IJCSES

Volume 6, No: 1
March 2023
ISSN 2513-8359

International Journal of Computer Science Education in Schools

Editors

Dr Filiz Kalelioglu

Dr Yasemin Allsop

www.ijcses.org

International Journal of Computer Science Education in Schools

March 2023, Vol 6, No 1

DOI: 10.21585/ijcses.v6i1

Table of Contents

A Brief History of K-12 Computer Science Education in Ireland

Roisin FAHERTY[1]
Karen NOLAN[1]
Keith QUILLE[1]
Brett A. BECKER[2]
Elizabeth OLDHAM[3]

[1]TU Dublin, Ireland
[2]University College Dublin
[3]Trinity College Dublin

DOI: 10.21585/ijcses.v6i1.148

Abstract

This paper unites the history of Computer Science (CS) Education in Ireland by plotting Ireland's roadmap leading to the implementation of formal Computer Science Education in schools. It first outlines the educational system in Ireland. The history roadmap starts in the 1970s with the first notions of introducing computing in post-primary school, and then continues up to the roll-out the CS curriculum in Ireland at the Senior Cycle level in 2018. The story is chiefly available in disparate publications and reports, so piecing together the entire story is often difficult. This paper collates the available literature, together with the authors' local knowledge of the process, into one paper that may be of interest locally and of value to other jurisdictions beginning their planning of national curricula. The paper describes the development and the current situation of the formal curricula in CS at second level. The current landscape of Computing Education at primary level, which at the time of writing is in the planning stages in Ireland, is described. Additionally, an investigation into the introduction of Computing Education in schools in the international jurisdictions that directly influenced the Irish roll-out takes place, to summarize any lessons learned that might provide insights for Ireland going forward.

Keywords: Computer Science Education, K-12, Computer Science Education Ireland

1. Introduction

The Irish K-12 education system has two levels, primary school and post-primary school. Primary school is for children aged between 4 years and 12 years approximately and is made up of eight educational years: Junior Infants, Senior Infants, and 1st to 6th Class inclusive. At the end of primary school children move into post-primary school. Post-primary schooling is six years in length divided into two sections: the Junior Cycle (1st to 3rd year) and Senior Cycle (4th, 5th and 6th Year). The 4th Year ("Transition year" or TY) is not a mandatory element of the Senior Cycle in Ireland. Schools can optionally offer this year which centres around personal growth of students (NCCA, Transition year, 2021). The Junior Cycle has terminal state assessment, until recently was called the Junior Certificate (JC). The Senior Cycle has a terminal state exam called the Leaving Certificate (LC), the results of which are used to determine university entry. These two educational levels in Ireland are the equivalent of K-12 in the US, and primary and secondary school in the UK. For a detailed comparison with other jurisdictions (see Figure 1) (Falkner, et al., 2019).

COUNTRY	AUSTRALIA (AUS)	ENGLAND (ENG)	IRELAND (IRL)	ITALY (ITA)	MALTA (MLT)	SCOTLAND (SCO)	US
AGE* (Years)							
2+				Pre-school	Kindergarten	Pre-school	Pre-school
3	Pre-school	Pre-school	Pre-school	Kindergarten	Kindergarten	Pre-school	Pre-school
4	Kindergarten	Pre-school	Junior Infants	Kindergarten	Kindergarten	Pre-school	Pre-school
4-5	Reception/ Foundation	Reception	Senior Infants	Kindergarten	Year 1	Primary 1	Pre-school
5-6	Year 1	Year 1	First Class	First class primary	Year 2	Primary 2	Kindergarten
6-7	Year 2	Year 2	Second Class	Second class primary	Year 3	Primary 3	Grade 1
7-8	Year 3	Year 3	Third Class	Third class primary	Year 4	Primary 4	Grade 2
8-9	Year 4	Year 4	Fourth Class	Fourth class primary	Year 5	Primary 5	Grade 3
9-10	Year 5	Year 5	Fifth Class	Fifth class primary	Year 6	Primary 6	Grade 4
10-11	Year 6	Year 6	Sixth Class	First class lower high school	Year 7	Primary 6	Grade 5
11-12	Year 7	Year 7	First Year	Second class lower high school	Year 8	Primary 7	Grade 6
12-13	Year 8	Year 8	Second Year	Third class lower high school	Year 9	S1	Grade 7
13-14	Year 9	Year 9	Third Year	First class higher school	Year 10	S2	Grade 8
14-15	Year 10	Year 10	Transition Yr.	Second class higher school	Year 11	S3	Grade 9
15-16	Year 11	Year 11	Fifth Year	Third class higher school	Sixth form lower	S4	Grade 10
16-17	Year 12	Year 12	Sixth Year	Fourth class higher school	Sixth form higher	S5	Grade 11
17-18		Year 13		Fifth class higher school		S6	Grade 12

Figure 1. School levels across different jurisdictions (Falkner, et al., 2019)

In Ireland, in 2020, there were 3107 primary schools (Department of Education, 2019) and 722 Secondary schools (Department of Education, 2019). The breakdown of the different types of these schools can be seen in Table 1 and Table 2.

Table 1. Primary School Breakdown

School Type	**N**
DEIS	688
Mainstream	2419
Male Only School	162
Female Only School	90
Mixed Schools	2855
Total Schools	**3107**

Table 2. Post-primary School Breakdown

School Type	**N**
DEIS	198
Mainstream	524
Male Only School	101
Female Only School	128
Mixed Schools	492
Total Schools	**722**

One of the most significant events in Ireland's formal Computer Science Education (CSEd) history was the introduction of Leaving Certificate Computer Science (LCCS) curriculum which became a reality in Ireland in September 2020, when any school could elect to offer the subject to students (NCCA, Leaving certificate computer science, 2019). The details on the specification of this subject will be covered in section 3.3. This moved Ireland a step closer to the realization of offering an education in CS to all students at all educational levels in Ireland. Phase one of the new LCCS subject was rolled out to 40 schools in 2018, completing in June 2020. It has taken almost fifty years of canvasing and effort to have this subject available formally at Senior Cycle level in Irish schools. The provision of such a subject is no small task given the changing nature of the area as well as the need to ensure teachers are well prepared to deliver such a new subject. In order to understand the landscape of CS in the Irish education system, this paper reviews the literature available to provide a single point of reference to the road taken, eventually leading to the provision of CS as a formal subject as part of the Senior Cycle in schools. It also highlights the current status of CSEd in primary schools in Ireland; while not yet formalized like the LCCS, this curriculum is progressing in a positive direction and its current situation is discussed in section 3.1. There is also an optional short coding course at Junior Cycle level which is discussed in section 3.2. The paper also includes a detailed description of the relevant international CS landscape in schools that directly influenced the Irish roll-out and collates any lessons that can be learned from these jurisdictions to aid the current Irish roll-out.

The paper has the following structure. Firstly, there is a short description of the education system in Ireland to provide context for international readers. Next follows a literature review (Section 2) which provides a summary of the literature detailing the history of Computer Science in Ireland. The paper

then examines the roll-out of the 2018 LCCS pilot subject (Section 3), as well as reviewing the Junior Cycle changes to incorporate optional computer courses for students and looking at the Computer Science status quo at primary level in Ireland, as indicated above. Lessons learned from international jurisdictions (Section 4) complete this review. Finally, the paper discusses the National roll-out in 2020 of LCCS as well as what the future might hold for Computer Science in Irish education.

1.1 Primary School Education in Ireland

In Ireland, children start their formal education at the age of 4 or 5 years. Traditionally this takes the form of starting in a primary school at Junior Infants. Primary education in Ireland is an eight-year cycle after which at approximately 12 or 13 years old the children move into post-primary school. Children in primary education in Ireland follow a prescribed curriculum covering many subject areas; however, the core subjects are English, Irish and Maths. Other subject areas include Physical Education, Social Environmental and Scientific Education, Social, Personal and Health Education, Art and Religion (NCCA, Primary curriculum, 2021). Some schools and teachers just use technology in the classroom to teach the curriculum in different subjects and others expose the students to some basic IT skills; some do introduce programming and allied activities. At present there is no formal curriculum for CS for primary school children in Ireland; however, the planning for this has started.

1.2 Junior Cycle

Typically, at the age of 12 or 13, an Irish child begins post-primary school, with the JC occupying the first three years, concluding with the exams that form part of the state assessment at this stage. Students normally sit these exams around the age of 15 (Citizens information, 2020). Students typically take a minimum of eight to a maximum of ten subjects for final examination/assessment and reporting in the JC (Department of Education, 2021), choosing from a range of subjects (see Figure 2). Ireland's JC is similar to the GCSE in the UK. A reform of the JC began in 2011 with the publishing by the DES of the document "Towards a framework for Junior Cycle" (NCCA, Towards a framework for junior cycle, 2011). This was followed by "A Framework for Junior Cycle" (Department of Education, 2015), published in October 2012. Prior to this point the JC was made up of discrete subjects which were offered by the schools, chosen by the students, and examined as part of the state exams. In 2012 there was a move towards introducing a group of subjects which while assessed are not examinable, allowed the student to acquire a level 3 qualification on the National Qualifications of Framework (NQF), through what was called a short course. Short courses were first offered in 2014 by schools and this was followed by the publication of the 2015 Framework for JC (Department of Education, 2015). This built on the 2012 publication, aiming to:

" set out a clear vision of how teaching, learning and assessment practices will evolve in the first three years of post-primary education to ensure a learning experience for our young people that is appropriate to the needs of the 21st century."

A JC short course is a 100-hour course (rather than one occupying 200 to 240 hours) and can be delivered at varying stages across the three years of the JC. The main JC short course that falls under the umbrella of CS is the Coding short course (NCCA, 2016). This has an emphasis on active learning where students provide evidence of their learning in a variety of ways, including digital media, audio recordings and written pieces. It is made up of three strands: CS Introduction; Let's get connected; and Coding at the next level. Thus, the JC offers opportunities for students to learn about CS through coding, but the course is optional and so is not taken by all students. In a report published by Lero in 2019 (Fleming & McInerney, JCCiA - interim report, 2019), some of the challenges faced by this short course included timetabling of the subject in schools, lack of necessary resources and time required to ensure teachers were up to date on the teaching methodologies needed.

1.3. Senior Cycle

Ireland's Leaving Certificate (LC) state exams are held at the end of the Senior Cycle (equivalent to the end of US grade 12) where typical students are 17-18 years of age. These are comparable to the UK A levels, more specifically the LCCS is comparable to the US AP (Advanced Placement) Computer Science A and Computer Science Principles exams, and the UK A level Computer Science subject. Students sitting the LC have a range of subjects to choose from (see Figure 3), but for university entrance purposes a minimum of six subjects is taken. While the only mandatory subject is Irish – compulsory except for students who have an exemption for some reason, such as recent arrival in the country – many schools require their students to take English and Maths (and indeed students would opt for them in any case) because they are required for entry to many third-level courses and for employment purposes. The standard is for students to sit seven subjects and count six of these towards the points for their university entrance (CAO, 2021). There are typically around 65,000 students who sit the LC each year, and for whom the results of this exam determine their university course. In 2018, three new LC subjects were added to the offering: Physical Education, Politics and Society and Computer Science. These new subject offerings also herald a move, albeit a small one, to a combination of exam and coursework assessment as opposed to just terminal examination which was the case for the majority of LC subjects. The top subjects sat by students for the LC in Ireland in 2019 (this year was chosen as it is prior to any effect of the COVID pandemic) were Irish, English and Maths (State Examinations Commission, 2021); however, apart from these, the top three choice subjects in 2019 were Biology (~34,000), Geography(~24,000) and French(~23,000) (State Examinations Commission, 2021).

SUBJECT	LEVEL
Irish	Higher, Ordinary and Foundation
English	Higher and Ordinary
Mathematics	Higher, Ordinary and Foundation
History	Higher and Ordinary
Geography	Higher and Ordinary
French	Higher and Ordinary
German	Higher and Ordinary
Spanish	Higher and Ordinary
Italian	Higher and Ordinary
Art, Craft & Design	Higher and Ordinary
Music	Higher and Ordinary
Science (Revised Syllabus)	Higher and Ordinary
Home Economics	Higher and Ordinary
Materials Technology (Wood)	Higher and Ordinary
Metalwork	Higher and Ordinary
Technical Graphics	Higher and Ordinary
Business Studies	Higher and Ordinary
Environmental and Social Studies (ESS)	Higher and Ordinary
Technology	Higher and Ordinary
Latin	Higher and Ordinary
Ancient Greek	Higher and Ordinary
Classical Studies	Higher and Ordinary
Jewish Studies	Higher and Ordinary
Religious Education	Higher and Ordinary
Civic, Social and Political Education (CSPE)	Common

Figure 2. Junior Cycle Subjects

SUBJECT	LEVEL
Irish	Higher, Ordinary and Foundation
English	Higher and Ordinary
Latin	Higher and Ordinary
Ancient Greek	Higher and Ordinary
Classical Studies	Higher and Ordinary
Hebrew Studies	Higher and Ordinary
Arabic	Higher and Ordinary
French	Higher and Ordinary
German	Higher and Ordinary
Italian	Higher and Ordinary
Spanish	Higher and Ordinary
History	Higher and Ordinary
Geography	Higher and Ordinary
Mathematics	Higher, Ordinary and Foundation
Applied Mathematics	Higher and Ordinary
Physics	Higher and Ordinary
Chemistry	Higher and Ordinary
Physics and Chemistry	Higher and Ordinary
Agricultural Science	Higher and Ordinary
Biology	Higher and Ordinary
Agricultural Economics	Higher and Ordinary
Engineering	Higher and Ordinary
Construction Studies	Higher and Ordinary
Technology	Higher and Ordinary
Design and Communication Graphics	Higher and Ordinary
Home Economics	Higher and Ordinary
Accounting	Higher and Ordinary
Business	Higher and Ordinary
Economics	Higher and Ordinary
Religious Education	Higher and Ordinary
Art (including crafts)	Higher and Ordinary
Music	Higher and Ordinary
Russian	Higher and Ordinary
Japanese	Higher and Ordinary

Figure 3. Senior Cycle Subjects

- Coding
- Civic, Social and Political Education (CSPE)
- Physical Education (PE)
- Digital Media Literacy (DML)
- A Personal Project: Caring for Animals (Level 2)
- Social, Personal and Health Education (SPHE)
- Artistic Performance
- CSI: Exploring Forensic Science (Level 2)
- Chinese Language and Culture
- Philosophy

Figure 4. Short Courses

2. The History of the Irish Landscape

This section will review available literature on the history of CSEd in Ireland: a long history, with computing in education dating back to the 1970s. The aim here is not to detail each step along this journey, but to provide an overview of the journey itself, where the focus will be on the main key turning points through the last 50 years. The history timeline, from the early 1970s to 2020, is summarised in Figure 5 which shows the main events that have defined the landscape today as nodes that are discussed in detail later in this section. This timeline will be broken into three phases, as defined by McGarr (McGarr O. , 2008), for discussion: Phase 1 - The Early Technophiles Stage (1971:1984), Phase 2 - The Keyboarding Phase (1985:1996) and Phase 3 - The Integration Stage (1997:2008).

In 2009 McGarr compiled a comprehensive history of information technology use in education in Ireland (McGarr O. , 2008). This report detailed the how the Irish educational system responded to changes in various ICT initiatives and policy changes from 1975 to 2008. Other documents covering parts of the story include that by Moynihan (Moynihan, 1986) (to the mid-1980s), Oldham (Oldham, 2015) (up to 1997) and McGarr and Johnston (from 1997 to 2017) (McGarr & Johnston, 2021), while Connolly et al. (Connolly, Byrne, & Oldham, 2022) give an overview of the whole period. We begin by summarising McGarr's three distinct phases of this history before detailing developments in CSEd from 2009 to present day.

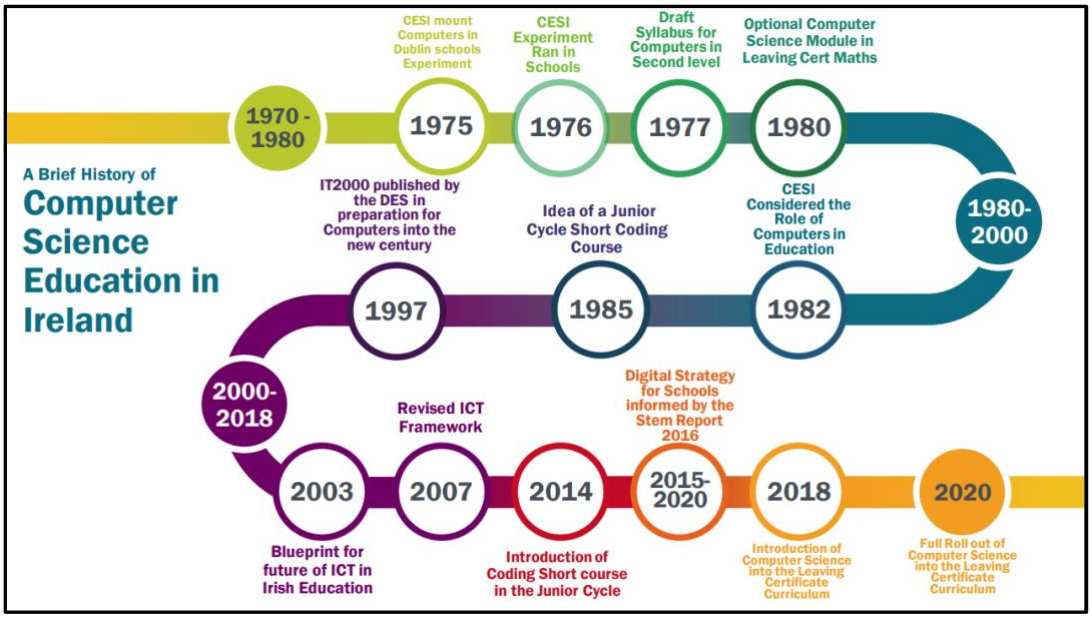

Figure 5. A Timeline of Computer Science in Ireland

2.1 Phase 1 - The Early Technophiles Stage (1971:1984)

In order to get a full picture of the history behind the introduction of CS into primary and post-primary schools in Ireland it is necessary to consider the landscape as far back as the 1970's (see Figure 6). The Computer Education Society of Ireland (CESI, now the Computers in Education Society of Ireland) – which is the official Teachers' Professional Network for CS in Ireland, while also advocating for the use of information technology in teaching and learning across the curriculum (Oldham, 2015) – was established in 1973 and the first chairman Jim Roche published a paper in 1975 detailing the CS training provision for teachers in Ireland (McGarr O. , 2008). Despite these training sessions being open to all teachers they were attended mainly by Maths teachers (Breathnach, 1987) and (McGarr O. , 2008). In 1973, Trinity College Dublin introduced a one-year, part-time diploma course on Computers in Education. This course, in addition to programming, covered a wide range of computer-related topics including the history of computers, problem solving & flow charting, modern computing, hardware, logic and computers in education (Moynihan, 1986). The aim of the course was to grow the base of teacher knowledge in computing. In July 1975, International Computers Ltd. approached CESI to run a trial course of CS in Dublin schools, and from January to April 1976 this course ran in seven Dublin schools (Moynihan, 1986). Moynihan (Moynihan, 1986) concludes that while this experiment was successful there was a requirement to increase the time given to this subject in order to fully embed it into the curriculum.

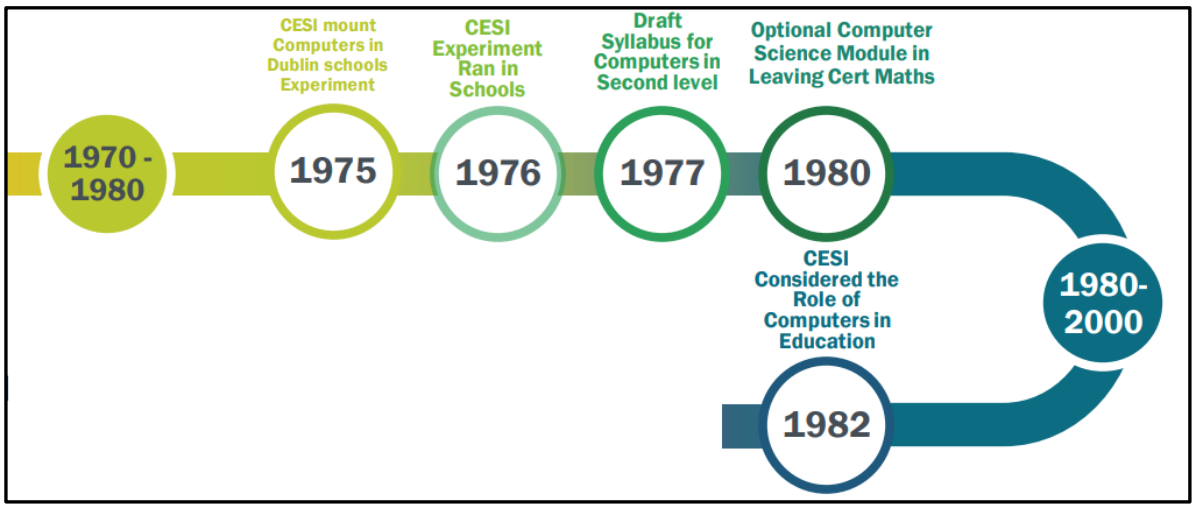

Figure 6. Phase 1 - The Early Technophiles Stage (1971:1984)

The state Department of Education released a white paper in 1980 on educational development. This paper is referred to by Professor John Coolahan in his report "A Review Paper on Thinking and Policies Relating to Teacher Education in Ireland" (Coolahan, 2007). Coolahan determines that the paper failed to focus enough on teacher training – general teacher training / education in Ireland. The lack of any mention of CS teacher training by Coolahan indicates its insignificance in Teacher Continuous Professional Development (CPD) during this time. There was however sustained pressure, mainly through CESI, for the development of a CS curriculum for the Senior Cycle at second level, and 1980 saw the introduction of computing to the Senior Cycle as part of the Maths programme. However, the computing element was not mandatory, and it was not examined as part of the LC state exam (Oldham, 2015) and (Connolly, Byrne, & Oldham, 2022). In addition to this, the syllabus was not set in stone by the Department; instead, schools applying to run the subject submitted their proposed syllabus to the Department as part of their application, allowing for changes over time (McGarr, The development of ICT across the curriculum in Irish schools: A historical perspective, 2008) and (Oldham, 2015). McGarr explains (though "chiefly" would be more accurate than "exclusively" (Oldham, 2015):

" The syllabus required the inclusion of issues such as: Careers in computing, structured diagrams, problem analysis and programming languages. This type of content was typical of the computer use in schools in the early years of the decade as the focus, at that time, was exclusively on learning about the new developing technology."

