2018), whereby children think a computer is like a person and will "know" more than that which has been programmed. In the same strand of research, there is a proposed “egocentrism bugs”, where children think that a programming command will do what they think it should, rather what it can do. With an overarching “superbug”, the idea that there is a “hidden mind” in the programming language that is intelligent and knows what is intended and may help you out (Grover and Basu 2017). As this potential misconception may have a wide-reaching impact on children’s views of computers and their competency as both users and programmers when things go wrong, it may be useful for curriculum design to incorporate learning outcomes that address the “superbug” misconception early.

Juha Sorva, is probably one of the most well-known researchers who has looked at misconceptions and the beginner programmer. Despite much of his research focusing on text-based programming languages and older children, his work provides a good starting point for consideration of this area for primary-aged children (Sentance *et al.* 2018). In conjunction with this, emerging research on block-based programming and younger children by Grover & Basu (2017) and Hermans *et al*. (2018) should also be considered.

Curriculum designers also need to consider the concepts and skills to be learned, the progression of mental models, expected ways to teach concepts and potential misconceptions that may be introduced, the impact of misconceptions and how and when they can be addressed.

Teacher Knowledge and Confidence

Teachers will need to be supported to develop their knowledge of the CP Model themes, how they can be combined, the learning outcomes, progression, misconceptions and context opportunities. Teacher confidence will be important, as teachers will be required to develop or adapt activities to meet children current competency levels and ensure that activities are accessible to all, fun, relevant, and culturally appropriate. Teacher confidence is often referred to as CS (computer science) self-esteem, self-concept or self-efficacy. Even in the literature, there is a lack of agreement on the construct meanings (Bailey 2003; Brown and Marshall 2006).

It has been shown that for children learning computer science, this construct of "confidence" relates strongly to performance (Quille and Bergin 2019) and that teachers have significantly lower computing self-esteem than children. More specifically, primary school teachers have been found to have lower computing confidence than second-level teachers (Vivian *et al*. 2020), which also should be considered when designing, developing and implementing professional development programmes.

For teachers to build confidence, it may take at least 4 to 5 years (Vivian *et al*. 2020). Age or teaching location (Metro, Urban, Rural and remote) did not highlight confidence differences, which is positive news for the Irish context with the sparse national location of primary schools. As with many previous studies at third level (Quille and Bergin 2019), female teachers report lower confidence than their male counterparts (Vivian *et al*. 2020).

All of this suggests that the success of new curricula could perhaps depend on developing confidence through professional development. The majority of teachers will not have formal computer science education training prior to the implementation of the curricula, with their first experience of formal computing education perhaps being a professional development programme.

5. Conclusion

The authors have reviewed and built upon four strands of earlier work and research to develop a primary digital technology competency model. These strands of work are:

- preceding Irish National for Council Curriculum and Assessment Coding in Primary Schools Initiative Reports (NCCA 2016; NCCA 2018) including outcomes from the school pilots of teaching coding (NCCA 2019)
- preceding Irish National Council for Curriculum and Assessment general research on competencies and primary curriculum reports (McGuinness 2018)
- research on digital technology competencies
- current digital technology curricula.

Rather than restricting the review to the understanding digital technologies, the authors looked at both knowledge and skills as it was not possible to separate the two when taking a competency-based approach.

As expert practitioners, policymakers and researchers work together to develop the curriculum different views will be heard which will need to be carefully evaluated. International research should take a high profile in supporting this evaluation. Although practically applied research in primary computing is limited, many new ideas are currently being developed to improve the teaching and learning of primary digital technologies, these should be taken advantage of in the creation of the Irish iteration of digital technology in the Primary School Curriculum.

References

Australian Curriculum, Assessment and Reporting Authority (2021a) *Digital Technologies in focus*, available: https://www.australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies/?year=12983&strand=Digital [accessed 25 September 2021].

Australian Curriculum, Assessment and Reporting Authority (2021b) *General capabilities* available: https://www.australiancurriculum.edu.au/f-10-curriculum/general-capabilities/ [accessed 25 September 2021].

