The Australian Digital Technologies Curriculum: Challenge and Opportunity

Katrina Falkner **Rebecca Vivian** **Nickolas Falkner**
School of Computer Science
The University of Adelaide
Adelaide, South Australia

Firstname.lastname@adelaide.edu.au

Abstract

There is a call for change in the treatment of ICT curriculum in our schools driven by the relatively recent acknowledgement of the growing importance of ICT in industry and society, and the need to empower youth as producers, as well as consumers, of technology. ICT curriculum in previous incarnations tended to focus on ICT as a *tool*, with the development of digital literacy as the key requirement. Areas such as computer science (CS) or computational thinking were typically isolated into senior secondary programs, with a focus on programming and algorithm development, when they were considered at all. New curricula introduced in England, and currently under debate within Australia, have identified the need to educate for *both* digital literacy and CS, and the need to promote both learning areas from the commencement of schooling, Foundation (F) to year 12.

In this paper, we discuss the main trends and learning objectives of these new curricula, identifying key areas requiring further research and development by the CS Education community. We undertake a review of current research in CS Education within the F-12 context, to identify research that can guide effective implementation and provide opportunities for further research.

Keywords: National curriculum, computer science, informatics, education, primary school, high school.

1 Introduction

Over the last decade, the need to rethink our education systems in terms of the treatment of computer science (CS) and information technology has gained global attention (Gander et al., 2013; Seehorn et al., 2011; The Royal Society, 2012). We struggle to attract potential students and to promote CS as a creative, engaging career, despite the growing need for CS professionals. Recent US statistics indicate that only 2% of SAT takers intending to pursue college degrees intend to major in CS (College Board, 2012). The "Shut down or restart?" report by The Royal Society (2012) states: "despite the near-ubiquity of computer technology, there is now a dwindling interest in studying Computing at school". Considerable research has explored the reasons behind this disparity, focussing on negative career perceptions, the identity issues caused by the confusion of CS with the simplistic application of ICT tools (Schulte et al., 2012), gender differences (Henwood, 2000) and other stereotypes (Jepson & Perl, 2002).

Over the past decade we have witnessed a transition in ICT education from ICT as a *tool* - with the development of digital literacy as the key requirement - moving toward understanding the underpinning concepts and workings of ICT. Areas such as CS or computational thinking were typically isolated into senior secondary programs, with a focus on programming and algorithm development, when they were considered at all. Despite the recognised need for CS education, schools are "failing to provide students with access to the key academic discipline of CS, despite the fact that it is intimately linked with current concerns regarding national competitiveness" (Gal-Ezer and Stephenson, 2009).

Recent reports from the US and Europe have argued that it is essential that children be exposed to CS concepts and principles from the very start of their education so that "every child [may] have the opportunity to learn Computing at School" (Gander et al., 2013; Wilson & Guzdial, 2010). This is a driver for CS to be taught in school, as early as the first year. Encouraging students to engage in current technologies and participate as creators of future technologies requires more than teaching the fundamentals of digital literacy – familiarity with the tools and approaches to interact with technology. We must also teach computational thinking, the problem solving processes and intellectual practices needed to understand the scientific practices that underpin technology. Without this, we face the risk of our youth being placed in the position of consumers of technology produced elsewhere, unable to actively participate as producers and leaders in this field (Gal-Ezer & Stephenson, 2009; Gander et al., 2013).

However, these reports stress that students would benefit from education in CS as an independent scientific subject on par with learning areas such as Mathematics or English (Gander et al, 2012). It is essential that our education systems evolve, requiring the clear articulation of CS as a *distinct* discipline, including integrating CS as a fundamental learning area across curricula, and exploring the societal and cultural impacts of technology.

New curricula introduced in England (Department for Education, 2013), Australia (ACARA, 2012), New Zealand and the new ACM CS standards (Seehorn et al.,

2011) have identified the need to educate for *both* digital literacy and CS, and the need to promote both learning areas from the commencement of schooling through to high school, to support youth in participating in an increasingly digital society. While this movement has many positive aspects, the introduction of such curricula poses many challenges for those involved: appropriate and inclusive development for teachers, research into pedagogy and approaches, and integration with current efforts in CS education that span primary-secondary education, and integration into further study.

In this paper, we provide an overview and discuss the core learning objectives of two new curriculum documents that introduce CS as a learning area: Australia's proposed Digital Technologies curriculum and England's computing curriculum. Additionally, we undertake a review of current research in CS Education within the primary and secondary context. Our goal is both to identify key sources of information that may be used to guide effective implementation, as well as identifying areas of research that have been insufficiently researched to date.

2 Next Generation ICT Curricula

Different terminology has been applied to identify the study of this discipline. For example *computer science* is used in the US (Seehorn et al., 2011), *informatics* in Europe (Gander et al., 2013), *computing* in England (Department for Education, 2013) and *computational thinking* or even *ICT* have been used in curriculum discussions. Australia introduces this learning area as the *digital technologies*. To demonstrate the variety of terminology, we draw on 71 articles later analysed in this paper, presenting the most frequent words (frequency increased by text size) used by authors to describe the discipline. For consistency, we have chosen to use the term computer science (CS), unless referring to particular curricula.