In 1984 the Department of Education sanctioned a subject on computer studies at Junior Cycle and a syllabus committee was set up to devise the content for this subject, which was introduced in 1985. However, it is worth noting that this subject, again, was not going to be assessed in the state exams

(Oldham, 2015) and (Connolly, Byrne, & Oldham, 2022). It was also around this time that a survey by the Association of Secondary Teachers, Ireland (ASTI, one of two post-primary teachers' unions) noted that a wide range of computer applications were being used in schools and by teachers. While organisations such as CESI were advocating both for a CS curriculum and for the use of computers in teaching and learning (Oldham, 2015), the practice on the ground seemed to be less CS-focused.

2.2 Phase 2 - The Keyboarding Phase (1985:1996)

McGarr notes that, during the keyboarding phase (see Figure 7), while there was investment in the technology for schools by the Department of Education, teachers seemed to be teaching a broad course in computer literacy rather than CS (Brady, 1987), and (McGarr, The development of ICT across the curriculum in Irish schools: A historical perspective, 2008). Notably, the JC Computer Studies course focused on use of applications packages as well as programming (Oldham, 2015). This is also addressed by Moynihan in his MSc. thesis titled "Computer Education in Ireland: A Case Study" (Moynihan, 1986). Moynihan had been part of CESI since 1976 and had been the chairman since 1978. He had advocated strongly for the introduction of CS into the school curriculum; however, this document (Moynihan, 1986) details the issues that arose as part of his campaign to introduce CS into the "rigid centralised Irish Educational System". The issues are summarised by Moynihan (Moynihan, 1986) as follows:

" The DES has not got the personnel, the structures or the expertise to provide the framework for the proper introduction of Computer Education into schools. The wider concept of Information Technology is simply not understood at official level."

During this time the Curriculum and Examinations Board was established. Their role was to oversee the design of new school curricula (McGarr O. , 2008). This board favoured the integration of ICT across the curriculum but mainly focused on Business and Technology subjects. In 1993 an EU evaluation on the use of computing at second level was undertaken. This report highlighted some of the inadequacies around the teaching of computing in Irish schools (McGarr O. , 2008). These inadequacies included: the use of standard applications such as word processing, little emphasis on CS and a lack of explicitly stated policy. This was followed by another large-scale study by Drury which backed the initial inadequacies, and further discovered that in the absence of national strategy, schools had developed informatics courses focused on computer applications software (Drury, 1995), and (McKenna, Brady, Bates, Brick, & Drury, 1993).

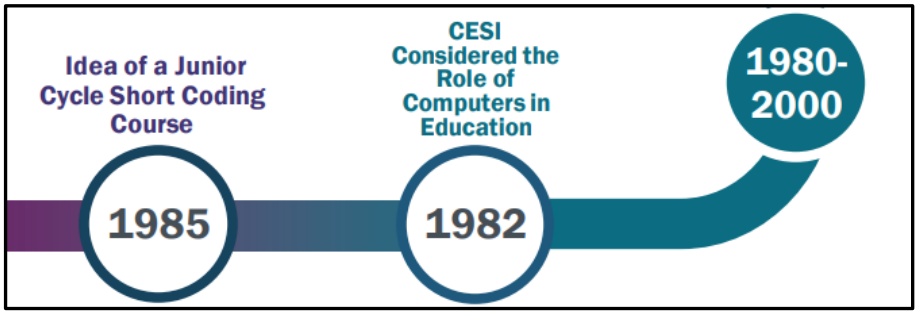

Figure 7. Phase 2 - The Keyboarding Phase (1985:1996)

These reviews through the 1990s shows that the lack of policy from the Department of Education in relation to computing in schools had led to a situation where computing in schools were largely being used for informatics classes (McGarr O. , 2008) and (Oldham, 2015) rather than CS. Bearing in mind that the initial stages of computer education of teachers was chiefly based around computer programming and CS, this was disappointing.

2.3 Phase 3 - The Integration Stage (1997:2008)

In 1996 an International Data Corporation (IDC) report ranked Ireland in the third division in relation to its state of preparedness for the Information Age. This sparked renewed efforts to incorporate ICT into schools in a meaningful way. McGarr states that this, the third phase (see Figure 8), is marked by the launch of the Schools IT2000 Initiative. The aim of this initiative was to increase student literacy in computing and to support teachers in developing the skills needed to support their students. This initiative improved the uptake of ICT across Ireland (McGarr O. , 2008). In 2001 an evaluation of the Schools IT2000 Initiative (Department of Education, 1997) found increased IT infrastructure in schools as well as increased uptake in teacher training, however it also found that a more defined policy was needed and that basic informatics type classes were still predominant in schools. McGarr notes that the absence of a clearly defined national CS policy for schools has been an ongoing problem in Irish schools over the past 30 years and this is also corroborated by Rinn, 1984, Kelly 1985, NCCA 1993, and Mulkeen 2002 according to McGarr's report (McGarr O. , 2008).

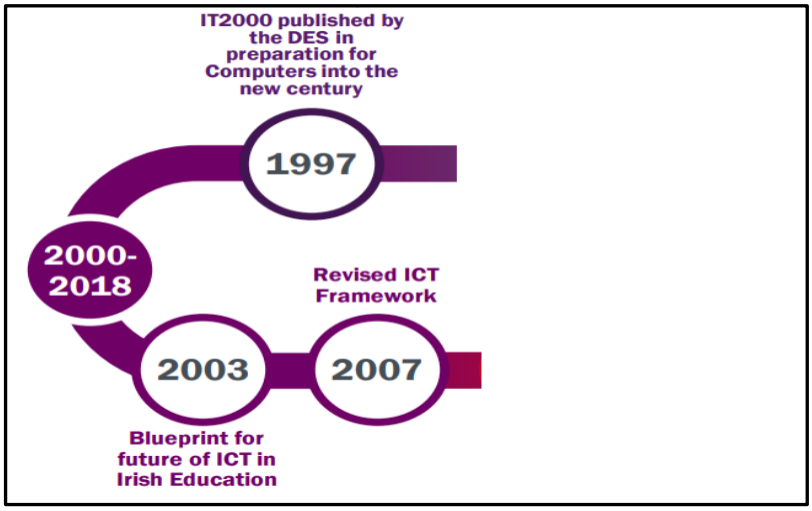

Figure 8. Phase 3 - The Integration Stage (1997:2008)

McGarr also notes the effect of a cabinet re-shuffle within three years of the Schools IT2000 Initiative where the appointment of a new Minister for Education, with different priorities, resulted in an ICT decline at this critical time. Having devised the three phases, as summarised above, McGarr concludes that CS at second level evolved rather than was developed. McGarr also states that in order for Ireland to continue to improve, lessons should be learnt from the past, whereby ICT initiatives and policies need to be presented as integrated teaching and learning with relevance for all teachers.

2.4 Developments mainly since 2008

The Schools IT2000 Initiative was the start of real change in the landscape of Ireland's approach to CS. McGarr and Johnston reviewed educational policy and how it has framed the CSEd in Ireland in a paper in 2017 entitled "Exploring the Evolution of Educational Technology Policy in Ireland: From Catching up to Pedagogical Maturity" (McGarr & Johnston, 2021). That paper reviewed a number of educational policy documents including:

- Schools IT2000 - A policy Framework for the New Millennium (Department of Education, 1997)
- Investing effectively in Information and Communication Technologies in Schools 2008 (Department of Education, 2013)
- The report of the Minister's Strategy Group (Department of Education, 2008)
- Smart Schools = Smart Economy: Report of the ICT in Schools Joint Advisory Group to the Minister for Education and Science (ICT Ireland, 2009)

- Digital Strategy for Schools 2015 - 2020: Enhancing Teaching, Learning and Assessment (Department of Education, 2016)

- Digital Strategy for Schools: Action Plan 2015 - 2020 (Department of Education, 2017)

McGarr and Johnston also explain four possible rationale categories for including technology education in schools. These are:

1. Educational - this justification for technology inclusion is based on enhanced learning for students.

2. Economic/Vocational - this justification for technology inclusion is to prepare students for roles is Science, Technology, Engineering and Maths (STEM) fields in order to advance the economic competitiveness of the country by producing more technically skills graduates (McGarr O. , 2008).

3. Social - this justification for technology inclusion is based on the issue if inequality and therefore looks at ensuring all students have access to adequate digital skills.

4. Catalytic - this justification for technology inclusion is based on the effect it can have on changing educational practices from teacher-led to student-led learning.

In examining these policy documents McGarr and Johnston (McGarr & Johnston, 2021) drew several conclusions. During the 20-year period examined in this paper the nature of computing in schools has changed significantly and the educational policy has responded to these changes bringing Ireland to a point where the educational system realised the need to facilitate the more independent, student centred learning for technology to be capitalised on effectively. The study also found that the impetus for these changes over the 20-year period were mainly Economic (McGarr & Johnston, 2021), where there is a body of data identified that would support this in the OECD reports from 2004, 2012 and 2015 suggesting that the trends in Irish schools in relation to technology was average or below average by international comparison. This drove the desire to use technology effectively rather than having it as a stand-alone subject in schools. Ireland was not alone in this Economic driving factor in Educational policy, as Canada and Northern Ireland also found that economic and political forces shaped educational policy (McGarr & Johnston, 2021) and (Connolly, Byrne, & Oldham, 2022).

3. Present Day Computer Science Education in Ireland

This section will review CSEd in Ireland from 2017 up to the introduction of the formal LCCS subject in 2018 (see Figure 9).

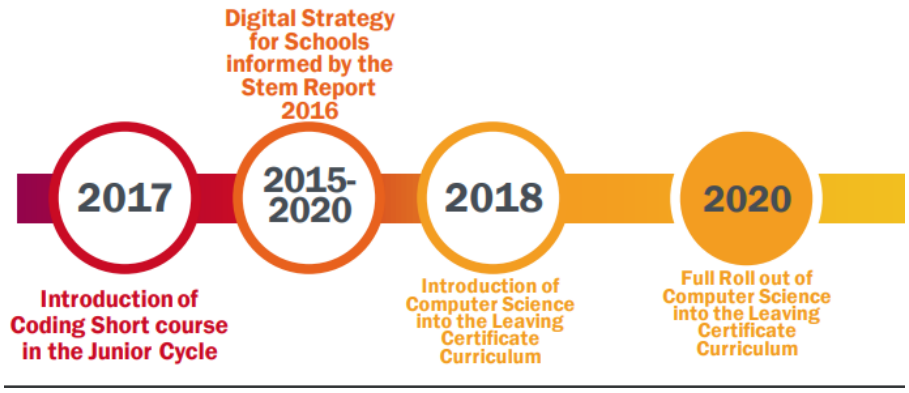

Figure 9. Current Phase

3.1 Primary Computer Science

In July 2016 the NCCA was asked by the then Minister for Education and Skills, Mr Richard Bruton TD, to consider approaches to integrating coding and computational thinking into the primary curriculum. The NCCA implemented several phases to help inform the development of the curriculum, before any work was done on the curriculum document itself, thus inverting the common approach of the developing the curriculum first (with multiple stakeholders) and investigating its suitability after the specification has been rolled nationally. This resulted in initial research into coding in primary schools in other jurisdictions (NCCA, 2019). From this the "Coding in Primary Schools Initiative" began with phase one in September 2017. This involved 15 primary schools, which were attained through an open call process (47 schools applied). The focus of phase one was to document coding practices in these 15 schools, all 15 schools had prior knowledge and teachers with experience of teaching coding and computational thinking in a primary classroom. The findings from this were then used to inform the development of materials for phase two. Phase two took place between May 2018 and February 2019, and consisted of 25 additional schools (153 schools applied), in addition to the 15 phase one schools. As there are 3107 primary schools in Ireland (Department of Education, 2019), ~1.3% of primary schools participated in phase two. The additional phase two schools did not have the prior knowledge and practices in place like phase one schools. Phase two focused on developing learning outcomes to use from Junior Infants to Sixth Class, that would allow for potential progression to the JC Coding short course and the LCCS course. The learning outcomes focused on physical computing and play-based pedagogical approaches to coding and computational thinking and were split into two separate groups: Junior Infants - Second Class, and Third - Sixth Class. Most teachers from phase two felt that the language used in the outcomes was too technical and that there needed to be clearer explanation of some of the terms. Some teachers also stated that they did not see a clear progression from the early years outcomes through to the senior years. When teachers were asked

what the main challenges might be, they identified curriculum overload, teacher confidence, CPD, and school infrastructure as being the most pressing issues (NCCA, 2019).

As of June 2019, approximately 3,180 primary teachers have participated in coding and computational thinking face-to-face workshops and, from 2014 - June 2019, a total of 3,463 teachers completed the online "Scratch for Learning" course, both facilitated by the PDST. The feedback from the teachers participating in phase two is that time will be needed to embed the concepts of coding and computational thinking in classrooms. This is part of a larger review of the entire curriculum planning (NCCA, 2019) to include the development of the final primary school curriculum, which at present, due to the COVID pandemic, is postponed.

3.2 Junior Cycle Computer Science

In 2014 the NCCA produced nine short courses which schools could include in their JC curriculum (Fleming & McInerney, 2019) one of which was the short Coding course for the Junior Cycle (NCCA, 2016). This course, which was part of the JC reform (NCCA, 2021), had three strands of learning associated with it: CS Introduction, Let's Get Connected and Coding at the Next Level. This course was piloted in 2016 with 22 schools taking part and released in 2017 with 52 schools taking part (Fleming & McInerney, 2019), and (NCCA, 2016). The aims of these strands were to introduce the learners to coding and the broader view of CS including algorithms, problem solving and testing code (NCCA, 2016). The purpose of this pilot was to examine the current provision and opportunities within schools for Information and Communications Technology. Its goal was to support and document the experiences of a small number of schools as they incorporate aspects of the Coding short course within their JC programme and to explore further options for support of schools and teachers offering the Coding short course. Lero, the Science Foundation Ireland Research Centre for Software, which brings together expert software teams from universities and institutes of technology across Ireland in a co-ordinated centre of research excellence with a strong industry focus, was chosen to conduct this research and create the report.

The study found that there was little difference in the gender breakdown of teachers teaching this course at 55%:45% male: female respectively. Expertise of the teachers delivering this course was predominately in Technology and Maths at a combined total of 45%. However, what was also discovered was the diverse time allocations across the surveyed schools to this course where 27% were seen to be allocating the required time but as high as 21% were not meeting the required time for the short course, which is 100 hours. This was compounded in the latter years as students drew closer to the formal JC state exams (Fleming & McInerney, 2019).

The statistics around student engagement, which was reviewed as part of a report by the NCCA (NCCA, 2019), in the short course showed very little gap between the male and female uptake at 51%

and 49% respectively, where 48% of the students had some previous exposure to coding. The previous exposure differed greatly from student to student with some students experiencing this at home and others attending summer camps. Perhaps the most interesting indicator from students was their enjoyment of the course. Given the concerning lack of female uptake in CS at third level, the breakdown of enjoyment is best looked at in this context. Here the NCCA report stated (NCCA, 2019) that 62% of males reported enjoying it while 48% of females did. Disappointingly, of the students who studied the Coding short course, only 20% of them said they would consider CS as an option at third level, with 47% saying they would not consider it.

There were a number of successes and challenges identified in the Lero report (Fleming & McInerney, 2019) from the introduction of short courses into the JC curriculum. The successes included the student interest and engagement, the development of student skills and the new teaching methodologies learned. Some of the challenges faced were timetabling of the subject, access to resources and the additional time requirement from teachers. Despite the challenges there was a large interest in delivering the Coding short course, in particular the continued interest of participating schools at 75% saying they planned to continue delivery in the 2017/2018 academic year. The introduction of the LCCS subject in 2018 has also had a positive effect on the acknowledgment across the educational sector of the need for these types of offerings at JC level. Continuing efforts need to be made by the initial cohort of schools from resourcing the course, supporting teachers and expanding communities of practice to allow for flexibility in order to ensure the sustainability of a coding for all approach.

3.3 Leaving Certificate Computer Science

In January 2017 the NCCA announced the new Leaving Certificate Computer Science subject. The Minister at the time, Mr. Richard Bruton TD, said.

" The introduction of Computer Science as a Leaving Certificate subject is part of the Government's overall commitment to embed digital technology in teaching and learning. The society our children will grow up in, will be one which has been fundamentally transformed by new technology. Our education system must prepare our children to thrive in such an environment by equipping them with skills in creativity, adaptability and problem solving."

In 2018, the LCCS subject was rolled out in a pilot phase to 40 schools around Ireland (phase one) (NCCA, 2019). The geographical spread of the phase one schools can be seen in Figure 10, where data was obtained from the LERO Interim report 2019 (McGarr, McInerney, Exton, & Power, 2019). These schools were selected based on an application submitted to the DES. The selection represented both a good geographical spread as well as being balanced in terms of gender numbers and DEIS School representation (McGarr, McInerney, Exton, & Power, 2019). Of the chosen schools, 31% were already

offering the JC Coding Short course (NCCA, 2016), with another 21% of schools selected offering the JC Digital Media Literacy course (NCCA, 2021). Of the participating schools, 38% of teachers were female. The teachers prior experience of this new subject area varied from teaching IT in the school previously and supporting IT in the school, to having CS industry experience (McGarr, McInerney, Exton, & Power, 2019).

Figure 10. Geographical Spread Phase One Schools (Fleming & McInerney, 2019)

The specification of the LCCS curriculum was designed assuming no previous experience in the area and prescribes 180 contact hours for the subject. The content is based around three interwoven strands consisting of: Practices and principles; Core concepts; and Computer science in practice (see Figure 11). Strand three is a group of applied learning tasks (ALTs) that provide an opportunity for skills based practical learning in the classroom. These practical skills are then further enhanced through the coursework part of the LC assessment.

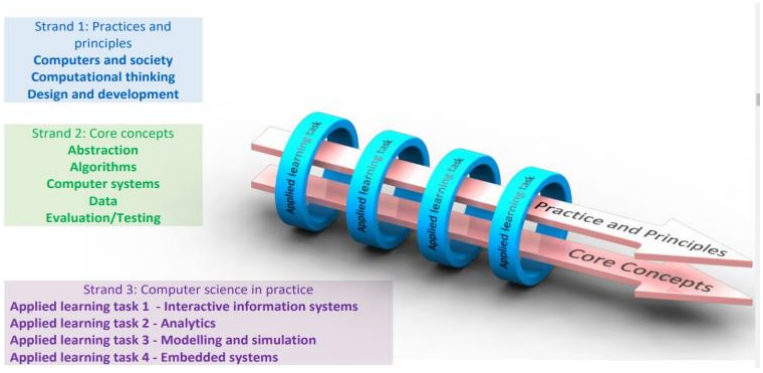

Figure 11. Leaving Certificate Strands

The aim of these strands is to enhance the learners' creativity and problem solving as well as facilitating independent and collaborative work. The strands can be completed in any order. In terms of

assessing the students there is a piece of continuous work (coursework), worth 30% of their marks, and one terminal exam in the subject, worth the remaining 70% of marks. The coding part of the course is taught using Python and JavaScript, although the aim is to regularly review this in line with industry developments (McGarr, McInerney, Exton, & Power, 2019).

Teachers were offered CPD training and continued support in the delivery of this subject from the NCCA. The training offered consists of a mix between National workshops, Skills workshops, Regional clusters and Webinars. The Lero report 2020 determined that both the National workshops and the Skills workshops were by far the most valued by the teachers (McGarr, McInerney, Exton, & Power, 2020). The feedback from the teachers while positive was diverse; several teachers felt that these training opportunities should focus more on content or practical skills to bring back to the classroom rather than pedagogical approaches (McGarr, McInerney, Exton, & Power, 2020), while others valued the pedagogical content more. This highlighted the different needs of teachers with some requiring content knowledge (CK) rather than pedagogical content knowledge (PCK). This emphases the requirement for teachers to feel confident in the CK before they feel confident to approach PCK.

Phase one completed in 2020, where students (739) that were due to sit the first ever LCCS exam in June 2020 received so-called "calculated grades" due the COVID pandemic (Department of Education, 2020). Students in this initial cohort were offered the opportunity to sit an exam in LCCS in November 2020, where a very small group of approximately 12 students completed this formal sitting. The NCCA continued with the national roll-out in September 2020 for this subject. The figure for the additional schools who started phase two in September 2020 was 52; this was lower than expected, perhaps due to COVID, where some schools might have decided to postpone the roll-out. This resulted in 92 schools offering LCCS in the 2020-21 academic year. Phase three commenced in September 2021 and there is expected to be approximately another 50 schools who opt to offer LCCS as part of this phase. Uptake for the first three phases is positive despite the setbacks that schools have faced due to the COVID pandemic, given that some 20% of post-primary schools in Ireland are offering LCCS. In terms of the student uptake in these schools at the end of phase one, 1.4% of senior cycle students were taking LCCS. If the same level of students take the subject in the phase two and phase three, Ireland will be on the right path to attempt to match the uptake for A Level Computer Science which was at 10.8% in 2019 (Ofqual, 2019).

The next section examines the international jurisdictions that specifically informed the Irish LCCS subject. It will observe their current CS landscape and how the roll-out of their CS in schools has progressed, with a specific focus on any lessons learned that might provide insights for Ireland going forward.

4. Lessons Learned from International Jurisdictions

The 2017 Irish commissioned report, the "Report on the Provision of Courses in Computer Science at the Senior Cycle level of Education Internationally" (Keane & McInerney, 2017) investigated the international provisioning of CS in five jurisdictions: England, Scotland, New Zealand, Ontario, and Israel across areas such are rationale and motivation, curriculum, assessment, participation, and teacher CPD, with the aim to help inform the provisioning of the LCCS roll-out. Given that these specific countries informed the current Irish Senior Cycle LCCS, their entire primary and second level CS initiatives would perhaps be noteworthy to readers interested in the Irish story. With the aim to summarize any common lessons learned that could provide insights for Ireland as it moves forward with its CS roll-out. Table 3 gives an overview of their CS offerings across primary and second level, which will be summarised and discussed in the following sections.

