LEARNING TO CODE: A NARRATIVE INQUIRY

"Without code we would probably be like cave people"

BY SHAHIRA POPAT

A thesis

submitted to Victoria University of Wellington

in partial fulfilment of the requirements for a

Master in Education

2018

Abstract

In 2018 there was educational change in New Zealand with the introduction of new curriculum content for digital technologies. A key component of the digital technologies curriculum content was computational thinking where all students from Years 1 to 10 were expected to learn core coding concepts. The reasons for introducing coding into schools reflected a range of ideologies including preparing children to contribute meaningfully to society in the digital age. This narrative inquiry aimed to explore the value of coding in the curriculum through the experiences of students in Years 7 and 8. The research questions to meet this aim were; Why do students think coding is taught in school? Do students use coding outside of school? Why do students want to learn how to code and how do students think coding might help them or be useful?

Curriculum ideologies underpinned this study as a theoretical framework to evaluate student experiences of coding across two case studies. The narratives were derived from focus group interviews held at two different schools. Similarities across the case studies included students' beliefs about the benefits of including coding in the curriculum. Students' felt confident that learning coding allowed them to; understand the digital world, create digital products, prepare for the future, teach others and fix broken technology. They could not comprehend what their lives would be like without technology and therefore coding. Some students believed that "without code we would probably be like cave people".

The main difference between the case studies was the level of teacher direction. This reflected a contradiction between competing curriculum ideologies and addressed the broader debate in education of 21st century skills versus powerful knowledge. The contradictions highlighted how the pedagogical design of coding in the curriculum could be effectively structured.

Traditional knowledge and teacher explanation were found to be important to students when learning more complex coding. However, globalisation is a key concept for education in a digital age. Therefore, opportunities can be created for students to build on knowledge and collaborate in new and challenging ways. Treating coding as a social practice by teaching students to connect with the wider community or to use programming for social good can engage them with experiences beyond their own. This

does not mean abandoning the elements of 21st century learning, such as students' own experiences or active learning. Drawing on the strengths of both traditional knowledge and 21st century learning approaches can lead to more powerful knowledge creation.

KEY WORDS

Coding Programming Computational thinking Digital technology

Curriculum change Curriculum ideology 21st century skills Powerful knowledge

Acknowledgements

Firstly, I would like to thank my supervisor and mentor Dr Louise Starkey. Not only did she teach me a lot about education in a digital age, she supported me throughout my study providing advice, direction, encouragement and most importantly to think critically and be a better writer. I always knew that I could depend on her when I needed her.

I would also like to thank the participants of this study as without them this narrative inquiry would not have been possible. Their eagerness to discuss their experiences of coding made my first time holding focus groups less challenging!

My dear parents, brother and sister. Not only for their financial assistance when making the decision to resign from my job but their continued love and understanding of how important it was for me to further my education. My sister's Snapchats which motivated me to keep up with the writing process particularly at times when I was struggling to stay on track. but also urging me to take breaks and have some "me" time when needed. My brother's invaluable advice on my writing style, thank you.

Leaving my job meant making new connections with people and I was fortunate to broaden my network of friends during this process. The people in my life, new and old, who showed an interest in what I was doing and provided me with the much-needed conversations I missed from working in a school every day.

Of course, I could not go without acknowledging my husband Simon. He had to work 100 times harder to support our family while I studied. Managing such a change to our lives was not easy. I am forever grateful that I was able to go back to University and that the flexibility of study allowed me to spend more time with my children. It was not only a journey for myself, it was for my family, particularly my daughters Ariana and Layla. They are growing up in a world so different to the one we grew up in and I feel like I understand that more now than I did before. Just last week Ariana came home and said she had learnt coding at school! This thesis is dedicated to them.

Contents

List of tables

List of figures

CHAPTER 1 INTRODUCTION

New curriculum content in New Zealand

A curriculum is determined through students' needs and wants, teachers' knowledge and expertise and/or government policies due to societal issues (Brown, 2006). As a curriculum is an end of a series of decisions made by people it is subject to constant review and revision. In 2018 there was educational change in New Zealand with the introduction of new curriculum content to ensure that students across all year levels could access learning aimed at building their digital skills and fluency (Ministry of Education [MOE], 2017a). Schools had a transition period of two years and the new curriculum is to be fully implemented from the start of 2020. All young people from Years 1 to 10 are to take part in digital technologies education and senior students can choose digital pathways to develop more specialised skills (MOE, 2017a). It is hoped the new curriculum content will prepare children to contribute meaningfully to society in the digital age.

The Ministry of Education invited submissions on the proposed digital technologies curriculum content over July and August 2017. A report summarising the consultation responses of individuals, businesses and organisations highlighted a concern. Many respondents felt that digital technology should not have its own learning area and should be integrated across the curriculum (Chen, 2017). It was felt that digital technology would become siloed and as the curriculum was already crowded, the new content would take away from other learning areas, particularly existing areas of technology. However, digital technology was included as a standalone area under technology in the New Zealand curriculum and it is anticipated that students will be encouraged to access knowledge and skills from other learning areas (MOE, 2017a).

A key area of digital technologies in the curriculum is computational thinking where students are expected to gain an understanding of the computer science principles that underlie all digital technologies. This is defined as, "being able to express problems and formulate solutions in a way that means a computer can be used to solve them" (MOE, 2017a, p11). It is envisaged that students will be able to take advantage of the capabilities of computers to become creators of digital products, not just the users (MOE, 2017a). Although the digital technologies curriculum content intended to be

relatively flexible for schools, a set of progress outcomes were identified. For computational thinking by the end of Year 10 students should be able to "independently decompose a computational problem into an algorithm that they use to create a program" (Te Kete Ipurangi [TKI], 2017, p.3). Coding and programming are used interchangeably in this study and encompass not necessarily writing code but understanding the processes involved in coding such as algorithms and the steps needed to get to a solution. Digital technology, computer science and computational thinking are referred to throughout this paper and when discussed they are linked or related to coding in the curriculum. Digital technology, computer science and computational thinking are not just about teaching coding. However, there is an element that should expose students to programming, because that is part of their world (TKI, 2017).

Views and beliefs about what should be taught and why have changed over time due to changes in society. Coding was taught in New Zealand in the 1970's as part of mathematics but disappeared within a decade as it lacked relevance to students lives (Bell, Andreae, & Robins, 2014). There is also a tendency for curricula to maintain some stability with continued links to tradition regardless of any changes incorporated in curricular reform (McGuiness Institute, 2016). In the past, education reforms in New Zealand have "struggled to achieve a balance between meeting both the needs of the individual and the growing demands of a changing economy, largely owing to rapid changes in technology and society" (McGuiness Institute, 2016, p.10). Can introducing coding into the curriculum achieve this balance? For example, will it provide opportunities for students which serve their interests but also support a more competitive economy?

The rationale for the research

Studies about what to include in a curriculum should not only focus on the subject itself but research should examine conceptions of curriculum (Brown, 2006). Conceptions of curriculum are shaped or derived from social ideologies, reinforced by views and beliefs about what should be taught and why (Adamson & Morris, 2014). Curriculum change in New Zealand has highlighted the importance for teachers to understand the nature of the curriculum (Brown, 2006). Interpreting curriculum through its manifestations of ideology and the planned, enacted or experienced curriculum will assist in evaluating

the implementation of curriculum change. In pluralistic societies, the curriculum is influenced by a combination of conceptions and ideologies which may contradict one another. The reasons presented for the introduction of coding also exhibit contention. For example, there is the entrepreneurial idea of coding, where students will become the producers of technology (MOE, 2017a). Another view is that students can develop a range of learning competencies such as thinking and collaborative skills through coding tasks (Falloon, Hale & Fenemor, 2016). However, in principle coding is not required to develop these skills (Sterling, 2016). Furthermore, the government and market analytics firms believe that learning code is important for career opportunities and employability. Nikki Kaye, the Minister of Education at the time of curriculum change stated that the future workforce will benefit from knowledge and skills relating to software development, digital media content and technology design (MOE, 2017a). Burning Glass Technologies (2016) analysed jobs from over 26 million online job postings in the United States of America [US] and reported the importance of learning to code as a skill in the labour market. The report identified 7 million job openings in 2015 which valued coding, and these were not just programming jobs. Therefore, including coding in the curriculum appears to be multi-factorial.

If young children are having to learn to code, their opinions on the value and usefulness of coding in the curriculum should be considered. During the consultation phase of the new curriculum only 1% of respondents were students (Chen, 2017). Students' thoughts, experiences and ideas are also important when the government makes decisions (Ministry of Youth Affairs, 2009). In the context of this study, the student voice allowed a comparison of curriculum intentions against how young people live today. Investigating students' experiences of coding and how these aligned with the ideological intentions of digital technologies in the curriculum may in turn influence the way individual educators think about coding in the curriculum. Interpreting the manifestations of ideology through students' experiences of coding will examine the variety of beliefs for including coding in the curriculum and identify any contradictions of importance to this study.

Locating myself within the research

I trained in Business Education in 2003 after completing a BA(Hons) in Business Administration. As I was teaching Year 10 to 13 commerce, I felt it was important to keep myself informed of technological advancements and how these changed the way people conducted their business. I was also keen to apply technological changes to my teaching in the classroom. During my experience as a classroom teacher, I witnessed more creative and less passive use of the internet. I investigated tools such as Skype in the classroom, how to implement 'Bring Your Own Device' successfully and using online learning such as OneNote. I was intrigued that the school I previously taught at had introduced coding into their primary school and students had started using 3D printers and robotics. I tried to understand the importance of digital technology in education. Including how it could be effectively used to provide knowledge and skills for the future, but I often reverted to traditional teaching of concepts, using technology as a tool to apply knowledge.

I am in awe of innovative and creative people and I read success stories of young children using coding for personal and social good. One that resonated with me was of Maru Nihoniho who was the first Māori female game developer in the world. She used her own resources to develop a game for which PlayStation offered a contract for in 2007. She also collaborated with the University of Auckland to create a game to help young people navigate depression which won awards from the United Nations (Kea, 2017). Nihoniho has been working on social projects to bring attention to mental health and to teach coding to New Zealand youth. She isn't alone in using digital technology or coding for social good. Students across the globe are starting to set up social enterprises that use coding technology to solve social issues and more organisations are providing funding for these projects (Bouwkamp, 2015; Kan & Ongchoco, 2017).

My daughters are growing up in a digital age and I want to understand ways young people are thinking about and using digital technology such as coding. Is it to design or create games or apps people can use? Is it for social good? Do school children want to learn code and if so, why? Do they know the possibilities coding could bring? I am particularly interested in students' own experiences of coding and the importance of this research to inform practice. Examining what students believe they are learning as they code and why they think it's useful or important, assisted in this.

Chapter 1 summary

The new digital technologies curriculum content in New Zealand came into effect in January 2018. It requires all students from Years 1 to 10 to start learning coding concepts by 2020. As students' must learn how to code, I wanted to understand how they use coding in their daily lives and if students think coding would be useful in the future. Interpreting how students' experiences compared with the intentions of the new curriculum and examining these through the various conceptions or ideologies aimed to determine the overall value of learning coding in schools.

The order of information in this thesis is; literature review, methodology, findings, discussion, and conclusion. The next chapter is in two parts; the first reviews existing literature related to curriculum theory and introduces a theoretical framework. The second focuses on coding in schools and explores reasons to include coding in the curriculum. Therefore, Chapter 2 identifies the current situation, gaps in the research thus far, and why further research on this topic is required. Chapter 3 describes the methodology and methods, including the research design. Chapters 4 and 5 present the findings from two case studies which include student narratives and a cross-case analysis. Chapters 6 and 7 include the discussion and conclusions.

CHAPTER 2 LITERATURE REVIEW

Introduction

This chapter reviews literature relating to curriculum theory and coding in schools. In the first part of this chapter I explain curriculum perspectives and how these reflect different emphases of what constitutes a curriculum. Curriculum ideologies underpin this study as a theoretical framework to evaluate the introduction of coding in the curriculum. Using previous examples from New Zealand I identify how curriculum ideologies can be distinguished from students' experiences of coding. In addition, I interpret the ideological intentions of digital technologies in the curriculum so that these can be compared against the findings of this study.

In the second part of this chapter, I consider the literature that investigated coding in schools using examples from New Zealand and globally. This provides context for the discussion of why coding is being considered as a compulsory part of the New Zealand curriculum. The main reasons presented are the development of competencies, or knowledge and skills considered useful in society and for work. There is research which noted the difficulty found in learning how to code (Webb et al., 2017a; Xia, 2017). This was mostly seen amongst undergraduate and high school students in their first year of a coding (Mow, 2008). To this extent, studies have examined ways that coding can be taught to increase enjoyment, self-efficacy, positive attitudes, and achievement in coding (Bishop-Clark, Courte, Evans, & Howard, 2007; Pellas & Peroutseas, 2017; Theodoraki & Xingalos, 2014). As this area had already been extensively researched it was not the focus of this thesis. However, the pedagogical design is important in influencing certain competencies and due to its significance, it is included within the literature review.

PART 1: CURRICULUM THEORY

In this section I explain curriculum perspectives to provide an insight into the changing nature of curriculum. These perspectives reflect different conceptions which are shaped by curriculum ideologies. I examine the ideologies to provide a framework by which to evaluate curriculum change; the introduction of coding into schools.

Curriculum perspectives

Curriculum is influenced by beliefs and traditions therefore change tends to be evolutionary rather than revolutionary (Marsh & Willis, 1995; McGuiness Institute, 2016). Constructivist theorists believe that to educate people a curriculum should be centred on individual and personal development where students' construct knowledge through their own experiences (Brooks, 1986; Dewey, 1916; Von Glaserfeld, 1995). Curriculum change in the digital age requires further innovation because of technology and globalisation. It means education can be more suited to individual tastes, interests and abilities. This is because students are able to collaborate with others and access courses and knowledge they may not be able to do so in traditional school settings (L. Starkey, personal communication, August 7, 2017). This takes learning beyond the students' own experiences.

Economists suggested that education in the 21st century, should focus on economic knowledge known as the knowledge society rather than academic knowledge (Drucker, 1989; Sharma, 2004). Academic knowledge encompasses understanding and explanation and protects the elite minority, but knowledge-based societies are groups of individuals that see knowledge as the primary source of all economic growth (Gilbert, 2005). Economic knowledge uses knowledge to produce something new. In the industrial age this was defined as the production of tangible assets, as most jobs were in the manufacturing industries. More recently, jobs are increasingly in the creative, technological and service based industries and it is this knowledge that is required for economic growth (World Economic Forum [WOE], 2016a). Politicians emphasised this utilitarian purpose of education as technology is a part of nearly every industry and most careers (Curran, 2018). Although New Zealand's technology sector is currently the third largest contributor to the economy, the government want to increase its stake to the

second largest contributor to GDP by 2025. One way to achieve this is by identifying opportunities within the game development sector. This is because gaming is the fastest growing example of creative technologies in New Zealand (Curran, 2018). Gaming and other interactive media can have a range of uses across a variety of sectors from economic development to job creation. The new curriculum content exposes students to digital technology, including game design (MOE, 2017a). When the goal of education is to support innovation, economic wealth and productivity a curriculum should focus on the knowledge and skills required to achieve this (Hakala, Uusikylä & Järvinen, 2015).

The skills required to thrive in society and the workforce today are referred to as 21st century skills. The pedagogical focus of a 21st century learning approach is on generic competencies (McPhail & Rata, 2016). The 2007 New Zealand curriculum showed full commitment to the 21st century skills inter-disciplinary curriculum design (McPhail & Rata, 2016). This was in the Ministry of Education's strategic plan and in the development of key competencies in the New Zealand Curriculum (MOE, 2014). The key competencies included 'managing self' and 'relating to others'. They correspond to the way students should *do* learning. Soon after implementation there was a shift from the content of learning areas being interpreted as a learning end to, "a means for teaching how to learn and for fostering lifelong learning" (McDowall & Hipkins, 2018, p8). The skills are not only to prepare students for employment but beyond that are skills for social life. Despite this, most people in government and education do not fully understand how the nature of knowledge is changing (Gilbert, 2005).

"To be able to interpret the curriculum in a way that values knowledge society rather than industrial age ways of thinking about education requires major changes" (Bull, 2009, p1) not only in *what* is known but *how* it is known.