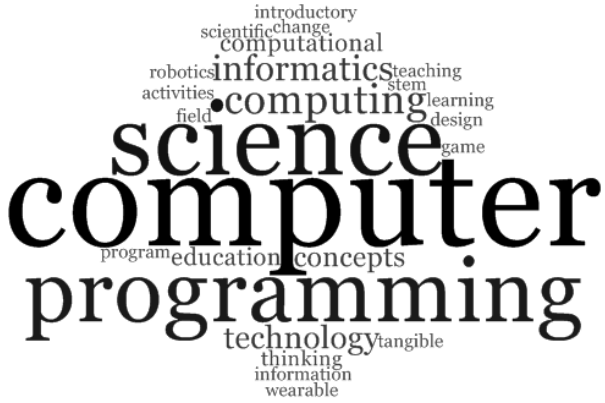

Figure 1: 25 most frequent words used to describe the discipline across 71 papers

2.1 The Australian National Curriculum

The Australian primary and secondary school system is undergoing a significant period of change, with the introduction of a National Curriculum. In Australia primary school includes the first year of school, called Foundation (F) followed by year 1, and so on, until year 6 or 7, (depending on the state) and secondary school (also known as high school) includes years 7 or 8 to year 12. In 2013, the Australian Curriculum Assessment and Reporting Authority (ACARA) released a series of draft curriculum standards for the national curriculum that is to be introduced across Australia in 2014. The curriculum introduces new learning areas with considerable effort committed in the definition of the curriculum and national achievement standards for each area. Some learning areas have achievement standards defined from F-12, while others, including ICT, have achievement standards defined from F-10, with decisions in the senior years of schooling to be defined at a later stage.

'The Shape of the Australian Curriculum' (ACARA, 2012), identifies that "rapid and continuing advances in ICT are changing the ways people share, use, develop and process information and technology, and young people need to be highly skilled in ICT". The ACARA documents include ICT awareness (digital literacy) as a key capability, embedded throughout the curriculum, and additionally introduce a new learning area, Technologies, combining the "distinct but related" areas of Design and Technologies and Digital Technologies (DT) (ACARA, 2013). DT explicitly addresses computational thinking and the use of digital systems and data, spanning representation, abstraction, algorithmic design, fundamental programming, requirements analysis and cultural impacts of technology.

An information report released by ACARA states that the DT curriculum does involve some (CS) knowledge and skills, as well as some digital solutions (possibly involving programming and CS concepts) but the intended focus is on developing computational thinking, logic and problem solving capabilities (ACARA, 2013). The DT curriculum is based on a *systems thinking* approach, designed to encourage students to understand the individual parts of the system, while also being capable of having a holistic view of the, including ethical, societal and sustainability considerations.

DT focuses on developing knowledge of digital systems, information management and the computational thinking required to create digital solutions. The core is the development of computational thinking skills: problem solving strategies and techniques that assist in the design and use of algorithms and models. The Australian Curriculum describes the nature of learners and curriculum across three broad year-groupings: Foundation to Year 2 (ages 5-7); Years 3 to 6 (ages 8-11); and Years 7 to 10 (ages 12-16).

Approaches to teaching vary according to these year-groupings. The development of both digital literacy and computational thinking commences in the F-2 band. In F-2, learning is based around directed play, facilitating students in developing an understanding of the relationship between the real and virtual worlds, the use of technology in communication, and the importance of precise instructions and simple problem solving in the digital world. In 3-6, students are guided to develop a wider understanding of the impact of technology, including family and community considerations, and are able to work on, and communicate about, more complex and elaborate problems. Across 7-10, students move beyond their initial community and are required to consider broader ethical and societal considerations. In this band, students should be able to solve sophisticated

problems using technology, and understand complex and abstract processes. This development from F-10 supports the understanding of the utility of technology, as well as the development of problem solving skills and an abstract understanding of CS.

The eight key concepts that underpin the DT curriculum are allocated to one of two strands: 'Knowledge and Understanding' and 'Processes and Production Skills'.

2.1.1 Knowledge and Understanding

The Knowledge and Understanding strand builds awareness of digital systems and digital information. This includes the impact of digital technologies upon societies and relationships between these technologies and a society, exploring ethical and cultural considerations, from both a local and global perspective. The following sequence of learning objectives explores how an understanding of digital representation is developed across the curriculum:

- F-2: *Recognise and play with patterns in data and represent data as pictures, symbols and diagrams.*
- 3-6: *Explain how digital systems represent whole numbers as a basis for representing all types of data.*
- 7-10: *Explain how text, audio, image and video data are stored in binary with compression.*

2.1.2 Processes and Production Skills

In Processes and Production Skills, students explore how to solve computational problems, involving developing skills in "formulating and investigating problems; analysing and creating digital solutions; representing and evaluating solutions; and utilising skills of creativity, innovation and enterprise for sustainable patterns of living" (ACARA, 2013).

The following presents an example sequence of learning objectives designed to introduce algorithmic planning:

- F-2: *Follow, describe, represent and play with a sequence of steps and decisions needed to solve simple problems*.
- 3-4: *Design and implement simple visual programs with user input and branching.*
- 5-6: *Follow, modify and describe simple algorithms, involving sequence of steps, decisions and repetitions that are represented diagrammatically and in plain English.*
- 7-8: *Develop and modify programs with user interfaces involving branching, repetition or iteration and subprograms in a general-purpose programming language.*
- 9-10: *Collaboratively develop modular digital solutions, applying appropriate algorithms and data structures using visual, object-oriented and/or scripting tools and environments.*

The processes and production strand encapsulates the key concepts of computational thinking and presents challenges to us as a community in how we develop relevant skills within the younger age-groups.

2.2 The National Curriculum in England

England's new National Curriculum, to be introduced in 2014, places the education of computing across two main learning areas: "computing", and the study of "design and technology". Computing as a discipline is a required study element across the curriculum, while the study of design and technology is a required component across Stages 1-3, addressing primary and junior secondary education. At Stage 4 (years 10-12) students may elect to study an information technology topic in-depth.

Computing: The Computing curriculum explicitly targets the development of CS skills, including the understanding of fundamental CS concepts, the ability to analyse problems and develop computer programs to solve those problems and the evaluation of information technology solutions. At Stage 1 (years 1-2), students will have direct exposure to programming languages, including skills in creating and debugging simple programs, as well as cyber-security and digital literacy. At Stage 2 (years 3-6), students develop more complex programming skills, including decomposition, iteration and selection, logical reasoning and error detection. At Stage 3 (years 7-9) move to a more abstract level, exploring computational abstractions that model real-world problems, sorting and searching algorithms, use of two or more programming languages, modularity and decomposition, and digital representation.