Bailey, J.A. (2003) The foundation of self-esteem, Journal of the National Medical Association, 95 5, pp. 388-93 available: https://www.semanticscholar.org/paper/The-foundation-of-self-esteem.Bailey/57b7f7b2b7430a631331a9f0311f66c4422ae863 [accessed 25 September 2021].

BBC (2021a) *Bitesize: Computational constructs*, available: https://www.bbc.co.uk/bitesize/guides/zcg9kqt/revision/7#:~:text=Repetition%20in%20a%20program%20means,you%20achieve%20the%20correct%20outcome [accessed 12 May 2021].

BBC (2021b) *Bitesize: Selection on programming*, available: https://www.bbc.co.uk/bitesize/guides/z2p9kqt/revision/1 [accessed 12 May 2021].

Bers, M.U., Flannery, L.P., Kazakoff, E.R. and Sullivan, A. (2014) Computational thinking and tinkering: Exploration of an early childhood robotics curriculum, *Computing. Education.*, *72*, 145-157, available: https://www.semanticscholar.org/paper/Computational-thinking-and-tinkering%3A-Exploration-Bers-Flannery/2b46c2e979f15170de15f6f372b944ac7c87c9a2 [accessed 12 May 2021].

Brennan, K. and Resnick, M. (2012) New Frameworks for Studying and Assessing the Development of Computational Thinking, *in proceedings of the 2012 Annual Meeting of the American Educational Research Association*, *April 2012*, p. 25, available: http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf [accessed 12 May 2021].

Brown, J. D. and Marshall, M. A. (2006). The three faces of self-esteem, in *M. Kernis (Ed.)*, *Self-esteem: Issues and answers*, pp. 4-9, New York: Psychology Press available: http://faculty.washington.edu/jdb/448/448articles/kernis.pdf [accessed 12 May 2021].

Code.org (2021) *Unit 2 - Web Development ('21-'22)*, available: https://studio.code.org/s/csd2-2021 [accessed 25 September 2021].

Computer Science Teachers Association (2020) *About CSTA's K-12 Standards*, *available*: K-12 Standards | Computer Science Teachers Association (csteachers.org) [accessed 25 September 2021].

Construct (2021) *MAKE GAMES WITH CONSTRUCT 3*, available: https://www.construct.net/en [accessed 18 January 2021].

Curzon, P., Bell, T., Waite, J. and Dorling, M. (2019) Computational thinking, in SA Fincher & AV Robins (eds), *The Cambridge Handbook of Computing Education Research*, Cambridge Handbooks in Psychology, Cambridge:

Jane Waite
Keith Quille

Cambridge University Press, pp. 513–546, available: https://qmro.qmul.ac.uk/xmlui/handle/123456789/57010 [accessed 12 May 2021].

Denning, P. (2003). Great Principles of Computing. *Communications of the* ACM, 46. 15-20, available: https://www.researchgate.net/publication/220426836_Great_Principles_of_Computing [accessed 12 May 2021].

Denning, P. (2009) The Profession of IT Beyond Computational Thinking, *Communications of the ACM*, 52. 28-30, available: https://www.researchgate.net/publication/220420930_The_Profession_of_IT_Beyond_Computational_Thinking [accessed 18 January 2021].

Digital Technologies Hub (2021) *Computational Thinking*, available: https://www.digitaltechnologieshub.edu.au/teachers/topics/computational-thinking [accessed 25 September 2021].

Du Boulay, B. (1986) 'Some Difficulties of Learning to Program', *Journal of Educational Computing Research*, 2(1), pp. 57–73. doi: 10.2190/3LFX-9RRF-67T8-UVK9. [accessed 18 January 2021].

Education Scotland (2021) *Experiences and outcomes*, available: https://education.gov.scot/education-scotland/scottish-education-system/policy-for-scottish-education/policy-drivers/cfe-building-from-the-statement-appendix-incl-btc1-5/experiences-and-outcomes/ [accessed 25 September 2021].

Future Learn (2021) *What is physical computing?* available: https://www.futurelearn.com/info/courses/physical-computing-raspberry-pi-python/0/steps/76165 [accessed 18 January 2021].

Google (2021) *Sites*, available: https://sites.google.com/new [accessed 18 January 2021].