Table 3. Jurisdictions of Interest to Ireland Delivering CS in schools, compiled from (Keane & McInerney, 2017)

Country List			
Country Name	Primary	Lower Secondary	Upper Secondary
England	Mandatory	Optional	Optional
Ireland	No	Optional Short Course	Optional
Israel	No	Mandatory and Optional Modules	Optional
New Zealand	Mandatory	Mandatory	Optional
Ontario	Integrated with Maths	No	Optional
Scotland	Mandatory	Mandatory	Optional

England

Interest groups such as Computing At School (CAS) led to England, in 2014, being the first country to introduce CS (which included coding) as a curriculum at primary level through a subject titled Computing. At second level, since September 2015, students in England can choose to continue to study CS through elective subjects at GCSE and A-Levels (both of which are named Computer Science). In 2017, the UK commissioned and produced a report titled "After the reboot: computing education in UK schools" (The Royal Society, 2017) which details CS curriculum and initiatives in schools across the UK. In summary the report stated that a majority of teachers felt they were

delivering a subject they were inexperienced with, and that typically they might be the only teacher in their school tasked with delivering the subject.

In 2018, it was reported that 70% of students in England attend schools offering GCSE CS, but only 11% of all students take GCSE CS. With 10.8% of A-Level students taking CS (Ofqual, 2019), which would imply that although there is a reduction in the overall number students who take the A-Levels (Department for Education, 2012), the ratio of students electing to take CS at this level remains at ~11% of the student population. However, a concern is that data shows that only 9.8% of the A-Level CS students were female (Keane & McInerney, 2017). The previous statistics initially suggest that delivery of CS to all primary school students has not yet dramatically increased the percentage of students, or the number females, opting to take this subject in senior years. However, it will be necessary to wait until a full cycle of primary students have been exposed to CS from entry to exit, for the final data to be analysed and adequate conclusions drawn.

Scotland

One of the four constituent nations in the UK is Scotland, where computing has been a subject delivered since the 1980s at second level (Jones, et al., 2011), although it was not delivered at primary school until 2010. While CS was available at primary school in Scotland before England (2014), it was introduced with the requirement only to ensure all students must at the very least be "exposed" to CS by the age of 14 (Keane & McInerney, 2017). In 2014, Scotland introduced Computing Science into the Junior Cycle curriculum, which leads to a National 5 (N5) qualification, the Scottish equivalent of England's GCSE, or Ireland's JC. While the number of entrants to the N5 CS subject from 2014 to 2017 was 7,000 students (Scottish Qualifications Authority, 2021), more recent years show a slow decline in the uptake at N5 (Scottish Qualifications Authority, 2021). The 6,221 students that elected to take the subject represented only just over 7.5% of the number of students who took the N5s in 2020, with only 1,256 (~20%) females. Additionally in 2020, just under half the number of students that elected to take the N5 exams elected to take CS at the Senior Cycle level, indicating a drop away of students (with only 17% females) (Scottish Qualifications Authority, 2021).

Scotland is further along in their CS curriculum roll-out than England, with the availability of CS as an optional subject at second level in some form since 1980s. This demonstrates it has not had the long-running issues with its development as Ireland and England seem to have had with the change in focus from CS to ICT usage. However, statistics show its uptake is still low and is decreasing at second level. Also a concern for Scotland is the 25% decrease in the number of computing teachers available compared to the previous ten years, with 17% of second level schools having no computing specialist to deliver the subject (The Royal Society, 2017). The number of first-year students on computing

initial teacher training courses dropped by 80% from 2007 to 2016, resulting in a number of universities dropping their Postgraduate Certificate in Education in CS (The Royal Society, 2017).

Israel

CS is not incorporated as a stand-alone subject at primary school. At secondary school in 2012, they introduced a new CS programme for the Junior Cycle in schools (Zur Bargury, et al., 2012, October). However, at Senior Cycle, Israel has offered an optional CS subject for the past five decades, where it has maintained and developed its current curriculum since 1998, but it has been an optional subject since the mid-1970s (Gal-Ezer, Beeri, Harel, & Yehudai, 1995). Its major principle is the ”zipper principle” that describes interweaving conceptual and experimental topics - essentially ensuring theory is delivered alongside hands on practical work. This closely mirrors the interweaving theme also seen in the Irish LCCS curriculum, which may be in part due to the fact the zipper principle was discussed and investigated as part of the Lero report (Keane & McInerney, 2017).

The average percentage of students who took CS at Senior Cycle level over a sixteen-year period was 18% of the total exam population (Keane & McInerney, 2017). Its long-standing implementation may also contribute to its sustained female participation rate of over 40% between 1995 and 2011 (Keane & McInerney, 2017). The fact that since 1998 individuals wishing to teach CS in high schools had to obtain a degree in CS coupled with formal teacher training, has been acknowledged as part of its success (Keane & McInerney, 2017).

New Zealand

In 2011, New Zealand, through the Digital Technologies curriculum introduced a number of optional Programming and CS modules for Senior Cycle level students (Keane & McInerney, 2017). These are part of The National Certificate in Educational Achievement (NCEA), the New Zealand equivalent of the Irish LC. These modules which became available in 2012 are assessed 100% on the students' course work, with no terminal examination paper which the majority of other NCEA subjects include. The sign-up initially has been low for the Programming and CS modules but there has been an upward trend in students taking the modules (Keane & McInerney, 2017). The New Zealand government updated the curriculum in 2017, where from January 2020 it is mandatory for all students in Years 1-10. However, a report titled "It's early days for the new digital technologies curriculum" from 2019 reported that only 7% of all the schools had a quite good understanding and enough knowledge and skills to start to implement the Digital Technologies curriculum. The majority of schools (88%) felt somewhat prepared (Education Review Office, 2019). The initial new Digital Standards that were introduced quickly over a two-year period from 2009-2011 resulted in teachers having little time to prepare, which may have impacted on the feeling of unpreparedness of teachers. This same issue might be replicated or - worse - dilated for the Year 1-10 mandatory roll-out. In the coming years it

will be easier to assess New Zealand's approach with its mandatory roll-out of CS to all students and preparing their teachers to deliver the new modules, but at the moment it is too early to draw conclusions.

Ontario

In September 2020 Ontario released a new maths curriculum at primary school for grades one through eight that integrates coding into the algebra strand of the curriculum, with clear coding expectations specified in the curriculum at each grade level. On entering grade nine, students can elect to take one introductory broad-based technology course called Exploring Technologies. This is followed by a subject called Computer Studies at the senior level, which has been in place since 2009, across grades 10-12. New entrant teachers teaching the subject are required to have an Ontario teaching licence and also a third level qualification in CS, software engineering or equivalent professional experience. Although the programme has been running for many years, the absence of publicly available data makes it difficult to conduct analysis of student uptake (Keane & McInerney, 2017).

4.1 International CS Rollout Discussion

CSEd at primary and secondary levels is expanding around the globe and is being formalised and integrated into school curricula internationally, with many schools moving in recent years to incorporating core CS subjects across their primary schools either as a mandatory stand-alone subject, or integrated with Maths as is in the case in Ontario, or as a set of modules in a wider programme, like New Zealand (see table 4 for summary). On examination of the current status of the international jurisdictions that informed the Irish LCCS, it is clear there needs to be a growing sense of urgency about progressing CS in Irish schools, not only at LCCS, but also as an examined subject in the JC, and integrated throughout primary school level. This is to ensure Ireland keeps abreast, and to ensure success and uptake of this subject by students in participating schools.

Table 4. Countries Delivering CS in schools that Informed Ireland

	Pre-University CS Offerings														
	Primary					Lower Secondary					Upper Secondary				
Country Name	Mandatory	Formal Curriculum	Standalone Subjects(s)	Module(s) within a wider	Integrated Subject	Mandatory	Formal Curriculum	Standalone Subjects(s)	Module(s) within a wider	Integrated Subject	Mandatory	Formal Curriculum	Standalone Subject(s)	Module(s) within a wider	Integrated Subject
England	*	*	*				*	*				*	*		
Ireland							*	*				*	*		
Israel						*	*		*			*	*		
New Zealand	*	*		*		*	*		*			*		*	
Ontario	*	*			*		*	*				*	*		
Scotland	*	*		*			*	*				*	*		

The 2017 Irish report entitled “Computer Science in Upper Second Level Education Internationally” (Keane & McInerney, 2017) looked in detail at the provision of CS in Israel, England, Scotland, Ontario and New Zealand. The report examined challenges faced, course content, learning outcomes and teacher CPD in each of these countries. It highlighted that other countries, such as England and New Zealand, also like Ireland had a long road to ensuring the curriculum is grounded in CS rather than on the use of computers. In particular, the English curriculum within the UK has informed the Irish curriculum, so their story is of interest and closely related to the Irish one as an insight of how another country rolled out CS to all. Specifically, their roll-out along with its success and/or failures can help guide Ireland as it progresses through its own roll-out.

Recent revisions in the English and New Zealand curriculum are taking place to ensure the core concepts of CS are now the foundations of their programmes. The report (Keane & McInerney, 2017) provided a pathway for the formal introduction of CS at the Senior Cycle in Ireland. Along with England and New Zealand, Scotland also recently revised their CS curriculum, the collective aim with

these revisions was not only to ensure the curricula were grounded in CS, but also to ensure that they supported student learning and scaffolding of the subject through all school years.

However, two key challenges stand out in all the international research, which Ireland needs to understand and learn from:

1. Low uptake, and in particular low female uptake of CS across countries, with the exception of Israel. Of particular note, Scotland, which has an established track record of CS in schools, has had notable decline in uptake over the past number of years.

2. Teacher Continuous Professional Development (CPD) to ensure teachers are supported is vital to the success of the subject's integration. In all countries where the uptake of the subject was low, the main prohibiting factor seemed to relate to the lack of teacher CPD and support. For example, New Zealand's short lead-in left teachers and schools unprepared and ill equipped to successfully deliver the CS subject. Also, in England teachers stated they felt they were delivering a subject they were inexperienced with, and that typically they might be the only teacher in their school tasked with delivering the subject. Where in Scotland there has been a 25% decrease in the number of computing teachers available compared to the previous ten years. To further support this theory, Israel who do not seem to suffer the same uptake issues, provide high quality teacher CPD and support.

Ireland needs to ensure it does not encounter these same issues faced by other countries as it integrates CS into primary level and cements its establishment at second level.

5. Discussion and Future Work

ICT in schools in Ireland has come full circle with the introduction of the LCCS in 2018, to focusing on CS rather than computer use which was in part due to lessons learned from the reviewed international countries. At this stage the only results that have been released from the SEC in this subject in Ireland are for those who sat the exam in November 2020. These results indicate that one third of students got in the top three grade bands (CAO, 2021) and other two thirds were in the next three grade bands with no one receiving grades lower than a H6 (CAO, 2021). Given the fact that in 2020 all LC results were predicted grades, the students who sat the exam in November are very low at 12 students and without public availability of the overall results including the predicted grades there are no conclusions, based on results, to be drawn at this stage. The introduction of the JC Coding short course is another positive step. This short course can help in a number of ways: It can prepare students for the LCCS course; it can educate students about CS; and it can also allow students realise that CS might not be for them. The NCCA are currently reviewing and redeveloping the primary school curriculum, and this work provides a timely opportunity to integrate coding and computational

thinking skills in the curriculum that aligns with progression into the JC short course and the LCCS. The possible introduction of formal CS at primary level, which will align with the JC short course and the LCCS will help ensure the success of CS at both primary and second level in Ireland. At a minimum the continuation of CS at second level is really important and the combination of the JC short course as well as the LCCS subject provides for this.

It is clear that CS at Senior Cycle level has taken many years to become a reality in Ireland and was greatly informed by the Irish report entitled "Computer Science in Upper Second Level Education Internationally" (Keane & McInerney, 2017) and the international countries this report reviewed. Ireland is on par with these international jurisdictions at Senior Cycle, and while it has an offering at Junior Cycle, this needs to progress to an examinable subject at JC to ensure its uptake by students. At Primary school, Ireland is lagging behind and needs to push forward with integrating CS at Primary level. All the while ensuring it does not encounter the same issues as the jurisdictions that informed Ireland did. From investigation into these jurisdictions, there are commonalities with their CS roll-out having been impacted with regards to participation levels, including low female participation levels and teacher CPD. Ireland is in such an early stage with their roll-out that it cannot be determined as of yet as to whether it will also suffer from these issues. Moving forward, Ireland needs to ensure that it not only increases its CS offerings at K-12 to keep on par internationally, but it also needs to ensure it does not encounter issues around participation rates and teacher CPD as it introduces formal CS into primary level and cements its establishment at second level.

The next number of years will see the national roll-out of the LCCS subject. The uptake of this will be interesting to watch and of course the output i.e., the student results, will also be important. Combined with the ongoing work looking at how to introduce computing concepts at primary level, the landscape of CS in schools in Ireland will be a rapidly evolving scene over the coming years. Ireland needs to also complete their CS offerings at primary school and in the enhance the offering at Junior Cycle of secondary school to ensure the same scaffolding for CS exists from entry to exit in the Irish school system.

The first two authors are involved in a body of work to help assist these goals and are conducting interventions and longitudinal studies with both students and teachers (over the coming years), where their wider team plan to share all of this work, founded on research nationally and internationally. Part of the future work will be to detail if Ireland's CS roll-out was successful and if it did encounter participation and teacher CPD issues.

6. Conclusion

In conclusion, the introduction of Leaving Certificate Computer Science (LCCS) in Ireland in 2018 marked a significant milestone in the country's formal Computer Science education history. The roll-

out of this subject followed many years of effort to offer Computer Science education at the Senior Cycle level in Irish schools. Although Computer Science education is not yet formalized at the primary school level, it is making positive progress. Additionally, there is an optional coding course offered at the Junior Cycle level. This paper reviewed the literature on the history of Computer Science education in Ireland and examined the 2018 LCCS pilot subject, the Junior Cycle changes, and the current status of Computer Science education at the primary school level. The paper also looked at international Computer Science education landscapes and drew lessons from these jurisdictions to aid the current Irish roll-out. The future of Computer Science education in Ireland is promising, as the national roll-out of LCCS in 2018 has opened opportunities for students to gain a formal education in this subject. The paper is an important contribution to the field of CS education and provides valuable insights for educators and policy makers.

References

Brady, M. (1987). Computers in secondary schools. *COMPASS—Journal of the Irish Association for Curriculum Development, 16*(1), 46-53.

Breathnach, P. (1987). Computer studies survey. *ASTI journal*, 14-18.

CAO. (2021, September 14). *Common points scale and grading system.* Retrieved from CAO: http://www2.cao.ie/downloads/documents/CommonPointsGradingSystem.pdf

CAO. (2021, September 31). *Irish leaving certificate examination points calculation grid.* Retrieved from CAO: https://www.cao.ie/index.php?page=scoring&s=lcepointsgrid

Citizens information. (2020, August 20). *Senior cycle and junior cycle*. Retrieved from Citizens information: https://www.citizensinformation.ie/en/education/primary_and_post_primary_education/going_to_post_primary_school/

Connolly, C., Byrne, J., & Oldham, E. (2022). The trajectory of computer science education policy in Ireland: A document analysis narrative. *European Journal of Education, 57*(3), 512-529.

Coolahan, J. (2007). *A review paper on thinking and policies relating to teacher education in Ireland.* Ireland: Teaching Council. Retrieved October 31, 2022, from https://www.teachingcouncil.ie/en/publications/research/documents/a-review-paper-on-thinking-and-policies-relating-to-teacher-education-in-ireland.pdf

Department for Education. (2012). *Subject progression from gcse to as level and continuation to a level.* United Kingdom: Department for Education. Retrieved from

https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/183942/DFE-RR195.pdf

Department of Education. (1997). *Schools IT 2000: full report.* Ireland: Department of Education. Retrieved September 13, 2021, from https://assets.gov.ie/24665/6bb0ea96e5944d12a39a0fe44ef14c61.pdf

Department of Education. (2008, March 26). *Department of education and science statement of strategy.* Ireland: Department of Education. Retrieved September 21, 2021, from Department of Education: https://www.gov.ie/en/organisation-information/dc5b86-department-of-education-and-science-statement-of-strategy-2008-2010/

Department of Education. (2013). *Investing effectively in information and communication technology in schools, 2008-2013.* Ireland: Department of Education. Retrieved October 31, 2022, from https://www.gov.ie/en/publication/4598cc-investing-effectively-in-information-and-communication-technology-in/

Department of Education. (2015). *A framework for junior cycle.* Ireland: Department of Education. Retrieved August 20, 2020, from https://www.documentcloud.org/documents/453392-a-framework-for-junior-cycle-full-report

Department of Education. (2016, November 2). *Digital strategy for schools 2015-2020: Enhancing teaching, learning and assessment.* Ireland: Department of Education. Retrieved October 31, 2022, from Department of Education: https://assets.gov.ie/25151/52d007db333c42f4a6ad542b5acca53a.pdf

Department of Education. (2017). *Digital strategy for schools 2015-2020: Action plan 2017.* Ireland: Department of Education. Retrieved October 31, 2022, from https://assets.gov.ie/24382/7b035ddc424946fd87858275e1f9c50e.pdf

Department of Education. (2019, November 1). *National school annual census 2019/2020*. Retrieved from Department of Education: https://www.gov.ie/en/collection/63363b-data-on-individual-schools/

Department of Education. (2020, September 1). *Leaving certificate 2020: Your questions answered - October 2020.* Retrieved September 14, 2021, from Department of Education: https://www.gov.ie/en/publication/ad9de-leaving-certificate-2020-your-questions-answered-september-2020/

Department of Education. (2021, November 29). *Department of education – circular cl0059_2021. Arrangements for the implementation of the framework for junior cycle with particular reference to the school year 2021/22.* Retrieved from Department of Education: https://assets.gov.ie/205684/b02e0580-53a8-406a-b8fa-c0487865cb4c.pdf

Drury, C. (1995). *Implementing change in education: The integration of information technology into Irish post-primary schools.* University of Leicester: Unpublished M. Sc. Thesis.

Education Review Office. (2019). *It's early days for the new digital technologies curriculum content.* New Zealand: Education Review Office. Retrieved January 20, 2021, from https://ero.govt.nz/sites/default/files/2021-05/Its-early-days-for-the-new-digital-technologies-curriculum-content.pdf

Falkner, K., Sentance, S., Vivian, R., Barksdale, S., Busuttil, L., Cole, E., . . . Quille, K. (2019). An international study piloting the measuring teacher enacted computing curriculum (metrecc) instrument. *ITiCSE-WGR '19 (Working Group Reports on Innovation and Technology in Computer Science Education)* (pp. 111–142). ACM: NY.

Fleming, Ú., & McInerney, C. (2019). *JCCiA - final report v1*. Limerick: Lero.

Fleming, Ú., & McInerney, C. (2019). *JCCiA - interim report.* Limerick: Lero. Retrieved September 31, 2021, from https://lero.ie/sites/default/files/2019_TR_04_Junior%20Cycle%20Coding%20in%20Action%E2%80%93%20a%20CPD%20initiative%20to%20support%20the%20introduction%20of%20the%20Junior%20Cycle%20short%20course%20Coding.pdf

Gal-Ezer, J., Beeri, C., Harel, D., & Yehudai, A. (1995). A high school program in computer science. *Computer*, *28*(10), 73-80.

Gill, T. (2017). *Uptake of gce a level subjects 2017 statistics report series no. 121.* Cambridge: Cambridge Assessment. Retrieved August 24, 2021, from https://www.cambridgeassessment.org.uk/Images/518880-uptake-of-gce-a-level-subjects-2017.pdf

ICT Ireland. (2009). *Smart schools = smart economy: Report of ICT in schools joint advisory group to the minister for education and science.* Ireland: Department of Education. Retrieved September 13, 2021, from https://assets.gov.ie/24670/be87ca90e27e4c898f0029cfded3fca5.pdf

Jones, S. P., Bell, T., Cutts, Q., Iyer, S., Schulte, C., Vahrenhold, J., & Han, B. (2011). Computing at school. *International comparisons, 7*, 2013.

Keane, N., & McInerney, C. (2017). *Report on the provision of courses in computer science in upper second level education internationally.* Ireland: Lero and NCCA. Retrieved November 20, 2022, from https://ncca.ie/en/resources/computer_science_report_sc/

McGarr, O. (2008). The development of ICT across the curriculum in Irish schools: A historical perspective. *British Journal of Educational Technology, 40*(6), 1094 - 1108.

McGarr, O., & Johnston, K. (2021). Exploring the evolution of educational technology policy in Ireland: From catching-up to pedagogical maturity. *Educational Policy, 35*(6), 841-865.

McGarr, O., McInerney, C., Exton, C., & Power, J. (2019). *Leaving certificate computer science teachers' cpd programme - interim report.* Limerick: Lero: SFI-funded project (Discover programme). Retrieved August 31, 2020, from https://www.lero.ie/sites/default/files/Leaving%20Certificate%20Computer%20Science%20CPD%20Interim%20Report%20October%202019.pdf

McGarr, O., McInerney, C., Exton, C., & Power, J. (2020). *Exploring teachers' professional development to support the roll-out of Computer Science in Irish second-level schools - Final report.* Limerick: Lero: SFI-funded project (Discover programme). Retrieved February 12, 2021, from https://lero.ie/sites/default/files/LCCS%20PD%20Final%20Report%20August%202020.pdf

McKenna, P., Brady, M., Bates, P., Brick, J., & Drury, C. (1993). *New information technology in the irish school system.* Luxembourg: Office for Official Publications (EC).