Design and Technology: At Stage 1 (1-2), students explore designing, making and evaluating technology, with an emphasis on physical structures and, where appropriate, ICT. At Stage 2 (3-6), digital literacy and CS become more prominent, incorporating the use of modelling tools and computer aided design, and the ability to programme in order to monitor and control products as a key technical knowledge component. At Stage 3 (7-9), this development is elaborated through elements of digital literacy (computer-based tool usage, digital presentations and modelling) and CS (applying their knowledge of computing to embed intelligence in products, with reasoning about explicit inputs and control outputs), along with a deeper understanding of the social and ethical impacts of technology, and consideration of culture and user needs within design.

2.3 Discussion

Both the Australian and English curricula integrate digital literacy and computational thinking from the Foundation year level. While the English curriculum focuses explicitly on programming and programming languages, the Australian curriculum introduces programming through a focus on the problem solving abilities required. In addition, the Australian curriculum introduces digital representation at an early point, with a stronger focus on understanding data. The English curriculum focuses on a stronger understanding of abstraction, and more advanced software decomposition and design methodology.

The challenges faced by both nations in the adoption of these curricula are extensive. Consultation with Industry, Community and Education within Australia (ACARA, 2013b) has identified significant concerns in relation to teacher development (particularly at F-7), appropriate pedagogy, and skills needed for integration of

DT learning objectives with the teaching of other learning areas. 55% of respondents indicated concern with the manageability of the implementation of the DT curriculum, while 45% of respondents did not think that the learning objectives were realistic.

Support for the professional development of teachers is crucial in expanding CS curricula, including the creation of community networks to share insights and pedagogical approaches and research (ACARA, 2013b; Gander et al, 2012). Bell, Newton, Andreae, and Robins (2012) describe the New Zealand experience of the rapid introduction of a senior secondary CS curriculum, and the need for extensive teacher development that addresses both content knowledge and pedagogical knowledge. Ragonis, Hazzan, and Gal-Ezer (2010) identify best practice as the development of a dedicated teacher development programme specifically addressing CS. They recommend that a critical element of such programs is to use empirical research to guide appropriate pedagogy for specific year bands, and learning objectives.

However, in addressing the learning of CS or computational thinking from the Foundation year onwards, do we as a community fully understand the pedagogy that is needed? As a community, there have been many efforts over recent decades devoted to exposing pre-tertiary students to CS and programming, via initiatives such as CS4HS (Google, 2013), CS4FN (CS4FN, 2013), Georgia Computes! (Georgia Tech, 2012), or with resources like CS Unplugged (Computer Science Unplugged, 2013). These efforts are often implemented with the aim of changing stereotypes and encouraging participation in non-traditional student groups. CS is a young field, and there is much to learn about how to integrate computational thinking principles and digital literacy concepts with traditional early education pedagogy. This presents a considerable challenge to the CS Education community, but also an opportunity for us to reassess the direction of our research and explore the open research questions ahead of us.

3 Computer Science Education Research

How can we use existing findings to inform the implementation of the DT learning objectives? Our approach is to review the existing literature within CS education in the context of F-12, exploring the following questions:

- What research exists to guide teaching CS to students aging from 5 years to 18 years?
- Which methodologies have researchers used?
- Which DT concepts do the studies investigate?

4 Methodology

There have been a number of surveys examining the literature in CS education. Fincher and Petre (2004) and Pears et al. (2007) explore the different subfields within CS education research. More recently, Malmi et al. (2010) have undertaken a review characterising CS education research according to the type of research undertaken, specifically exploring associated theories and frameworks, research purpose and data collection. Sheard, Simon, Hamilton, and Lönnberg (2009) report on a survey of CS education within introductory programming, identifying common trends and limitations of the current research. They identify that investigating student learning in terms of established theories of learning are rare, and deserving of more research attention. Most relevant to this work is the methodological review of Randolph (2008) of program evaluations in F-12, published prior to 2005, which resulted in the identification of 29 reports. The majority of the evaluation reports related to US studies, and only 3 of the reports were set within the F-6 context.

We adopted Simon's classification system as it was suitable for our purposes, has been applied to a number of computing education conferences (Simon, 2007, 2008; Simon, Carbone, et al., 2008; Simon, Sheard, et al., 2008). The approach has been validated previously with fairly consistent results, with the exception toward difficulty in identifying 'topic' (also referred to as 'theme' in Simon, Carbone, et al., 2008). In the following section we describe the instruments used and elaborate on the classification processes along with our search process.

4.1 Analysis Procedure

We have reviewed existing research papers about CS Education implemented for children between the ages of 5 and 18. We undertook a semi-systematic literature approach to review each paper 1) by classification, using Simon's system (Simon, 2007) to determine context, topic, scope and nature; 2) identify the subject matter taught that aligns with the Australian key concepts for the Digital Technologies curriculum; 3) to identify the age group studied; and 4) to identify data collection methods reported. We used software tools EndNoteX5 and NVivo 10 to organise our classifications and to "code" papers.

While Simon's system has been broadly applied across CS-related conference proceedings, we have a particular focus on research that appear in journals and conference proceedings about CS Education for 5-18 year olds. We explain how our specifications relate to Simon's process below and identify those that emerged in our analysis of the field in the Results section. We briefly describe each dimension in the system, however, for a thorough description of Simon's classification see Simon (2007).