Grillenberger, A., & Romeike, R. (2018) Developing a theoretically founded data literacy competency model, Proceedings of the 13th Workshop in Primary and Secondary Computing Education, available: https://www.semanticscholar.org/paper/Developing-a-theoretically-founded-data-literacy-Grillenberger-Romeike/c5e45f6231d9fdd3b0689f4df157829244d41972 [accessed 12 May 2021].

Grover, S. and Basu, S. (2017) Measuring Student Learning in Introductory Block-Based Programming: Examining Misconceptions of Loops, Variables, and Boolean Logic, *in Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (SIGCSE '17), New York ACM, pp. 267–272, available: https://doi.org/10.1145/3017680.3017723 [accessed 25 September 2021].

Hermans, F., Swidan, A., Aivaloglou, E. and Smit, M. (2018) Thinking out of the box: comparing metaphors for variables in programming education. *Proceedings of the 13th Workshop on Primary and Secondary Computing Education* (online), available: https://doi.org/10.1145/3265757.3265765 [accessed 25 May 2021].

Hubwieser, P., Armoni, M., Giannakos, M. and Mittermeir,R. (2014). Perspectives and Visions of Computer Science Education in Primary and Secondary (K-12) Schools, *ACM Transactions on Computing Education*, 14. 7:1-9, available: https://www.researchgate.net/publication/264626324_Perspectives_and_Visions_of_Computer_Science_Education_in_Primary_and_Secondary_K-12_Schools [accessed 12 May 2021].

Izu, C., Schulte, C., Aggarwal, A., Cutts, Q., Duran, R., Gutica, M., Heinemann, B., Kraemer, E., Lonati, V., Mirolo, C. and Weeda, R. (2019) Fostering Program Comprehension in Novice Programmers - Learning Activities and Learning Trajectories, in Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR '19), New York: ACM, NY, pp. 27–52. DOI: https://doi.org/10.1145/3344429.337250 [accessed 18 01 2021].

K–12 Computer Science Framework (2016) *The K–12 Computer Science Framework, led by the Association for Computing Machinery, Code.org, Computer Science Teachers Association, Cyber Innovation Center, and National Math and Science Initiative in partnership with states and districts, informed the development of this work*, available: https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf [accessed: 18 January 2021].

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. and Werner. L. (2011). Computational thinking for youth in practice, ACM Inroads 2, 1, pp. 32–37, available: https://doi.org/10.1145/1929887.1929902 [accessed 12 May 2021].

Lister, R, (2011) Concrete and other neo-Piagetian forms of reasoning in the novice programme, *Conferences in Research and Practice in Information Technology Series*, pp. 9 – 18, NSW: Open Publications of UTS Scholars, available: https://opus.lib.uts.edu.au/handle/10453/17580 [accessed 12 May 2021].

Machine Learning for Kids (2021) About, available: https://machinelearningforkids.co.uk/#!/about [accessed 18 January 2021].

Magenheim, J., Nelles, W., Rhode,T., Schaper,N., Schubert.S and Stechert, P. (2010) Competencies for informatics systems and modeling: Results of qualitative content analysis of expert interviews, *IEEE EDUCON 2010 Conference*, 513-521, available: https://ieeexplore.ieee.org/document/5492535?reload=true [accessed 12 May 2021].

Massachusetts Institute of Technology - MIT (2015) *Logo programming*, Boston: MIT, available: https://el.media.mit.edu/logo-foundation/what_is_logo/logo_programming.html [accessed 18 January 2021].

Massachusetts Institute of Technology - MIT (2021) *Scratch programming*, Boston: MIT, available: https://scratch.mit.edu/ [accessed 18 January 2021].

Maton, K. (2013) Making semantic waves: A key to cumulative knowledge-building, Linguistics and Education, 24, 1 pp. 8-22, available: https://www.sciencedirect.com/science/article/abs/pii/S0898589812000678 [accessed 12 May 2021].

McGuinness, C. (2018) *Research-Informed Analysis of 21st Century Competencies in a Redeveloped Primary Curriculum - NCCA key competencies McGuinness Final Report*, Dublin: NCCA, available: https://ncca.ie/media/3500/seminar_two_mcguinness_paper.pdf [accessed 15 June 2021].