There was a common misunderstanding when the key competencies were first implemented. Schools thought that these skills should replace the learning of fundamental pre-determined or prescribed knowledge. This led to a "watering down" of the enacted curriculum (McDowall & Hipkins, 2018, p8). There was also a misconception that a learner-centred curriculum meant that key learning decisions were left to students (Bolstad, Gilbert, McDowall, Bull, Boyd, & Hipkins, 2012; McDowall & Hipkins, 2018). The 2007 curriculum intended that teachers would find meaningful ways to integrate the key competencies into learning areas so that knowledge was not

just about *doing* or *knowing* but about *doing, knowing and being* (Bolstad et al., 2012; McDowall & Hipkins, 2018). An example of this is students using academic or disciplinary knowledge in new contexts, combinations or innovative ways. This can be achieved by creating connections or collaborating online with the wider community, but by also drawing on the strengths of both teacher and individual to support and personalise learning (Bolstad et al., 2012; McDowall & Hipkins, 2018).

These curriculum perspectives reflect various conceptions of curriculum or emphases of what constitutes a curriculum. They focus on the goals and content of education. Three conceptions of curriculum are evident, these are; 1. developing students' personally and individually, 2. developing skills useful in society and 3. developing subject knowledge in new contexts. A fourth conception exists but is not as apparent in the curriculum perspectives; that a curriculum should focus on improving society.

"Making decisions about curriculum, is understood better as an exercise in exploring and understanding alternative possibilities, rather than in reaching consensus by excluding alternatives" (Marsh & Willis, 1995, p4). Examining all four conceptions of the curriculum in the context of coding ensures that all perspectives are taken into consideration. Conceptions are derived from ideologies which have distinct components. Therefore, these components can be analysed through students' experiences of curriculum (Adamson & Morris, 2014). This research examines the ideologies of what should be taught and why and interprets how students' experiences of coding may reflect each of the ideologies. Recognising curriculum ideologies in this context aims to assist in understanding the reasons for the curriculum decision to introduce computer programming as a compulsory subject into schools.

Theoretical framework

Five curriculum ideologies were derived from the conceptions; academic rationalism, social and economic efficiency, social reconstructionism, progressivism and cognitive pluralism (Adamson & Morris, 2014). Similar ideologies have been identified across literature related to curriculum theory.

Academic rationalism

Academic rationalism underpins the importance for students to gain knowledge of academic disciplines and therefore a curriculum should be designed to transmit what is already known about these subjects to students (Adamson & Morris, 2014; Eisner & Vallance, 1974). It can thereby enhance students' intellectual capabilities and teach students how to learn the concepts associated with the discipline. For example, in the New Zealand Curriculum the learning areas; English, the arts, health and physical education, learning languages, mathematics and statistics, science, social sciences and technology should deliver time-tested knowledge and content based on these disciplines. In the context of coding in digital technologies, if students emphasise the difference of coding in the curriculum rather than cross-curricular connections we can say that coding sits within the academic discipline of computer science or digital technology. In the past this view of the curriculum was inherited from the 19th century and assumed that knowledge was a given and beyond debate. However, most curriculum theorists today reject this fixed view of knowledge and accept that a modern academic curriculum can be contestable and subject to change (Erekson, 1992; Young, Lambert, Roberts C., & Roberts, M, 2014). Technology is a prime example of a discipline that has changed and evolved over time. Therefore, "academic rationalism includes the perspective that knowledge can be created and the systems for disciplined inquiry are an integral part of the theoretical rationale" (Erekson, 1992, p8).

Social and economic efficiency ideology

Social and economic efficiency ideology, like academic rationalism, is focussed on knowledge and skills; however, it is concerned with these in terms of importance to future employment (Adamson & Morris, 2014; Schiro, 2008). Therefore, emphasis is placed on students applying relevant knowledge and skills. Under this ideology a curriculum is designed to prepare responsible citizens who have what it takes to contribute to the growth of the economy. This was based on the practical abilities of students which were taught through schooling. According to social and economic efficiency, teaching is a "moulding exercise" (Adamson & Morris, 2014, p268). Therefore, teachers enforce what students are supposed to learn, why they are learning it and how they are supposed to learn it. This ideology was noticeable in the very early 20th century. During this time, the New Zealand primary school curriculum consisted of a formal syllabus covering morals, forming good habits and manners, patriotism and health, which were taught alongside academic subjects (McGuiness Institute, 2016). Later this ideology was evident in vocational training. In the 21st century, social and economic efficiency ideology means a curriculum could prepare students to use and create with digital technology, as this is what is essential for economic growth. Therefore, if students believe coding provides the knowledge and skills required by society, and learn to apply these skills, we can say the digital technology curriculum is underpinned by social and economic efficiency ideology.

Social reconstructionism

Social reconstructionism positions curriculum as an agent for social reform and change. With this view of the curriculum teachers would make students aware of social issues and use these as the focus. Social reconstructionism is also known as critical theory. Paulo Friere was one of the most fervent educational theorists of the critical theory approach to education (Darder, 2015). Friere said that education should embrace students as, "political and communal beings" who can not only achieve academically but "with innate potential to transform the concrete conditions that erode their well-being" (Darder, 2015, p63). A curriculum influenced by social reconstructionism ideology would encourage students to work together and actively go out into the community, investigating a problem and identifying solutions. In the mid-20th century,

the recognition of Māori who were socially and economically disadvantaged to other New Zealanders spread ideas of social reform. Therefore, a more socially relevant curriculum was developed (McGuiness Institute, 2016). However, changes in schools focused on inclusive education rather than what should be taught. Technology educators have used activities based on social reconstructionism enabling students to participate in society essentially by improving it (Zuga, 1992). For this study, students would be aware of social issues and use coding for social good. Teachers may incorporate cause-related learning and projects into coding activities, for example, developing an app that solves a social issue. We could also see students going out into the community or teachers bringing the community into the school to demonstrate how coding can be used to resolve social issues.

Progressivism

As a more socially relevant curriculum in New Zealand evolved, advocates were concurrently in support of constructivist beliefs of learner-centred pedagogy to effectively meet individual needs. Learner-centred ideology positions education as contributing to the development of every individual so that they might contribute to the development of society. Learner-centred ideology is also known as progressivism (Adamson & Morris, 2014). A progressive curriculum does not focus on the needs of society. Instead the curriculum is focused on the needs, abilities and interests of individuals. Educators believe that individuals are actively involved in their own learning and teachers are facilitators of this learning. However, it is worth noting that the constructivist theory pre-dates the digital era and therefore it may lack relevance to the digital age (Starkey, 2010). This is because even though learning is active, it is based on students' individual experiences, not the value of what is being learned. In addition, individuals cannot experience everything so other people's experiences would create knowledge (Siemens, 2005). If students see coding as an opportunity to enhance their personal and intellectual development then it could be considered learner-centred.

Cognitive pluralism

The fifth and final ideology used in this study is cognitive pluralism. Like progressivism, this ideology believes students to be active in their own learning. However, a curriculum based on this ideology aims to provide a wide range of competencies and attitudes focused on the process of learning. The curriculum should therefore cater to multiple forms of intelligence like those identified by Gardner (1983). These include; linguistic, logical-mathematical, spatial, kinaesthetic, musical, interpersonal, intrapersonal and naturalist intelligences. The New Zealand Curriculum 2007 identified five key competencies; thinking, relating to others, using language, symbols, and texts, managing self, and participating and contributing. Providing a curriculum based on competencies is believed to be because of rapid change and innovation in society – students need to be able to learn in many ways to cope with environments that are forever changing. "The key competencies take account of these changes – they put today's students at the centre and bring a future-focused perspective to teaching and learning" (MOE, 2017b, para. 3). Therefore, if students view learning coding as developing a range of competencies then they would experience a cognitive pluralism perspective of coding in the curriculum.

Previous literature identifies the contradictions between academic knowledge and learner-centred ideology (Adamson & Morris, 2014; Eisner & Vallance 1974; Schiro, 2008). Studies note the pluralist tolerance of multiple perspectives which may be incompatible and suggest that a curriculum should be organised using the conceptions. This would help educators to understand the conflicting areas and those that are important when deciding on what students should learn at school (Brown, 2006).

Organising students' experiences of coding in the curriculum with curriculum conceptions and ideologies will help to identify conflicting areas and areas of importance to form a basis for discussion.

The New Zealand curriculum

The New Zealand Curriculum Framework which was published in 1993 and the revised New Zealand Curriculum published in 2007, contain many elements of the previously outlined conceptions and ideologies (Brown, 2006). The learning areas such as technology and English are compulsory and provide *academic subject knowledge.* Principles, values and key competencies express conceptions that provide *knowledge and skills useful to society* such as, sustainability, and *skills for personal development*, like confidence. Throughout the curriculum there are elements of *social justice* including equality in education. The 2007 curriculum therefore supports multiple conceptions, and although this can lead to improved instruction, it can also be difficult for those involved in education to agree on the nature and purpose of the school curriculum. This can cause conflict as there is "not one systematic approach to deciding what should be taught" (Brown, 2006, p3). Interpreting the ideological intentions of digital technologies in the curriculum for computational thinking identifies why coding is included in the prescribed curriculum (interpreted policy perspective).

Ideological intentions of digital technologies in the curriculum: Prescribed curriculum

The summary report of the consultation responses written for the Ministry of Education stated that, "the digital technologies curriculum content intended to *strengthen the digital competencies* of learners, so they can *participate, create and thrive in this fast-moving digital world*. It is about supporting learners to develop the *confidence and skill* to not only use digital technologies, but to design and build digital systems." (Chen, 2017, p10). The consultation document also stated that the intention of the new content was "for students to be able *to understand and create digital technologies* to succeed in further education and the world of work. By the end of Year 10, all learners should be *digitally capable – able to use and create digital technologies* to solve problems and take advantage of opportunities. They will be equipped to *apply their understanding of digital technologies to all aspects of their lives and careers*, whatever path they follow" (MOE, 2017a, p.5). These documents were the basis for interpreting the intentions of computational thinking for digital technologies for Years 7 and 8, and therefore coding in the curriculum.

I interpreted these into three generic conceptualisations;

1. Understanding and creating digital technology through core programming concepts
2. Knowledge and skills required to be digital citizens and digitally capable now and, in the future
3. Equipping students to participate in society

The learning progressions in digital technologies represent "the skills, knowledge and attitudes of a digitally capable learner at the end of Year 10" (MOE, 2017a, p18). One of the learning progressions for computational thinking includes coding concepts such as debugging, algorithms, sequences and loops. It is intended that students understand these and use them to create a computer program for example, creating an animated character that moves to music. Being able to understand and create digital technology would therefore provide the knowledge and skills for students to become digital citizens and digitally capable. The purpose of schooling is to educate students to be active participants in the society in which they live (Dewey, 1916). The knowledge and skills learnt through coding are intended to equip students to participate in society. This is by applying what has been learnt in coding to their daily lives and future careers.

These interpretations distinctly relate the conception of social utility where the digital technologies curriculum content is "oriented towards the subjects that are considered most useful for life in contemporary society" (Adamson & Morris, 2014, p265), for example, coding. Social utility is derived from social and economic efficiency ideology where the curriculum is designed to prepare students to contribute to the well-being and growth of the economy. The intentions of the prescribed curriculum are compared against the enacted and experienced curriculum throughout the discussion chapter to evaluate the introduction of coding in the curriculum.

PART 2: CODING IN SCHOOLS

In the previous section I focused on literature relating to curriculum theory and the curriculum perspectives underpinning the introduction of coding. In this section I focus on research that investigated coding in schools to provide context to the introduction of coding and identify existing reasons for including coding in the curriculum.

Coding entered schools in the 1970's. From 1974 to 1985, programming was taught in New Zealand schools through mathematics (Bell, Andreae, & Robins, 2014). Research conducted by seminal authors in the 1980's suggested that coding allowed students to become independent learners and develop skills such as problem-solving (Mayer, 1988; Papert, 1980). The teaching of coding became less prominent within a decade as it lacked relevance to classroom work and students' lives, therefore computer skills such as word processing took over. Coding re-entered curricula globally from 2010 initially as part of computer science, an optional subject at senior level. Computer science is defined as "the study of computers and algorithmic processes, including their principles, their hardware and software designs, their implementation, and their impact on society" (Tucker, 2003, p. 6). There are parts of computer science that involve coding. Coding has been defined as computing language such as syntax or how to write a set of instructions. However, developments in learning to program software such as the visual programming language, Scratch (https://scratch.mit.edu) can take the "chore" out of coding (Wilson & Moffat, 2010). This means that students no longer need to type in the coding instructions and instead can focus on the problem-solving (Buie & Seith, 2012) and creating new knowledge (Webb et al., 2017a).

England was one of the first countries to make computer programming compulsory in both primary and secondary public schools (Balanskat & Englehart, 2015). In 2014, sixteen EU countries were found to have coding in their curricula at varying levels of integration. Denmark made basic coding a compulsory part of the physics, chemistry and maths curricula. In contrast, Slovakia integrated compulsory programming at all levels of school (Balanskat & Englehart, 2015). US-led initiatives such as Code.org and the "Hour of Code" spread worldwide and coding clubs popped up as extra-curricular or holiday programmes (Tuomi, Multisilta, Saarikoski, & Suominen, 2018).

The US states and school districts felt that computer science had driven innovation in every field, but the notion of computer science was quite new (Nelson, Sahami, &

Wilson, 2016). In response to parent demand, along with the computer science community, a complete school grade framework was developed to provide guidance for those looking to implement computer science into their curriculum (Alano et al., 2017; Nelson et al., 2016). In 2017, New Zealand planned to make digital technology a mandatory part of the school curriculum for Years 1 to 10. Digital technology, which included programming, was recognised as a part of the technology learning area from the beginning of 2018.

Reasons for including coding in the curriculum

In the early 21st century, perhaps due to the contemporary and politicised view of coding, research has been conducted on students learning coding and on integrating computer science and coding into school curricula. Research conducted between 2010 and 2016 provided three key reasons for introducing computer science into a curriculum (Webb et al., 2017a). These were; 1. Economic; to produce a workforce that can keep up with technological change and support innovation. 2. Social; to encourage less passive and more creative use of technology. 3. Cultural; to enable "cultural change rather than having change imposed by technological developments" (Webb et al., 2015, p61). These all signal towards knowledge and skills required for the human capital needs of a society and therefore social and economic efficiency views of education.

Knowledge of coding is the only way to fully understand the digital world in which we live (Bell & Roberts, 2016; Burke, O'Byrne, & Kafai, 2016; Kafai, 2016). This is because it takes us beyond what is happening on the screen and allows us to look at what is happening behind it. However, much of this discussion appears to be based on reflections rather than empirical evidence (Sterling, 2016). Student voice in curriculum change is important (Ministry of Youth Affairs, 2009). There is little evidence of students' in-depth views of coding in the curriculum in the academic papers found, but a typical manifestation of the experienced curriculum are student's cognitive processes (Adamson & Morris, 2014). A range of cognitive skills or outcomes are identified from learning to code.

Outcomes of learning to code

Researchers found 21st century skills such as computational thinking, creativity, critical thinking and problem-solving to be outcomes from learning coding (Bell & Roberts, 2017; Hagge, 2017; Moreno-Leon, Robles, & Román-González, 2015; Tuomi et al. 2007; Webb et al., 2017a). Further learning outcomes included; interdisciplinary curricular concepts other than 21st century skills (Falloon et al., 2016; Hagge, 2017; Lambic, 2010; Saez-Lopez, 2015) and coding concepts (Bell et al., 2014; Lambic, 2010; Lye & Koh, 2014; Theodoraki & Xinogalos, 2014). All three can be encouraged through the enacted curriculum and therefore pedagogical design (Popat & Starkey, 2017, submitted to Computers & Education, under peer review).

21st century skills

21st century skills are generic competencies such as critical thinking, collaboration and creativity and are how things are done. The skills tend to be subjective and have an interdisciplinary curriculum focus where they can be used across several learning areas. A 21st century learning approach is compatible with constructivist pedagogy (McPhail & Rata, 2016); it is learner-centred and relevant to the students' experiences. For example, students take an active role in working with others. Coding is a means for practising and developing 21st century skills. I identified two significant 21st century skills from the previous studies on the outcomes of learning coding. These were; social skills and higher order thinking skills [HOTS]. I thought these would support the findings of my thesis.

Social skills

The social skills identified in literature included communication and collaboration, within and beyond the classroom environment. Communication and collaboration are strands in the Australian and Polish computer science curriculum (Webb et al., 2017b) and a core concept of the K-12 computer science framework in the US (Alano et al., 2017). However, it is unclear how these skills were developed through computer science or coding. Communication and collaboration can be encouraged through pedagogical design. For example, students working in small groups to complete coding tasks. They develop collaborative skills through peer feedback which is enabled by teachers who model the phrasing of questions to ask peers. (Falloon et al., 2016). Sometimes communication and collaboration can be organic skills that students demonstrate when learning to code (Fessakis et al., 2013; Kalelioğlu, 2015). An example of this type of communication and collaboration is students' sharing their code with others who need help.