Moynihan, M. D. (1986, January 1). Computer education : Ireland : a case study. Loughborough University, Loughborough University, UK. Retrieved August 31, 2020, from https://hdl.handle.net/2134/10837

NCCA. (2011). *Towards a framework for junior cycle.* Ireland: NCCA. Retrieved August 20, 2020, from https://ncca.ie/media/2466/towards_aframework_juniorcycle.pdf

NCCA. (2015). *Framework for junior cycle.* Dublin: Department of Education and Skills. Retrieved August 20, 2021, from https://ncca.ie/media/3249/framework-for-junior-cycle-2015-en.pdf

NCCA. (2016, June 1). *Short course coding specification for junior cycle.* Retrieved October 31, 2022, from DES: https://www.curriculumonline.ie/getmedia/cc254b82-1114-496e-bc4a-11f5b14a557f/NCCA-JC-Short-Course-Coding.pdf

NCCA. (2019, April 1). *Leaving certificate computer science.* Retrieved from NCCA: https://www.curriculumonline.ie/getmedia/a4feb183-69fe-4144-9add-56a13a5aa3f8/CS-Coursework-Assessment-Guidelines-201904_en.pdf

NCCA. (2019, May 1). *Primary curriculum review and redevelopment information for schools.* Retrieved September 14, 2021, from NCCA: https://ncca.ie/en/resources/primary-curriculum-review-and-redevelopment-information-for-schools

NCCA. (2019). *Primary developments: final report on the coding in primary schools initiative.* Ireland: NCCA. Retrieved from https://ncca.ie/media/4155/primary-coding_final-report-on-the-coding-in-primary-schools-initiative.pdf

NCCA. (2021, September 13). *Digital media literacy*. Retrieved from NCCA: https://curriculumonline.ie/Junior-cycle/Short-Courses/Digital-Media-Literacy/

NCCA. (2021, September 13). *Framework for junior cycle*. Retrieved from NCCA: https://ncca.ie/en/junior-cycle/framework-for-junior-cycle/

NCCA. (2021, August 31). *Primary curriculum*. Retrieved from NCCA: https://curriculumonline.ie/Primary/

NCCA. (2021, August 30). *Transition year*. Retrieved from NCCA: https://ncca.ie/en/senior-cycle/programmes-and-key-skills/transition-year/

Ofqual. (2019, August 1). *Selected a level subjects 2019.* Retrieved August 20, 2020, from Ofqual: https://assets.publishing.service.gov.uk/government/uploads/system/uploads/attachment_data/file/825429/A-Level-infographic_current___3_.pdf

Oldham, E. (2015). Setting the context: developments in the Republic of Ireland prior to schools IT 2000. In E. Oldham, D. Butler, K. Marshall, & L. Margaret (Eds.), *Shaping the Future: How Technology Can Lead to Educational Transformation* (pp. 19 – 45). Dublin: The Liffey Press.

Scottish Qualifications Authority. (2021, April 29). *Annual statistical report: advanced higher: Computing science.* Retrieved from sqa: https://www.sqa.org.uk/sqa/94723.html

Scottish Qualifications Authority. (2021, April 29). *Annual statistical report: n5: computing science.* Retrieved from sqa: https://www.sqa.org.uk/sqa/94723.html

State Examinations Commission. (2021, August 31). *State examinations statistics*. Retrieved from State examinations commission: https://www.examinations.ie/?l=en&mc=st&sc=r19

The Royal Society. (2017). *After the reboot: computing education in uk schools.* London: The Royal Society, Computing Education. Retrieved August 15, 2020, from https://royalsociety.org/-/media/policy/projects/computing-education/computing-education-report.pdf

Zur Bargury, I., Muller, O., Haberman, B., Zohar, D., Cohen, A., Levy, D., & Hotoveli, R. (2012, October). Implementing a new computer science curriculum for middle school in Israel. *2012 Frontiers in Education Conference Proceedings* (pp. 1-6). IEEE.

A. Appendix

A.1. Table of Irish Educational Terminology

The following table gives an overview of the key terminology used in this paper. The paper is written in the Irish context and as such the authors have used Irish terminology to refer to the school levels and exams. This is may not be a familiar jurisdiction for all readers so this table should help readers in most other jurisdictions.

Table 5. Table of Terminology - Part 1

Terminology Table		
Term	Description	Definition
K-12	Kindergarten to 12th Grade	All school levels
Primary School	Ages approx. 4 to 12 years	Eight years from Junior Infants up to 6th class
Secondary School	Ages approx. 12 to 18 years	Split into Junior Cycle and Senior Cycle
Junior Cycle	Lower Second Level (US), GCSE levels (UK)	Ages 12 years to 15 years
Senior Cycle	Upper Second level (US), A levels (UK)	Ages 15 years to 18 years

LC	Leaving Certificate	State Exam before University
JC	Junior Certificate	State Exam after Junior Cycle (now abolished in favour of Junior Cycle
CESI	Computer Education Society of Ireland (later Computers in Education Society of Ireland)	The main Teacher Professional Network for CS in Ireland
DES	Department of Education	Government Body for Education in Ireland
TES	Teacher Education Services	Promotes the quality of teaching and learning through quality teacher training programmes
LERO	Science Foundation Ireland Research Centre for Software	Proving insights and reports on the implementation of the national CS curricula
ASTI	Association of Secondary Teachers, Ireland	One of the post-primary teachers' unions in Ireland
PDST	Professional Development Service for Teachers	Provides Teacher Continuous Professional Development (CPD) for teachers

Table 6. Table of Terminology - Part 2

Terminology Table		
Term	Description	Definition
NCCA	National Council for Curriculum and Assessment	Provides research informed curriculum and assessment
DEIS	Delivering Equality of Opportunity in Schools	Schools eligible for extra state supports
NFQ	National QualificationsFramework	The Irish National Framework of Qualifications (NFQ) is a 10-level system used to describe qualifications in the Irish education and training system
CS	Computer Science	The term used to describe the Computer Science subject in the

		Irish Educational system
GCSE	General Certificate of Secondary Education	The General Certificate of Secondary Education is an academic qualification in a particular subject, taken in England, Wales, and Northern Ireland.
EU	European Union	The European Union (EU) is a unique economic and political union between 27 European countries.
SEC	State Exams Commission	The State Examinations Commission is responsible for the development, assessment, accreditation and certification of the second-level examinations of the Irish state.
Short Courses	Short Educational Courses at Junior Cycle	Students can acquire a level 3 qualification on the NQF

Analyzing Computational Thinking Studies in Scratch Programming: A Review of Elementary Education Literature

William H. STEWART[1]

Kwanwoo BAEK[2]

[1]Hankuk University of Foreign Studies, Korea

[2]University of Southern California, USA

DOI: 10.21585/ijcses.v6i1.156

Abstract

Computational Thinking (CT) has become popular in recent years and has been recognized as an essential skill for everyone in the digital age. CT literature, however, is at an early stage of development, and there is no consensus among researchers/scholars in the field. To date, many have been unable to concretely explain what CT is, or how to teach and assess this broad skill set. This is particularly evident in different educational contexts and settings such as higher education versus elementary education. The purpose of this cumulative literature review is to examine papers that focus on CT in terms of elementary education, elementary-aged learners, and related issues/considerations in order to provide a better understanding of the CT in an elementary context. An inductive qualitative content analysis was conducted on 58 papers set in elementary school settings about CT from 2010-2020. Five main themes emerged from the review: exploiting tangible blocks in a physical coding environment, integrating *Scratch* into various disciplines through programming, *Scratch* gaming for computational thinking, evaluating computational thinking skills through *Scratch* projects, and teaching and learning methods/factors affecting CT in children. Implications for practice and directions for future research are discussed.

Keywords: Scratch, Computational thinking, Programming, Coding, Elementary Education

1. Introduction

The world has become saturated with digital and computer technology in the 21st century, which in turn has made effective computational tool use a necessary professional and economic skill set (Angeli et al., 2016; Bers, 2010; English, 2016; Miaoulis, 2010; Yadav et al., 2011). Further obfuscating this issue is the simple fact that the logic or principles behind computational tool use, which is known as Computational Thinking (CT), manifest differently in practice depending on the subject matter (Weintrop et al., 2015; Wing, 2017; Yang et al., 2018). In recent years, the discussion on CT has evolved into one not just focused on how CT manifests itself, but also a discussion with a growing call to view CT anew from additional participatory, community, and maker perspectives (Kafai, 2016; Rode et al., 2015), a universal metaphor for reasoning (Henderson et al., 2007), as well as one that envisions CT as an all-encompassing 21st century literacy rather than just a discrete set of skills (diSessa, 2018; Jacob & Warschauer, 2018).

1.1 What is Computational Thinking?

While computational tools (e.g., robotics, programming, simulations, computers, music, maker spaces, etc.) are diverse (perhaps seemingly disparate), underlying the effective use of such tools is CT. CT, however, is an umbrella term for a problem-solving process that encompasses numerous sub skills such as abstraction, decomposition, and simulation (Brennan & Resnick, 2012; Henderson et al., 2007; Wing, 2017). While CT is recognized in broad terms, no consensus exists on exactly how it should be defined, or what skills ultimately constitute CT (Barr & Stephenson, 2011; Barr et al., 2011; Bocconi et al., 2016; Grover & Pea, 2013; Kalelioglu et al., 2016; Lammi et al., 2018; Selby & Woollard, 2013; Weintrop et al., 2015, Yadav et al., 2016). Although CT has been brought to the foreground of the discussion in STEM fields and STEM integration over the last 15 years (see Wing, 2017), CT itself is not new; the origins of CT can be traced back much further to the 1970s and the work of Seymour Papert's LOGO programming and procedural thinking (Bers, 2010; Grover & Pea, 2013; Lye & Koh, 2014; Sengupta et al., 2013; Weintrop et al., 2015; Yadav et al., 2011). Nevertheless, while an effort has been made to promote CT in high school by the computer science community, no analogous effort exists in primary or middle schools (Angeli et al., 2016; Jacob & Warschauer, 2018). Barr and Stephenson (2011) suggested that there may be difficulties transferring CT from a development context situated in higher education, to a K-12 context where CT is applied differently. For example, Lee et al. (2011) noted that there are multiple possible domains (e.g., web design, mobile app development, robotics) that can be used to help develop CT processes/skills in students but these domains may not be widely available in K-12 whereas they are far more common in higher education.

Barr and Stephenson (2011) suggested that any definition of CT should be accessible and framed in terms of the classrooms in which it will take place (versus an overly technical definition). Barr et al. (2011) proposed that CT is a unique combination of cognitive skills that enable a novel form of problem-

solving. This process is also closely tied to various tools (e.g., computers) and can make aspects of problem-solving (i.e., testing, iteration) far more accessible to learners since they can be automated or enacted at a wide scale. Grover and Pea (2013) discussed an overview of the commonalities of various definitions which included abstraction, systematic information processing, symbol systems and representation, and algorithmic concepts such as flow and control. Selby and Woollard (2013) synthesized a definition of CT based on whether or not there was consensus in the literature regarding a specific skill. They suggested that CT can be defined in terms of thinking abstractly, algorithmically, and in terms of decomposition, generalizations, and evaluations. Rode et al. (2015) included aesthetics, creativity constructing, visualizing, and understanding, whereas Jacob and Warschauer (2018) suggested that CT can be re-defined as a new literacy built on programmatic logic. Voogt et al. (2015) elaborated on this definition, describing computational thinking as a universal attitude and skill set that includes decomposition, abstraction, algorithmic thinking and pattern matching and many more. Consequently, computational thinking is considered as a thought process critical for solving problems in a technology-driven society (Kale et al., 2018). Modern digital technology, which is reliant upon programming and coding, is a domain where CT is thought to be necessary. One popular way to explore CT through programming/coding has been through MIT's visual block-based coding platform, *Scratch*.

1.2 What is Scratch Programming/Coding?

Scratch is a block-based visual coding language created by MIT. Although *Scratch* has primarily been associated with a young learning audience (e.g., Chou, 2020; Rose et al., 2020), it's a user-friendly visual interface where students stack and fit blocks together, rather than write code via complex and technical syntax. This block-building metaphor for programming and coding, however, can encourage CT for beginners in the domain regardless of age (e.g., Dolgopolovas et al., 2015; Korkmaz, 2016; Romero et al., 2017). With Scratch, users can learn the fundamental principles of programming (e.g., sequences, loops, conditional statements, etc.) by creating their own projects, such as games or animated videos. By providing an accessible learning environment where young learners can think about such concepts and engage in various cognitive processes, Scratch is particularly impactful for developing problem-solving skills (Berikan & Özdemir, 2019; Donley, 2012; Korkmaz, 2016; Topallia & Cagiltayb, 2018). The cognitive benefits of Scratch include the development of logical, analytical, mathematical, and creative thinking skills as a means to approach complex problems in computer programming (Korkmaz, 2016). As a problem-solving process by extension, these skills not only overlap with CT, but are critical in practice such as abstraction, algorithmic thinking, problem solving, pattern recognition, and design-based thinking (Kalelioglu et al., 2016). As these skills are logical and mathematical in nature, they are heavily implemented in programming environments, and Scratch is no exception–it is rather a question of how and why Scratch affects computational thinking.

1.3 Why and How Scratch Affects Computational Thinking?

Scratch facilitates the process of thinking through higher mathematical understanding, problem-solving strategies, and analytical thinking skills (Korkmaz, 2016). Calao et al. (2015) found that an experimental group of 6th grade students (who have received training in Scratch) had shown statistically significant improvement in mathematical knowledge with respect to modeling, reasoning, and problem-solving. Given that the observed skills in the experiment coincided with some of the skills in the domain of computational thinking, it is appropriate to conclude that the improvement of mathematical processes with the assistance of a visual programming environment (Scratch) can also help facilitate the development of computational thinking. Other research, however, has noted no such relationship. Kalelioglu et al. (2016), for example, concluded that the effects of Scratch on the problem solving-skills of 5th grade students did not yield conclusive results. Merely providing a learning environment was insufficient with regards to teaching effectively and observing students' performance gains. Nevertheless, many students in this study enjoyed and wanted to learn more programming since they were able to utilize their creativity to create games. Thus, while there were no discernable effects in terms of statistically significant results, Scratch did have a noticeable effect on the desire of the students to improve on their programming skills. The increase in motivation or desire, however, can still promote computational thinking.

Although computational thinking is not equivalent to programming, computational thinking can be seen as a problem-solving method utilized by a computer scientist or programmer. As Wing (2010) elaborated, computational thinking is "the thought process involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent" (p. 1). In other words, computational thinking is a way of figuring out how to solve a problem and processing information in a method that is concerned with the realm of computer science. However, what this looks like in practice and in certain contexts is poorly described or conflated in the literature (Weintrop et al., 2015; Wing, 2017; Yang et al., 2018) and the impetus for this review Thus, to comprehensively contextualize CT in Scratch for teaching and learning, as well as explore the assessment of CT, in primary school (i.e., K-9) classrooms, we performed a cumulative literature review (see Templier & Paré, 2015) in order to answer the following research questions:

RQ1: How is CT defined/conceptualized in the context of Scratch in elementary education?

RQ2: How is CT taught and assessed in Scratch in the context of elementary education?

2. Method

Since CT literature is at an early stage of development, there is no consensus among researchers/scholars in the field and to date, many have been unable to concretely explain what CT is, or how to teach and assess this broad skill set. Therefore, the manner of this review was cumulative in nature where the goal is to "compile empirical evidence to map bodies of literature and draw overall conclusions regarding particular topics of interest" (Templier & Paré, 2015, p. 120). Further, we also employed a semi-scoping

approach in which the review is not only one that accumulates a body of evidence, but one that also can "examine and clarify definitions that are used in the literature" (Munn et al., 2018, p. 3). We also followed various systematic procedures based on several other reviews (see Baek et al., 2020; Hamari et al., 2014; Levy & Ellis, 2006; Nakano & Muniz, 2018; Ramdhani et al., 2014). A summary of the overall process is illustrated in Figure 1. Rather than being linear, it is a recursive approach to examining and synthesizing various literature sources.

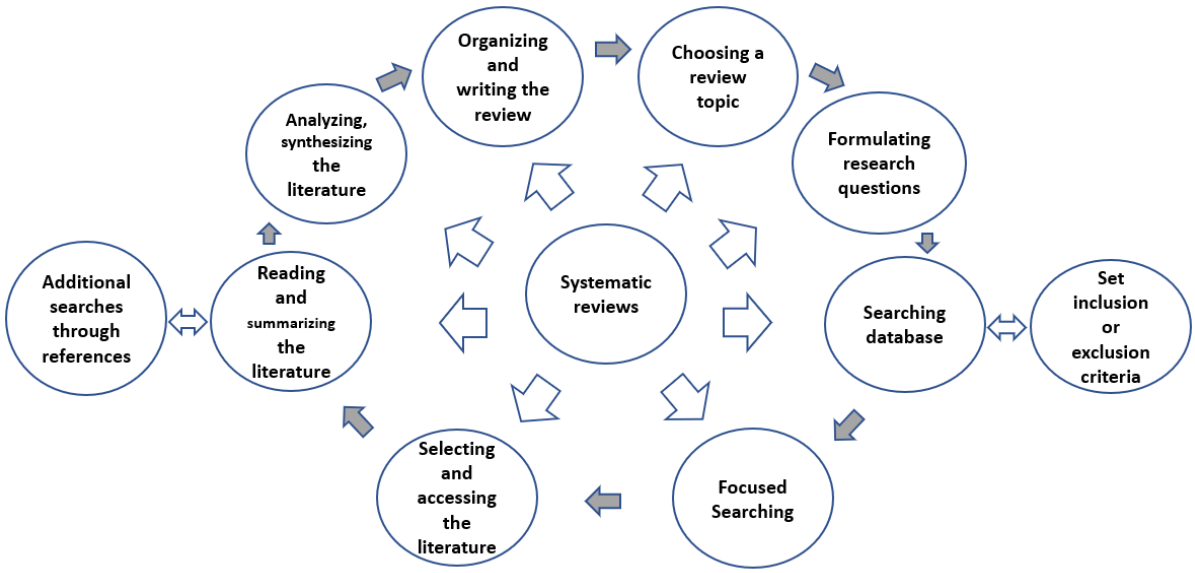

Figure 1. Review Procedure

2.1 Step 1: Search Terms and Databases

Since our investigative target was the use of *Scratch* in elementary educational settings in regards to computational thinking, we used the following keywords, *Scratch*, *computational thinking*, and *education*. We searched Ebscohost, ScienceDirect, Web of Science, Springer, IEEE Digital Library, ACM Digital Library, Google Scholar, and ProQuest for relevant literature displaying these keywords in their titles or abstracts. This initial search yielded 551 articles.

2.2 Step 2: Inclusion and Exclusion Criteria

To focus the scope of this review, we required that articles be written in English, be published from 2010-2020, and be conducted in elementary education. Studies that involved pre- or in-service elementary teacher training were excluded to refine the results to elementary school student performance. This resulted in 125 papers.

2.3 Step 3: Assessing the Literature

Papers were screened again by sorting them into two categories: conceptual articles and empirical studies. Conceptual papers discussed the general features of Scratch, provided a theoretical framework or suggested instructional practices for Scratch programming into education. Empirical studies tended to test and justify specific interventions and measure(s) of computational thinking via qualitative, quantitative research, or mixed-methods research designs. This reduced the number of articles to 79.

We further refined the dataset to papers that specifically looked at CT skills (e.g., characteristics and processes, models, assessments, interventions), in addition to including experimental and non-experimental study designs. The research was further scrutinized for the quality of the research designs such as excessive statements or assumptions, tangential CT focus, outcomes other than measures of CT skills, pilot studies, or studies with samples of 10 or less that could not produce valid statistical outcomes. In regards to excluding studies with small sample sizes, scholarship does suggest that the outcomes reported in small studies are more variable than large-study counterparts, ultimately making the results of larger studies more reliable and where the greatest emphasis should be placed (Slavin & Smith, 2009). Therefore, the exclusion of these studies in this review is not to state (or even imply) that such studies are inferior (they are not), rather simply that for the purpose of summarizing evidence for a literature review where broader generalizability of the findings was the goal (Templier & Paré, 2015), the larger studies were preferred for drawing conclusions from (see Slavin & Smith, 2009). Additional assessment of the literature included examining the papers' topical, historical, and methodological relevance, as well as gap analyses. This ultimately produced 58 papers, an overview of which is presented in Figure 2 and 3.

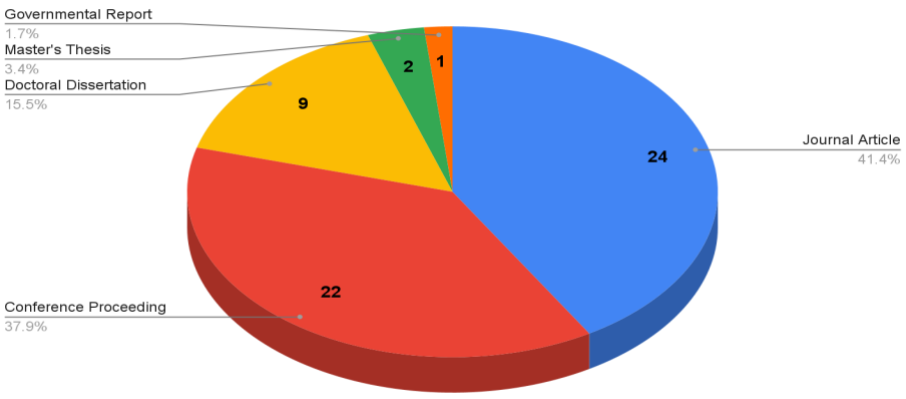

Figure 2. Overview of Review Articles by Type

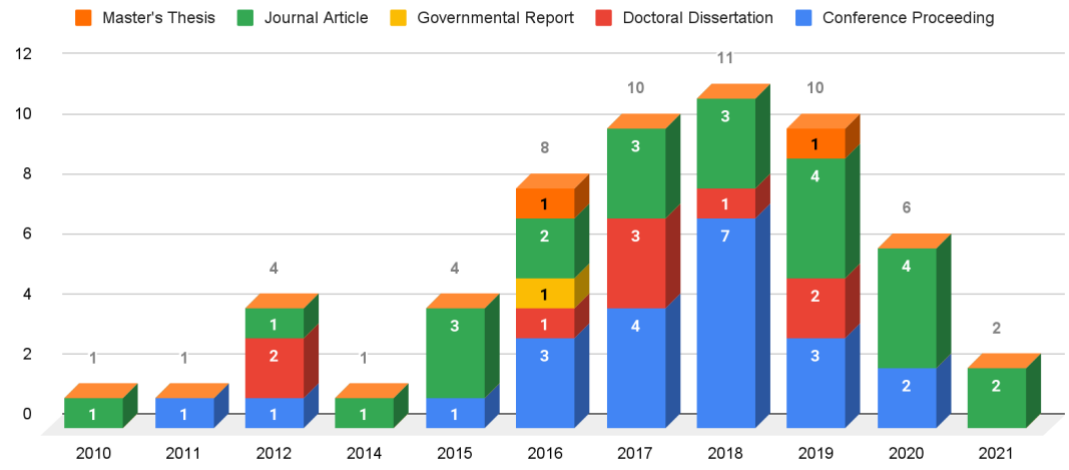

Figure 3. Overview of Review-articles by Year

2.4 Analyzing and Organization

We performed an inductive qualitative content analysis whereby the content discussed in papers (i.e., topics, findings, issues) were assigned keywords or phrases (i.e., codes). These were then aggregated into larger categories where vertical and horizontal relationships appeared (Braun & Clarke, 2006). The researchers discussed areas where thoughts diverged and ultimately, this process produced five core themes which are used to structure the findings section of the review: exploiting tangible blocks in a physical coding environment, integrating *Scratch* into various disciplines through programming, *Scratch* gaming for computational thinking, evaluating computational thinking skills through *Scratch* projects, and teaching and learning methods/factors affecting CT in children.