Microsoft (2021a) *Using functions and nested functions in Excel formulas*, available: https://support.microsoft.com/en-us/office/using-functions-and-nested-functions-in-excel-formulas-3f4cf298-ded7-4f91-bc80-607533b65f02 [accessed 18 01 2021].

Microsoft (2021b) *Micro:bit*, available: https://makecode.microbit.org/ [accessed 18 January 2021].

Millwood, R., Bresnihan, N., Walsh, D., and Hooper, J. (2018) *Review of Literature on Computational Thinking*, Dublin: NCCA, available: https://ncca.ie/en/resources/primary-coding_review-of-literature-on-computational-thinking/ [accessed 21 May 2021].

Minecraft (2021) *Makecode*, available: https://minecraft.makecode.com/ [accessed 15 June 2021].

National Council for Curriculum and Assessment (2016) *Desktop audit of coding in the primary curriculum of 22 jurisdictions*, Dublin: NCCA, available: https://ncca.ie/en/resources/primary-coding_desktop-audit-of-coding-in-the-primary-curriculum-of-22-jurisdictions/ [accessed 21 May 2021].

National Council for Curriculum and Assessment (2018) *Investigation of curriculum policy on coding in six jurisdictions*, Dublin: NCCA, available: https://ncca.ie/en/resources/primary-coding_investigation-of-curriculum-policy-on-coding-in-six-jurisdictions/ [accessed 21 May 2021].

National Council for Curriculum and Assessment (2019) *Final report on the Coding in Primary Schools Initiative*, Dublin: NCCA, available: https://ncca.ie/en/resources/primary-coding_final-report-on-the-coding-in-primary-schools-initiative/ [accessed 21 May 2021].

National Council for Curriculum and Assessment (2020) *Draft Primary Curriculum Framework*, Dublin: NCCA, available: https://ncca.ie/en/resources/ncca-primary-curriculum-framework-2020pdf/ [accessed 15 June 2021].

New York: *AMC*, available https://doi.org/10.1145/3265757.3265766 [accessed 12 May 2021].

Nouri, J., Zhang, L,. Mannila, L. and Norén, E. (2020) Development of computational thinking, digital competence and 21st century skills when learning programming in K-9, Education Inquiry, available: https://www.tandfonline.com/doi/full/10.1080/20004508.2019.1627844 [accessed 21 May 2021].

Papert, S, A. (1980) *Mindstorms: Children, computers, and powerful ideas*, New York: Basic books Inc publishers, available: https://mindstorms.media.mit.edu/ [accessed 18 Jan 2020].

Quille, K. and Bergin, S. (2019) CS1: how will they do? How can we help? A decade of research and practice, Computer Science Education, 29:2-3, pp. 254-282, DOI: 10.1080/08993408.2019.1612679

Raspberry Pi (2021) Pedagogy Quick Reads Improving program comprehension through Parson's Problems, available: https://raspberrypi-education.s3-eu-west-1.amazonaws.com/Quick+Reads/Pedagogy+Quick+Read+13+-+Parson's+Problems.pdf [accessed 21 May 2021].

Redfern (2021) *The Crumble Controller*, available: https://redfernelectronics.co.uk/crumble/ [accessed 15 June 2021].

Rodríguez, J., Moreno-León, J., Román-González, M. and Robles, G. (2020) LearningML: A Tool to Foster Computational Thinking Skills Through Practical Artificial Intelligence Projects Revista de Educación a Distancia (RED), available: https://www.researchgate.net/publication/340584246_LearningML_A_Tool_to_Foster_Computational_Thinking_Skills_Through_Practical_Artificial_Intelligence_Projects/citation/download [accessed 15 June 2021].

Schulte, C., Magenheim, J., Müller, K. and Budde, L. (2017) The design and exploration cycle as research and development framework in computing education, *2017 IEEE Global Engineering Education Conference (EDUCON)*, available: https://www.researchgate.net/publication/317418777_The_design_and_exploration_cycle_as_research_and_development_framework_in_computing_education, [accessed 18 Jan 2021].

Scratch (2021) *Create stories, games, and animations Share with others around the world*, available: https://scratch.mit.edu/ [accessed 7 May 2021].