There is yet to be a consensus on the importance of collaborative learning in computer science or programming (Webb et al., 2017a) but pedagogy should reflect the use of technology that transforms learning. To transform learning, pedagogy should move away from what has already been discovered and prescribed as knowledge and focus on knowledge creation and learning through connections. Starkey (2011) developed a creativity model for the digital age which depicted how learning should be modified to include access to information and connections beyond the immediate environment. The model showed that if ideas are shared with people online through social networking this can allow feedback in the development of an idea thereby creating an original product that provides a new reality.

In coding, students should be provided with the opportunities to share their work. Deeper levels of learning occur when ideas are shared and compared with other learners and when communication is two-way. New knowledge is created through this interaction and when feedback is considered and analysed (Starkey, 2011). Visual programming websites such as Scratch have a creative online learning community where students can share their projects. Students can make digital products based on their own interests and talent such as digital stories, maths bulletin boards and music videos and share these with the online programming community (Fields, Vasudevan, &

Kafai, 2010; Hagge, 2017). Learning is personalised and allows students to create, leave comments on each other's work, and to seek and provide help to improve their work. The external audience plays a key role in encouraging groups to deepen their media designs. School leavers in the digital age should be confident in their ability to make connections, create and share knowledge (Starkey, 2012). This view supports a connectivist perspective rather than constructivist, where knowledge is debatable and develops through connections beyond our own experiences (Siemens, 2005).

Higher order thinking skills [HOTS]

The creation and sharing of knowledge also demonstrate HOTS. These require high levels of cognitive processing such as creative and critical thinking (Anderson & Krathwohl, 2001).

Computational and critical thinking both involve problem-solving and use higher level cognitive thinking skills such as analysis, synthesis, evaluation and inference (Kules, 2016). There is a long research history exploring the benefits of learning programming to develop problem-solving skills (Webb et al., 2017a). Problem-solving is recognised as a crucial part of mathematics and in this context the solutions are completely analysable. Students demonstrate enhanced mathematical problem-solving through computer programming (Bernardo & Morris, 1994; Kalelioğlu, 2015; Palumbo & Reed, 1991; Psycharis & Kallia, 2017). This is because schools use coding tasks that offer these foundational skills, for example primary school students solving a series of problems such as navigating an object through a maze in as few commands as possible (Falloon, 2016; Fessakis et al., 2013; Kaleioglu, 2015. These basic skills allow students to apply concepts of orientation, angle rotation, counting and measuring to analyse and solve a problem.

People naturally engage in computational thinking and problem-solving processes in their daily lives when exploring and interacting with the world around them. Computational thinking refers to the thought processes involved in formulating problems and their solutions (Cuny, Snyder & Wing, 2010). As such, a computer is not needed to demonstrate computational thinking. However, it can be difficult to assess this in authentic contexts (Alano et al., 2017; Schoenfeld, 2013; Webb et al., 2017a). Therefore, computational thinking is best developed through computer science and

programming because students can apply the concepts in a tangible way and can check their thinking (Webb et al., 2017a). Programming is no longer exclusively about learning how to write code, but more about understanding the fundamentals of why instructions need to be written in a certain way (Bell & Roberts, 2016). A computer will follow a set of instructions exactly, if the instructions are correct this demonstrates sound computational thinking (Bell & Roberts, 2016; Papert, 1980). If students have a solid understanding of applying computational thinking through programming one could assume that they would be able to use these skills across a variety of examples and subjects (Webb et al., 2017a). Transferring certain problem-solving skills to non-programming domains has only been found when the skills are similar (Bernardo & Morris, 1984; Mayer, 1988). The length of time spent on programming may also influence this (Mayer, 1988)

The curriculum intends that students demonstrate increasingly critical, reflective and creative thinking as they evaluate and critique technological outcomes (MOE, 2007a). This is encouraged through the design of coding activities. For example, students' receiving or giving critique through the online programming community, allows them to use this information to evaluate and make changes to their work (Fields et al., 2010; Hagge, 2017). Another example, is encouraging students to reflect on and evaluate work by finding more efficient solutions than students who had completed the task before them (Fessakis et al., 2013). Similarly, instructing students to test, evaluate and modify code (Falloon, 2016). Teachers are key in uncovering students reasoning and thought processes (Alano et al., 2017) therefore in developing HOTS.

Interdisciplinary curriculum concepts

Visual programming websites such as Scratch can combine two or more academic disciplines, for example, mathematics and coding. Knowledge of mathematics is essential in computer programming (Lambic, 2010) and therefore coding can be used to teach concepts such as angles and distance (Falloon et al., 2016; Fessakis, Gouli, & Mavroudi, 2013; Kaleioglu, 2015). Certain programming activities combine science, technology and principles of art and design (Hagge, 2017). Therefore, coding can be used in subjects such as English, science and art history (Lye & Koh, 2014; Saez-Lopez, 2015). In these instances, increased coding ability along with understanding of

academic knowledge of the subjects learnt through coding is possible. Students can also develop a more positive disposition towards these subjects (Ke, 2014).

Learning coding concepts

Knowledge and understanding of coding concepts is a key educational outcome of teaching programming. There are two reasons why the technical skill of coding is important;

To create tangible computer applications;

Students' can use their knowledge and skills in programming to create something tangible such as games (Haden, 2006; Lye & Koh, 2014; Theodoraki & Xingalos, 2014). Using programming to make games is a highly motivating way for students to learning coding concepts (Theodoraki & Xingalos, 2014). In addition, as the gaming industry becomes more profitable, introducing games programming into the curriculum intends to set students up for employment in the industry (Haden, 2006).

For career opportunities;

Understanding coding concepts is not only useful for employment in the games industry, but for career opportunities in other technology related fields (Balanskat & Englehart, 2015; Bell et al., 2014; Tuomi et al, 2018). Many students will be using computer science in their future careers, not just in science, technology, engineering and mathematics [STEM] but also in non-STEM related careers (Alano et al., 2017). There is a clear link to learning coding for future careers, but this was not necessarily the technical skill of coding but how the process of coding develops 21st century skills (Lye & Koh, 2014; Saez-Lopez, 2015; Theodoraki & Xinogalos, 2014). For example, the technical skill of coding is valuable for careers such as information technology, data analysis, art and design; engineering and science (Burning Glass Technologies, 2016). However, social skills such as teaching others will be in higher demand and other 21st century skills such as; active learning, creativity, mathematical reasoning and critical thinking will be core skill requirements for many industries (World Economic Forum, 2016a).

Pedagogical design in coding

Pedagogical design is important to develop the outcomes identified. For example, students can gain knowledge and skills in programming when applying these in a relevant way (Haden, 2006; Lambic, 2010; Theodoraki & Xinogalos, 2014). Visual programming websites connect students to programming as they focus on creating and sharing a range of programmable media online, for example games, digital stories and interactive art projects rather than just building code. This means that computational thinking has shifted to computational participation (Kafai, 2016). Computational participation expands on the definition of computational thinking and involves "solving problems, designing systems, and understanding human behaviour in the context of computing" (Kafai, 2016, p26). However, the costs of digital technologies and online services were identified as the biggest barriers to their use by schools (Johnson, Wood & Sutton, 2014). To offset the costs of the purchase of digital tools schools often use free online resources where possible. Schools that only use the free visual programming resources available, may still connect students to programming in a relevant way but there could be a limited range of coding concepts learnt. This is because free resources tend to be too basic and students' can only reach a certain point after which there is a financial cost (Curran, 2017).

Another aspect of pedagogical design that is important is the way students work on or are encouraged to work on coding tasks. If students do not work in small groups or use an online learning community, collaboration and connections may not be evident. On the other hand, if students are encouraged to reflect on, test and evaluate their work this can encourage deeper levels of thinking. Academic knowledge other than coding can also be achieved through pedagogical design. For example, when coding is integrated into subject areas such as mathematics or art history (Fessakis et al., 2013; Sáez-López et al., 2016).

A blended learning environment is important in learning coding. A blended learning environment enhances communication and interaction with others leading to better participation and learning outcomes (Pellas & Peroutseas, 2017). Coding combined with teacher guided instruction and group work allows students to apply mathematical concepts for problem-solving (Lambic, 2010) and although coding provides opportunities for self-directed learning, explicit instruction should be provided when

attempting to integrate course content into Scratch experiences (Hagge, 2017). Overall, the evidence for learning with digital technologies supports that teacher scaffolding is important. Without teacher instruction, time is consumed with learning to manipulate new digital tools rather than meeting the intentions of the digital activity (Evans, 2014). This is because not all students are skilled technology users as stereotypically believed.

Literature review summary

The literature review identified two main areas important to this research;

1. Previous research on coding investigated reasons for, and ways to teach coding to students. The main reason presented was to develop 21st century skills important in life and the workplace. Skills on their own cannot be an adequate basis for a curriculum (Young et al., 2014). Skills tackle *how* students are doing things rather than *what* they are doing (Rata & McPhail, 2016). It is the *what* that takes students beyond their own experiences and empowers them to challenge their existing ideas. This is "how knowledge is theorised in the powerful knowledge curriculum design type" (Rata & McPhail, 2016, p59). Powerful knowledge is a synthesis of two competing perspectives or ideologies of the school curriculum; progressivism (based on individuals) and academic rationalist (a traditional view based on subject disciplines). In addition, globalisation and connecting beyond the classroom environment is a key concept for education in a digital age (Starkey, 2011). Curriculum ideologies and curriculum perspectives are used in this study and form a basis for the discussion.

2. Because of the issues and challenges associated with coding in schools (Webb et al., 2017) we should learn from the past by looking at why students are interested in programming, under what circumstances they do it and how (Kafai, 2016). What seemed to be lacking from the relevant literature found is, why or if, students thought it was important to learn how to code and in the New Zealand context how the experienced curriculum aligned with the intentions of the new digital technologies content.

Examining how students use coding and how it may be useful in the future as well as what students believe they learn as they code and why, assists in understanding students' experiences and perspectives of coding. Investigating how these align with the intentions of digital technologies in the curriculum aims to assess the overall value of coding in the curriculum. The research questions to meet this aim are below. The significance of my findings considering the existing literature can be found in the discussion chapter.

Research questions

The aim of this research is to explore students' experiences and perspectives of coding and its value in the curriculum. This research uses a narrative inquiry and the research questions to meet this aim are;

1. Why do students think coding is taught in school?
2. Do students use coding outside of school?

   If so, how, or where? If not, why not?
3. Why do students want to learn how to code?
4. How do students think coding might help them or be useful?

These questions investigate current experiences of coding and look forward to the possibilities coding could bring, particularly at a time where coding is becoming a more prominent part of the national curriculum (MOE, 2017a). The research questions intend to assist in understanding the unique positions, experiences, and perspectives of coding through those who have lived it. The answers to the questions, guided by curriculum ideologies, intend to explain students' understanding of computer programming and their perceptions of coding in the curriculum.

CHAPTER 3 METHODOLOGY AND METHODS

Methodology

Aims and objectives

The aim is to explore students' experiences of coding and their perspectives of its role in the curriculum. There was little in-depth evidence of students' perceptions of coding in the curriculum in existing research. The research focuses tended to be on the outcomes of learning to code or ways to teach coding. For example, gaining learning outcomes such as problem-solving or using a blended learning approach. The primary aim of this thesis is for students to share their overall experiences of coding and their perspectives of the importance of these rather than measuring (pre-determined) learning outcomes. Curriculum ideologies are imposed on the study as a theoretical framework. This approach provides a different context to research found on the topic thus far.

Research paradigm

The primary research follows an interpretivist paradigm. Interpretivism aims to understand, whereas other paradigms such as positivist, aim to generalise, predict and control. (Johnson & Christensen, 2017). The research endeavours to understand students' experiences of coding and to ascertain how these aligned with curriculum ideology. Rather than starting with a theory, the research gathers evidence of student experiences and generates a theory. Therefore, meanings of the data emerge at the end of the research. An assumption of interpretivism is that, individuals attach their own meanings to things and act towards them based on these meanings, these may also be influenced by other people (O'Donoghue, 2017; Wilson 2013). Researching students' "meanings" of coding in terms of curriculum ideology provides a different perspective of understanding the value of coding in the curriculum.

Theoretical perspective

The theoretical perspective, hermeneutic phenomenology is used as the interpretivist approach. This is because, hermeneutic phenomenology finds meaning in experiences, (O'Donoghue, 2007) and therefore, students real-life experiences of coding in the curriculum are studied. Subjective interpretation is the only way to understand the phenomena under observation (Wilson, 2013). Subjectivity is sometimes seen as being invalid or inadequate because results would more likely to have bias. However, when investigating coding in the curriculum through students' perspectives it is necessary to take account the subjective meanings of the students' experiences. In interpretivist research this is relevant, as when trying to understand the social world through personal experiences, we are trying to make sense of everyone's subjectivities when making judgements. However, this does not need to be too rigidly followed as some objectivity is always good as our own biases are a problem (Wilson, 2013).

Qualitative approach

Qualitative research is used to collect the primary data for this study, as subjective understanding is at the core of qualitative research. Using qualitative research seeks understanding of coding in the curriculum rather than quantitative data which looks at trends or relationships between coding and outcomes. A narrative using case study design can be used to inquire into student's experiences as a storied phenomenon (Johnson & Christensen, 2017). Narrative inquiry seemed to be the appropriate qualitative research method to portray a group of individuals' experiences of coding. Several studies that use hermeneutic phenomenology in education use narrative as their method of research (Friesen, Henriksson, & Saevi, 2012). Through students' experienced descriptions and narratives, this research can highlight some of the aspects of coding which may be overlooked in previous research. Hermeneutic phenomenology can provide teachers and others interested in coding in the curriculum, different knowledge and deeper understanding of what coding means to students.

Research Method

Participants

Existing studies particularly in New Zealand schools, predominantly focus on students in Year 1 and 2 or senior school students (Years 11 and 12). Some schools in New Zealand introduced coding into the curriculum at Years 7 and 8 so the participants in this study were students in Years 7 and 8. Schools were selected through purposeful sampling. This included; those geographically accessible to the researcher, schools willing to take part in the research and schools that included coding as part of the curriculum at Year 7 and/or Year 8. The sample did not only consider convenience but sources that were expected to yield the depth of information required to meet the overall aim. A search was completed using site: school.nz "coding" to identify schools that potentially taught coding. Reviewing these websites identified schools where students had learnt coding at Year 7 and/or Year 8. Eight schools in the region were identified from this search.

Ethical considerations

The research conformed to the Human Ethics Policy of Victoria University Wellington [VUW] and the VUW Human Ethics Committee approved this research [Reference number: 0000025436]. Principals were contacted after ethics approval had been gained. Two schools agreed to take part. The Principals consented to the study and put me in touch with the relevant teachers to discuss the research. These schools were both co-educational but quite different in their demographics and curriculum design. The teachers also agreed to take part and distributed information on my behalf to the students who they thought would be suitable for the research. Six students from each school returned consent forms signed by themselves and a parent. Therefore, one focus group of six participants was held at each school. Six participants were deemed enough to generate conversation, engage and give all students the time to speak. Using case studies employing narrative methods means that only a small number of participants in each case is required to identify the theoretical concepts of the curriculum and for intricate analysis (Wells, 2011).

Data gathering

Within the interpretivist paradigm, the researcher can use guided semi-structured or open-ended questions with people in their own surroundings, aimed at understanding a phenomenon (O'Donoghue, 2007). Therefore, in this study a focus group was held in the students' school setting. Focus groups were chosen due to the age of the students of interest as it was seen to be less intimidating than an individual interview. A focus group was also thought to allow experiences of coding to unfold naturally. The focus groups were guided by questions (Appendix 1). The questions were piloted on a student, who was not part of either focus group. The pilot testing helped to confirm that some questions were unnecessary, and some words needed changing or questions rephrasing to be better understood. The guided questions were amended accordingly. The focus groups were recorded using two digital audio recorders. In addition, I took notes so there were points where I could clarify what the participants said and where I checked the accuracy of my understanding. Additional questions to the interview guide were asked in a flexible manner, for example, to build on a comment made. There were points during the focus groups where I asked students to break into smaller groups of two or three to discuss and write down or draw ideas. This was to engage the students further. I retained these pieces of paper which were used to support the findings. Each focus group lasted approximately 60 minutes. I transcribed the narrative data shortly after the focus group was held. I also collected data to verify the enacted curriculum. Although teachers were not interviewed, I asked if they would like to share any relevant documentation about the teaching programme for coding, and they did. One teacher sent an email regarding the resources used and the other provided worksheets and discussed the resources they used. I also found information about the coding programme for Years 7 and 8 on their school website however this may not have been as reliable and up-to-date as the information received first-hand, therefore it was only used if it supported the information provided by teachers.