3. Results

3.1 Exploiting Blocks in A Coding Environment

Various studies compared Scratch with other programming environments, particularly with their relationship to developing computational thinking skills. Smith (2019) compared Scratch with Cozmo, a robotics-based coding environment While both of these coding and robotics based programming environments shared the same content and instructional features, curriculum with Scratch was often more computer-based whereas the Cozmo curriculum was made of animated emotional-educational robotic activities. Smith (2019) similarly found that students were more engaged when using Cozmo over Scratch, though both programming environments were equivalent in developing CT skills. When Scratch was compared to Lego Mindstorms or C++ environments, Scratch was more effective at developing logical thinking skills (Korkmaz, 2016). Other studies (i.e., da Cruz Alves et al., 2019; Park & Shin, 2019; Quitério Figueiredo, 2017) compared Scratch and App Inventor and found that Scratch projects scored higher when being evaluated on parallelism, synchronization, and flow control whereas

App Inventor projects displayed higher scores on user interactivity and data representation. This implies that Scratch, as a tool for coding and enhancing computational thinking skills, has been established, however its efficacy in facilitating other CT skills/knowledge domains is less certain. da Cruz Alves et al. (2019) not only supported this point but cautioned that the evaluation of CT skills is heterogeneous; there is no consensus on exactly what criteria should be used or how to evaluate them. Thus, the comparisons between Scratch, App Inventor, Cozmo, and other coding environments may not be so useful; multiple tools can develop and improve computational thinking (Quitério Figueiredo, 2017; Turchi & Malizia, 2016). Nevertheless, research has shown that visual block-based coding environments do reduce the difficulty of abstract programming concepts and complex syntax by converting them into tangible (metaphorically and physically) and accessible elements for students to manipulate and interact with (Rose et al., 2017). This makes it a valuable tool when integrated into other subject areas and learning environments.

3.2 Integrating Scratch into Various Disciplines Through Programming

Scratch has been shown to be an effective way of developing computational thinking skills when integrated in other disciplines (Moreno-León & Robles, 2016; Olabe et al., 2011; Ruthmann et al., 2010; Scullard et al., 2019). Ruthmann et al. (2010) discussed the potential for developing CT through live musical coding in Scratch by approaching programming as music notation. Olable et al. (2011) noted how Scratch was useful when applied to robotics as the interface between digital code and the real world manifestation of it via a robot, which can provide immediate feedback to learners. Even when Scratch is used by students who are not pursuing conventional computer science or STEM related fields, the use of Scratch can influence the development of computational thinking skills such as abstraction or logical thinking (Harimurti et al., 2018). In short, there are numerous (even unexpected) benefits to integrating Scratch into other disciplines across K-12 such as math, writing, science, or English (Moreno-León & Robles, 2016).

3.2.1 Coding with Scratch for CT enhancement

There is no doubt that coding with Scratch is effective at developing CT. The implementation of Scratch for this has even been refined to include specific sequences of programming projects that progressively challenge learners to think computationally. This kind of curricular structure then requires learners to compose problems, recognize patterns, collect and represent data, and ultimately develop code to solve a particular task or challenge (Swaid & Suid, 2019). Progress design scenarios such as this help students to learn not only core programming concepts such as sequences, loops, and events, but also computational concepts such as abstracting, modularizing, and debugging (Zhang & Nouri, 2019). Chou (2020) even found that third-grade students improved their CT competence when engaged in weekly Scratch activities. Moreover, parents' active involvement in take-home assignments influenced students' long-term CT competence retention. Fagerlund et al. (2020)'s evaluation of Scratch projects

found that students' work indicated CT skills and knowledge in diverse ways, and that nearly all students' projects showed knowledge of patterns, abstraction, collaboration, and logical operators, though less than half of the projects used algorithmic procedure, automation, synchronized parallel scripts, recursive solutions, and boolean logic. Thus, simpler programming and computational thinking tend to be far more prevalent in Scratch projects than more advanced ones; developmental levels need to be considered carefully when implementing Scratch for the purpose of CT development.

Similarly, since computational thinking is a skill related to coding, teaching a subject with coding can also show increased achievement in a subject matter (Calao et al., 2015). Calao et al. (2015) utilized Scratch to see whether it could enhance mathematical understanding among sixth grade students. The results showed that students who received Scratch training gained an increase in the understanding of mathematical processes in modeling process and reality phenomena, reasoning, problem formulation and problem solving, and comparison and execution of procedures and algorithms. Thus, coding in mathematics class is assumed to help develop computational thinking skills. This assumption was later supported by Rodríguez-Martínez et al. (2020) where coding and math performance significantly improved. However, some activities in math could be infused with Scratch coding. For example, Vinayakumar et al. (2018) developed computational exercises facilitating the learning of both fractal geometry and computational thinking through tree drawings, which stimulated learners to think computationally about iteration and size change, leading to the concept of 'parallelism, conditionals, and operators. Nevertheless, while mathematics and programming are perhaps logical and obvious areas to use Scratch in to develop CT, there are other creative ways that Scratch is being used such as storytelling.

3.2.2 Storytelling with Scratch

Storytelling involves both reading and writing, and Scratch has been documented in literature as a novel way to promote computational thinking skills, especially with younger/early grade students (Burke, 2012; Lowe & Brophy, 2019; Smith & Burrow, 2016; Von Gillern, 2017).

Smith and Burrow (2017) analyzed five and seven-year old children's use of CT skills and observed looping actions, debugging, remixing, and expression as the students generated ideas and content for their story, which are all examples of concrete CT skills. Similarly, Lowe and Brophy (2019) concluded that computational thinking seems to be most valuable in young learners when it is grounded in concrete activities such as storytelling. They also argued that students can benefit from spending time in abstract story planning since this bears connection to decomposition and algorithm design. Burke (2012) also noted the potential benefits of storytelling in Scratch which could contribute to enhancing computational thinking skills, arguing that the creative functionality of algorithms accentuates the connection between coding and writing. For this reason, digital stories in Scratch embody the technical and the creative elements of composition (Von Gillern, 2017).

3.3 Scratch Gaming for CT

3.3.1 Playing Scratch games for CT

Rose et al. (2017) designed a computer application, Pirate Plunder, which is a block-based programming game that teaches its players how to use Scratch's coding blocks. It does so by focusing on helping children learn decomposition and abstraction skills. This is primarily done under the assumption that practicing loops and procedures in Scratch can promote students' abstraction and decomposition skills, thereby improving computational thinking as a result.

Rose et al. (2019) found that Pirate Plunder was effective in using custom blocks, procedures, and clones in Scratch with children aged 10-11. On Scratch abstraction/decomposition and computational thinking tests, students who played Pirate Plunder showed significantly higher scores than other groups' students. In related studies, Rose et al. (2018) ultimately concluded that Scratch game-based learning can increase children's procedural abstraction in Scratch projects as well as their computational thinking skills, in addition to the development of procedural abstraction skills as a result of controllable success conditions and difficulty levels (Rose et al., 2020).

3.3.2 Creating Scratch Games for CT

Computational thinking development for elementary aged children though the creation of games in Scratch is frequently discussed in the literature (e.g., Fadjo, 2012; Serbec et al., 2018; Ternik et al., 2017; Topallia & Cagiltayb, 2018). Nančovska Šerbec et al. (2018) performed a study that compared how primary school students aged eight to 12 thought versus prospective teachers of computer science. They compared two groups' projects and found the differences in the category of logic, synchronization, and parallelism which were explained by the differences in reasoning, complexity, and understanding of simultaneous events. No differences were found in the conceptual categories of flow control, data representation, abstraction, and user interactivity. These results implied, however, that computational thinking skills of elementary students can be promoted with game programming making activities through guided instruction.

Ternik et al. (2017) analyzed a maze game developed by 17 primary students aged between eight and 10 years-old. Their primary goal was to improve students' basic computational thinking skills by making a maze-game in Scratch. After teaching concepts such as sequences, loops, events, and conditionals, the participants developed their own maze-game. According to the neo-Piagetian theory of cognitive development, four students (out of 17) reached a concrete operational stage in their programming abilities. The participants displayed different rates of progress from the sensorimotor to preoperational to concrete operational stages of reasoning within Scratch. Even if they could determine different levels of understanding and abstract thinking, most students reached the developmental level. When using Scratch programming, educators should consider students' level of cognitive development. More

specifically in the context of computational thinking skills, instructional methods are also important. For example, step by step practical exercises with the concepts of parallelism and synchronization in tandem with other advanced concepts should be done (Fadjo, 2012; Serbec et al., 2018; Ternik et al., 2017; Topallia & Cagiltayb, 2018).

Fadjo (2012) found that sixth and seventh grade students who received pre-written Scratch code analogues (i.e., visual novel form) created in Scratch when compared to students who were instructed with only code in a virtual environment developed more computational thinking skills and concept knowledge (such as conditional logics and operator patterns). Rose (2019) also tested the assumption that the earlier children begin to develop expertise in computer science, the faster they will be able to develop a holistic understanding of code, including more abstract programming principles like selection, repetition, debugging, variables and procedures. She found that primary school children can understand abstract computer science concepts if the instructional method utilized a structured level progression, ultimately highlighting the importance of synergy between instructional method, learner characteristics, and certain Scratch-based tools. Such tools have been developed over the last 10 years to support educators' assessment of student programming and development of computational thinking skills. While Scratch, the platform, is often the most visible component of programming education, tailored programming tools have been developed to further unlock its learning and educational potential; one example of such a tool is Dr. Scratch.

3.4 Evaluating Computational Thinking Skills through Scratch projects

3.4.1 Using Dr. Scratch

Dr. Scratch is a tool that automates analysis of Scratch programs, detecting the presence/absence of certain target characteristics (e.g., conditional statements) of students' work. In addition to identifying these traits, Dr. Scratch then extrapolates and assigns a CT score to projects, thus providing feedback to both educators and learners about the CT skills present in their work (Moreno-León et al., 2015, Moreno-León et al., 2017). What makes Dr. Scratch a powerful tool, however, is the general consensus on its effectiveness (Browning, 2017; Lawanto, 2016; Oluk & Korkmaz, 2016).

Lawanto (2016), for example, concluded that Dr. Scratch was well suited to assess computational skills, which in turn helped teachers understand students' strengths and weaknesses in programming. Browning (2017) conducted a pre/posttest study in which two groups of 5th-6th graders had a treatment group assessed by Dr. Scratch for the presence of CT skills, and where the control group was assessed by other CT tests. They found that the development in students' programming skills in Scratch would relate to similar increases in their computational thinking skills or improvements in their computational thinking levels. Nevertheless, Dr. Scratch, as a tool, is not without its own limitations which both Lawanto (2016) and Browning (2017) noted, chiefly in the area of formative assessment. That is, Dr. Scratch is a summative assessment by nature and not one that provides feedback during the learning process. While

this may be an accurate description of the intent behind how Dr. Scratch was designed, other scholars such as Moreno-León et al. (2015) have, in fact, used Dr. Scratch for formative assessment purposes.

For example, Moreno-León et al. (2015) used Dr. Scratch's analysis output on students' projects as a stimulus to encourage students to keep improving their programming skills. They asked students aged 10 to 14 years-old to read Dr. Scratch output (displayed as feedback) and then try to improve their projects using the guidelines and tips offered by the tool. As a result, the students' computational thinking scores increased and students displayed improved coding skills. This was especially noticeable for students with an initial medium (i.e., developing) CT score rather than for students with a high one. Browning (2017), however, took a somewhat different approach to utilizing Dr. Scratch for formative assessment purposes. The difference was in taking certain components of programming into account (i.e., easier versus more difficult tasks or skills). Browning reported that there was a significant increase in abstraction, which they also suggested could be significant in flow control with a larger sample. In terms of other CT skills, there were no significant differences in logic scores, and little variation in data representation scores. Differences in the purpose of assessment aside, other limitations have been noted in the literature.

Brennan and Resnick (2012), for example, discussed the fact that the use/presence of a particular Scratch block (or group of blocks) was not necessarily a strong indicator of any particular mastery or fluency in a CT concept. In other words, students may not really understand why they need to use one block over another, or what a more efficient and/or effective sequence of blocks might be when compared to their own code. Further, the use of a single project (i.e., a single data point) to extrapolate CT scores may be skewed; multiple projects from a single student would need to be evaluated for validity in the assessment results. Moreover, certain key CT skills cannot be measured or assessed by examining the source code of the project alone -for example- debugging code in a project would not necessarily be evident in the final version of the code. Similarly, the creativity involved in remixing an existing project would not necessarily be obvious without comparing/contrasting it with the original.

3.4.2 Other Assessment Tools

In addition to the use of Dr. Scratch, other studies have documented the use of other tools to assess computational thinking skills in other ways. For example, Chou (2020) used a test developed by Strawhacker et al. (2018) to measure CT skills, which focused on debugging and fixing a program, circling the blocks, matching the program, and reverse engineering (reverse engineering is a battery of video-based programming tests). This assessment requires students to view programming questions via video clips and then asks students to provide solutions/answers on a structured answer sheet. Another example is from Saez-Lopez et al. (2016) who developed and used a visual block creative computing test to assess elementary students' CT competence after receiving instruction in Scratch. Zhang and Nouri (2019) examined the computational thinking skills that can be learned by K-9 students through

Scratch, based on Brennan and Resnick's (2012) framework. Brennan and Resnick's (2012) framework consists of the three key dimensions of CT: computational concepts, computational practices, and computational perspective, which is one that ultimately Brennan and Resnick's (2012) framework places students as designers of interactive stories, games, and simulations in a holistic assessment approach.

Fagerlund et al. (2020) created a framework using three formative assessment processes to identify areas of CT in Scratch projects: what to teach and learn (i.e., clarifying learning objectives), estimating students' current level of understanding, and analyzing their conceptual encounters with CT. This framework made it possible to perform formative assessments by integrating coding patterns, code constructs, and the extent to which students had conceptual encounters with CT through Scratch projects in elementary classrooms. Fagerlund et al. (2020) is also an example of in-depth insight of students' experiences with diverse areas in CT. It is also one that sets future directions of CT assessment in facilitating students' learning CT through students' Scratch projects when compared with earlier approaches.

3.5 Teaching & Learning Methods /Factors affecting CT of Children

The type of instruction and the context of the instructional materials play a significant role in students' development of CT skills and concept knowledge. In a relatively large study across six schools in the United States (222 K-2 students), Strawhacker et al. (2018) found that educators who demonstrated flexibility in lesson planning and who were responsive to students' needs had a positive effect on students. Further, educators that made a positive impact were also skilled in their use of technology, and were concerned about developing students' independent thinking skills. This highlights the importance of sound pedagogy and teaching in addition to the use of proper tools when developing CT skills and the use of technology. Quality teaching practices aside, more specific pedagogical approaches have been noted in the literature in terms of efficacy.

Fadjo (2012) used a grounded embodied pedagogy called "instructional embodiment" when teaching abstract concepts through the use of direct and imagined embodiments (embodiment refers to physical motion or activity). Fadjo (2012) found statistically significant effects for students who physically embodied (or acted out) predefined instructional materials such as speech and motion blocks. In addition to physical embodiment, imagined embodiment, is another technique found to be useful for teaching and developing CT. With imagined embodiment, students mentally simulate and construct imaginary worlds. The benefits of this approach were students' implementation of more computational structures in their projects. Similarly, using familiar contexts had a significant effect on identifying and implementing the CT skill pattern recognition, although learning CT concepts from an unfamiliar context had a significant positive effect on the implementation of both broadcast/receive couplings and conditional logic and operator patterns. Pérez et al. (2020) used metaphors to teach Scratch programming to children aged nine to 12 and concluded that using metaphors improved knowledge of programming

concepts. For example, when teaching loops, they used the metaphor of a hand mixer, and for conditionals, they used an intelligent fridge. One limitation that was noted, however, was that using metaphors could not be definitively correlated with enhanced students' CT since the CT test was not applicable to students who are younger than 10 (Pérez et al., 2020).

Student collaboration in Scratch was also found to improve CT skills. Hamelburg (2019) found that when sixth graders designed games with help from peers during collaborative coding, their knowledge of CT improved. While some students experience difficulty in collaborating, students often assisted their peers by making them feel more comfortable during the challenge. The results of this study corroborate the findings of Chowdhury (2017) where collaborative coding was similarly found to improve computational thinking skills. Other studies, however, did not find any positive effect from collaboration on Scratch programming or on CT skills from (Donley, 2012). Marcelino et al. (2018) also obtained similar results as Donley (2012) with adult (teacher trainees) learners of Scratch.

4. Discussion

4.1 A Variety of CT Definitions

There is a general consensus that programming skills are closely related to CT skills, but more importantly the distinction that programming skills are a subset of CT. The literature consistently describes that CT includes all concepts that computer scientists use to solve computational skills. While this conceptual hierarchy is clear, numerous issues arise when considering which concepts are more appropriate to learn at the elementary school level. For example, pedagogical approaches, content/skill scope and sequence, etc. For this reason, some studies emphasize CT concepts differently in the context of elementary school children. Thus, in regards to our first research question regarding how CT is defined and/or conceptualized, we had difficulty in interpreting previous studies and making generalizations from their results given the extant variety. Thus, the interpretation of results from CT studies needs careful and well-reasoned considerations as the variables being manipulated and/or outcomes being measured/assessed are not necessarily the same. To this point, Rose et al. (2017) proposed that future research in this area should focus on the individual concepts involved in computational thinking to get a deeper understanding of CT for elementary students.

4.2 Assessment & Evaluation

To assess CT skills in Scratch programming, research to date has predominantly relied on code analysis of students' projects. While this approach can provide CT competency feedback in the form of a score (Alves et al., 2019), automated calculation in the CT assessment lacks context that observations or interviews can provide. In other words, more comprehensive evaluation and assessment strategies are needed. Further, automated and performance-based approaches also lack explicit suggestions or tips on how to improve code such as in its efficiency or complexity. In Scratch assessment, the use of formative

assessment, such as students' explaining parts of their projects and finding mistakes in their code, would be beneficial for developing their CT skills (Ternik et al., 2017). At present, assessment is varied; we suspect that the diversity in assessment and evaluation is strongly correlated with the variety of operational definitions of CT. While this pedagogical/curricular relationship is not novel or unique to CT, it does highlight an ongoing challenge and obstacle for practitioners and researchers. Some variables that are connected to how CT is taught and consequently assessed A number of variables were noted in the literature that affected Scratch performance.

4.3 Variables Affecting CT

Current key CT variables ranged from pedagogical approaches (see Strawhacker et al., 2018), learner gender (e.g., Chou, 2018), previous programming experience, and math skills. However, findings from Longi (2016) regarding college students' competence in learning programming may provide insight regarding the potential factors influencing CT competence for elementary school aged learners. In Longi's (2016) systematic literature review, two major factors surfaced: namely students' background information and psychological characteristics in terms of performance in programming courses. These two variables may be related to age or a proxy for age, thus, there is a gap in the literature in terms of sampling that warrants additional research with elementary aged learners. In terms of gender, some studies (i.e., Oluk & Korkmaz, 2016) have assessed both Scratch programming (via Dr. Scratch) and CS skills (via the Computational Thinking Levels Scale which includes 5 factors: a) creativity, b) problem solving, c) algorithmic thinking, d) collaboration, and e) critical thinking). Dr Scratch has a gender parameter in its evaluation process however results showed no difference in gender or time online, although there was a significant relationship between programming skill and computational thinking. Further research regarding the effects of gender on CT and/or programming skills, especially across various related knowledge domains, is warranted.

5. Conclusion

CT is developing practice, field, and area of study that has emerged over the last 20 years. In addition to the inevitable growing pains of a nascent topic, discussion and debate has also emerged regarding the principles that ultimately constitute CT and how to develop CT skills. Similarly, how to assess or measure CT has evolved over time from more rudimentary and limited constructs to more holistic approaches. Equally important, however, is the ongoing inquiry into how CT skill teaching, learning, and development manifests for learners of different ages and in different contexts. The findings from this review highlight how exploiting tangible blocks in a physical coding environment can be particularly beneficial for young/novice learners, and that integrating Scratch into various disciplines through programming (not just programming alone) demonstrates increased learning gains and CT skill development. Additionally, more pedagogically sound ways of teaching CT skills and Scratch have emerged with demonstrated learning effectiveness, and in tandem how CT and programming skills are

assessed is starting to evolve in more holistic and sophisticated ways than in years prior. Similarly, more discrete variables affecting CT skill development in children have been identified and are starting to be researched. However, given the emergent or nascent character of CT as a field of practice and inquiry, ongoing research is warranted in several areas.

Current literature has only begun looking into how students interact with the Scratch interface. While the visual block-building metaphor has been effective since Scratch's inception for young learners when compared to learning syntax, some research has shown that younger learners interact differently with programming interfaces such as Scratch, Scratch Jr., and Lightbot. While research findings have generally been in line with the pedagogical underpinnings of Scratch and Scratch Jr., future research can investigate the different approaches that students, particularly young ones, take to programming through visual interfaces.

In this review we found only limited literature that looked at how elementary school teachers are learning about Scratch and CT, and particularly the most effective ways of teaching it as subject matter as well as a critical thinking skill. There is burgeoning discussion about teaching methods that moved beyond just the pedagogical foundations of Scratch as a platform (i.e., constructivism) and into best practices with the platform (e.g., teaching metaphors, generative strategies [instructional embodiment], robust assessment methods, etc.). As current literature is only beginning to investigate and describe more deliberate uses of formative assessment with Scratch and CT, in addition to holistic approaches, research is warranted in this area in general, and specifically with elementary aged learners.