Simon's scheme classifies papers across four dimensions, which include: *topic*, *context*, *scope* and *nature*. The *topic* dimension describes what the paper is about, for example 'ability/aptitude', 'curriculum' or a 'teaching/learning tool'. The *context* dimension includes the subject area in which the paper is situated, such as the area of programming or group work. Where topic and context differ is that a paper may be in the area of 'programming', but the topic of focus is specifically student 'aptitude/ability'. Although the previous studies have identified a number of topics and contexts covered, we intend to see those relating to CS education at the schooling level, so do not expect to see work on 'capstone projects' or 'work experience' (contexts) or 'tutors and demonstrators' (topics). Instead we expect the emergence of topics and contexts particular to this review. *Scope* describes the breadth of the paper, such as within a subject, an institution, a department/program or across multiple institutions. Many efforts to teach CS in primary and high school contexts are currently situated within initiatives, camps, or programmes inside or outside

of the classroom and so we have included another scope called 'intensive program/initiative'. The *nature* dimension describes the type of paper. Simon's classification includes four: 'experiment' and 'analysis' (which, combined constitute 'research' papers), 'reports' and 'position' papers. An 'experiment' examines a specific research question or hypothesis and collects data to test or answer the research question. An 'analysis' is a paper that analyses existing data and a 'report' is a report on something that has been done, possibly in conjunction with a basic survey. Our analysis excludes position papers as these are not fully implemented or evaluated.

We included a further classification named *age band.* The possible bands within this classification align with the Australian curriculum (ACARA, 2011) and include year levels grouped as: Lower primary: F-2 (ages 5-7) and 3-4 (ages 8-9), Middle: Year 5-6 (ages 10-11) and 7-8 (ages, 12-13) and Upper/HS: 9-10 (ages 14-15) and Year 11+ (16+).

Additionally, we created a broad-level classification for studies conducted across multiple year levels. Where articles targeted a specific age range or a number of age ranges, we classified according to the 'best fit' (e.g. for an article about ages 13-15, band 9-10 was selected).

To determine the variety of CS concepts found in the papers we used the ACARA document (ACARA, 2012, pp. 63- 64) as a guide to code content identified in the papers as being the object of study in the activities being researched or reported. We created a document based on the desired key concepts on page 63-64, including a description and "content terms to look out for". If the subject content were mentioned in the paper it was coded to the relevant 'key concepts' nodes in NVivo.

Methodology was another aspect of interest in our review. We coded any mention of data collection techniques to particular nodes we created in NVivo (e.g. interviews, focus groups) and classified each article as being 'mixed' methods or 'qualitative' or 'quantitative'.

4.2 Process and procedure

We searched Google Scholar and the ACM Digital Library database for articles about the F-12 CS Education, limited to 2003- 2013. Google search terms included those associated with 'computer science' ('informatics', 'programming', 'computing') and words such as 'education', 'activities', 'learning', and 'lesson'. Year-level search terms used included 'schooling', 'high school', 'primary', 'elementary', 'F-10', 'F-12' and their derivatives. As the ACM Digital Library has a CS focus, we wanted to source articles with a F-12 and lesson focus and searched the database using the terms 'school', 'activities', 'lessons', 'students' and their derivatives.

Inclusion	Exclusion
2003- 2013 F-12 (ages 5- 18) Research papers and reports About the implementation of activities for teaching CS-related concepts Situated within any context Student-focused	Before 2003 University/college Position papers Theoretical papers Teachers and PD programs (other than design and implementation of lessons/initiatives)

Table 1: Inclusion/exclusion criteria

Relevant papers matching our inclusion/exclusion criteria (Table 1) were entered into EndNote X5 with the PDF as an attachment. The Endnote file was exported to NVivo version 10 for classification and coding.

5 Results

The search for articles returned 71 results that matched our inclusion criteria. Table 2 describes the descriptives of the articles sourced using Simon's classification.

5.1 Summary of research articles

Nature	Book	Conference	Journal	**Total**
Analysis	0	1	0	**1**
Experiment	2	18	10	**30**
Report	5	29	6	**40**
Total	**7**	**48**	**16**	**71**

Table 2: Cross-tabulation of papers by nature & type

Some 40 papers were reports: discussing the outcomes of a particular activity or outreach program, using researcher experiences, observation or a basic end-of-course questionnaire. 30 papers were based on experiments (or a study) where researchers used research methods to gather data to answer a particular research question. Although these were also about outreach programs or activity outcomes, the researchers used a combination or more rigorous use of methods. However, many measured student engagement or interest, rather than pedagogical effectiveness or students' achievement. Use of existing data of students' work was classified as 'analysis'.

Table 3 demonstrates that the majority of studies were conducted in the United States (US; 39), followed by European regions and Asia.

US	EU	Asia	UK	AU	NZ	Other	Total
39	15	9	2	1	1	4	**71**

Table 3: Number of articles by origin

When examining the publish dates for each of the articles in Table 4, starting from 2004, we can see that they grow significantly in 2009 and continue to increase. As the search was conducted in 2013, we expect the number of published articles to continue to rise.

'04	'05	'06	'07	'08	'09	'10	'11	'12	'13
1	2	3	1	2	12	10	16	17	7

Table 4: Number of articles by published year

The scope of the paper identifies the range of the sample and context in which the paper describes. We present a cross-tabulation of context and scope in Table 6. Although university institutions run many of the initiatives and research, we can see that most of the articles were about intensive programs and initiatives, so we created a category to recognise this. A number also specifically referred to research that was conducted within a single case study so we classified these as 'single cases'.

Context/Topic	Ability/ aptitude	Assess. techniques	Assess. tools	Curriculum	Perceptions/ interest	T/L	T/L techniques	T/L tools	Total
Broad-based				3	2				**5**
Computational thinking	1		2				3	3	**9**
Curriculum					1			2	**3**
Data structures							1		**1**
Gaming					2		1		**3**
Hardware/architecture								1	**1**
Information systems		1							**1**
Integrated curriculum				1	2		3	1	**7**
Introduction to IT					2				**2**
Mathematics		1					2	1	**4**
Programming	11				4	1	7	8	**31**
Project				1	1		2		**4**
Total	**12**	**2**	**2**	**5**	**14**	**1**	**19**	**16**	**71**

Table 5: Cross-tabulation of context and topic of papers

Intensive program/ initiative	31
Single case	17
Multi-institutional (different schools)	10
institution (within school)	9
Not applicable	4

Table 6: Scope

Of the papers, we classified them according to the type of research design used according to mixed methods (25), qualitative methods only (32) and quantitative methods (14), with 3 being 'other'. Table 7 presents the range of data collection methods used across the 71 papers. Some papers used more than one method. 'Other methods' included collecting data by involving the students as researchers, for example, by producing journals about their processes. The most commonly used methods were questionnaires and interviews, measuring student engagement and interest after the activity or intervention. Other common methods involved the collection of student work that was examined or analysed, usually in the form of student games that they had programmed.