Data analysis

Students narratives were first arranged to answer each of the four research questions. This was to convert the students' experiences into condensed, displayed analysable text that could be used to draw and verify conclusions (Miles, Huberman, & Saldaña, 2014). The research information was combined within focus groups rather than individual interviews and confidentiality was protected throughout the process. In vivo coding identifies short phrases from transcripts (Miles et al., 2014). This was the first stage of data analysis where phrases were identified to answer each of the research questions. Then evaluation coding was used. Evaluation coding "assigns judgements to the merit, worth or significance of programs" (Miles et al., 2014, p101). For the second stage, I evaluated the students' experiences of coding in the curriculum to identify areas of importance. For example, their reasons for including coding in the curriculum. Abductive reasoning was used as it adopts an interpretivist ontology therefore it was approached from the viewpoint of its participants. There are two common ways to approach interpretive research (Chamberlain, 2006). One approach is for the researcher to have an open mind and reactively obtain raw data from the participants. This however runs the risk of failing to generate useful data as the information may not be able to be related to any situation but the one being described. Due to this risk, this research adopted the second approach which uses a theoretical framework, in this instance curriculum ideologies, and imposes it on the research, for example, students' experiences of coding. This means that once data is coded it can be approached in a logically coherent way (Chamberlain, 2006). I used pattern coding for the cross-case analysis to summarise the data into themes of each type of curriculum ideology. I aligned students' experiences and perceptions of coding in the curriculum with curriculum ideologies to identify and understand the contradictions and areas of importance. These formed a basis for the discussion. As the scope of this study was small I coded data into themes manually rather than using CAQDAS (Computer Assisted Qualitative Data Analysis Software).

The process of abduction begins from the data then seeks to explain what is found by using theory and existing literature (Flick, 2013). This means that abduction occurs and then deduction and induction come after (Chamberlain, 2006; Flick, 2013). In this

research, first new ideas were generated (abduction), these were evaluated (deduction) and then justified (induction) to form a most likely reason or explanation.

Research validity

Verbatim the students' language, known as a low-inference descriptor, used in findings improves the interpretive validity (Johnson & Christensen, 2017). I checked the understanding of what was said during the focus groups and I used verbatim when transcribing to ensure the accurate portrayal of what was meant. This research was conducted with twelve different participants in two different schools, over two focus groups, therefore the data gathering and analysis process could be replicated in different contexts. The methodology was transparent and interview questions included in the appendix. Although additional questions were asked to explore areas further when required, another researcher would be able to repeat the research method. Subjectivity was inevitable in this research as implicated by hermeneutic phenomenology (Frieson et al., 2012). Researchers of hermeneutic sensibility would not want to set aside their experience and understanding. Instead they would want to recognise their pre-existing beliefs and how these might impact the research process and findings. This is termed researcher reflexivity, where there should be continual self-awareness and reflection of your own assumptions and biases and recognition of any limitations (Johnson & Christensen, 2017). A final strategy to improve the validity was peer review of evidence which provided useful challenges and insights. Three reviewers read the thesis in its entirety and made their own notes of questions and some suggestions for improvements. I went through each of these and made amendments as necessary.

Limitations

Generalisation is not in the nature of qualitative research but a greater insight into student understanding would be gained with a bigger sample. In addition, multiple-cases can improve the validity and trustworthiness of the research. However, in qualitative research "an underlying theory is being matched rather than the larger universe" (Miles et al., 2014, p58). Therefore, using two different schools allowed for similarities or differences in aspects of coding to be identified but these could not be generalised to the entire population. Facilitating focus groups in two schools with twelve participants meant multiple data sources and some diversity of experience to improve the validity of the research. Ideally more than one researcher should conduct the primary research to further improve validity by replicating the method in a different context, I was the only researcher and I was a novice. However, I did consult and seek advice from my supervisor who was able to mentor me through the process. There is the potential to mismatch the researcher's reality and that of the participants by forcing a theoretical framework on to a social situation (Chamberlain, 2006). This could lead to bias and inaccurate data coding; however, it does eliminate the risk of findings that do not relate to anything. Thoroughly examining information aimed to rule out alternative explanations than the ones provided.

The students were selected by teachers therefore it is likely to be a biased sample as all participants are enthusiastic and have a considerable interest in computers and/or coding. In addition, the "group effect" could impact the findings. This is if participants are influenced by what others in the focus group said before them. However, the opinions of "enthusiastic" students are still important and provide in-depth information. There are also benefits of the group effect which observe consensus and diversity amongst participants views. The methodology is based on hermeneutic phenomenology, data gathering through narrative inquiry and data analysis using abductive reasoning. However, partly due to using a theoretical framework and the qualitative nature of the research, it could be argued that the approaches used are not in their purest form. It is only natural in qualitative research to no longer adhere solely to the boundaries of one philosophical approach and that this makes for more deliberate and diligent analysis (Miles et al., 2014).

Chapter 3 summary

I conducted a qualitative narrative inquiry to provide an insight into students' experiences and perceptions of coding in the curriculum. I held two focus groups consisting of six purposefully selected participants from Years 7 and 8. Students were asked open, semi-structured questions and narratives were constructed for each case study to answer the research questions. A cross-case analysis was used to summarise the findings into themes of curriculum ideology. Every effort was taken to ensure the validity of the research by using the students' own language and checking the accuracy of my interpretations when possible. The methodology was transparent.

CHAPTER 4 FINDINGS: NARRATIVES

In this chapter, student narratives were pooled together for each case study and provided insights into students' perspectives and experiences of coding in the curriculum. Each case starts with a description of the school profile, the participants and details of the enacted curriculum. Next, student narratives are used to answer each of the four research questions. In the second findings chapter there is a cross-case analysis applied to the theoretical framework. I took care in ensuring the confidentiality of the school, the teacher of coding and the participants.

CASE STUDY 1

Introduction

Case study 1 is a full primary school, with students from Years 1 – 8. It is in a rural town and has a total school population of between 150 and 200 students. The school is classified as a decile 2 school. Decile is a measure of the socio-economic position of a school's student community relative to other schools throughout the country. Decile 1 schools are those with the highest proportion of students from low socio-economic communities. The students are predominantly Pākehā and around a third of the students identify as Māori.

The school curriculum has a major focus on reading, writing and maths. Other learning emphases include Te reo Māori, inquiry and PE/Health. Students are organised into year group learning hubs with two teachers across Years 6, 7 and 8. Each child has access to a Chromebook for learning in class. There are no specialist teachers for classes, this means the students' own class teachers deliver the school curriculum. Students have visited the local College for technology. During Term 4 2017 the learning hub focused on digital technology and science. This included video/movie editing, green screen recording technology, coding, physics and friction. In Term 1 2018, if students had finished their reading or writing work, they could complete 10- 15 minutes of online coding activities or tutorials at the end of class as a "treat". Focus groups were conducted near the end of Term 1 2018.

There were six participants in the focus group interview, three female and three male students and an equal number of Year 7s and Year 8s. Students talked about their past experiences of coding in the curriculum which included experiences from Year 6. Students in case study 1 seemed excited to share their thoughts and experiences of coding with me. They were asked to speak one at a time and therefore raised their hand when they had something to add. Students were also called upon if they had not had an opportunity to speak. This meant that each student participated equally in most parts.

Narratives

Students views on why coding is taught in school

Students first learnt coding at school using the free online programming language Scratch which was used to create games and animations. After Scratch, students used other visual programming websites including; Tynker, Hour of Code and madewithcode.com. These provided introductory coding activities and involved "drag and drop" blocks which manipulated the way things looked and/or sounded, showing the written code along the way. Students used these to create a variety of digital products, including making music or icons that were used on a phone. There were some more advanced projects that students completed where they learnt how to program a character to navigate through a game. Coding was a separate learning focus and a cross-curricular approach was not used; "we haven't used (coding) in maths, science or English, we don't do it every day"

There were three main reasons why students in case study 1 thought coding was taught in school. These were; to provide a basic understanding of coding and therefore how digital technologies work, to enable them to create games or apps and to prepare them for future employment prospects.

Students saw coding as what made technology work and everything that required a computer required some level of coding.

> *I think coding is pretty much a source of life for computers. So, you have to code a computer to work, with numbers and symbols or letters otherwise it wouldn't work, and you wouldn't be able to use a computer or the system itself.*

Students were conscious that technology was constantly evolving and updating; it was a big part of their lives. Because of this they felt that coding was taught so that they understood the basic concepts. Students were asked to split into pairs and write down or draw on paper what they had created using coding in school (Figure 1). They included making games or other digital products such as emojis, neon dresses or Snapchat filters. Students felt that learning coding meant that they could make digital technology work, "say there was a game, you have to code something to make it do something"

Figure 1. Students drawings of what they have created in school using coding

The discussion then moved to the importance of coding for employment as being a reason that it was taught in school. Students felt that most jobs required coding or that knowledge of coding would be helpful or useful for jobs and therefore being taught the concepts now would mean "you are prepared for when you are older". When I probed further about the types of jobs coding would be useful for they were able to relate this to designing games (Figure 1), graphic design or other examples of creating technology related products.

> *There's tons of jobs in the future that aren't there yet, that could include coding and stuff. Jobs are upgrading… sometimes computers seem like they are taking over humans' jobs, it's kind of like scary, they make cars, what they are doing now, with robots and stuff, computers can drive cars.*

There was a consensus that knowledge of coding would provide more job opportunities. This included the opportunity to change jobs. They all saw coding as fun and said knowing how to code would allow people to change their profession to one which included programming if they wanted. Students were also aware that if coding wasn't taught in school they may have little or no knowledge of it or find it harder to learn later.

Overall, students felt that coding was taught in school so that they could gain an understanding of the basic concepts of how digital products work. This would allow them to create digital products which would provide a solid foundation for employment in the future.

How do students use coding outside of school?

Four out of six students in case study 1 used coding outside of school. One student who did not use coding outside of school said;

> *I like doing it and I enjoy it, but it's not something I sit and home and always do.*

This student perceived learning to code as important for future employment but despite this they did not see coding as a school subject, they saw it as something fun they got to do in their time outside of what they considered to be more academic disciplines such as reading, writing and maths, "it's like a game, you get to code". This was because coding was used as "a treat" when they had finished their reading or writing.

Those students who did use, or even talked about coding outside of school, were often encouraged to do so by family members. Students said that their parents explained coding or supported them at home to create video games or animations instead of just playing or watching them. One student would watch videos on how to build robots and their mum bought them a kit so that they could make a robot which they programmed to bring it to life. Another student who was a keen gamer was urged by a family member to create a game outside of class;

> *My mum and I did a little activity, we cut the screen in half or something like that and we made two little games and we each had, I don't know how long it was, but maybe fifteen minutes to make a game or animation and she did one first, showing me how to do it. Eventually when I finished I made this rocket*

game where we had to dodge all these little white meteors and stuff. I normally use it to make games.

Four of the students wanted to learn more about coding. They thought that coding at school provided basic knowledge, but wanted to take this further and would therefore learn more complex coding at home;

For me personally, I've been trying to learn how to do written code which is a bit more confusing than the drag and drop code, so I was learning that for a bit. I first found coding just playing games…So, I was like I want to learn how to do this instead of just the simple Scratch or just that.

Therefore, students were mainly using coding outside of school for gaming however instead of just playing games they were making their own. This was supported by family who would encourage their children or create digital products with them.

Why students want to learn how to code

The main reason that students wanted to learn how to code was because they saw other people making digital products such as games or animations on YouTube and they were inspired to try and make these themselves. Students agreed that programming could be used to create anything when using technology which empowered them to be creative. Four of the students had a genuine interest in how digital technologies work, particularly how to make characters move in a game and would therefore seek information on the internet that would help them learn more. Coding in school was limited to specific tasks and therefore students would find information, mainly online but sometimes from other students, that were more suited to their needs and interests;

I learnt movement through watching YouTube videos. Because I was like "how do I do this thing?" so I watched a YouTube video and learnt that and also physics, it's not just you can press the space bar and jump up, if it were to be in a game you would stay there in mid-air and not drop down, so you have to figure out so they make it so they can drop down onto the ground again.

In some instances, the reason students wanted to refine and develop their skills was because they envisaged a career that involved programming. Half of the students in focus group 1 recognised that they wanted a career where they would need coding knowledge. One student wanted to use coding to create an app, or software or social

media websites like Facebook or Instagram. However, mostly the students wanted to make games for other people and considered this to be “pretty cool”.

> *Technically nearly half of my life I’ve been interested in video games and how they work and all of that sort of stuff, so I’ve always wanted to be a game designer and stuff like that but the only way I am going to learn how to do that is to learn at least the basics of coding. Making games would be a good job.*

The students who did not see a career in coding still wanted to learn to code as they found it enjoyable. Students liked that the products they created were individual. They could make them based on their own interests and no one would have the same design.

Figure 2. Student drawing of a personalised emoji

A student explains Figure 2;

> *We did this Made with Code thing and you had to make your own emoji and you could customise it to what colour hair. If it was up or down and then you could also add some of your interests, like I had music. I added, because I want to be like a doctor, I added the thing that they have around their necks and then I also want to be an artist so I want to do that stuff, so in my hand I had like an artist pallet and you can do some writing and stuff like that too, it’s really cool.*

Students also enjoyed the problem-solving challenge of making code work. From the discussion it was evident that students didn’t necessarily use instructions or code that

worked the first-time around, but they would eventually get it to work through experimentation. I explored how students learnt to code. Students felt that they learnt it themselves or through watching others and on occasion would ask other students for help. They agreed; “we work individually but help each other”.

> *We kind of just had to experiment because the teacher told us to go on it and have some fun, so we kind of experimented with everything to do, and we watch videos of other people doing it.*

However, there were also limitations of learning through experimentation as more complex processes were too difficult for students to learn on their own or with the resources available. A student said that they looked at other peoples’ games thinking it would be easy to reproduce but found that this was too hard to complete on their own.

Students wanted to learn how to code because they were interested in how games or animations worked. They saw others creating these which inspired them to do the same. For some students this is because they envisaged a career that would require knowledge of coding, for others, it was because they enjoyed coding and the challenge of getting code to work. Students worked individually and used online resources or help from others. They did not acknowledge teacher guided instruction.

How students think coding might help them or be useful

> *It’s kind of a skill you can pass on to other people and teach them to do it so that they don’t have to go through all the stuff you have, when you can just teach them.*

Whilst students said they taught themselves to code through online resources, they felt this was a time-consuming process. They were confident that once they knew the basics they could easily teach others what they knew about how to code. They thought this useful because other students wouldn’t have to go through the trial and error or time spent looking for how to do things that they went through. Learning individually through experimentation and personally finding the most suitable resources was regarded as a challenging process. One student commented, “I tried to learn how to do written code, but I couldn’t find any apps to just learn how to do the coding”.

Students also recognised that having learnt coding at school it was now easier for them to understand how apps or games worked. Therefore, they were better at using them or found them easier to use. Students also felt that knowing generic coding would make

them more confident to fix problems with computers if required and that coding was useful for coming up with new ideas and creating them (Figure 3).

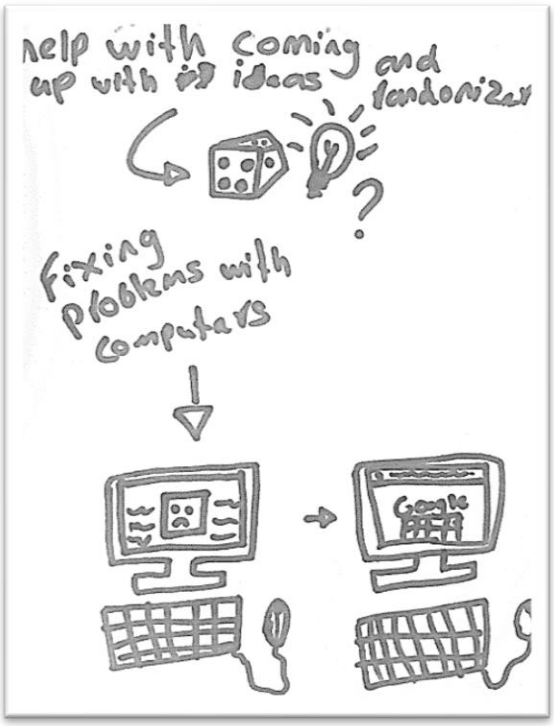

Figure 3. Student drawing of how coding could be useful

However, there were times where the lines between learning code to create, and generally using technology (that perhaps wouldn't exist without code) were blurred. Some of the comments from the focus group demonstrated that students couldn't always distinguish between learning coding and using a computer application.