References

Alves, N. D. C., Von Wangenheim, C. G., & Hauck, J. C. (2019). Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. *Informatics in Education*, *18*, 17–39. http://doi.org/10.15388/infedu.2019.02

Angeli, C., Voogt, J., Fluck, A. E., Webb, M., Cox, M. J., Malyn-Smith, J., & Zagami, J. (2016). A K-6 computational thinking curriculum framework - Implications for teacher knowledge. *Educational Technology & Society*, *19*, 47–57. https://www.jstor.org/stable/pdf/jeductechsoci.19.3.47.pdf

Baek, Y., Min, E., & Yun, S. (2020) Mining educational implications of Minecraft. *Computers in the Schools*, *37*, 1–16. http://doi.org/10.1080/07380569.2020.1719802

Barr, D., Harrison, J., & Conery, L. (2011). Computational thinking: A digital age skill for everyone. *Learning & Leading with Technology*, *38*, 20–23. https://files.eric.ed.gov/fulltext/EJ918910.pdf

Barr, V., & Stephenson, C. (2011). Bringing computational thinking to K-12. *ACM Inroads*, *2*, 48–13. http://doi.org/10.1145/1929887.1929905

Berikan, B., & Özdemir, S. (2019). Investigating "problem-solving with datasets" as an implementation of computational thinking: A literature review. *Journal of Educational Computing Research, 58,* 502–534. https://doi.org/10.1177/0735633119845694

Bers, M. U. (2010). The TangibleK Robotics program: Applied computational thinking for young children. *Early Childhood Research & Practice, 12,* 1–20. https://files.eric.ed.gov/fulltext/EJ910910.pdf

Bocconi, S., Chioccariello, A., Dettori, G., Ferrari, A., & Engelhardt, K. (2016). *Developing computational thinking in compulsory education: Implication for policy and practice*. Joint Research Center (JRC) Science for Policy Report. https://ec.europa.eu/jrc

Braun, V., & Clarke, V. (2006). Using thematic analysis in psychology. *Qualitative Research in Psychology, 3,* 77–101. https://doi.org/10.1191/1478088706qp063oa

Brennan, K., & Resnick, M. (2012, April). New frameworks for studying and assessing the development of computational thinking. In *Proceedings of the 2012 Annual Meeting of the American Educational Research Association* (pp. 1–25). AERA. http://scratched.gse.harvard.edu/ct/files/AERA2012.pdf

Browning, S. F. (2017). *Using Dr. Scratch as a formative feedback tool to assess computational thinking*. [Master's thesis, Brigham Young University]. https://scholarsarchive.byu.edu/etd/6659

Burke, W. Q. (2012). *Coding & composition: Youth storytelling with Scratch programming* (Publication No. 3510989) [Doctoral dissertation, University of Pennsylvania]. ProQuest Dissertations Publishing.

Calao, L. A., Moreno-León, J., Correa, H. E., & Robles, G. (2015). Developing mathematical thinking with Scratch. In G. Conole, T. Klobučar, C. Rensing, J. Konert, & É. Lavoué (Eds.) *Design for teaching and learning in a networked world* (pp. 17–27). Springer International Publishing. http://doi.org/10.1007/978-3-319-24258-3 2

Chou, P.-N. (2018). Smart technology for sustainable curriculum: Using drone to support young students' learning. *Sustainability, 10,* 3819. https://doi.org/10.3390/su10103819

Chou, P.-N. (2020). Using ScratchJr to foster young children's computational thinking competence: A case study in a third-grade computer class. *Journal of Educational Computing Research, 58*, 570–595. https://doi.org/10.1177/0735633119872908

Chowdhury, B. T. (2017). *Collaboratively learning computational thinking*. Unpublished doctoral dissertation, of Virginia Polytechnic Institute and State University, USA.

da Cruz Alves, N., Gresse Von Wangenheim, C., & Hauck, J. C. (2019). Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study. *Informatics in Education*, *18*, 17–39. https://doi.org/10.15388/infedu.2019.02

diSessa, A. A. (2018). Computational literacy and "The Big Picture" concerning computers in mathematics education. *Mathematical thinking and learning*, *20*, 3–31. https://doi.org/10.1080/10986065.2018.1403544

Dolgopolovas, V., Jevsikova, T., Savulionienė, L., & Dagienė, V. (2015). On evaluation of computational thinking of software engineering novice students. In *Proceedings of the IFIP TC3 Working Conference "A New Culture of Learning: Computing and next Generations* (pp. 90–99). https://core.ac.uk/download/pdf/42583209.pdf#page=98

Donley, K. S. (2012). *Coding in the curriculum: learning computational practices and concepts, creative problem solving skills, and academic content in ten to fourteen-year-old children* (Publication No. 10842428) [Doctoral dissertation, Temple University]. ProQuest Dissertations Publishing.

English, L. D. (2016). STEM education K-12: Perspectives on integration. *International Journal of STEM Education*, *3*, 1–8. http://doi.org/10.1186/s40594-016-0036-1

Fadjo, C. L. (2012). *Developing computational thinking through grounded embodied cognition* (Publication No. 3506300) [Doctoral dissertation, Columbia University]. ProQuest Dissertations Publishing.

Fagerlund, J., Häkkinen, P., Vesisenaho, M., & Viiri, J. (2020). Assessing 4th grade students' computational thinking through scratch programming projects. *Informatics in Education*, *19*, 611–640. https://doi.org/10.15388/infedu.2020.27

Grover, S., & Pea, R. (2013). Computational thinking in K–12: A review of the state of the field. *Educational Researcher*, *42*, 38–43. https://doi.org/10.3102/0013189X12463051

Hamari, J., Koivisto, J., & Sarsa, H. (2014, January). Does gamification work? – A litera-

ture review of empirical studies on gamification. *Proceedings of the 47th Hawaii International Conference on System Sciences* (pp. 3025–3034). Waikoloa, Hawaii, USA. http://doi.org/10.1109/HIS.2014.377

Hamelburg, N. (2019). *Coding, collaboration, and computational thinking* (Publication No. 10183306) [Master's thesis, Hofstra University]. ProQuest Dissertations Publishing.

Harimurti, R., Qoiriah, A., Ekohariadi, E., & Munoto, M. (2018, July). Implementation of computational thinking concepts in ICT learning using Scratch programming. In *International Conference on*

Indonesian Technical Vocational Education and Association (APTEKINDO 2018) (pp. 105–109). Atlantis Press. https://doi.org/10.2991/aptekindo-18.2018.23

Henderson, P. B., Cortina, T. J., & Wing, J. M. (2007). Computational thinking. In *Proceedings of the 38th SIGCSE Technical Symposium* (pp. 195–3). ACM Press. http://doi.org/10.1145/1227310.1227378

Jacob, S. R., Warschauer, M. (2018). Computational thinking and literacy. *Journal of Computer Science Integration, 1,* 1–21. http://doi.org/10.26716/jcsi.2018.01.1.1

Kafai, Y. B. (2016). From computational thinking to computational participation in K-12 education. *Communications of the ACM*, *59*, 26–27. http://doi.org/10.1145/2955114

Kale, U., Akcaoglu, M., Cullen, T., Goh, D., Devine, L., Calvert, N., & Grise, K. (2018). Computational what? Relating computational thinking to teaching. *TechTrends*, *62*, 574–584. https://doi.org/10.1007/s11528-018-0290-9

Kalelioglu, F., Gülbahar, Y., & Kukul, V. (2016). A framework for computational thinking Based on a systematic research review. *Baltic Journal of Modern Computing*, *4*, 583–596. http://acikerisim.baskent.edu.tr/handle/11727/3831

Korkmaz, Ö. (2016). The effect of Scratch- and Lego Mindstorms Ev3-based programming activities in academic achievement, problem-solving skills and logical-mathematical thinking skills of students. *Malaysian Online Journal of Educational Sciences*, *4*, 73–88. https://ajap.um.edu.my/index.php/MOJES/article/download/12658/8149

Lammi, M., Denson, C., & Asunda, P. (2018). Search and review of the literature on engineering design challenges in secondary school settings. *Journal of Pre-College Engineering Education Research*, *8*, 1–19. http://doi.org/10.7771/2157-9288.1172

Lawanto, K. N. (2016). *Exploring trends in middle school students' computational thinking in the online scratch community: A pilot study* (Publication No. 10183306) [Master's thesis, Utah State University]. ProQuest Dissertations Publishing.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., et al. (2011). Computational thinking for youth in practice. *ACM Inroads*, *2*, 32–7. http://doi.org/10.1145/1929887.1929902

Levy, Y., & Ellis, T. (2006). A systems approach to conduct an effective literature review in support of information systems research. *Informing Science: The International Journal of an Emerging Transdiscipline, 9*, 181–212. http://doi.org/10.28945/479

Longi, K. (2016). *Exploring factors that affect performance on introductory programming courses* (Unpublished master's thesis). Department of Computer Science, University of Helsinki, Finland.

Lowe, T. A., & Brophy, S. P. (2019, June). Identifying computational thinking in storytelling literacy activities with Scratch Jr. In *2019 ASEE Annual Conference & Exposition*. https://peer.asee.org/32913

Lye, S. Y., & Koh, J. H. L. (2014). Review on teaching and learning of computational thinking through programming: What is next for K-12? *Computers in Human Behavior*, *41*, 51–61. http://doi.org/10.1016/j.chb.2014.09.012

Marcelino, M. J., Pessoa, T., Vieira, C., Salvador, T., & Mendes, A. J. (2018). Learning computational thinking and Scratch at distance. *Computers in Human Behavior*, *80*, 470–477. https://doi.org/10.1016/j.chb.2017.09.025

Miaoulis, I. (2010). K-12 engineering – The missing core discipline. In *Holistic engineering education* (pp. 37–51). Springer New York. http://doi.org/10.1007/978-1-4419-1393-7_4

Moreno-León, J., & Robles, G. (2016, April). Code to learn with Scratch? A systematic literature review. In *2016 IEEE Global Engineering Education Conference (EDUCON)* (pp. 150–156). IEEE. https://doi.org/10.1109/EDUCON.2016.7474546

Moreno-León, J., Robles, G., & Román-González. (2015). Dr. Scratch: Automatic analysis of Scratch projects to assess and foster computational thinking. *RED. Revista de Educación a Distancia, 15*, 1–23 https://www.um.es/ead/red/46/moreno_robles.pdf

Moreno-León, J., Robles, G., & Román-González, M. (2017). Towards data-driven learning paths to develop computational thinking with Scratch. *IEEE Transactions on Emerging Topics in Computing*, *8*, 193–205. https://doi.org/10.1109/TETC.2017.2734818

Munn, Z., Peters, M. D., Stern, C., Tufanaru, C., McArthur, A., & Aromataris, E. (2018). Systematic review or scoping review? Guidance for authors when choosing between a systematic or scoping review approach. *BMC Medical Research Methodology*, 18, 1–7. https://doi.org/10.1186/s12874-018-0611-x

Nakano, D., & Muniz, J. Jr., (2018). Writing the literature review for an empirical paper. *Production*, *28*, e20170086. http://doi.org/10.1590/0103-6513.20170086

Nančovska Šerbec, I., Cerar, Š., & Žerovnik, A. (2018). Developing computational thinking through games in Scratch. *XI Национална конференция Образованието и изследванията в информационното общество 2018* [*XI National Conference "Education and Research in the Information Society 2018*]. http://pefprints.pef.uni-lj.si/5141/1/Serbec_Developing.pdf

Olabe, M., Basogain, X., Maíz, I., & Castano, C. H. (2011). Programming and robotics with Scratch in primary education. In A. Méndez-Vilas (Ed.), *Education in a technological world: Communicating current and emerging research and technological efforts* (pp. 355-363). Formatex Research Centre.

Oluk, A., & Korkmaz, Ö. (2016). Comparing students' Scratch skills with their computational thinking skills in terms of different variables. *Online Submission*, *8*, 1–7. http://doi.org/10.5815/ijmecs.2016.11.01

Park, Y., & Shin, Y. (2019). Comparing the effectiveness of Scratch and App Inventor with regard to learning computational thinking concepts. *Electronics*, *8*, 1269. https://doi.org/10.3390/electronics8111269

Pérez, D., Hijón-Neira, R., Bacelo, A., & Pizarro, C. (2020). Can computational thinking be improved by using a methodology based on metaphors and scratch to teach computer programming to children? *Computers in Human Behavior*, *105*, 105849. https://doi.org/10.1016/j.chb.2018.12.027

Quitério Figueiredo, J. A. Q. (2017). How to improve computational thinking: A case study. *Education in the Knowledge Society*, *18*, 35–51. https://doi.org/10.14201/eks20171843551

Ramdhani, A., Ramdhani, M. A., & Amin, A. S. (2014). Writing a literature review research paper: A step-by-step approach. *International Journal of Basic and Applied Science*, *3*,47–56. http://digilib.uinsgd.ac.id/5129/1/08IJBAS%283%29%281%29.pdf

Rode, J. A., Booker, J., Marshall, A., Weibert, A., Aal, K., Rekowski, von, T., et al. (2015). From computational thinking to computational making. In *2015 ACM International Joint Conference* (pp. 401–402). ACM Press. http://doi.org/10.1145/2800835.2800926

Rodríguez-Martínez, J. A., González-Calero, J. A., & Sáez-López, J. M. (2020) Computational thinking and mathematics using Scratch: an experiment with sixth-grade students. *Interactive Learning Environments*, *28*, 316–327. https://doi.org/10.1080/10494820.2019.1612448

Romero, M., Lepage, A., & Lille, B. (2017). Computational thinking development through creative programming in higher education. *International Journal of Educational Technology in Higher Education*, *14*, 1–15. https://doi.org/10.1186/s41239-017-0080-z

Rose, S. P. (2019). *Developing children's computational thinking using programming games* (Publication No. 27771989) [Doctoral dissertation, Sheffield Hallam University]. ProQuest Dissertations Publishing.

Rose, S. P., Habgood, M. P. J., & Jay, T. (2017). An exploration of the role of visual programming tools in the development of young children's computational thinking. *The Electronic Journal of e-Learning*, *15*, 297–309. https://doi.org/10.34190/ejel.15.4.2368

Rose, S., Habgood, J., & Jay, T. (2018). Pirate Plunder: Game-based computational thinking using Scratch blocks. In *Proceedings of the 12th European Conference on Games Based Learning* (pp. 556–

564). Academic Conferences and Publishing International Limited. https://core.ac.uk/download/pdf/160276026.pdf

Rose, S. P., Habgood, M. J., & Jay, T. (2019, May). Using Pirate Plunder to develop children's abstraction skills in Scratch. In *Extended Abstracts of the 2019 CHI Conference on Human Factors in Computing Systems* (pp. 1–6). https://core.ac.uk/download/pdf/189171289.pdf

Rose, S., Habgood, J., & Jay, T. (2020). Designing a programming game to improve children's procedural abstraction skills in Scratch. *Journal of Educational Computing Researc*h, *58*, 1372–1411. https://journals.sagepub.com/doi/pdf/10.1177/0735633120932871

Ruthmann, A., Heines, J. M., Greher, G. R., Laidler, P., & Saulters, C. (2010, March). Teaching computational thinking through musical live coding in scratch. In *Proceedings of the 41st ACM Technical Symposium on Computer Science Education* (pp. 351–355). https://dl.acm.org/doi/abs/10.1145/1734263.1734384

Saez-Lopez, J., Roman-Gonzalez, M., & Vazquez-Cano, E. (2016). Visual programming languages integrated across the curriculum in elementary school: A two year case study using Scratch in five schools. *Computers & Education, 97*, 129–141. https://doi.org/10.1016/j.compedu.2016.03.003

Scullard, S., Tsibolane, P., & Garbutt, M. (2019). The role of Scratch visual programming in the development of computational thinking of non-is majors. In *Proceedings 2019 Pacific Asia Conference on Information Systems (PACIS)* (pp. 79*)*. https://aisel.aisnet.org/pacis2019/79

Selby, C., & Woollard, J. (2013). *Computational thinking: The developing definition*. Corwin Press. http://doi.org/10.4135/9781506313214

Sengupta, P., Kinnebrew, J. S., Basu, S., Biswas, G., & Clark, D. (2013). Integrating computational thinking with K-12 science education using agent-based computation: A theoretical framework. *Education and Information Technologies*, *18*, 351–380. http://doi.org/10.1007/s10639-012-9240-x

Serbec, I. N., Cerar, Š., & Zerovnik, A. (2018). Developing computational thinking through games in scratch. In *Proceedings at 11th National Conference with International Participation, Education and Research in the Information Society* (pp. 21–30). Plovdiv, Bulgaria

Slavin, R., & Smith, D. (2009). The relationship between sample sizes and effect sizes in systematic reviews in education. *Educational Evaluation and Policy Analysis*, *31*, 500–506. https://doi.org/10.3102/0162373709352369

Smith, S. M. (2019). *A comparison of computer-based and robotic programming instruction:Impact of scratch versus cozmo on middle school students' computational thinking, spatial skills, competency*

beliefs, and engagement (Publication No. 27602977) [Doctoral dissertation, Kent State University]. ProQuest Dissertations Publishing.

Smith, S., & Burrow, L. E. (2016). Programming multimedia stories in Scratch to integrate computational thinking and writing with elementary students. *Journal of Mathematics Education, 9,* 119–131. https://educationforatoz.com/images/2016_Commentary_6.pdf

Strawhacker, A., Lee, M., & Bers, M. U. (2018). Teaching tools, teacher's rules: Exploring the impact of teaching styles on young children's programming knowledge in Scratch Jr..*International Journal of Technology and Design Education, 28,* 347–376. https://doi.org/10.1007/s10798-017-9400-9

Swaid, S., & Suid, T. (2019, December). Computational thinking education: Who let the dog out?. In *2019 International Conference on Computational Science and Computational Intelligence (CSCI)* (pp. 788–792). IEEE. http://doi.org/10.1109/CSCI49370.2019.00150

Templier, M., & Paré, G. (2015). A framework for guiding and evaluating literature reviews. *Communications of the Association for Information Systems, 37,* 112–137. https://doi.org/10.17705/1CAIS.03706

Ternik, Ž., Koron, A., Koron, T., & Šerbec, I. N. (2017). Learning programming concepts through maze game in Scratch. In *Proceedings at 11th European Conference on Games Based Learning* (pp. 661–670). Academic Conferences International Limited.

Topallia, D., & Cagiltayb, N. E. (2018). Improving programming skills in engineering education through problem-based game projects with Scratch. *Computers and Education, 120,* 64–74. https://doi.org/10.1016/j.compedu.2018.01.011

Turchi, T., & Malizia, A. (2016). Fostering computational thinking skills with a tangible blocks programming environment. *2016 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* (pp. 232–233). IEEE. https://doi.org/10.1109/VLHCC.2016.7739692

Vinayakumar, R., Soman, K. P., & Menon, P. (2018, July). Fractal geometry: Enhancing computational thinking with MIT Scratch. In *2018 9th International Conference on Computing, Communication and Networking Technologies* (ICCCNT) (pp. 1-6). IEEE. https://doi.org/10.1109/ICCCNT.2018.8494172

Von Gillern, S. (2017). *Young children, computer coding, and story creation: An examination of first- and second-grade children's multimodal stories and literacy practices when engaged with a multimedia coding application* (Publication No. 10269304) [Doctoral dissertation, Iowa State University]. ProQuest Dissertations Publishing.

Voogt, J., Fisser, P., & Good, J. (2015). Computational thinking in compulsory education: Towards an agenda for research and practice. *Education and Information Technologies*, *20*, 715–728. https://doi.org/10.1007/s10639-015-9412-6

Weintrop, D., Beheshti, E., Horn, M., Orton, K., Jona, K., Trouille, L., & Wilensky, U. (2015). Defining computational thinking for mathematics and science classrooms. *Journal of Science Education and Technology, 25*, 127–147. https://doi.org/10.1007/s10956-015-9581-5

Wing, J. M. (2010, November 17). Computational thinking: What and why. *The Link*. http://www.cs.cmu.edu/link/research-notebook-computational-thinking-what-and-why

Wing, J. M. (2017). Computational thinking's influence on research and education for all.*Italian Journal of Educational Technology*, *25*, 7–14. https://doi.org/10.17471/2499-4324/922

Yadav, A., Hong, H., & Stephenson, C. (2016). Computational thinking for all: Pedagogical approaches to embedding 21st century problem solving in K-12 classrooms. *TechTrends*, *60*, 565–568. http://doi.org/10.1007/s11528-016-0087-7

Yadav, A., Zhou, N., Mayfield, C., Hambrusch, S. E., & Korb, J. T. (2011). Introducing computational thinking in education courses. In *Proceedings of the 42nd ACM Technical Symposium on Computer Science Education* (pp. 465–470). ACM Press. http://doi.org/10.1145/1953163.1953297

Yang, D., Swanson, S., Chittoori, B., & Baek, Y. (2018). Integrating computational thinking in stem education through project-based learning. In *Proceedings of the 5th STEM in Education Conference*. ASEE. https://par.nsf.gov/biblio/10106769

Zhang, L., & Nouri, J. (2019). A systematic review of learning computational thinking through Scratch in K-9. *Computers & Education, 141*, 103607. https://doi.org/10.1016/j.compedu.2019.103607

Semantic Analyses of Open-Ended Responses from Professional Development Workshop Promoting Computational Thinking in Rural Schools

Amber GILLENWATERS[1]

Razib IQBAL[1]

Diana PICCOLO[1]

Tammi DAVIS[1]

Keri FRANKLIN[1]

David CORNELISON[1]

Judith MARTINEZ[1]

Andrew HOMBURG[1]

Julia COTTRELL[1]

Melissa PAGE[2]

[1]Missouri State University, Springfield, MO, United States of America

[2]The Evaluation Group, Simpsonville, SC, United States of America

DOI: 10.21585/ijcses.v6i1.136

Abstract

In this paper, an application of open-ended textual feedback is presented as a tool to evaluate the perceptions and needs of teachers tasked with implementing computational thinking in the K-12 curriculum. Semantic analysis tools, including sentiment analysis and thematic analysis, facilitated the identification of common themes in open-ended textual feedback. Results show that semantic analysis techniques can be useful in evaluating formative assessment data or open-ended feedback to discover response patterns, which may aid in determining actionable insights related to adult learner perceptions,

interests, and self-efficacy. Formative assessment data were collected from a unique professional development workshop to promote computational thinking and curriculum integration in core subjects, including writing, math, science, and social studies, with the goal of discovering the barriers that rural teachers face in developing and implementing lesson plans for grades 3-8 teachers in a rural midwestern state in the USA to promote computational thinking and curriculum integration in core subjects, including writing, math, science, and social studies, with the goal of discovering the barriers that rural teachers face in developing and implementing lesson plans.