Method	No.
Questionnaire	24
Student work or artefact	18
Interview	17
Observation (by researcher)	17
Test (of knowledge/ability)	14
Researcher reflection	12
Questionnaire incl. open qu.	11
Focus Group	7
Video	6
Course materials or curriculum document	2
Student grades	1
Other	6

Table 7: Data collection methods used across 71 papers

5.2 Research Topics

Table 5 presents a cross-tabulation of topics and contexts for the papers analysed. The table indicates that, similar to previous analysis using Simon's classification with CS education research, these research papers were also most commonly situated within a 'programming' context. Within this context, the papers explored topics such as students' ability or aptitude to do programming activities or the extent they applied CS concepts and knowledge to their programming. Other topics included exploring teaching and learning techniques for CS concepts or delivery of activities, trialling new teaching and learning tools and student perception and interest in programming. Other popular contexts included integrating CS within other learning areas, such as the Humanities.

In Table 8 we grouped articles by year level bands to allow the examination of types of paper topics explored within each band. From Most articles addressed children in the middle school or high school. In these year levels, the articles focused on student perceptions about doing CS activities, their ability to undertake CS tasks and teaching and learning techniques used within these age groups. Minimal research exists about students in the lower primary levels but for those articles we did source, investigated whether young children could engage in programming or computational thinking and also explored new tools that could be used to teach CS activities for children.

Topic	Lower (5-9yrs)	Mid/Upper (10- 14yrs)	HS (14>)	Broad-age
Ability/aptitude	4	6	2	
Assess. technique			2	
Assess. tools	1			1
Curriculum			4	1
Perceptions/ interest		9	3	2
T/L			1	
T/L techniques	1	10	6	2
T/L tools	4	6	4	2
Total	**10**	**31**	**22**	**8**

Table 8: Topic compared to year level band

	Lower (5- 9yrs)	Mid/Upper (10- 14 yrs)	HS (14+)	Total
Tangible programming tools	5	14	2	21
Other resources/tools	3	1	5	9
Curriculum resources (CS Unplugged)	0	3	4	7
Electronics	0	4	2	6
Non-digital activities	0	2	3	5
Robotics	1	3	1	5
Game creation environments	0	4	0	4
Java and java-programming tools	0	2	2	4
Game or PC puzzle	0	2	1	3

Table 9: Tools and resources used in the 71 papers

Table 9 demonstrates that the majority of F-7 research within CS addresses the use of tangible programming tools (21), followed by the use of existing CS activities (with all but one involving the use of *CS Unplugged*; the other a German version *Informatik erLeben*). Scratch makes up the majority, with 10 cases, followed by three research papers examining the use of Alice

An examination of the articles according to the DT key concepts (ACARA, 2012) in Table 10, reveals that most of the articles implemented activities that involved algorithms, implementation and specification: essentially those involved through teaching programming activities. Another commonly taught topic was data representation and interpretation. In the younger years, this involved activities such as understanding binary through tactile games or in the upper years it extended to more complex activities such as manipulating digital images. Some of the papers also discussed multiple topics within one article and we coded these as broad-based. These typically involved reporting on the success of a set of activities that covered many of the DT concepts.

Communication of Problems and Solutions	
Algorithms (following and describing)	29
Implementation (translating and programming)	34
Specification (descriptions and techniques)	18
Data	
Data collection (properties, sources and data collection)	3
Data interpretation (patterns and context)	8
Data representation (symbolism and separation)	12
Digital systems (hardware, software, and networks on the Internet)	
Hardware and software	7
Networks and the Internet	9
Interactions (people and digital systems, data and processes)	8
Broad-based concepts	8
Abstraction (hiding irrelevant details)	7
Impact (impacts and empowerment)	1

Table 10: Articles according to DT key concepts

There was only one article that explored CS careers. With the Australian and English curricula addressing social and ethical impact, research is required that investigates the teaching of such content, in addition to programming skills and computational thinking. This will also be an important area for consideration if we are to make computational thinking and programming activities relevant to the lives and future careers of students.

6 Limitations

We acknowledge that we have not possibly captured all existing literature about CS education in years F-12. In our initial study, we provide a preliminary guide to current existing research and trial our analysis approach so that we can review and implement our approach on a larger scale. Our future work will continue to refine and expand databases and terms.

We also realise that CS Education research may exist within other discipline areas that were not discovered in searches, such as society and environment, design and technology, mathematics, or science because of its versatile nature and the ability for CS concepts and approaches to be applicable in other fields, as we saw with the use of programming as a tool for story telling and learning about storyboarding (Burke & Kafai, 2010). This offers opportunities for future research to identify cross-curricula use of CS within other learning areas.

Similar to Simon, Carbone, et al. (2008), we also encountered difficulties in deciding the context and topic of papers, however, using Simon's suggestion, we made our decision on what topic or context was the 'best fit' when more than one possible topic existed. We acknowledge that others may classify some papers within different areas, but nonetheless the classification still provide sound guidance for what research currently exists in F-12 CS Education and what research is required.

7 Discussion

After review, three significant areas emerged that provide guidance for future research. We will discuss how these guide approaches to future research and lead into the conclusion.