> *We use an app called Seesaw and we record our writing to say what we made and post it back to our parents who can listen to it, because I don't like talking in crowds, so I can do that instead.*

Another student referred to an activity completed when using madewithcode.com (Figure 4). They used blocks to design a dress and wrote down that this meant coding could help in a designing job, when asked to clarify they referred to becoming a fashion designer.

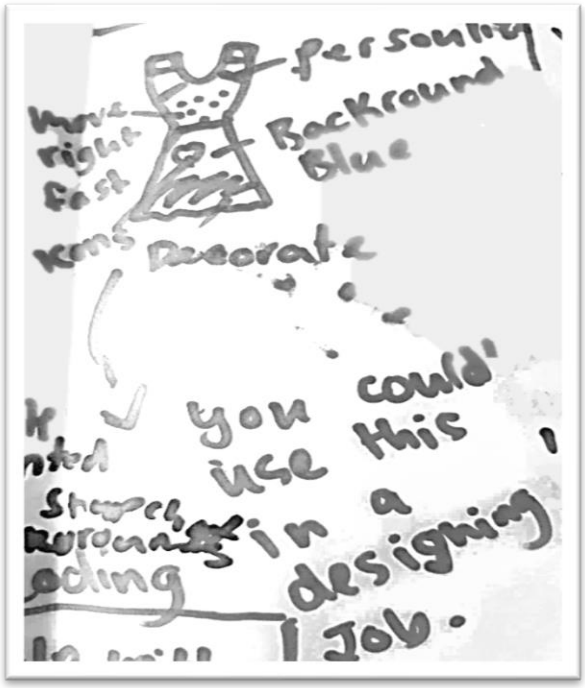

Figure 4. A student view of coding to design clothes

> *When you create dresses and coding and that, this could help you be a designer like designing clothes and stuff.*

Despite this, some students recognised that the skills learnt in coding could be used or applied in other areas. “I’ve just been adjusting a lot of the skills you need since I’ve started learning graphic design”.

Student views of how coding may be helpful or useful to them were mainly to understand how technology worked. This allowed them to use it more efficiently and pass on what they had learnt so that others could also learn from or use this knowledge.

The essence of case study 1

Students saw coding as a way of understanding how digital technology worked. They believed this to be useful so that they could create their own games or other digital products or maybe even fix broken technology. In the future students felt that knowledge of coding would be beneficial for job opportunities as most jobs would require people to code. Students genuinely enjoyed learning coding and wanted to learn more about it to create products.

> *I would like to be able to do that written code stuff and I would like to make games and make things for other people, like make games other people can play*

Some students actively sought out other online resources suited to their needs and interests to independently learn more. However, they found more complex coding hard to learn without help.

> *Games are way easy to make unless you are making them yourself.*

Students were encouraged or inspired to learn more about coding by peers, people online and family members. They were also confident that they could teach others to code.

CASE STUDY 2
Introduction

Case study 2 is an intermediate school comprising of students in Years 7 and 8. It is in an urban area with a multicultural population of between 600 and 700 students. As a decile 8 school it has a much lower proportion of students that live in low socio-economic households when compared to case study 1.

The school curriculum encompasses eight learning areas defined as prescribed by the New Zealand Curriculum (English, the arts, health and physical education, learning languages, mathematics and statistics, science, social sciences and technology). Technology is a specialist subject and all Year 7 students take part in all areas including computer technology. In Year 8, students opt whether to take computer technology or an alternative subject within technology or the arts. For the purpose of this study, it was assumed that computer or "digital technology" learning would be a compulsory subject for Years 7 and 8 when the prescribed curriculum is fully implemented by 2020.

There is a specialist teacher for computer technology and coding is taught as part of this course. For those who take computer technology at Year 8, coding and STEM (science, technology, engineering and mathematics) are a major part of the programme for 2018. There is also a voluntary coding club which was held at lunchtime for those students who want to participate. Students have access to computer suites and a range of tablets including iPads and iMacs.

There were six students in the focus group discussion, which included one female and five male students, all in Year 8. All six participants had elected computer technology as one of their specialist subjects in Year 8. The specialist teacher of computer technology had mentioned the group included some who love coding and others who love computers and everything computer related but were not fans of coding, these students were not identified individually. The focus group was conducted near the end of Term 1, 2018. All students had first learnt about coding in their previous school in Year 6 but during the focus group they mostly talked about their experiences of coding in Year 7. It was clear from the start that students were all keen to contribute to the discussion as they all raised their hand to answer the first few questions. To ensure all

students contributed equally the focus group was conducted on a round table basis. On occasion students would provide further insight at a point where they felt they had more to add or when asked for clarification.

Narratives

Students views on why coding is taught in school

In case study 2, free websites were used for coding (Hour of Code, Studio Code, Code Combat and Khan Academy). Students used these to learn how to make games and navigate characters to make specific patterns on the screen. However, before students could use computers to apply coding concepts they were given worksheet tasks to manually demonstrate computational thinking. An example of such a task was where a machine that served juice had some cups coming out the wrong way around. Students had to use a set of instructions provided to turn all the cups the right way up in as few steps as possible, for example, one step only swapped the position of the 1st and 4th cups whereas another turned the 1st and 5th cups over. Students manually turned the cups and then wrote down the steps taken. This worksheet was used to test problem-solving strategies, another explored prime numbers and algorithms (encryption).

Students in case study 2 thought that coding should be taught at school because they believed that everything required a computer or the internet, which required coding to operate. The students seemed to feel strongly that without code they would not have a lot of the resources they used at school and at home. Therefore, due to the perceived enormity of coding they felt it was important to learn how to do it;

> *Coding is in our everyday lives like in the recording thing* (referring to the voice recorder)*, there is code in it. And without code we would probably be like cave people…like so maybe get a worksheet and its printed out, that printer needs code to work and the website where we printed that educational paper needed code to actually create it so that's why I think code is quite big.*

Two students felt that coding might be taught in school because future jobs would require coding. However, most of the other students thought that learning to code at school was not necessarily for jobs. For example, although they said it was used in many areas such as programming aircrafts to developing a webpage, they did not

associate coding (in these areas) as a profession. Instead students thought that they were learning more of a skill that they could use if required (Figure 5).

Coding can be used in many different ways because it's not just like for jobs, it can also be used on maybe you can use it for stuff like with aircrafts or something. It can be used in many different ways. It's really hard if you don't know what you are doing so that's why we have to learn.

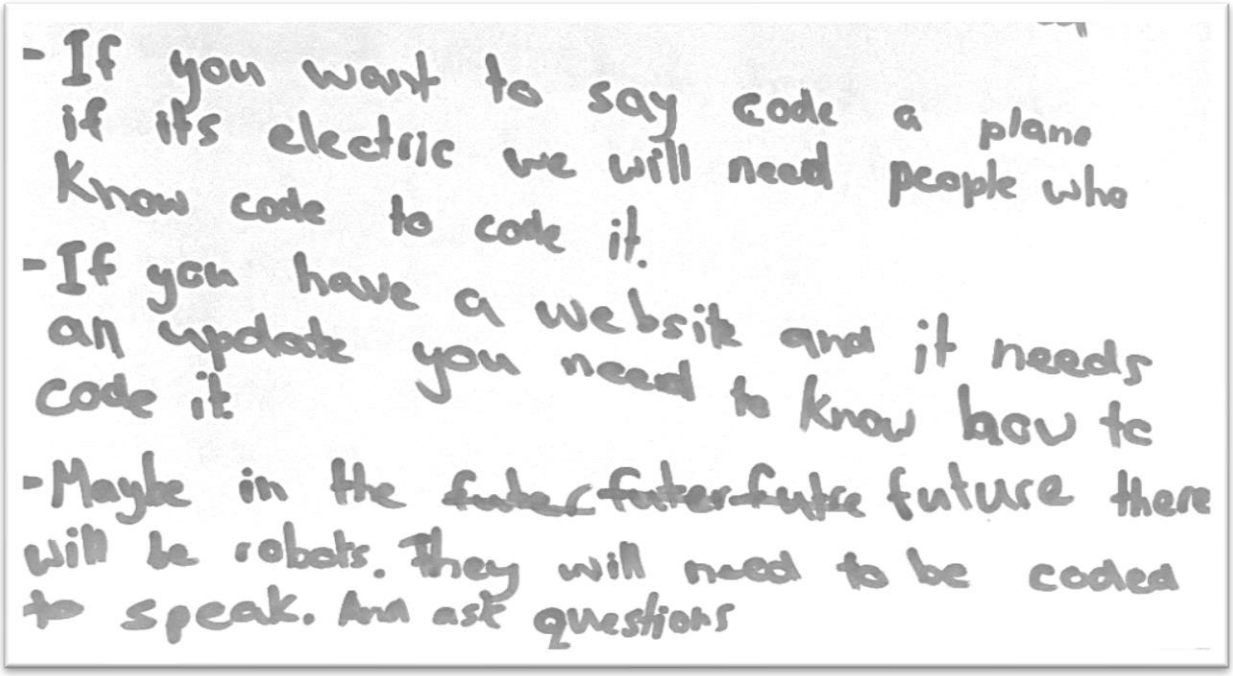

Figure 5. Student views on how they could use coding

Students also agreed that they should have some knowledge of coding;

> *It gives you a good understanding of how things work and that all these things that people make they are all made by coding and you think wow cool.*

In case study 2, coding was not only a skill that could be used in future careers. Students' thought coding was a skill they should learn because they were dependent on technology that was coded, therefore they should have some knowledge of how to make this technology work.

How do students use coding outside of school?

All six students had used coding outside of school. They did this by going on the visual programming websites that they knew from school. For example, Scratch and code.org to "have a play" on the resources available and to learn more about coding. In particular, Code.org allowed students to make animations and games with characters that move and to make simple web pages. Students enjoyed using the game lab or making a sprite move on the visual programming websites and as in case study 1, two students said they were encouraged by family members;

> *I was addicted to it when I was at home and I literally got home and had my snacks, did my homework and then I was straight on it, on Scratch and my mum and dad liked to see what I came up with so basically I was on it 24/7 if I was able to I probably would be but I'm not exactly allowed up past half past eight.*

Another student said that their parents got them into coding on Khan Academy because of the demand for coding in the labour market, "because basically everything is made out of code". This generated a discussion on how relevant coding was now;

> *What's the percentage of people, all the population, in the world, it's like 1 billion out of 7 billion, 1 billion only do code when we need half the population doing it…we learnt that in computer tech.*

However, students felt that more people were using coding than in the past and getting a feel for what it was. The consensus was that people used coding every day when they went onto websites. Although people aren't specifically writing code, they were using webpages which were made using code. Students said that if they learnt more about how to code they would be able to make a webpage that people would use because everyone uses websites.

Students used coding outside of school by going on the visual programming websites that they were aware of and played around with these to get a feel for coding and learn more but also because they found it fun. Some students were encouraged by their parents.

Why students want to learn how to code

Students wanted to learn coding as they enjoyed the process of writing code and found it interesting. They also saw coding as educational in that it allowed them to learn new things and that it was challenging.

Students in case study 2 liked that learning coding gave them the freedom to be independent in their own learning and that it was "not like writing, reading or any other subject you do at school, it's something different". The notion that coding was "something different" was a common theme in the student discussion;

> *I enjoy doing coding because it's something else, who knew that today we would have WIFI and stuff because my house it's a real tech house, without WIFI I don't know what we would do, and so we are pretty much all on technology and stuff and yeah so it was something new rather than like Instagram and Snapchat and social media and stuff, yeah and it's fun.*

Despite enjoying being able to choose their own activities and independently play around and understand concepts themselves, they recognised that there were times where they needed help;

> *So we had to choose a game and I chose the Frozen game and if we needed help we could go to somebody who knew how to code and all that and we could ask them but if we don't have anyone there, we could go to a teacher and they could guide us to solve the pattern and then once you've got it you just keep going until you have finished.*

Students mentioned that sometimes the teacher explained new concepts to them. One student in case study 2 thought coding was better being taught rather than learning how to do it on your own because if you got stuck you would be able to ask a peer or the teacher.

There was another student who wanted to become a computer programmer in the future, and this was the reason that they wanted to learn how to code.

The main reasons that students wanted to learn how to code was that they found it interesting and they could learn new things, students also enjoyed being able to figure

things out themselves but recognised that they also needed help from other students or the teacher.

How students think coding might help them or be useful

Students felt that coding was a useful skill as it gave them some understanding of how digital technology worked. They felt that as a skill it could help them in the future if they needed to use it or in employment;

> *It's useful for our future like what's going to happen, it may come up, like some of it may not but it's there just in case when we need it, say if I worked somewhere and I had to programme a computer, if I didn't know what the heck I was doing but if we learnt that at school and then we could use that knowledge in the future, so what we learn now has a big impact on our lives.*

Although most students did not believe coding was taught in school because of future employment. There was a sense that students believed that most, if not all jobs, in the future would be technology related and that every job would need code. Students imagined that there would be robots serving as waitresses and self-driving cars coming out soon, so knowledge of coding now would be useful in the future. For example, if these innovations broke down or needed reprogramming. Students had found information about technological advancements online and in some instances, students questioned the authority of the information they had found.

One student was concerned why most of the people in the focus group were "worrying about using coding in their jobs which is still a long time away". They felt that coding as a skill would be useful next year, in high school;

> *Like if the teachers' computers broken down they don't know how to turn it on. We can also use it if we want to study coding in Uni then we have a bit of knowledge or an idea of what we are doing*

As students saw coding as a skill, they felt it would be useful if they needed to programme something in the future. Similarly, to case study 1, students wanted to learn more about coding so that they would be able to help others to learn it. Students also felt that if they didn't get their dream job, knowledge of coding would mean that they could go into computer programming because they knew how to do it from learning it in

school. Students also explored ideas they thought would be possible for computers but were unrealistic;

> *Like imagine computers that don't need WIFI or phones that don't need WIFI like that would be the best thing in the world because it would have been someone that programmed the computer that when you ask a specific question it would come up as a file instead of just a website.*

Students wrote down and explained in more detail some of the projects they had made with code (Figure 6)

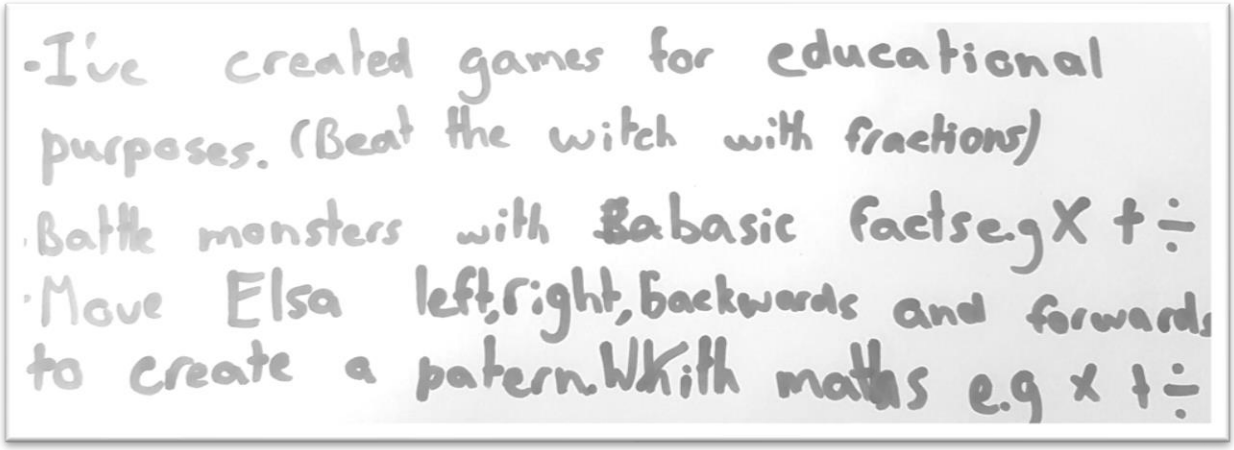

Figure 6. What students had created in coding that they couldn't before

Students had made games that used maths concepts. They had not used coding in maths or any other subject areas apart from computer technology however they recognised that mathematics and coding went hand in hand;

> *When you are coding, you need to like know how to do maths, like if you want to extend the sun by 3.5% or something, or maybe you want to move the character 30 pixels so you can move the character like 30 pixels but maybe the screen isn't big enough to move it so you have to know maths to know how much the screen can hold and the pixels and stuff.*

One student also mentioned that coding and the way of thinking could be used to understand how things not controlled by technology worked;

You can even think about things that are made with fuel because you kind of use cogs and you think okay if that's going to move around, then this is also going to move around until it can go somewhere specific.