Keywords: K-12 education, rural classroom, formative assessment, teaching strategy, feedback, semantic analysis, thematic analysis, sentiment analysis

1. Introduction

According to the U.S. Bureau of Labor Statistics (2021), the demand for workers with applied knowledge in Computer Science (CS) is high and expected to continue growing at a rapid pace. However, the percentage of students participating in high school CS courses nationwide ranges from only 1-4% (Guzdial & Hill, 2019). Early access to computer education is lower among rural communities (Education Commission of the States, 2019). Teachers and students in rural areas face unique educational challenges, including high rates of poverty and unemployment (Marré, 2017). Racial and ethnic minorities from rural areas experience even higher rates of poverty and more structural barriers to pursuing CS (Wang et al., 2017) and are half as likely to obtain a college degree (Marré, 2017). Research suggests that exposure to CS education can increase the likelihood of choosing a STEM career path by five times (Lamb et al., 2019), particularly among members of underrepresented groups (Mahadeo et al., 2020). Moreover, recent studies indicate that CS education can help students beyond computing and improve problem-solving abilities (Brown & Brown, 2020; Salehi et al., 2020).

Therefore, K-12 educators, especially in rural areas, need to play a synergistic role in preparing students adequately for CS professions. According to a recent report published by Code Advocacy Coalition (2021), many states have established K-12 CS performance standards. However, K-12 CS programs across the United States vary widely in terms of content and programming tools and languages offered (Hubwieser et al., 2015). In addition, teaching resources and standardized knowledge assessments are still in development. Chetty et al. (2014) indicated that there are shortages of qualified teachers who understand the concept of computational thinking. Therefore, offering CS education to all students in K-12 school systems remains a challenging goal due to a lack of teacher education training in the areas of computational thinking and software development skills. Although professional development (PD) workshops with the goal of providing K-12 teachers with appropriate computer science training across the USA are gaining traction (Code Advocacy Coalition, 2021), there are factors present that impede the implementation of computing knowledge and related activities in the course activities. As computer science professional development in K-12 is relatively new, the literature suggests that the aggregation

of open-ended responses can be particularly useful when an area is understudied (Clarke & Braun, 2014), which motivated us to develop a streamlined approach to analyze the open-ended feedback from the teacher PD workshops. The aim was to provide evidence for the following research questions:

RQ1: Are the semantic analysis techniques a suitable and practical approach to evaluating open-ended feedback from teachers participating in computer science professional development?

RQ2: What are some motivations, barriers, and areas of concern present in this group of rural teachers?

The goal of this paper is to apply the semantic analysis techniques to the feedback provided by a population of rural teachers participating in professional development as they implement and develop computer science educational material in their classrooms. The semantic analysis approach is intended to reveal the impact of the experience of rural teachers participating in a PD workshop in applied computational thinking to demonstrate the efficacy of these proposed techniques.

2. Textual Data Analysis

Given the overall lack of teachers knowledgeable in CS, PD initiatives have been taken to engage teachers in computing workshops to familiarize them with CS concepts and block-based programming skills (Liu et al. 2011). As K-12 teaching programs are pioneered throughout the United States, data collection and high-quality data analysis throughout the development process are essential to ensure teachers' needs are met to the greatest extent possible. As noted by Clarke and Braun (2014), qualitative semantic analysis of open-ended responses is particularly useful when a topic area is not well-specified in theory or the setting requires a contextualist approach, both of which are applicable in this case, as the optimal instructional methods to prepare grade 3-8 teachers to integrate CS concepts and computational thinking in the core curriculum is not well-studied or clearly defined.

2.1 Formative Assessment

A variety of formative assessments were used to collect feedback from teachers each day during the workshop resulting in a wide range of open text responses. Formative assessments are open-ended prompts that are used to monitor learning. This feedback can be used to determine the level of understanding and confidence in the material and to help identify appropriate corrective adjustments in instruction (Guskey, 2005). Evidence suggests these strategies encourage self-reflection in learning (Black & Wiliam, 2009), increase engagement (Benotti et al., 2017), and are associated with higher scores on summative knowledge assessments (Hashemi, et al., 2016). The formative assessment prompts used in the analysis were strategies supported with moderate to strong evidence in the Institute of Education Sciences What Works Clearinghouse Practice Guide called Teaching Secondary Students to Write Effectively (Graham et al., 2016). Formative assessment strategies included K-W-L-S (Ogle, 1986; Steele & Dyer, 2014), 3-2-1 (Djudin, 2021), Exit Card Reflection (Patka, et al., 2016), and Plus/Deltas (Helminski, 1995). In brief, K-W-L-S ask learners to describe what they know, want to

know, learned, and still want to know. The 3-2-1 is an informational and persuasive prompt in which learners are asked to write three things they learned, two things they would like to learn more about, and one question they have on the topic. Exit card reflections requested details on ideas gained, insights relating to participation, suggestions for improvement, and questions. Plus/deltas prompted thoughts on positive aspects and suggested changes. See Appendix for full item text. Teachers received a presurvey prior to the workshop and submitted their suggestions/feedback online at the end of each day as part of the workshop's activities.

2.2 Semantic Analysis

Semantic analysis refers to the general study of the syntax of open-text data to derive meaning. Qualitative data analysis is useful in determining similarities and contiguity in textual data, as per Maxwell and Chmiel (2014). Increasingly, semantic analysis is being applied as a promising means to assess learners throughout the learning process. For instance, Masood et. al (2022) describe a series of preprocessing steps used to evaluate informal student feedback followed by sentiment analysis. Another recent application evaluated the sentiment of students impacted by isolation during quarantine as instruction was changed unexpectedly to an online format for a long period of time (Pastor, 2020). Chen, Li, & Huang (2020) tested an approach based on word segmentation and word clusters to provide output instantly associated with online discussions. Gottipati, Shankararaman, and Lin propose a series of preprocessing and analysis steps to automate qualitative text from end-of-course feedback (2018). In this case, thematic analysis and sentiment analysis were applied.

2.2.1 Thematic Analysis

Thematic analysis is a form of qualitative semantic analysis technique where a set of procedures is applied to either induce or deduce common themes from qualitative data to identify patterns of response (Clarke & Braun, 2014). Thematic analysis helps provide flexibility in handling complex data, particularly when the interpretation of semantic meaning may vary widely while maximizing adequate data representation, especially when outcomes are not easily predicted (Nowell et al., 2017). In this case, classifying responses using thematic analysis was expected to provide information on teacher perceptions vital to optimal instruction by the workshop team. For instance, it is possible that teachers may question the benefit of spending valuable classroom time teaching computational thinking. In that case, it would be wise to provide evidence and a rationale to the teachers on the advantages for their students. On the other hand, if teachers understand the benefits and are eager to learn more, spending more workshop time on applied computing skills would be appropriate.

Thematic analysis can be accomplished with either an inductive or deductive approach. Inductive methods of thematic analysis incorporate semantic clustering allowing themes to emerge. An inductive method is helpful in cases where unexpected patterns of response may appear (Nowell, et al., 2017). For example, this can help extract the major topics or themes that teachers most often describe in their

responses to the prompts. Deductive methods of classification derive themes based on prior expectations or theory. The most appropriate deductive schema for the classification of open-ended questions that elicited expressions of concern based on the context and types of response was the Concern-Based Adoption Model (CBAM) from Newlove and Hall (1976). CBAM is a well-established framework for classifying open-ended statements of concern from teachers tasked with implementing a new program in their schools. Chamblee and Slough (2004) provide a comprehensive meta-analysis using CBAM in technology implementation. Gabby et al. used CBAM to gather feedback from chemistry teachers introducing technical content (2017). The coding model is paraphrased from the original in Table 1.

Table 1. Concerns-based adoption model classification

Label	Primary Concern Expressed in Statement
Awareness	Little concern or involvement is indicated.
Informational	Substantiative aspects in a selfless manner such as general characteristics/requirements.
Personal	Uncertainty relating to demands, personal role, and personal adequacy.
Management	Processes and tasks involve the best use of information and resources.
Consequence	Relevance for students, e.g., student outcomes and performance and competency evaluation.
Collaboration	Coordination and cooperation with others regarding use.
Refocusing	Universal benefits. Individual has definite ideas about alternatives to proposed or existing form.

Classifications are paraphrased from "A manual for assessing open-ended statements of concern about an innovation" by Newlove and Hall (1976).

2.2.2 Sentiment Analysis

Sentiment analysis refers to the methodological comparison of widely available tools to determine the polarity of emotional valence, generally either a positive or negative affect and sometimes including a measure of intensity associated with individual words, statements, or sets of statements (Feldman, 2013). Sentiment analysis can help identify statements where words associated with positive feelings or negative feelings are included. The advantage of sentiment analysis is to identify indications of strong

emotions within answers. Expressions with a negative rating may indicate high emotionality and stress relating to the learning experience, something the workshop leaders would want to acknowledge and help with as soon as possible. It can also help identify strong motivators and reasons for excitement during the learning process.

3. Method

3.1 Research Setting

Motivated by initiatives like Liu et al. (2011) and the other challenges mentioned above, a multi-year project was initiated to provide instruction and support for grade 3-8 teachers in a rural midwestern state to implement computational thinking as a problem-solving framework in their daily instruction. A week-long PD workshop was launched in the Summer of 2021 with support from a federal grant. This program is an interdisciplinary effort developed and led by a diverse group of eight university professors representing multiple departments, fields of study, and backgrounds. The project aimed to incorporate writing in introducing CS while aligning with state performance standards in English, Math, Science, Social Studies, and CS. The workshop aimed to develop teachers' content knowledge and efficacy in integrating CS with STEAM (science, technology, engineering, arts, and math), project-based learning in Alice and Scratch (grade-level appropriate programming tools) and using Raspberry Pi in their current curriculum. The teachers were compensated for their participation in this workshop and have committed to continuing the project throughout the year in exchange for further compensation for their time and support from a federal grant program.

3.2 Participant Characteristics

In order to best support schools serving high-needs students, participants in the first-year teacher launch workshop consisted of twenty-nine teachers recruited from schools in a rural midwestern state with a Small Rural School Achievement (SRSA) and/or a Rural Low-Income Schools (RLIS) designation also carrying a Title I designation. Although information on race was not collected, the sample was predominantly white. The presurvey included 33 participants, whereas the actual participants in the workshop included 23 females and 6 males. The midwestern U.S. state from which the sample originated has been characterized as having moderate levels of rural poverty and a primarily agricultural property tax base, according to Showalter, et al. (2017). Inductive thematic analysis showed that the group represented a broad range of years of experience teaching (Figure 1), and their level of education varied (Figure 2).

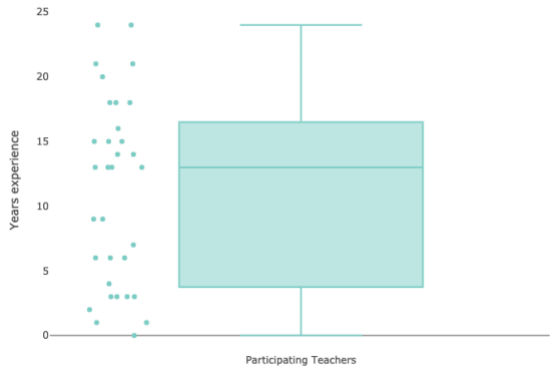

Figure 1. Years Experience

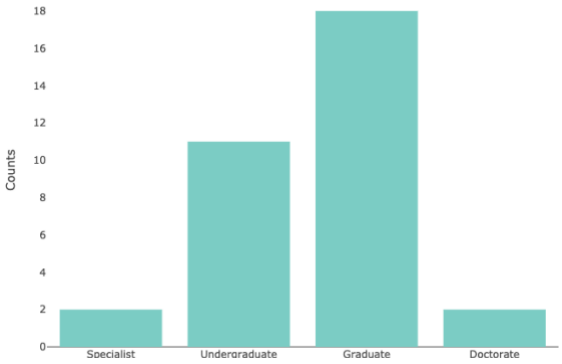

Figure 2. Level of Education

3.3 Thematic Analysis

Open-text feedback was collected before, during, and immediately after the workshop in response to multiple types of formative assessments, including K-W-L-S, 3-2-1, exit card, and plus/delta. To prepare the data for analysis, the following steps were taken. After spellchecking, a basic tokenization process was applied, splitting each textual response into individual alphanumeric units using standardized parsing and using regular expressions [\string^[:alnum:][:space:]’’]. Next, a list of standard English stop words from NLTK was used to identify and label stop words. For the frequency charts only, know, want, learn, and the numbers 3, 2, and 1, were added to the stop word list since these terms were within prompts and appeared often in the replies, and then the stop word list was excluded. This resulted in a word set of 5,041 tokens. In addition, simple plurals were changed manually to the same token as the singular form, e.g., “computer” and “computers” were changed to “computer(s)”, and proper nouns such as “Raspberry” and “Pi” were considered single term “Raspberry Pi” for aggregation. Inductive coding for some statements was determined using researcher triangulation; that is, responses were clustered into groups by two independent raters. These items related to the level of education (Specialist, Undergraduate, Graduate, or Doctorate), years of experience, and experience coding, which was open-text to allow for flexible self-report, as well as prompts requesting general comments or suggestions. For deductive analysis, two independent raters interpreted and applied CBAM classification to all open-ended responses with a third rater to resolve ties. Some items appeared to elicit statements involving concern more than others, such as what teachers learned, hoped to learn, and what unique needs their students had. A selection of several of these items is summarized in the results. For sentiment analysis, the preprocessing steps above were applied. This resulted in a word set of 9,961 words.

3.4 Sentiment Analysis

Sentiment analysis was derived using two tools, the SocialSent Programming domain-based lexicon from Hamilton, et al. (2016) and NLTK VADER (Natural Language Took Kit Valence Aware

Dictionary for Sentiment Reasoning, Bird, 2006). These tools are well-established, free, and readily available online. Both tools incorporate word embedding which refers to the likelihood of positive or negative sentiment based on the co-occurrence of neighboring words, after training on large corpora to provide context-specific results. SocialSent was derived from the top 250 Reddit forum subcommunities in 2014, while VADER was trained on historical Twitter data. All lexicons considered were tested for suitability with an inner join, matching terms from the test set and each lexicon. The match ratio is the number of words from the open-response data that were also present in the lexicon after stop word removal. The SocialSent 2014 Programming lexicon was selected as it demonstrated a high word match ratio (82%) after stop word removal. NRC (Mohammad & Turney, 2013), Bing (Hu & Liu, 2004), AFINN (Nielsen, 2011), Loughran & Mcdonald, 2011) were excluded prior to analysis because the word match ratio was 15% or lower.

VADER's polarity score function results in a compound score that has undergone normalization, with each statement being assigned a number ranging from -1 representing a negative polarity and +1 indicating a positive expression. A value close to zero indicates relative neutrality. The VADER tool is capable of handling various pitfalls in textual analysis, such as negation or more informal speech, by considering the semantic meaning of the piece of text in full (Bird, 2006). For example, the statement beginning with "If I'm being really honest here, the first thing that stood out to me was how much passion, excitement, and connectedness there already was. . ." resulted in a compound sentiment score of 0.99. Whereas a negative statement that began with "It is frustrating that we can't necessarily use Alice with our students if we don't have hard drives. . . "resulted in a compound sentiment score of -0.68. An example of an emotionally laden word from the SocialSent lexicon present in open-ended responses included "great" with a mean sentiment rating value assignment of 4.26, whereas "slow" had a mean sentiment value assignment of -3.12. For comparison with NLTK VADER, a summative score was calculated, adding the scores of the words by the response. Using more than one tool allowed for comparison and for some degree of cross-validation.

4. Results

In this section, a selection of outcomes is described after the use of inductive and deductive thematic analysis and a comparison of sentiment analysis with the SocialSent programming lexicon and NLTK VADER results applied to the open-ended questions from the workshop (see Appendix).

4.1 Inductive Thematic Analysis

Inductive thematic analysis was used to summarize participant characteristics, including the number of years of teaching experience (Figure 1) and level of education (as seen in Figure 1 and Figure 2), to determine participant classroom needs from open-text data. In response to "How much have you coded?", almost all (93%, N = 29) respondents reported either having not coded at all (n = 17) or described feeling like a beginner with only a superficial understanding of block-coding and computer

concepts (n = 6). Two reported teaching CS classes and felt somewhat confident. This form of inductive coding worked particularly well for suggestions and comments. For instance, responses to "What can we do to make tomorrow better?" included requests for certain foods or drinks (n = 5), environmental preferences (e.g., room temperature, n = 4), and accessibility improvements (n = 4), such as the number of breaks, pacing, and improving visibility. Otherwise, this item was left blank (n = 6) or complemented the experience overall, using the terms "good" or "great" n = 11). Keeping communication open and making sure learners are as comfortable as possible was important for the research team and seemed to improve the experience of the workshop participants. Upon review of token frequencies in this group, there was high word commonality within the K-W-L-S (Figure 3) and 3-2-1 strategies (Figure 4). Summarizing responses in this manner is helpful in determining topic areas that stood out the most for learners and in seeing common themes at a glance. This allowed the team to review the growth in the same participants as learning progressed. It is expected that over time responses will likely become more unique as the teachers begin to understand and feel more comfortable with the field of computer science.

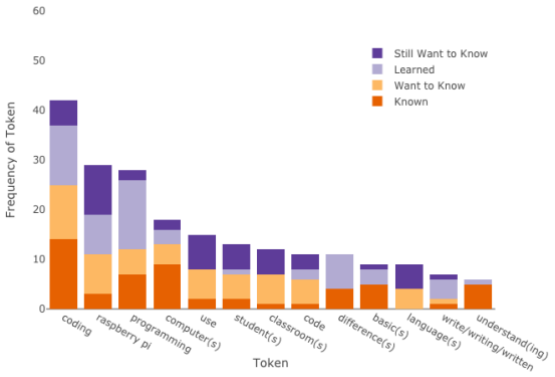

Figure 3. K-W-L-S Token Count

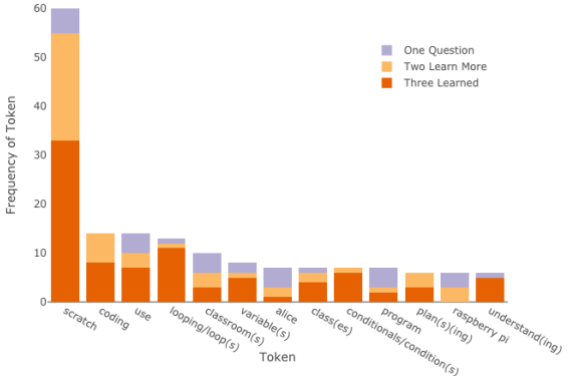

Figure 4. 3-2-1 Token Count

4.2 Deductive Thematic Analysis

In Figure 5, CBAM analysis is presented as applied to the open-ended responses given during the workshop, with criteria summarized in Table 1. This analysis demonstrated that teachers are highly motivated by meeting the needs of their students, as represented by the category of response labeled "Consequence". When asked what they hoped to learn during the week, teachers overwhelmingly expressed wanting to learn to better serve their students, e.g., "I am hoping to get more ideas to get kids excited to code and use logical reasoning!" When asked "How we can best support you as a learner. . ." , responses were often self-reflective, and tended to incorporate a sense of self-doubt in their efficacy in learning and applying CS content, as represented by the category labeled "Personal". Examples of these statements were as follows: "I will need lots of patience and hands-on experience" and "Lots of grace, coding is not my strong suit, but I will do my best". The "Unconcerned" category was applied to

responses like “None at this time”. On Day 1 of the workshop, “Informational” concerns were higher, where the focus is primarily on details and facts. For instance, on the want-to-know item, one teacher said, “What is Raspberry Pi capable of”. On Day 3, more teachers were concerned with “Consequences”, that is, relevance for their students and classrooms, e.g., “I want to know how Scratch is used with core content.” and “Personal” responses describing the content in relation to themselves.

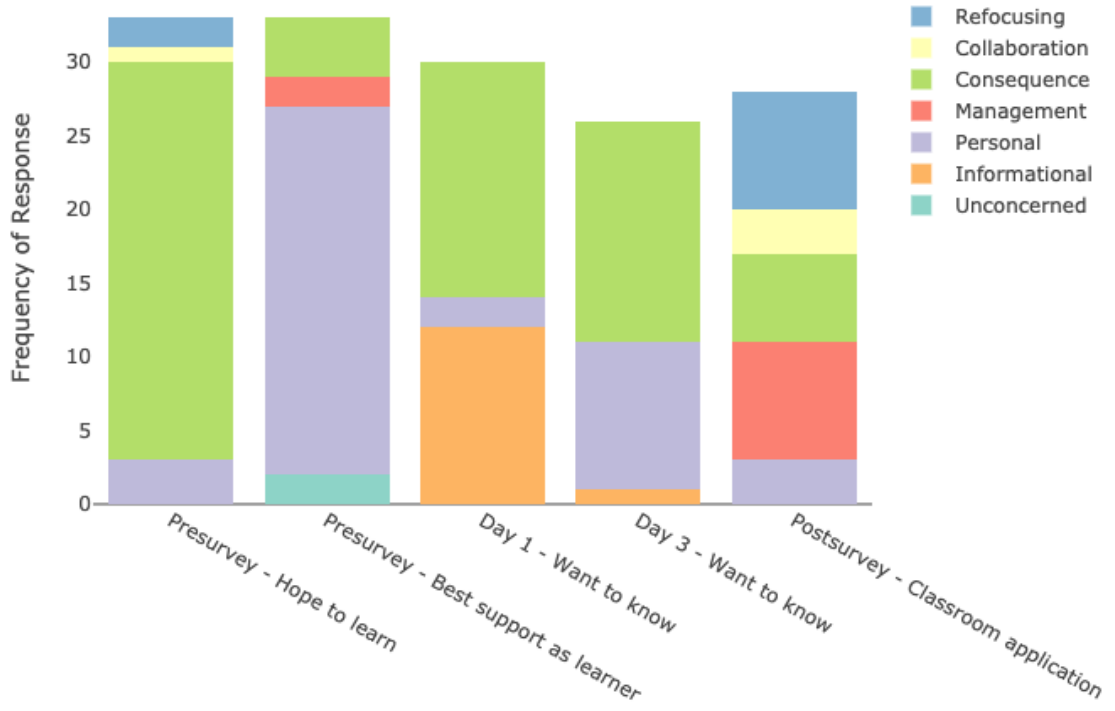

Figure 5. CBAM Deductive Analysis

4.3 Sentiment Analysis

A histogram of output was reviewed to check for normality across all individual statement scores, as seen in Figure 6. The SocialSent lexicon demonstrated a normal distribution, while the NLTK VADER result was leptokurtic, with most statements classified as close to neutral in tone. While NLTK VADER was generally in agreement with more extreme scores, results from SocialSent are summarized here. SocialSent results were evaluated using summary scores by prompt, as demonstrated in Figure 7. Teachers used words with highly positive emotional valence most often in answering what they hoped to learn. Responses to Plus +, or what they felt was working, were also high. Delta, which requested feedback on opportunities for improvement in presented content, was the lowest scoring item. Relatively low scorers also included thoughts on classroom applications and the unique needs of their students. SocialSent appeared useful for evaluating expressions on an individual response level basis as well.