7.1 CS F-10 Pedagogy

While there has been considerable research into CS within the F-12 context, it is typically focussed on years 5-12 with much less research at the F-4 level. Most of the research that has been done is situated within outreach programs, focussed on sharing teaching techniques aimed at motivating students to study CS, to address negative perceptions of the discipline, stereotypes and to increase diversity in our student cohorts. Computer games and the creation of games through tangible programming tools also play a significant role in current approaches to engaging younger students in CS, however as highlighted by Denner (2011), the majority of studies in this domain explore the potential for computer games to motivate students to study CS, rather than exploring what they are able to learn. This is of increasing importance with the emerging focus on computational thinking and the development of computational problem solving skills. There is a whole field of possibilities for pedagogical exploration in F-10 CS education and to investigate specific techniques for early education within CS, including small-group ability levels, inquiry-based learning, and play-based learning.

Compare this with the field of Mathematics education, with its rich history of deep exploration of Mathematics pedagogy. Some interesting recent examples that highlight potential areas for related CS research include: analysis of symbolic number sense and impact upon mathematics achievement (Jordan et al, 2009); analysis of core concepts and student understanding (Knuth et al, 2011); gender-based stereotypes and achievement (Beilock et al, 2010); and emergent mathematical thinking in play environments (van Oers, 2010).

Similarly, there are opportunities for exploring how the use of CS tools influences learning processes. Papert's (1980) work in programming environments for children introduces the idea of constructionist programming environments: places where children can create concrete digital constructs from abstract ideas, and then reflect over those to develop understanding. Many of the constructionist programming environments are focused on years 3-7, including Scratch, Alice and Kodu. In this emerging field, there is early work that demonstrates that children who are exposed to constructionist environments are able to learn

computational thinking concepts (Bers & Horn, 2010; Kazakoff & Bers, 2012; Kazakoff, Sullivan, & Bers, 2013; Lai & Yang, 2011). In contrast, a study by Meerbaum-Salant, Armoni, and Ben-Ari (2011) identifies that use of Scratch engenders specific poor programming habits, at odds with both accepted practice and the learning objectives of the proposed curricula.

This also applied to the lesson resources that currently exist, such as CS Unplugged. These resources are helpful, especially for teachers who have limited or no experience in CS and are able to be implemented in classrooms with no technology. However, we must be clear on the goals of a program such as CS Unplugged. Taub, Ben-Ari, and Armoni (2009) state the three main aims of CS Unplugged as changing students' views on the nature of CS, promoting views that CS is a career for women and changing views about CS as a profession. An analysis of the CS unplugged resources to determine approach, coverage of explicitly addressed CS concepts and whether the aims were addressed identified that only some of the objectives were addressed in the activities. After trialing activities, year 7 students did change their understanding of the nature of CS, but held less attractive perceptions of CS as a career. Similarly, Feaster, Segars, Wahba, and Hallstrom (2011) found implementation of a semester long outreach program using the resources had no significant impact on attitudes toward CS or content understanding. Once again, activities like CS Unplugged have typically been assessed in terms of their effectiveness to change attitudes and perceptions, rather than learning progress. There are new opportunities for evaluating existing CS activities in terms of student achievement, learning objectives and improved computational thinking processes.

7.2 Methodology, Sample and Scope

Many studies were conducted with small sample sizes or were pilot studies due to being situated within the work of intensive programs or initiatives and because many were about show-casing and sharing teaching and learning techniques or tools (Kordaki, 2011; Lewis, 2011). Furthermore, the studies are usually conducted outside of conventional classroom settings and authors identify that it is difficult to make a comparison to classroom environments (Lode, Franchi, & Frederiksen, 2013). If studies were conducted in-class they were typically one-off sessions, out of the context of the regular curriculum, which authors cautioned may have result in students and teachers being 'less committed' (Meerbaum-Salant, Armoni, & Ben-Ari, 2010).

Another limitation was that students who were the subject of study were usually involved because they volunteered to participate in after school or holiday programs (Denner, Werner, & Ortiz, 2012; Lau, Ngai, Chan, & Cheung, 2009; Magnenat, Riedo, Bonani, & Mondada, 2012). As volunteers, the participants may come to the classes out of interest: a different frame of mind to students who are in classrooms out of duty. Other studies selected students based on their achievement, for example in a study by Curzon, McOwan, Cutts, and Bell (2009) participants were identified as being in the top 5% of the school and participants in research by Feaster, Ali, and Hallstrom (2012) involved high achievers. In classroom environments, teachers typically have to cater to students with a whole range of capabilities, interests and achievement levels making this a challenge for teachers to overcome.

The actual effectiveness of teaching techniques are often not known because researchers have not measured before and after (Meyers, Cole, Korth, & Pluta, 2009) and because researchers experienced difficulty in identifying ways to formally assess goals and outcomes of projects (Settle et al., 2012). Ultimately, research in this area will need to be rigorous, replicable and explicitly defined.

7.3 Teacher Experiences and Development

Our review of the literature was focused on students and the implementation of the lessons, rather than teacher ability and training, but one important aspect that arose was in regard to who was implementing the activities that were the object of study. In many cases, activities were conducted by researchers from CS institutions or by those with significant experience in teaching CS. For example, in Meerbaum-Salant et al. (2011) the teacher had 15 years experience with teaching CS and in Taub et al. (2009) one teacher taught mathematics and programming and the other teacher had one year's experience teaching CS Unplugged. Robertson & Nicholson 2007 involved a specialist IT teacher and three researchers; and in a study by Stoeckelmayr, Tesar, and Hofmann (2011) the activity was conducted by a CS academic from a university with the support of undergraduate students. These situations are vastly different to a single generalist teacher implementing classroom activities without support.

Authors, Settle et al. (2012), recognise the difficulty in translating materials into existing curriculum, when unfamiliar with the tools. In one study, when teachers used guiding activity resources for their CS lessons, they were apprehensive about using teaching methods such as group work (Curzon). The teachers also felt that because they were unfamiliar with the topic, considerable preparation would be required. Meerbaum-Salant et al (2011) identified that although the teacher was experienced in CS, adding new tools created anxiety, causing deviation from lesson plans. Tinapple, Sadauskas, and Olson (2013) further comment on the challenge for teachers, where expected software and/or hardware are not easily available.