Most students in case study 2 felt that they were hindered in how much they could learn in school and wanted to spend a lot more time learning code. They felt that as there were a lot of subjects at school, they spent their time rotating around and would perhaps only spend 30 minutes each day on a specific area. They recognised that computer science was an option in high school but did not see the time spent in the subject increasing in Years 9 or 10;

I don't think we will be doing much coding, as I don't think teachers focus on coding as much as maths, English and science.

Overall, students felt like coding was a useful skill if they wanted or needed to program something, particularly if they needed to fix technology that wasn't working. They also believed that most jobs in the future would need code. Coding helped students to understand how things worked but they wanted more time on it so that they could learn more and help others to learn.

The essence of case study 2

Students in case study 2 saw coding as a skill they needed. They were dependent on technology that used coding. Therefore, students felt they should understand and know how the technology worked. They thought this knowledge may be useful in the future should they need to programme something. For example, if technology was broken. Students also thought most jobs in the future would require code.

Students in case study 2 would visit the websites they knew to get more of a feel for coding and the different ways to code (game versus webpage). They enjoyed being able to solve problems themselves but recognised they also needed help from other students or the teacher at times. Students wanted to learn more so that they could help others to code. Some students were encouraged to learn more about coding by their parents. They found coding interesting and liked learning new things. They considered coding to be different from any other subject. "It's good to learn heaps of things in school so that you know heaps of stuff". This variety in their learning meant students felt they could

change their career or study path in the future. However, they also recognised that limited time was spent on coding at school and wanted extra time to learn more.

Summary narrative description

I found similarities and differences between the two case studies. Digital technology and therefore coding was taught as a separate subject in both contexts and students said that a cross-curricular approach was not used in either school. Students in both case studies believed coding was essential for future employment as most jobs would require the ability to code. They felt they could apply their knowledge of coding now but even more so in the future. Students in both focus groups said they would be able to fix computers if they were broken or create digital products others could use. The main difference between the two case studies seemed to be the approach of the teacher. In case study 1 students said they were told to experiment with various online resources and could choose the activities they found fun and interesting. In case study 2 however the teacher explained coding concepts such as algorithms and students applied these both with and without the use of a computer. In some instances, students in case study 2 worked individually or with the guidance of their teacher. Students in both case studies said that they would turn to other students for help if required. However, they also felt confident that they would be able to teach other students coding or help others learn to code once they had learnt the concepts themselves. In both case studies, parents had a role in supporting or encouraging programming at home, overall, five out of the twelve participants discussed parental involvement. From these discussions their parents had some experience or an interest in IT or coding. Their involvement included, explaining coding, the demand for coding jobs and urging students to create using code.

CHAPTER 5 FINDINGS: CROSS-CASE ANALYSIS

Introduction

This chapter applies a theoretical framework to examine and compare the findings of both case studies. I adapted the components of the five curriculum ideologies from Adamson & Morris, (2014) to place them in the context of coding (Table 1). These components include elements of the pedagogical design which are integral to the curriculum perspectives. I read through the transcripts again and when I found evidence of a component, I highlighted the relevant words and added a comment (Figure 7). I summarised the findings in Table 1.

Table 1: A summary of the findings applied to curriculum ideologies and their components

Curriculum ideologies and their components in the context of coding *(Adapted from Adamson & Morris, 2014)*	**Case Study 1**	**Case Study 2**
Academic rationalism;		
Coding is part of the technology learning area only	X	X
Knowledge of coding	X	X
Teacher explanation of coding		X
Passive learning of coding		
Generate knowledge through inquiry	X	
Social and economic efficiency;		
Knowledge and skills relevant to future employment	X	X
Apply coding skills	X	X
Social reconstructionism;		
Coding for social good		
Awareness of social issues		X
Interaction and group work	X	X
Community activities		
Progressivism;		
Learner centred (focused on individual needs, interests, abilities)	X	
Personal and intellectual development	X	X
Teacher as facilitator	X	X
Cognitive pluralism;		
Learn coding in different ways	X	X
Diversity of outcomes	X	
Teacher as facilitator	X	X

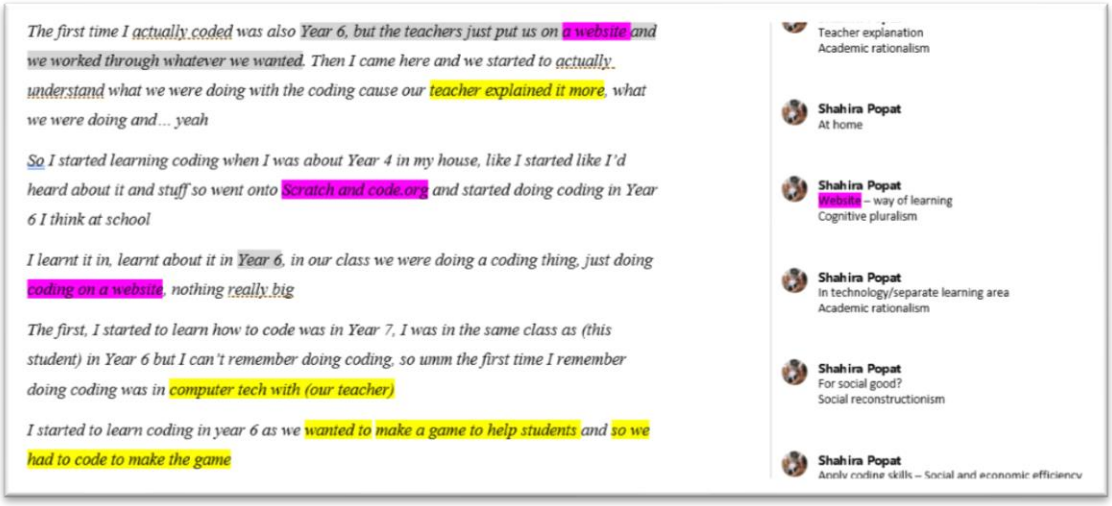

Figure 7. Coding transcripts into components of the curriculum ideologies

Although there was evidence of components of each type of ideology from the student conversations I could see varying levels of significance of each of these by the number of times students mentioned them within their discussions. I synthesised the number of times components within an ideology were talked about (Table 2). Students discussions placed more emphasis on components under social and economic efficiency and progressivism ideologies (Table 2). In this chapter I explain the components of each ideology in terms of the students' comments and identify the similarities and differences between the case studies.

Table 2: Enumeration of students' views aligned with curriculum ideologies

	Number of times a component under each ideology was discussed and represented as a percentage			
	Case Study 1		**Case Study 2**	
Curriculum Ideology	**No.**	**%**	**No.**	**%**
Academic rationalism	5	8	10	13
Social and economic efficiency	22	33	33	42
Social reconstructionism	2	3	4	5
Progressivism	20	30	20	26
Cognitive pluralism	17	26	11	14

Evidence of academic rationalism

A key component of academic rationalism is a view that the subject area is a separate discipline in the curriculum. Students in both case studies recognised digital/computer technology and coding as a different or separate discipline to other subjects. Students in both case studies said that they had not used or learnt coding or coding concepts in other subject areas and referred to maths, English and science when discussing this. Students did say that knowledge of maths was essential for coding but rather than developing mathematical knowledge students referred to using existing knowledge. There was a slight difference in the way that students in each case viewed coding. This was attributable to the differences in the enacted curriculum. A student in case study 1 said, "with coding, I don't see it as a school subject, we do it for like fun, it's like a game, you get to code". This contrasts with students in case study 2 who saw coding as an "educational" subject.

Academic rationalists believe that the curriculum should develop knowledge. Students discussions on the types of activities they completed showed that they gained knowledge of coding concepts. There were some similarities and differences in what was learnt and the way they learnt it. Students in case study 1 felt that they had learnt some basic coding concepts in school as they referred to ideas such as adjusting values (variables) and the order of instructions (sequences). Students in case study 1 did not refer to coding specific terminology and most of the activities completed in school involved "drag and drop blocks". Students in case study 2 however said what they do "is a lot harder" than using block-based Scratch programming which most had tried in Year 6. Students felt they had "learnt lots of different ways to code" at intermediate school. Students in case study 2 said that their teacher would explain new coding concepts to them first, whereas students in case study 1 did not mention teacher explanation. Instead they were asked to go on specific websites and work through online tutorials. Students in case study 2 did not refer to specific coding related terminology either. Both ways of learning could be passive ways of learning. For example, a student watching or listening to an explanation and demonstration. However, they put what they had learnt or been taught into action. Therefore, it was not considered to be passive learning.

Academic rationalists do not limit their view of the curriculum to the transmission of existing knowledge. Furthermore, academic rationalism can also include the perspective that knowledge can be created through inquiry. Students in case study 1 discussed learning through inquiry, where they sought out new information about how to complete coding activities or ways to code. However, in some instances this was not in school and was at home. Students in case study 1 would find new understandings about coding through additional online resources or other students. In case study 2, the tasks provided by the teacher promoted inquiry. Students had to answer questions, however it wasn't clear from the student discussion whether they learnt coding concepts through inquiry.

Regardless of how knowledge was generated both case studies showed some evidence of academic rationalists' views of coding in the curriculum. Academic rationalism in case study 2 was conceived as a separate and different subject to any others. Students in case study 2 referred to tasks completed in "computer tech" when talking about their experiences of coding in the curriculum. Students in case study 1 also discussed coding being different to other subjects but referred to seeking out more knowledge about coding. However, this was mostly discussed as an independent task or learners' activity rather than promoted through the teacher. Students in both case studies gained knowledge of coding, for example, variables, sequences and how to write coding instructions but did not refer to coding specific terminology. In most instances, students gained knowledge about "doing" coding not just knowing about coding and the coding knowledge was applied. This can be linked to social and economic efficiency ideology.

Evidence of social and economic efficiency

In social and economic efficiency ideology it is the curriculum that determines what skills are essential for society's needs and it is designed to prepare students to contribute to the growth of the economy**.** Students in both case studies deemed coding as relevant for future employment and that coding would be required in most jobs in the future. Students in case study 1 saw this as a reason why coding was taught in schools and associated coding mainly with careers in creating digital products. What students said about the utilitarian value was derived from what they had learnt, for example, how to make games. Students in case study 2 saw coding more as a skill that should be taught and that could be applied across many different areas or be useful in any profession that

required a computer. Students in both case studies were able “to code something to make it do something” (Student, Focus Group 1).

In case study 1, students were responsible for their own decisions when it came to learning coding rather than being provided with set tasks to master and apply specific skills. Students in case study 2 were directed by the teacher to complete a worksheet on a coding concept and apply the skill manually before being allowed to code on a computer. Students in case study 2 also talked about how their teacher had told them there was a shortage of people who could code in the world which is why it was important to learn.

Components under social and economic efficiency were brought up the most regardless of what the guided questions in the focus group asked. Students in both case studies could apply coding skills, mainly through making games. Students also viewed that people needed to code as we live in a society controlled by technology, even more so in the future. Students in case study 1 referred to knowledge of coding being important for future employment. Students in case study 2 shared this view but mostly commented on coding as a necessary skill, with one student stating, “basically because if you don’t have code in the future you’re going to get nowhere probably”.

Evidence of social reconstructionism

Students in case study 2 recognised the lack of people who knew how to code in the economy as an issue, and therefore to learn coding was important. However, there was no discussion on the use of coding for social good. Students in case study 2 seemed keen to make a webpage or an app that they could sell for money but there were no suggestions on what the content of this would be. Students in case study 1 focused on ideas of making games or social media websites. Students in both case studies did not mention apps that others have created that could help people.

There was some evidence of interaction and group work however not in the view of social reconstructionism. Students in case study 1 visited their local college for technology classes. A student commented on Bee-Bot robots at the college which students programmed to move in different directions. However, when discussing robots’ students in both case studies referred to making jobs easier such as self-driving cars rather than helping with societal problems. A student in case study 1 said; “but they

can't do that *(referring to robots driving cars)*, because, just take everything, because we would just become lazy people that just sit down all day", therefore not seeing the use of technology for social good such as reducing speeding related deaths.

Students in both case studies interacted with other students and worked in groups, either at the local college (case study 1) or on more complex coding projects (case study 2). However, social reconstructionism views the curriculum as promoting social change and as students in both case studies did not comment on using coding to solve societal problems there is very little evidence to support this ideology. Therefore, it could be said that using coding for social reform was not experienced in either case study.

Evidence of progressivism and cognitive pluralism

This section has grouped evidence of progressivism and cognitive pluralism as both placed emphasis on students being active in their own learning and the teacher as facilitator. Both case studies showed evidence of progressivism and cognitive pluralism ideology. Case study 1 presented slightly stronger views as students said that they could work on whatever they wanted when they were coding and felt that they could focus on their own needs, interests and abilities. Therefore, one student would focus on making characters move where another would work on drag and drop blocks to create dresses, emojis or Snapchat filters. This also meant that there was diversity in their learning outcomes. Students in case study 1 discussed how they would explore and develop coding knowledge independently by looking at further resources and although the teacher was present they did not seem to rely on them for help. In case study 2, although students said they had the opportunity to choose certain computer-based coding activities in school, they were working towards the same or similar outcomes. For example, understanding algorithms or writing instructions to solve problems.

Students in both case studies learnt coding in different ways including; number, symbols, text, online tutorials, watching others, websites, creating games, puzzles, music and in case study 2 also creating apps, webpages and coding manually using worksheets. Some of this was at home rather than part of coding in class. Both case studies supported teacher facilitation. In case study 2 students discussed having to work through problems themselves first and then they would ask another student for help. If they still couldn't complete the task they would ask the teacher who would "guide us to solve the pattern". Students in case study 1 said that the teacher told them to go on to

visual programming websites and have fun playing around. They said several times that they had the flexibility to work on coding activities of their own choice. There was an overall feeling in both case studies that students thought they developed personally by being able to help others and intellectually through their knowledge of coding and thought processes.

In focus group 1, students placed a lot of emphasis on the diversity of the learning outcomes between each student hence supporting cognitive pluralism ideology. Students in case study 2 were however often working towards the same outcomes and mostly commented on learning in different ways. Components under progressivism were revealed an equal number of times in case study 1 and 2, calculated as 30% and 26% of the interview transcripts respectively. In case study 1, students pointed out that they independently worked through tasks and saw themselves as being autonomous in their own learning. Students in case study 2 referred to this also but most of their comments centred around developing personally and intellectually through coding in the curriculum. There is a connection between progressivism and cognitive pluralism which is synergistic. For example, when students focused on their own needs and interests this often led to diversity of outcomes.

Chapter 5 summary

There was evidence of components of each type of ideology from the student conversations but this to varying degrees based on the number or percentage of times students mentioned them. There was almost an even balance between students' remarks that were applied to progressivism and social and economic efficiency ideologies in case study 1. These two ideologies have one main contradiction. Social and economic efficiency, like academic rationalism, views the curriculum to be more teacher-directed, whereas progressivism, like cognitive pluralism sees the curriculum as learner-centred. This is also true of the broader debate in education between powerful knowledge and 21st century skills. Powerful knowledge views learning as a teacher supported process (McPhail & Rata, 2016) and 21st century learning means students drive their own learning (Bolstad et al., 2012). I discovered other elements of competing ideologies that contradicted one another which are of importance to this study. These are summarised in Figure 8.