Sentance, S., Barendsen, E. and Schulte, C. (2018) *Computer Science Education: Perspectives on Teaching & Learning in Schools*, London: Bloomsbury Publishing, available: https://www.bloomsbury.com/uk/computer-science-education-9781350057111/ [accessed 18 01 2020].

Sentance, S., Waite, J. and Kallia, M. (2019) *Teaching computer programming with PRIMM: a sociocultural perspective*. Computer Science Education (online), 29(2), pp.136-176, London: available: https://www.tandfonline.com/doi/abs/10.1080/08993408.2019.1608781 [accessed 7 May 2021].

Sharples, M., Adams, A., Ferguson, R., Gaved, M., McAndrew, P., Rienties, B., Weller, M. and Whitelock, D (2014). Innovating Pedagogy, Online: The Open University, available: https://www.learntechlib.org/p/149392/ [accessed 18 Jan 2021].

Torsten, B., Hermann, P. Schulte, C. (2009) Bridging ICT and CS: educational standards for computer science in lower secondary education. *14th annual ACM SIGCSE conference on Innovation and technology in computer science education (ITICSE 2009)*, available: *DOI:10.1145/1562877.1562965* [accessed 7 May 2021].

Trimble (2021) *SketchUp*, available: https://www.sketchup.com/ [accessed 18 01 2021].

TTS (2018) 'Bee-Bot - a teacher's guide', *The Blog*, 18 July, available: https://www.tts-group.co.uk/blog/2018/07/18/bee-bot-a-teachers-guide.html [accessed 15 June 2021].

Turkle, S., and Papert, S. (1990) Epistemological Pluralism: Styles and Voices within the Computer Culture. Signs, 16(1), 128–157, Chicago: University of Chicago Press, available: http://www.jstor.org/stable/3174610 [accessed 20 July 2020].

Van Merriënboer, J. J. G., Clark, R. E. and de Croock, M. B. M. (2002) Blueprints for complex learning: The 4C/ID-Model, *Educational Technology Research and Development*, 50(2), 39–64, available: https://psycnet.apa.org/doi/10.1007/BF02504993 [accessed 20 July 2020].

Vivian, R., Quille, K., McGill, M.M., Falkner, K., Sentance, S. Barksdale, S., Busuttil, L., Cole, E., Liebe, C. and Maiorana. F. (2020) An International Pilot Study of K-12 Teachers' Computer Science Self-Esteem, in Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '20). Association for Computing Machinery, New York: AMC, pp. 117–123, available: https://dl.acm.org/doi/abs/10.1145/3341525.3387418 [accessed 7 May 2021].

Waite, Curzon, P., Marsh, W. and Sentance, S. (2017b) K-5 Teachers' Uses of Levels of Abstraction Focusing on Design, Proceedings of the 12th Workshop on Primary and Secondary Computing Education (WiPSCE

'17). Association for Computing Machinery, New York: AMC, 115–116. DOI: https://doi.org/10.1145/3137065.3137068 [accessed 20 July 2020].

Waite, J. (2017) *Pedagogy in Teaching Computer Science in schools: A Literature Review* (After The Reboot: Computing Education in UK Schools), available: https://royalsociety.org/~/media/policy/projects/computing-education/literature-review-pedagogy-in-teaching.pdf [accessed 5 October 2021].

Waite, J., Maton, K., Curzon, P. and Tuttiett, L. (2019) Unplugged Computing and Semantic Waves: Analysing Crazy Characters, available: https://www.researchgate.net/publication/335527816_Unplugged_Computing_and_Semantic_Waves_Analysing_Crazy_Characters [accessed 15 June 2021].

Weintrop, D. (2019) Block-based programming in computer science education, *Communications of the ACM, ACM*, 62, 8, pp. 22–25, available: https://dl.acm.org/doi/fullHtml/10.1145/3341221 [accessed 15 June 2021].

Wing, J. (2006) Computational Thinking, *Commun. ACM*, 3, pp. 33,35, New York: Association for Computing Machinery, available: https://doi.org/10.1145/1118178.1118215 [accessed 7 May 2021].

Jane Waite
Keith Quille