Black et al. (2013) describe a survey of UK computing teachers in relation to their suggestions on improving CS education, and teacher development needs. Their results highlighted teacher training, and the need for a network and community to support resource development. Black *et al*'s survey identifies that teachers focus more on fun activities rather than providing opportunities for deep learning of computational thinking, focussing on impressive technology, physical computing and programming in constructionist environments. These forms of activities can complicate the learning environment further by placing additional stress on teachers inexperienced with technology.

8 Conclusions

The expected changes in the teaching of Computer Science represent a significant challenge for our schooling systems. Computational Thinking and

Computer Science will form part of the Australian standard curriculum from F-12 from 2014. In this paper, we have presented the key learning objectives of both curricula, and have identified the key challenges that arise from these changes, specifically, the need to teach computational thinking as a standalone concept; the introduction of computational thinking and computer science from Foundation onwards, and the need to develop and understand appropriate pedagogy that integrates with existing early childhood approaches.

We have undertaken a preliminary review of existing CS education research within the F-12 context, identifying key themes (outreach, programming, tangible programming tools, CS activities, senior secondary) and also gaps (F-7, computational thinking, CS concepts). We have identified a distinct lack of rigorous research within the F-7 context, including relevant pedagogy and assessment practices within conventional classroom settings. This represents an outline of needed research requiring greater collaboration between representatives in primary and secondary school education, education researchers, and higher education CS departments. With greater collaboration between each group it may better ensure the development of a research agenda that encompasses the expertise and needs of both groups.

9 References

ACARA. (2011): 3.1 School structures. National Reporting on Schooling in Australia 2009, http://www.acara.edu.au/reporting/national_report_on_schooling/schools_and_schooling/school_structures.html, Accessed 4 Sep 2013.

ACARA. (2012): The shape of the Australian curriculum: technologies. Sydney, NSW: ACARA, http://www.acara.edu.au/curriculum_1/learning_areas/technologies.html, Accessed 29 Aug 2013.

ACARA. (2013): The Australian curriculum: technologies information sheet. Sydney, NSW: ACARA, http://www.acara.edu.au/curriculum_1/learning_areas/technologies.html, Accessed 20 Sep 2013.

Bell, T., Newton, H., Andreae, P., & Robins, A. (2012): The introduction of computer science to NZ high schools: an analysis of student work. *Workshop in Primary and Secondary Computing Education*, Hamburg, Germany, 5-15.

Bers, M., & Horn, M. (2010): Chapter 4: Tangible programming in early childhood. In I. Berson & M. Berson (Eds.), *High-Tech Tots: Childhood in a Digital World (HC)*, 49- 68, Information Age Publishing.

Black, J., Brodie, J., Curzon, P., Myketiak, C., McOwan, P., & Meagher, L. (2013): Making computing interesting to school students: teachers' perspectives. *Proc. ITiCSE*, Canterbury, England, 255-260.

Burke, Q., & Kafai, Y. B. (2010): Programming & storytelling: opportunities for learning about coding & composition. *Proc. International Conference on Interaction Design and Children*, Barcelona, Spain, 348- 351.

Computer Science Unplugged. (2013): Computer Science Unplugged, http://csunplugged.org/, Accessed September 2013.

CS4FN. (2013): About cs4fn, http://www.cs4fn.org/about.php, Accessed 25 Aug 2013.

Curzon, P., McOwan, P., Cutts, Q., & Bell, T. (2009): Enthusing & inspiring with reusable kinaesthetic activities. *SIGCSE Bulletin* **41**(3): 94- 98.

Denner, J. (2011): What predicts middle school girls' interest in computing? *International Journal of Gender, Science and Technology* **3**(1): 53-61.

Denner, J., Werner, L., & Ortiz, E. (2012): Computer games created by middle school girls: can they be used to measure understanding of computer science concepts? *Computers & Education* **58**(1): 240-249.

Department for Education. (2013): The national curriculum in England. Cheshire, UK: Crown.

Feaster, Y., Ali, F., & Hallstrom, J. (2012): Serious toys: teaching the binary number system. *Proc. ITiCSE*, Haifa, Israel, 262- 267.

Feaster, Y., Segars, L., Wahba, S., & Hallstrom, J. (2011): Teaching CS unplugged in the high school (with limited success): *Proc. ITiCSE*, Darmstadt, Germany, 248- 252.

Fincher, S., & Petre, M. (2004): Computer Science *Education Research*. Psychology Press.

Gal-Ezer, J., & Stephenson, C. (2009): The current state of computer science in US high schools: a report from two national surveys. *Journal for Computing Teachers*, Spring: 1- 5.

Gander, W., Petit, A., Berry, G., Demo, B., Vahrenhold, J., McGettrick, A., Boyle, R., Drechsler, M., Mendelson, A., Stephenson, C., Ghezzi, C. & Meyer, B. (2013): Informatics education: Europe cannot afford to miss the boat ACM Europe: Informatics Education Report. New York.

Google. (2013): What is CS4HS? Google: computer science for high school, from http://www.cs4hs.com/

Georgia Tech. (2012): Georgia Computes!, http://gacomputes.cc.gatech.edu/, Accessed 15 Sep 2013.

Henwood, F. (2000): From the woman question in technology to the technology question in feminism rethinking gender equality in IT education. *European Journal of Women's Studies* **7**(2): 209- 227.

Jepson, A., & Perl, T. (2002): Priming the pipeline. *ACM SIGCSE Bulletin* **34**(2): 36- 39.

Kazakoff, E., & Bers, M. (2012): Programming in a robotics context in the kindergarten classroom: the impact on sequencing skills. *Journal of Educational Multimedia and Hypermedia* **21**(4): 371- 391.