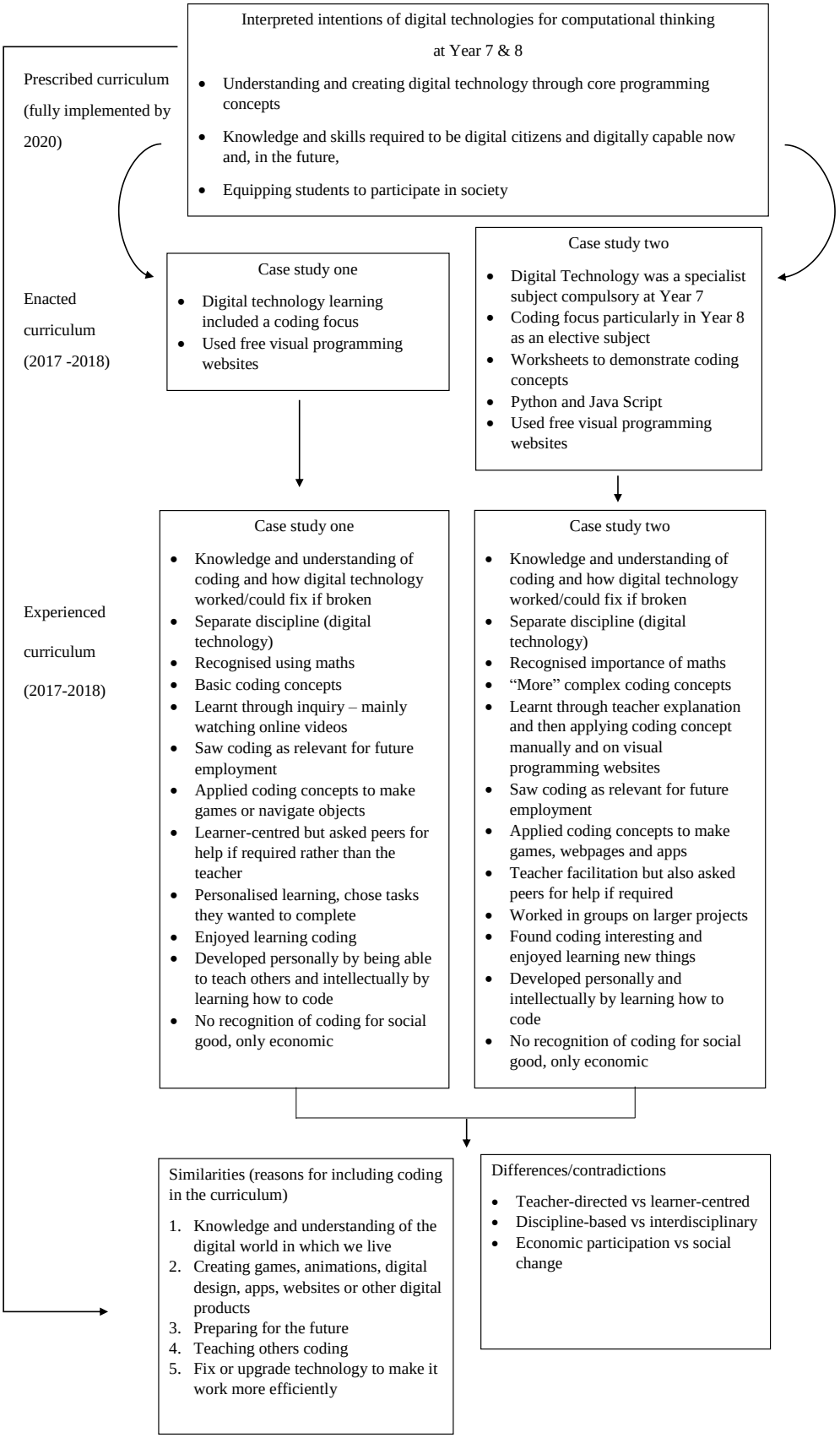

Figure 8. Summary: Prescribed, enacted and experienced curriculum

I also learnt from the findings that some components of competing ideological perspectives can be interrelated to one another. For example, students felt that they developed both personally and intellectually from learning new things such as coding, this included knowledge and skills. Therefore, elements of progressivism connected to academic rationalism and social academic efficiency.

Figure 8. summarises the findings of the experienced curriculum from Chapters 4 and 5. These are aligned with the interpretations of the computational thinking for digital technologies (prescribed curriculum) from the literature review and the enacted curriculum derived from the teachers' resources and school website. The similarities of students' views of coding across the case studies support multiple perspectives of curriculum ideologies and provide the students' overall reasons for including coding in the curriculum (Figure 8). These match with the intentions of digital technologies in the curriculum. The differences between case studies and contradictions between competing curriculum ideologies suggest how the teaching of coding could be effectively structured. I explain these ideas in the discussion chapter with reference to previous research literature.

CHAPTER 6 DISCUSSION

Introduction

The digital technologies curriculum content was recognised as part of the technology learning area from the beginning of 2018 with an expectation that New Zealand schools should have it fully implemented by 2020. Some schools had already introduced digital technology and coding into their school curriculum. As such, this study aims to determine the value of coding in the curriculum through the experiences of students. In chapter 4, I answered each of the research questions from the students' experiences of coding in the curriculum. In chapter 5, I identified the similarities and differences across the case studies linked to a theoretical framework. The findings were summarised in Figure 8 along with the prescribed and enacted curriculum. In this chapter, I take an interpretive perspective, also known as the hermeneutic perspective to analyse and explain the findings. I evaluate the interpretive outcomes of this study against previous research on coding, curriculum perspectives and curriculum ideology. This chapter is structured into two areas of importance to assist with the discussion of the research. The first of these addresses the main aim of the research which was to explore students' experiences and perspectives of coding and its value in the curriculum. The similarities across the case studies included students' beliefs about the benefits of including coding in the curriculum. Therefore, highlighting the reasons for and perceived value of coding in the curriculum. The second area focuses on the contradictions between the competing curriculum ideologies distinguished from students' experiences of coding in the curriculum. Pedagogical design was found to be important in the literature review. The contradictions identified in this research suggest how experiences of coding in the curriculum could be effectively structured and link to broader debate in education of 21st century skills versus powerful knowledge. This provides schools with the basis of an action plan that could be adopted when considering the new curriculum content and it may influence the way educators think about coding in the curriculum.

Reasons for including coding in the curriculum: Students' perspectives

Previous literature which explored coding claimed that coding skills are important 21st century skills and referred to a range of competencies including; problem-solving, critical thinking, creativity, collaboration, and communication (Balanskat & Englehart, 2015; Lye & Koh, 2014; Tuomi et al., 2017). In this study, students did indicate they were developing these skills. For example, I asked what was useful about learning to code, a student in case study 2 said coding had allowed them to think more logically about how things in general worked. Students in both case studies also commented on problem-solving. For example, being able to "figure things out" and if they didn't get things right the first time they could go back and correct their mistakes.

Another area of discussion that related to 21st century skills was being creative. Although students used existing resources on visual programming websites when creating games, puzzles or emojis, they still changed and customised these to meet their own needs, interests and capabilities. Students also discussed helping each other out and working in groups which demonstrated effective communication and collaboration. These skills may have been put into practice or even developed during coding, however, it is argued that teachers can create opportunities for students to acquire these skills in ways other than coding (Hayes & Stewart, 2016). These competencies were not the focus of students group discussions. The skill that students felt was important was the technical skill of being able to code. Previous studies have not recognised that the aim has been to bring programming back after it disappeared from classrooms in the mid-80's (Kafai, 2016). The computer science community has learnt from the past and programming came back worldwide because of the visual programming languages such as Scratch where students could create applications without learning complex syntax (Moreno-León et al., 2015; Wilson & Moffat, 2010). This chapter discusses the opportunities that students felt learning coding could bring and therefore their perceived value of coding in the curriculum.

Five reasons for including coding in the curriculum were identified from the findings which were aligned with the interpreted intentions of digital technologies in the curriculum;

1. Knowledge and understanding of the digital world
2. Creating digital products
3. Preparing for the future
4. Teaching others coding
5. Fixing or upgrading technology

Knowledge and understanding of the digital world

Students in both case studies felt that learning to code meant that they understood how digital technology worked and because they lived in a world that was reliant on digital technology it was important for them to learn. Getting students to look at what is happening behind the screen empowers them to understand the world in which they live, and coding is the most powerful way to work on a computer and establish a presence in the digital world (Bell & Roberts, 2016; Burke et al., 2016; Kafai, 2016). Students felt learning coding meant that they knew that they could influence what was on the screen or even create things behind the screen themselves.

When I first thought about understanding how digital technology worked and "core programming concepts" (MOE, 2017a) I envisaged students learning technical programming language and algorithms. In the new curriculum content, core programming concepts are referred to as debugging, sequences, loops, algorithms and binary digits. However, computational thinking has shifted to computational participation (Kafai 2016). An example of this shift was reflected in the focus group discussions where students in each of the case studies did not refer to specific coding concepts or use programming language but demonstrated them. For example, students said they went back to instructions when they went wrong and corrected their mistakes (debugging). Computer participation is explained by coding in the curriculum moving away from writing or building code and to creating applications such as games (Burke et al., 2016). Students used coding skills in a relevant way which means it is not an

academic *learned* discipline; not academic rationalism. Coding in the curriculum focuses on the interests of the individual, supporting progressive views. Creating using coding concepts allowed students to participate in digital activities by interacting with the content and taking advantage of motion and animation. When aligning this with the prescribed curriculum the interrelationship between progressivism and social and economic ideology is demonstrated. The prescribed curriculum intends that students understand and create through core programming concepts and apply this to their lives and careers. Therefore, taking account of individual interests relevant to students' lives but also societies needs in terms of knowledge and skills relevant to future employment.

Students in case study 1 found that the free visual programming websites such as Scratch, Tynker and Made with Code they used in class only provided them with basic knowledge of coding. Creating games was only a starting point for them. Most students in case study 1 wanted to learn the technicalities of programming language that were not seen to be possible through drag and drop blocks. One of the main limitations of using free visual programming software was the resources were too basic and students could only reach a certain point with coding. This was an obstacle to more meaningful and enriching participation. Some students would therefore try to learn more complex coding at home but struggled without the help of a teacher. Teacher explanation is discussed further under conflicting ideologies. However, there is another way to broaden and deepen computational participation. This is by treating the use of digital tools, in this case coding, as a social practice by engaging with communities and sharing work for others to comment on and/or change (Kafai, 2016; Starkey, 2012). In a digital age, due to globalisation, this interaction can be beyond the classroom or even the town, city or country. Students in the case studies worked with others in their class and helped each other out with coding. However, apart from sharing work with parents, they did not discuss communicating with the wider community or beyond. This is discussed further when considering conflicting ideologies.

Creating digital products

Students in both case studies said that they had created things that they could not make before learning coding in school. Creating digital products was combined with understanding digital technology in the prescribed curriculum. For example, students should understand and create digital technologies to succeed in further education and the world of work (MOE, 2017a). In the case studies, students understood how digital technology worked and discussed this mainly through using visual programming languages and creating applications. The enacted curriculum in both case studies consisted of free visual programming websites which provided the resources for students to create games, animations, digital designs, apps and webpages. To help educators understand the computational thinking progress outcomes, MOE (2017a) provided sample exemplars illustrating the learning described. The examples show how coding concepts which have been taught in class can be applied by using Scratch block-based resources to create a computer program. This is similar to what was experienced by students in case study 2.

Students' in both case studies were excited to talk about the games they had made and wanted to make more complex animations using code and apps or websites that other people could use. Students played games in their daily lives. Through creating games or animations students learn coding knowledge and skills in a relevant way (Haden, 2006; Lambic, 2010). Students agreed that it was important to learn about coding in school now so that they had an idea about what it was. Some students said they would not know what coding was if they had not learnt it in school. This supports the view that learning coding at a young age breaks down the stereotypes that it is just for those who want to go into programming. In addition, coding could be an intimidating subject if learnt later in life (Gardiner, 2014). One student in case study 2 recognised that the knowledge they had now would be useful if they wanted to study coding at University.

The findings of this study suggest that students enjoyed making games and did not regard coding as something that was just for people who wanted to go into programming. Learning coding was fun, interesting and enjoyable for students. In addition, it is a way for them to make and be in the digital world (Kafai, 2016). Some students felt that game design would be a good job and learning to code would allow them to pursue this as a career.

Preparing for the future

Students felt that jobs in the future would require knowledge of coding. Some students also believed coding was a skill that would be useful not only in jobs but in life when needed. Students referred to technological advances being part of the reason, for example, robots and self-driving cars. Therefore, they seemed to recognise that the world is rapidly changing due to technology and so were jobs. This aligns with the political agenda of introducing coding into the prescribed curriculum. Nikki Kaye who was the Minster of Education when the change was introduced said that students need to be prepared to adapt to technology and jobs that have not yet been invented such as those involving robotics and artificial intelligence (MOE, 2017a). Teaching coding means having a generation of students who know how to code and can take these skills into the workforce. However, students brought in the idea that with a basic level of coding they would be able to change their career to a programming related job or code "things" if required. Is this realistic or is it likely to be problematic in the future? Students wanted to learn more complex coding and the prescribed curriculum intends that they will continue to take part in digital technologies learning until Year 10. However, the new curriculum content is considered to be too limited and "not enough focus on components, frameworks and high-level tools" (Chen, 2017, p12). Students demonstrated awareness of what is possible with computers, but mostly related these to their own experiences.

They recognised that technology could take over some peoples' jobs, therefore it would be useful for them to know how to code. Burning Glass Technologies (2016) found that although programming jobs were growing faster than the job market, coding skills were not just for programmers. Creating and developing digital technology through coding was interpreted as an intention of the prescribed curriculum and is said to be a key requirement for innovation and success in several industries (Alano et al., 2017; MOE, 2017a). Coding is without doubt useful for career opportunities in the technology sector (Alano et al., 2017; Balanskat & Englehart, 2015; Bell et al., 2014; Tuomi et al., 2018). The view is that coding can be useful in areas other than STEM fields (Alano, 2017). Despite this, most research links the usefulness of coding to developing 21st century skills (Balanskat & Englehart, 2015; Lye & Koh, 2014; Tuomi et al., 2017). Rather than problem-solving or creativity, how would technical coding be useful for health workers

or chefs? Students in case study 2 believed that coding could be useful in many areas for example, in the aviation industry by programming a plane or any website as it could be created and updated.

What students said about the social utilitarian value of coding was derived from what they were taught or what they had learnt or read. For example, they provided examples of how coding can be useful when using computers or applications in general and creating games, apps, websites or innovations that were being developed or that did not exist yet. Students were aware that coding allowed them to have an understanding how digital technology worked and that items they used at home and at school required coding. They said they were able to understand applications better, including the capabilities and limitations of the technology.

Teaching others coding

Students in both case studies said that it was useful to learn coding in school so that they could teach others how to code. I did not find this reasoning in any previous studies. Teaching others is a social skill and previous studies noted that 21st century skills such as these were core concepts in computer science and/or coding (Alano et al, 2017; Falloon et al, 2016; Fessakis, 2013; Kalegliogluu, 2015; Webb et al, 2017b). Researchers found that students would help other students with coding concepts by sharing what they had completed (Fessakis, 2013; Kalelioğlu, 2015). This was also evident in the focus groups discussions where students helped other students or were encouraged to ask peers for help. However, students in both case studies specifically said they would be able to teach others how to code or they wanted to help other people to learn how to code. This is learner-centred and meant that students believed that they were confident and capable in coding and could pass this knowledge on to others. For these students, coding supported them to develop confidence and skill, therefore to some extent, to practice digital capability and digital citizenship, as intended by the prescribed curriculum.

Fixing or upgrading technology

Students had an awareness that digital technology was run by software which could be replaced with better software so there was a requirement for people to know how to fix it or update it. Students in both case studies felt that because of coding, they had knowledge of how digital technology worked. Therefore, they believed they would be able to fix issues with computers or upgrade software to make it work more efficiently. A student in case study 2 even said that if a teacher had trouble with a computer they would be able to fix it, implying that a teacher would need their help. No previous research was found on this as an outcome of learning how to code at school. Students felt confident and capable but does their expectation match reality? Students in case study 1, were only learning basic coding concepts from free resources on visual programming websites. Students searched for more complex coding activities and were aware that they could find most things they were looking for online. This could be the same for fixing technology related issues. Students can follow what other people have tried or ask people within the wider community. Students particularly in case study 1 have some experience in finding information online. However, the approach of the teacher would be important to support and encourage. This is because students in case study 1 commented on not being able to find what they were looking for online all the time and students in case study 2 only used resources they were aware of. Teacher direction is discussed further in the competing ideologies section.

The reasons presented for including coding in the curriculum demonstrate the value of coding in the curriculum from the students' perspectives. The reasons support the interpreted intentions of digital technologies in the curriculum which is underpinned by social and economic efficiency ideology. However, the findings reveal aspects of learner-centred or progressivism ideology. This is because students are learning knowledge and skills that they believe to be useful in employment but which they can also apply to their daily lives. In addition, students felt that they were developing personally and intellectually through learning the technical skill of coding and knowing how digital technology worked. Coding was something they believed they could teach others. This suggests that the introduction of coding into the curriculum provides a

balance between supporting a more competitive economy and the needs and interests' students.