Both tools (CBAM and VADER) appeared to help identify areas of strength and specific problems to be addressed.

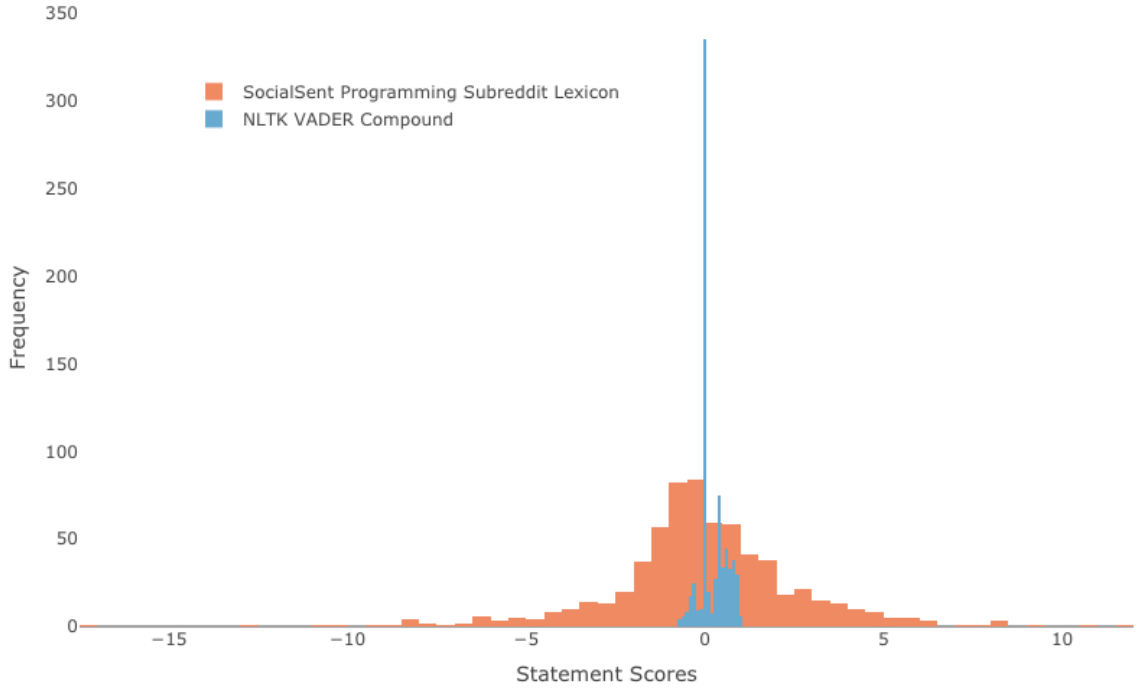

Figure 6. Comparison of SocialSent with NLTK

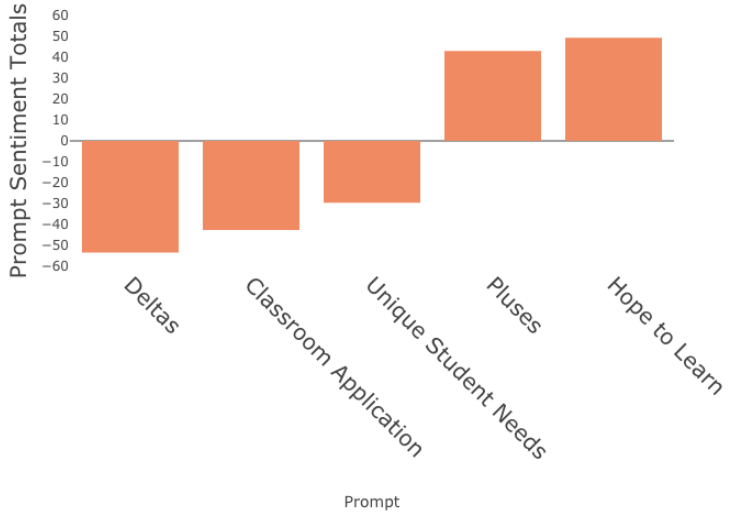

Figure 7. SocialSent Items with High Emotionality

5. Discussion and Conclusion

In this paper, various methods of semantic analysis are presented to evaluate the utility and practicality of handling open-ended textual responses from teachers in PD workshops. These methods included lexicon-based and word-embedded sentiment analysis, using both inductive and deductive approaches to derive patterns representing the learning experience of the participants. This approach is novel as a practical implementation suitable for regular use in that the study design utilizes open-source and readily available tools, an important factor since time and resources are often strained in a classroom setting. While open-response prompts are common, the volume of data can quickly become difficult to manage and more prone to subjectivity without the use of proper data analysis techniques. There is no existing standardized or streamlined process to make use of this form of valuable data.

Inductive methods of thematic analysis served to allow workshop organizers to intervene and correct issues that might otherwise decrease the likelihood of teacher success. For example, during the workshop, the team became aware that a sufficient number of computer monitors with HDMI capability were not available in some schools to be used with Raspberry Pi, and Alice was not usable on the only device available, Google Chromebooks. Collecting and aggregating open-response data from participants with these types of issues allowed the team to make adjustments and follow up to assist with resource management which would otherwise pose a serious barrier to successful implementation. Deductive methods using CBAM analysis provide information on areas of concern and level of comfort with material for teachers incorporating a new curriculum. In this case, participating teachers represented a wide range of comfort levels, with some describing feeling eager to move forward as fast as possible, while others felt the presentation was at the right pace. Therefore, self-paced activities and paired programming seemed helpful in meeting group needs.

CBAM proved to be an effective diagnostic tool to determine barriers that participating teachers face in classroom implementation to help the team to adjust the presentation and content of materials in the workshop and beyond. For example, on the first day of data collection, almost all teachers expressed fear and self-doubt. Whereas, by the last day of data collection, about one-third of the participants communicated excitement and described specific ways in which they plan to incorporate the material learned into their class, labeled “Refocusing”. An example of a Refocusing statement is “Help my science students create models to represent what they are learning or problem-solve scenarios through [S]cratch”. A few teachers expressed interest in collaborating with others in their schools which was classified as “Collaboration”. An example of a Collaboration statement is “I already teach coding but would def [definitely] love to collaborate with other teachers and bring some of their hands-on projects alive with coding”. A number of individuals discussed time management and process as well, which was classified under “Management”: “My main goal is to start by integrating coding into a core subject one day per week. I hope to increase this as I feel comfortable”.

In this group, CBAM was effective in assessing changes in confidence over time. CBAM revealed that initially, self-doubt among teachers was prevalent. Providing high support and sufficient time to practice appeared to be an effective intervention until confidence began to increase. When confidence in the form of interest in collaboration and/or refocusing ideas was high, the teachers expressed readiness to learn more conceptual information and maintain high levels of engagement. CBAM may help inform more specific lines of follow-up questioning in the future in the target group.

The outcomes of sentiment analysis using SocialSent suggest that using words with strong emotionality in either direction of polarity may be associated with learners being overwhelmed or being ready to learn more. An example of a low-scoring response was "Shorter breaks and more frequent. Think breaks, so much information in a short time. Almost overloading." Another participant listed several specific frustrations relating to a lack of hardware at their school and expressed the feeling this was overlooked by the team. Unique needs included "High poverty, lack of access at home and lack of experience" or mention of large class sizes. A positive high-scoring statement was "Great introduction to Coding. Very positive experience. Well-organized and planned. Love how there are different presenters" or "I am just excited to bring a new program full of possibilities to my students". Sentiment analysis is useful to target learners expressing high emotionality and can assist PD facilitators to implement timely interventions among learners with the most need of support.

There are some limitations to the interpretation based on the nature of this study. The generalizability of these methods depends on the data used and the context. The small sample size may influence how representative these data are of rural teachers from the target group. In sentiment analysis, generally, questions relating to the unknown tended to be more negative in tone. However, this skew may be partially due to associations from the Reddit programming sub-community from which the lexicon was derived that may not reflect the true feelings of the teachers. For example, the word "teacher" has a particularly low score in the SocialSent programming lexicon, but teachers using the word "teacher" are not likely to share that negative connotation. While teachers were extremely hard-working and enthusiastic throughout the week, mental fatigue after an intensive workshop with a daily commute may have influenced their responses. In addition, the SocialSent lexicon was highly applicable in this particular use case but may be somewhat dated since the set was derived from a 2014 forum. Language relating to computing and user attitudes are apt to change over time as technology progresses. Finally, the attitudes expressed may be transitory in nature or influenced by the wording in the questions themselves. More research is needed to confirm any apparent trends within the findings reported here and to follow up appropriately as the team supports the teachers in the process of curriculum integration during the academic year.

This paper aimed to demonstrate whether semantic analysis is a practical approach to finding common themes among participants in a PD workshop. It was anticipated that this could act as a diagnostic tool for teams to gain information helpful to organize teacher PD workshops, identify resources needed in

their school settings, as well as potential areas of improvement. Overall, the textual analysis of open-ended formative assessments showed promise to quickly identify common themes and identify learners needing a challenge or support. Therefore, the methodology described can be used to effectively analyze open-ended responses from participants in similar PD initiatives, without which the organizers may fail to address barriers teachers face in successfully implementing the workshop objectives.

Acknowledgments

Research reported was supported by a $4 million grant from the Early-Phase Education, Innovation and Research (EIR) Program of the U.S. Department of Education (USDE) under award number S411C200085. The content is solely the responsibility of the authors and does not necessarily represent the official views of the funding agency.

References

Benotti, L., Martinez, M. C., & Schapachnik, F. (2017). A tool for introducing computer science with automatic formative assessment. *IEEE Transactions on Learning Technologies, 11*(2), 179-192. Retrieved from https://doi.org/10.1109/TLT.2017.2682084

Bird, S. (2006, July). NLTK: The natural language toolkit. In *Proceedings of the COLING/ACL 2006 Interactive Presentation Sessions* (pp. 69-72). Retrieved from https://aclanthology.org/P06-4018.pdf

Black, P., & Wiliam, D. (2009). Developing the theory of formative assessment. *Educational Assessment, Evaluation and Accountability (formerly: Journal of Personnel Evaluation in Education), 21*(1), 5–31. Retrieved from https://doi.org/10.1007/s11092-008-9068-5

Brown, E., & Brown, R. (2020). The Effect of Advanced Placement Computer Science Course Taking on College Enrollment. *West Coast Analytics*. Retrieved from https://www.cuny.edu/wp-content/uploads/sites/4/page-assets/about/administration/offices/oira/policy/seminars/APHighSchoolManuscript_JHRNotypset.pdf

Chamblee, G., & Slough, S. (2004). Using the concerns-based adoption model to assess changes in technology implementation. In *Society for Information Technology & Teacher Education International Conference* (pp. 864-871). Association for the Advancement of Computing in Education (AACE). Retrieved from https://www.learntechlib.org/primary/p/13584/

Chen, C. M., Li, M. C., & Huang, Y. L. (2020). Developing an instant semantic analysis and feedback system to facilitate learning performance of online discussion. *Interactive Learning Environments*, 1-19. Retrieved from https://doi.org/10.1080/10494820.2020.1839505

Chetty, R., Friedman, J. N., & Rockoff, J. E. (2014). Measuring the impacts of teachers II: Teacher value-added and student outcomes in adulthood. *American Economic Review, 104*(9), 2633–79. Retrieved from https://doi.org/10.1257/aer.104.9.2633

Clarke, V., & Braun, V. (2014). Thematic analysis. In *Encyclopedia of Critical Psychology* (pp. 1947–1952). Springer, New York, NY. Retrieved from https://doi.org/10.1007/978-1-4614-5583-7_311

Code Advocacy Coalition (2021). State of computer science education: Accelerating action through advocacy. Retrieved from https://advocacy.code.org/2021_state_of_cs.pdf

Djudin, T. (2021). Promoting students' conceptual change by integrating the 3-2-1 reading technique with refutation text in the physics learning of buoyancy. *Journal of Turkish Science Education, 18*(2), 290–303. Retrieved from http://www.tused.org/index.php/tused/article/view/734/668

Education Commission of the States (2019). Stem content: Give more united states students early access to computer science. Retrieved from https://vitalsigns.ecs.org/state/united-states/curriculumprogram-grade-stats

Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM, 56*(4), 82-89. Retrieved from https://doi.org/10.1145/2436256.2436274

Gabby, S., Avargil, S., Herscovitz, O., & Dori, Y. J. (2017). The case of middle and high school chemistry teachers implementing technology: Using the concerns-based adoption model to assess change processes. *Chemistry Education Research and Practice*, *18*(1), 214-232. Retrieved from https://www.learntechlib.org/primary/p/13584/

Gottipati, S., Shankararaman, V., & Lin, J. R. (2018). Text analytics approach to extract course improvement suggestions from students' feedback. *Research and Practice in Technology Enhanced Learning*, *13*(1), 1-19. Retrieved from https://doi.org/10.1186/s41039-018-0073-0

Graham, S., Bruch, J., Fitzgerald, J., Friedrich, L. D., Furgeson, J., Greene, K., ... & Smither Wulsin, C. (2016). Teaching Secondary Students to Write Effectively. Educator's Practice Guide. What Works Clearinghouse.™ NCEE 2017-4002. *What Works Clearinghouse*.

Guskey, T. R. (2005). Formative Classroom Assessment and Benjamin S. Bloom: Theory, Research, and Implications [Paper presentation]. *Annual Meeting of the American Educational Research Association*, Montreal, Canada. Retrieved from https://files.eric.ed.gov/fulltext/ED490412.pdf

Guzdial, M., & Hill, R. K. (2019). Getting high school, college students interested in CS. *Communications of the ACM, 62*(12), 10–11. Retrieved from https://doi.org/10.1145/3365581

Hamilton, W. L., Clark, K., Leskovec, J., & Jurafsky, D. (2016, November). Inducing domain-specific sentiment lexicons from unlabeled corpora. In *Proceedings of the Conference on Empirical methods in Natural Language Processing*. NIH Public Access. Retrieved from https://doi.org/10.18653/v1/D16-1057

Hashemi, A., Mobini, F., & Karimkhanlooie, G. (2016). The impact of content-based pre-reading activities on iranian high school EFL learners' reading comprehension. *Journal of Language Teaching and Research, 7*(1), 137. Retrieved from https://doi.org/10.17507/jltr.0701.15

Helminski, L. (1995). Total quality in instruction: A systems approach. In H. V. Roberts (Ed.), Academic initiatives in total quality for higher education (pp. 309–362). Milwaukee, WI: ASQC Quality Press.

Hu, M., & Liu, B. (2004, August). Mining and summarizing customer reviews. In *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (pp. 168-177). Retrieved from https://doi.org/10.1145/1014052.1014073

Hubwieser, P., Giannakos, M. N., Berges, M., Brinda, T., Diethelm, I., Magenheim, J., ... & Jasute, E. (2015). A global snapshot of computer science education in K-12 schools. In *Proceedings of the 2015 ITiCSE on Working Group Reports* (pp. 65-83). Retrieved from https://doi.org/10.1145/2858796.2858799

Lamb, R., Annetta, L., Vallett, D., Firestone, J., Schmitter-Edgecombe, M., Walker, H., ... Hoston, D. (2018). Psychosocial factors impacting STEM career selection. *The Journal of Educational Research, 111*(4), 446–458. Retrieved from https://doi.org/10.1080/00220671.2017.1295359

Liu, J., Lin, C. H., Hasson, E. P., & Barnett, Z. D. (2011, March). Introducing computer science to K-12 through a summer computing workshop for teachers. In *Proceedings of the 42nd ACM technical symposium on computer science education* (pp. 389-394). Retrieved from https://doi.org/10.1145/1953163.1953277

Loughran, T., & McDonald, B. (2011). When is a liability not a liability? Textual analysis, dictionaries, and 10-Ks. *The Journal of Finance, 66*(1), 35-65. Retrieved from https://doi.org/10.1111/j.1540-6261.2010.01625.x

Mahadeo, J., Hazari, Z., & Potvin, G. (2020). Developing a computing identity framework: Understanding computer science and information technology career choice. *ACM Transactions on Computing Education (TOCE), 20*(1), 1–14. Retrieved from https://doi.org/10.1145/3365571

Marré, A. (2017). Rural education at a glance, 2017 edition. *U.S. Department of Agriculture, Economic Research Service, Economic Information Bulletin, 171*. Retrieved from https://www.ers.usda.gov/webdocs/publications/83078/eib-171.pdf?v=3841.6

Masood, K., Khan, M. A., Saeed, U., Al Ghamdi, M. A., Asif, M., & Arfan, M. (2022). Semantic Analysis to Identify Students' Feedback. *The Computer Journal*, *65*(4), 918-925. Retrieved from https://doi.org/10.1093/comjnl/bxaa130

Maxwell, J. A., & Chmiel, M. (2014). Notes toward a theory of qualitative data analysis. *The SAGE Handbook of Qualitative Data Analysis*, 21-34.

Mohammad, S. M., & Turney, P. D. (2013). Crowdsourcing a word-emotion association lexicon. *Computational Intelligence*, *29*(3), 436-465. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1111/j.1467-8640.2012.00460.x

Newlove, B. W., & Hall, G. E. (1976). A Manual for Assessing Open-Ended Statements of Concern About an Innovation. Retrieved from https://eric.ed.gov/?id=ED144207

Nielsen, F. Å. (2011, March). AFINN: A new word list for sentiment analysis on Twitter. *Informatics and Mathematical Modelling*, Technical University of Denmark. Retrieved from http://www2.compute.dtu.dk/pubdb/pubs/6010-full.html

Nowell, L. S., Norris, J. M., White, D. E., & Moules, N. J. (2017). Thematic analysis: Striving to meet the trustworthiness criteria. *International Journal of Qualitative Methods, 16*(1). Retrieved from https://doi.org/10.1177/1609406917733847

Ogle, D. M. (1986). K-W-L: A Teaching Model That Develops Active Reading of Expository Text. *The ReadingTeacher, 39*(6), 564–570. http://www.jstor.org/stable/20199156

Patka, M., Wallin-Ruschman, J., Wallace, T., & Robbins, C. (2016). Exit cards: creating a dialogue for continuous evaluation. *Teaching in Higher Education*, *21*(6), 659–668. Retrieved from https://doi.org/10.1080/13562517.2016.1167033

Salehi, S., Wang, K. D., Toorawa, R., & Wieman, C. (2020, February). Can majoring in computer science improve general problem-solving skills? In *Proceedings of the 51st ACM Technical Symposium on Computer Science Education* (pp. 156–161). Retrieved from https://doi.org/10.1145/3328778.3366808

Showalter, D., Klein, R., Johnson, J., & Hartman, S. L. (2017). Why Rural Matters 2015-2016: Understanding the Changing Landscape. A Report of the Rural School and Community Trust. *Rural School and Community Trust*. Retrieved from https://files.eric.ed.gov/fulltext/ED590169.pdf

Steele, J., & Dyer, T. (2014). Use of KWLS in the online classroom as it correlates to increased participation. *Journal of Instructional Research, 3*, 8–14. Retrieved from https://files.eric.ed.gov/fulltext/EJ1127637.pdf

U.S. Bureau of Labor Statistics (2021, May). Occupational Outlook Handbook: Computer and information technology occupations. Retrieved from https://www.bls.gov/ooh/computer-and-information-technology/home.htm

Wang, J., & Hejazi Moghadam, S. (2017, March). Diversity barriers in K-12 computer science education: Structural and social. In *Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education* (pp. 615-620). Retrieved from https://doi.org/10.1145/3017680.3017734

Appendix

Items Used in Analysis	*Semantic Analysis*
Presurvey	
What is your highest level of education?	Researcher Triangulated (Fig. 1)
Including this school year, how many years have you been employed as a teacher?	Researcher Triangulated (Fig. 2)
What do you hope to learn by participating in the CODERS project? (Hope to learn)	CBAM (Fig 5.), Sentiment Analysis (Fig. 6, Fig. 7)
How can we best support you as a learner during the CODERS summer workshop? (Best support as learner)	CBAM (Fig 5.), Sentiment Analysis (Fig. 6, Fig. 7)
How much have you coded?	Researcher Triangulated (Sec. 4.1)
Day 1 (K-W-L-S)	
K - What I know or think I know (Known)	Token Frequency (Fig. 3), CBAM
W - What I want to know (Want to know)	Token Frequency (Fig. 3), CBAM (Fig.5)
L - What I learned (Learned)	Token Frequency (Fig. 3), CBAM
S - What I still want to know (Still want to know)	Token Frequency (Fig. 3), CBAM
What do we need to know to make tomorrow better?	Researcher Triangulated (Sec. 4.1)
Day 2 (Pluses/Deltas +/Δ)	
Identify things that are working (Pluses +)	Researcher Triangulated, Sentiment Analysis (Fig. 6, Fig. 7)
Identify opportunities for improvement (Deltas Δ)	Researcher Triangulated, Sentiment Analysis (Fig. 6, Fig. 7)
Day 3 (3-2-1)	
What are three things you learned? (Three Learned)	Token Frequency (Fig. 4), CBAM, Sentiment Analysis (Fig. 6)

What are two things you want to learn more about? (Two Learn More/Day 3 - Want to know)	Token Frequency (Fig. 4), CBAM, Sentiment Analysis (Fig. 6)
What's one question you have? One Question	Token Frequency (Fig. 4), CBAM, Sentiment Analysis (Fig. 6)
Day 4 (Exit Card) As a result of my participation, here are:	
Ideas I have gained	CBAM, Sentiment Analysis (Fig. 6)
Insights I have about what helped me learn, process, and/or fully participate	CBAM, Sentiment Analysis (Fig. 6)
Suggestions I have for how to make the work stronger	Researcher Triangulated, Sentiment Analysis (Fig. 6)
Questions I have	Researcher Triangulated, Sentiment Analysis (Fig. 6)
Postsurvey	
Describe your thoughts about applying computational thinking and programming skills using Alice and/or Scratch in your classroom (Classroom application)	CBAM, Sentiment Analysis (Fig. 6, Fig. 7)
What unique needs do your students have that you think CODERS needs to know about? (Unique student needs)	CBAM, Sentiment Analysis (Fig. 6, Fig. 7)
What suggestions do you have to improve the teacher launch in the future?	CBAM, Sentiment Analysis (Fig. 6)


www.ijcses.org