Kazakoff, E., Sullivan, A., & Bers, M. (2013): The effect of a classroom-based intensive robotics and programming workshop on sequencing ability in early childhood. *Early Childhood Education Journal* **41**(4) 1- 11.

Kordaki, M. (2011): A computer card game for the learning of basic aspects of the binary system in primary education: design and pilot evaluation. *Education and Information Technologies* **16**(4): 395- 421.

Lai, A.-F., & Yang, S.-M. (2011, 16-18 Sept. 2011): The learning effect of visualized programming learning on 6th graders' problem solving and logical reasoning abilities. *Proc. International Conference on Electrical and Control Engineering* (ICECE), Yichang, 6940- 6944.

Lau, W., Ngai, G., Chan, S., & Cheung, J. (2009): Learning programming through fashion and design: a pilot summer course in wearable computing for middle school students. *SIGCSE Bulletin* **41**(1): 504- 508.

Lewis, C. (2011): Is pair programming more effective than other forms of collaboration for young students? *Computer Science Education* **21**(2): 105- 134.

Lode, H., Franchi, G., & Frederiksen, N. (2013): Machineers: playfully introducing programming to children. *Proc. Human Factors in Computing Systems*, Paris, France, 2639- 2642.

Magnenat, S., Riedo, F., Bonani, M., & Mondada, F. (2012, 21-23 May 2012): A programming workshop using the robot "Thymio II": The effect on the understanding by children. *Proc. Workshop on Advanced Robotics and its Social Impacts,* Munich, Germany, 24- 29.

Malmi, L., Sheard, J., Bednarik, R., Helminen, J., Korhonen, A., Myller, N., Sorva, J. & Taherkhani, A. (2010): Characterizing research in computing education: a preliminary analysis of the literature. *Proc. ICER*, San Diego, 3-12.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2010): Learning computer science concepts with scratch. *Proc. International workshop on computing education research*, Denmark, 69- 76.

Meerbaum-Salant, O., Armoni, M., & Ben-Ari, M. (2011): Habits of programming in scratch. *Proc. ITiCSE,* Germany, 168- 172.

Meyers, A., Cole, M., Korth, E., & Pluta, S. (2009): Musicomputation: teaching computer science to teenage musicians. *Proc. ACM conference on Creativity and cognition,* Berkeley, California, 29-38.

Papert, S. (1980): *Mindstorms: children, computers, and powerful ideas*. New York, Basic Books, Inc.

Pears, A., Seidman, S., Malmi, L., Mannila, L., Adams, E., Bennedsen, J., Delvin, M. & Paterson, J. (2007): A survey of literature on the teaching of introductory programming. *SIGCSE Bulletin* **39**(4): 204- 223.

Ragonis, N., Hazzan, O., & Gal-Ezer, J. (2010): A survey of computer science teacher preparation programs in Israel tells us: computer science deserves a designated high school teacher preparation! *Proc. ACM technical symposium on computer science education,* 401- 405.

Randolph, J. (2008): A methodological review of the program evaluations in K-12 computer science education. *Informatics in Education- An International Journal* **7**(2): 237- 258.

Schulte, C., Hornung, M., Sentance, S., Dagiene, V., Jevsikova, T., Thota, N., Eckeral, A. & Peters, A.-K. (2012): Computer science at school/CS teacher education: Koli working-group report on CS at school. *Proc. Koli Calling International Conference on Computing Education Research*, Koli, Finland, 29- 38.

Seehorn, D., Carey, S., Fuschetto, B., Lee, I., Moix, D., O'Grady-Cunniff, D., . . . Verno, A. (2011): CSTA K-12 computer science standards The CSTA Standards Task Force. New York: Computer Science Teachers Association, Association for Computing Machinery.

Settle, A., Franke, B., Hansen, R., Spaltro, F., Jurisson, C., Rennert-May, C., & Wildeman, B. (2012): Infusing computational thinking into the middle-and high-school curriculum. *Proc. ITiCSE*, Haifa, Israel, 22- 27.

Sheard, J., Simon, Hamilton, M., & Lönnberg, J. (2009): Analysis of research into the teaching and learning of programming. *Proc. ICER*, Berkeley, California, 93- 104.

Simon. (2007): A classification of recent Australasian computing education publications. *Computer Science Education* **17**(3): 155- 169.

Simon. (2008): Koli calling comes of age: an analysis. *Proc. Baltic Sea Conference on Computing Education Research (Koli Calling),* Koli National Park, Finland, 119- 126.

Simon, Carbone, A., de Raadt, M., Lister, R., Hamilton, M., & Sheard, J. (2008): Classifying computing education papers: process and results. International *Proc. Workshop on Computing Education Research,* Sydney, Australia.

Simon, Sheard, J., Carbone, A., de Raadt, M., Hamilton, M., Lister, R., & Thompson, E. (2008): Eight years of computing education papers at NACCQ. *Proc. Annual Conference of the National Advisory Committee on Computing Qualifications*, Auckland, New Zealand, 101- 107.

Stoeckelmayr, K., Tesar, M., & Hofmann, A. (2011): Kindergarten children programming robots: a first attempt. *Proc. International conference on robotics in education*, Vienna, Austria, 185- 192.

Taub, R., Ben-Ari, M., & Armoni, M. (2009): The effect of CS unplugged on middle-school students' views of CS. *Proc. SIGCSE conference on innovation and technology in computer science education*, Paris, France, 99- 103.

The Royal Society. (2012): Shut down or restart? The way forward for computing in UK schools. London.

Tinapple, D., Sadauskas, J., & Olson, L. (2013): Digital culture creative classrooms (DC3): teaching 21st century proficiencies in high schools by engaging students in creative digital projects. *Proc. International Conference on Interaction Design and Children*, New York, 380- 383.

Wilson, C., & Guzdial, M. (2010): How to make progress in computing education. *Communications of the ACM* **53**(5): 35- 37.