Contradicting areas of competing ideologies

Education also has a role in creating an equitable society through the curriculum which prescribes what students should learn and be taught. However, there are conflicting perspectives that exist particularly between an academic focus and generic skills or competencies. This highlighted some inequities in students' experiences of coding. The main contradicting areas or areas of disagreement identified were;

1. Level of teacher direction against how actively students were involved in their own learning (learner-centred). This difference existed between case study 1 and case study 2 and the competing curriculum ideologies.
2. Discipline-based and interdisciplinary which was identified as a conflict between competing curriculum ideologies. There was little evidence of interdisciplinary knowledge or skills in the enacted curriculum.
3. Students recognised being able to participate economically but is tackling social issues not important? Social change was not evident in students' experiences or curriculum documents

Each of the contradicting areas related to how learning could be structured and therefore the pedagogical design of coding in the curriculum. I discussed these individually in consideration of the findings and literature. This led to an explanation of the relationship between coding, the wider academic discipline, curriculum perspectives and powerful knowledge. I was then able to connect these three contradicting areas to form a theory.

Teacher-directed versus learner-centred

One of the main differences between case study 1 and 2 was students' discussions regarding teacher explanation. Students in case study 1 said their teacher told them to go on to visual programming websites and experiment with the activities and did not mention teacher explanation. Students in case study 2 said that their teacher would explain new coding concepts first and then they would apply these independently. This was also a contradiction between the competing curriculum ideologies. Academic rationalism and social and economic efficiency view teachers as responsible for delivering the curriculum whereas progressivism and cognitive pluralism view students as responsible for their own learning. This reflects the broader debate in education of 21st century skills versus powerful knowledge. It was originally envisaged that programming would allow students to learn by discovery (Papert, 1980). Although there is an element of active learning which students enjoyed, much of the research states that teacher explanation in a blended-learning environment was key in ensuring learning in coding (Hagge, 2017; Lambic, 2015; Saez-Lopez, 2015). Students in both case studies said they needed teacher help or explanation to carry out more complex coding.

When considering skills for a curriculum in the digital age, the evidence also suggests we need to teach capability and therefore it cannot be assumed that children can learn the skills themselves (Evans, 2014). By being born in the digital age, our students are assumed to be digital natives therefore competent speakers of digital language and teachers are assumed to be digital immigrants. If this was true students would know more about digital technologies than teachers and therefore do not need to be taught. In the case studies most of the students also had support from their parents who were not born into the digital world but at some later point have adopted many aspects of technology. Therefore, growing up in a digital age does not mean that students are taught about technology or that all children learn independently. Most young children do not naturally migrate online. For example, asking students to share work through online programming communities may not meet the intended purpose of deep and meaningful computer participation if students are not taught the strategies to cope with the vulnerability of sharing their work for others to comment on or change (Kafai, 2016).

Good pedagogy uses a range of methods, therefore can include students' everyday experiences (a learner-centred approach) to link them to coding in the curriculum (Young et al., 2014). However, students cannot experience everything, therefore learning should also take them beyond their own experiences (Young et al., 2015; Siemens, 2005). This means that learning should be a teacher-supported process. This is because if learning to code is focused on the student it limits them from accessing more powerful knowledge as there is better knowledge beyond their own experiences. It should be the role of the school to allow this. The digital technologies consultation document said, "a curriculum is effective only if we equip skilled teachers to deliver it" (MOE 2017a, p3). Teachers are expected to help students to develop generic understandings in Years 1-10 and then students should be able to work more independently during their senior years with the teacher as facilitator. However, with advances in digital technology and more specialised areas it may not be possible for subject teachers to know all innovations in a developing field of study. In addition, in the context of a new subject in schools, professional development for teachers had not been launched at the time of this study (MOE, 2018). In case study 1, students had a generalist teacher, whereas in case study 2 students had a specialist teacher. Will students have equitable access to teachers with the specialist knowledge of coding? The areas that cannot be adequately addressed by the subject teacher would require a range of people with the different types of expertise to be accessible to students.

When comparing the experienced curriculum of both case studies, although the enacted curriculum was different, the other experiences were not too dissimilar. The main difference was connected to the approach of the teacher. Students from case study 2 felt they knew more complex coding than the Scratch-based drag and drop boxes and understood concepts because of the teacher. Students in case study 1 however had a greater diversity of outcomes as their learning was driven by their own interests and capabilities. One outcome is not necessarily better than the other, but we should draw on the strengths and knowledge of each approach to support learning. This would then give all the students the access to the foundations of powerful knowledge (Young et al., 2014).

Discipline-based versus interdisciplinary

All students have the right to access all learning areas of the New Zealand curriculum (McDowall & Hipkins, 2018) and powerful knowledge starts with the idea of equality. Therefore, students should not be limited on their own interests, assumed capabilities or any other factor (Young et al., 2014). The intention of the digital technologies curriculum content is for all students from Years 1 -10 to learn coding concepts (MOE, 2017a). Therefore, digital technology was introduced as a compulsory subject as prescribed by the New Zealand curriculum. Students in both case studies experienced coding in the curriculum as a discipline-based area of digital technology as intended. However, this highlights another contradiction between the competing ideologies.

Academic rationalism ideology views curriculum content as derived from the academic discipline whereas progressivism focuses on knowledge as being holistic and therefore interdisciplinary (Adamson & Morris, 2014). Although students said a cross-curricular approach was not used, they said maths was important in coding and saw mathematical competence as essential. Coding allows students to apply mathematical concepts in a relevant way (Falloon, 2016; Fessakis, 2013; Haden, 2006; Lambic, 2010). Combining coding with other learning areas not only improves programming ability but also academic knowledge of these learning areas (Falloon, 2016; Hagge, 2017; Lye & Koh, 2014; Saez-Lopez, 2015). Some students in case study 1, mentioned physics and design when discussing the activities which they had completed. For example, making a character drop to the ground after jumping up. These students learnt about both digital technology and physics in Term 4, 2017, which could be why they were able to discuss gravity in relation to coding a character. They did not say that their academic subject knowledge of physics or maths improved, just that they used their existing knowledge. The digital technology curriculum had said that; “students should be encouraged to access relevant knowledge and skills from other learning areas” (MOE, 2017a, p13). There was little evidence of this deliberately occurring in either case study. However, activities on visual programming websites tend to combine coding with other subject areas.

Powerful knowledge is often associated with a disciplinary approach as it is concept specific and objective (McPhail & Rata, 2016). However, interdisciplinary enquiry is not ruled out by the powerful learning approach as long as it does not take for granted

substantial subject-based knowledge. Based on the literature review, focus group findings and discussion thus far, substantial subject-based knowledge of coding would involve the technicalities of programming language or engaging in two-way communication with the wider community. However, visual programming websites have removed complicated syntax from coding activities (Moreno-León et al., 2015; Wilson & Moffat, 2010). The enacted curriculum for students in case study 1 only used visual programming language, whereas in case study 2 the Year 8 course includes Python and Java Script. This again highlights the inequities between having a generalist teacher versus a specialist teacher or a learner-centred approach versus teacher direction. In addition, students in case study 2 said there are a lot of subjects which they must rotate around, therefore they do not get as much time on the things they are really interested in, such as coding. If coding is used across existing learning areas this could alleviate the risk of digital technologies being siloed and allow students to spend more time on coding (Chen, 2017). It could also assist in encouraging students to make connections across the curriculum not only with what they are doing but how they are doing it.

Combining two or more learning areas is not the only way for interdisciplinary learning to occur. 21st century skills are interdisciplinary concepts that can be applied across subjects. This was an area of contention between powerful knowledge and a 21st century learning approach because substantive knowledge was often separated from skills such as problem-solving and communication (McDowall & Hipkins, 2018). It was common practice in New Zealand education to detach objective or intellectually powerful knowledge from socially advantageous knowledge which led to using either a disciplinary or interdisciplinary approach (Bolstad et al., 2012). Many of the respondents during the consultation phase of the new curriculum content discussed, "the extent to which there should be more focus on the underlying capabilities rather than content knowledge. That is, creativity, collaboration, resilience, problem solving, critical thinking, communication and self-management, rather than coding" (Chen, 2017, p2). However, coding does not only involve the technical skill but also generic skills such as creativity rather than one or the other. It is however difficult to assess the transferability of these skills as they are dependent other factors, such as the length of time of study and the similarity of the skills to non-programming domains (Mayer, 1988).

Economic participation versus social change

Being able to participate in society economically was a seen as an important reason for including coding in the curriculum both by students in the case studies and the prescribed curriculum. However, participating in society also means learning values and attitudes beneficial to society (Adamson & Morrison, 2014). If the focus of learning these is on economic growth it still comes from a social and economic efficiency perspective, however if the focus is on social issues and social change this is at the core of social reconstructionism (Schiro, 2008). Students in the case studies did not discuss social reconstructionist views of coding in the curriculum. Students perceptions of coding were derived from their experiences. Therefore, they may encounter social issues when they are a little older and have more life experience, for example, when students learn how to drive. However, a powerful knowledge approach would take students beyond their own experiences. This provides an example of how ideology and the ideals around the purpose of education come into play when delivering a curriculum. If students are not aware of social issues and the purpose of education is not to elicit social change then students will not experience social reconstructionism. When developing a curriculum, it is beneficial to look at all possibilities than exclude these completely (Marsh & Willis, 1995). Therefore, this study will look at coding from a social reconstructionism perspective rather than omit it because it was not experienced.

When coding began in the 1970's in schools the focus was on theory, mathematics, logical thinking and practical use of computers (Bell et al., 2014). For example, breaking instructions down into the smallest logical steps for a computer to follow. The same processes are required by programmers today but there is a lot more creativity involved for example by designing apps to use in daily life. Technology educators have used activities motivated by social reconstructionism (Zuga, 1992) and there are many examples of people creating apps or games for social good in press releases and coding for social good initiatives (Bouwkamp, 2015; Kan & Ongchoco, 2017). Although there is little evidence of social reconstructionism in the curriculum, todays coders are some of the first who can use coding for social good, to solve social issues (Bouwkamp, 2015). In a pure social reconstructionist curriculum, teachers would only teach the technical skills required to solve the social issue and ideally the social purpose should be left to the choice of the students (Zuga, 1992). However, a combination of ideologies

may form a curriculum and there are few schools or learning areas that are singular in curriculum perspective or ideology. Students in case study 1 said that they were inspired by others to seek out more information about coding. The experience of solving a problem such as making a game to navigate depression could inspire students to seek out the knowledge and skills needed to take on additional problems. This would involve using what has been learnt in new contexts, combinations or innovative ways.

Chapter 6 summary

Students in the case studies understood how digital technology worked. They had confidence in their coding capability and they applied these skills to create digital products. However, they brought in the idea that with their little knowledge of coding they would be able to get a job or to fix technology. Students also felt confident they could pass on what they had learnt about coding to others.

The aim of the digital technologies curriculum content is that all students have access to learning that builds their digital skills and fluency (MOE, 2017). This supports the idea of equal citizens and that students have an equal entitlement to knowledge (Young et al., 2014). Introducing digital technology as a standalone compulsory subject was a way of ensuring all students have access to it. However, the contradictions highlighted an inequity in the enacted and experienced curriculum between the case studies. The approach of the teachers and parental involvement impacted the students' experiences and therefore their perceptions of coding in the curriculum. The different curriculum perspectives and ideologies also identified a conflict between the ideological intention of digital technologies in the curriculum and education in a digital age. Globalisation is a key concept for education in a digital age. For example, social participation should increase contact and connections between people (Kafai, 2016; Starkey, 2011). It also allows students to get involved with communities both nationally and globally. This reinforces that New Zealand education should not only look at what is known but how it is known (Bull, 2009). The digital technologies curriculum content intends for students to become digital citizens who can participate in society. However, coding was not taught as a social practice in either case study.

Curriculum perspectives and powerful knowledge

I summarised the areas of competing ideologies and perspectives in Figure 9. The findings suggest that rather than a single curriculum perspective that is teacher-directed or learner-centred, drawing on the strengths of each would bring the concepts that students understand into a new abstraction. This explained the relationship between the curriculum perspectives and powerful knowledge. I used an example of creating a digital product for social good to explore the possibility and related this to examples from the findings of this research which provided a way of how coding in the curriculum could be effectively structured.

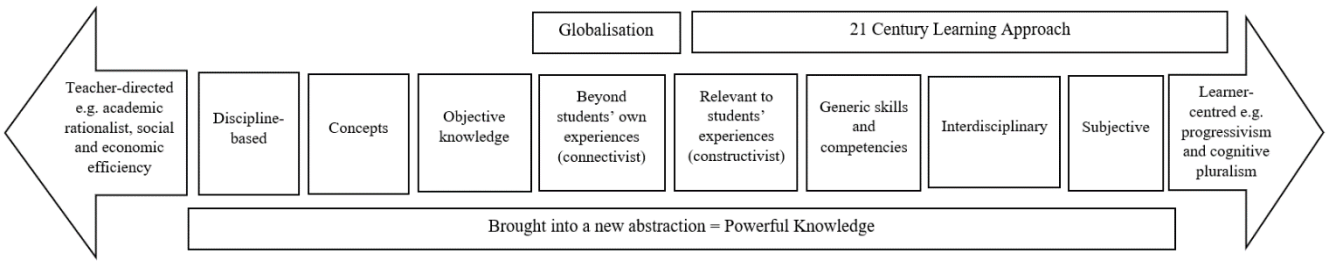

Figure 9. Curriculum perspectives and powerful knowledge

The students in both case studies said that they found it difficult to learn complex coding on their own. Teaching the technical skills involved in coding does not mean that students would be following directions all the time. In addition, learner-centred does not mean students have to make all the decisions regarding their learning. It would be a teacher-supported process where students can learn the concepts and objective knowledge required, for example, how to create a game, app or website.

Students could still be actively involved in choosing the social problem relevant to their own experiences. They may even determine how they would like to approach this issue and solve it using digital technology. Students would use generic skills and competencies such as creativity and problem-solving. They could be encouraged to draw on skills from learning areas other than digital technology such as science, English, maths or art and design. The teacher could connect students with experts in the field or those who have experienced the issue or tackling issues in a similar way. This

would link students to the wider community and rather than finding information online from anywhere is the world, students would be supported in two-way communication and feedback to deepen their designs and broaden their computational participation.

Academic knowledge of concepts and ideas are still important for students to understand core programming concepts. However, in this example opportunities are created for students to build knowledge and collaborate in new and challenging ways. Engaging students with different experiences, perspectives and ideas will draw on the strengths of both traditional knowledge and 21st century learning approaches which can lead to more powerful knowledge creation.

CHAPTER 7 CONCLUSION

Interpreting the new curriculum content in New Zealand through its manifestations of ideology and the enacted and experienced curriculum allowed me to evaluate the implementation of this curriculum change. The focus of this study was on coding, which was an important part of digital technologies in the curriculum.

In this narrative inquiry twelve Year 7 and 8 students who had already experienced coding in the curriculum shared their experiences. They provided examples of why they believed coding was important to learn and how they found it useful. The reasons and experiences across both case studies were similar. Students felt that coding allowed them to understand the digital world in which they lived, and that coding was useful for the future, including employment. Their reasons aligned with the interpreted intentions of digital technologies in the curriculum which supported economic participation and social and economic efficiency ideology. However, this study also found coding supports learner-centred ideology. Students believed coding was a useful skill, they were able to create digital products, wanted to teach others coding and believed they could fix problems with computers if required. This meant that students felt they developed personally and intellectually through learning new things through coding.

There was a significant difference between the two case studies which reflected a contradiction between competing ideologies. The main difference was the approach of the teacher. This difference at classroom level corresponded with the broader debate in education of 21st century skills versus powerful knowledge. The debate between the two approaches and the contention between teacher-directed or learner-centred methods need not exist. Students in both case studies said that they needed help from others to learn more complex coding. Drawing on the strengths of academic knowledge, for example learning coding concepts through teacher instruction, and using 21st century skills, students can use information produced by others to actively build new knowledge. Globalisation is a key concept for education in a digital age. Treating coding as a social practice by teaching students to use the online learning communities, to connect with the wider community or to use programming for social good are examples of how coding in the curriculum could be effectively structured.

The students in this study were born in a digital age and cannot comprehend what their lives would be like without technology. They appreciate what a computer does and want to learn more about *how* it does it. Most students want to be able to use and control technology as intended by the prescribed curriculum. Students in the case studies enjoyed learning coding and found it interesting, this is because they saw it as relevant to their lives. Coding is part of the world they know; therefore, “without code we would probably be like cave people”.

References

Adamson, B., & Morris, P. (2014). Comparing curricula. In M. Bray, B. Adamson, & M. Mason (Eds.), Comparative education research: Approaches and methods (pp. 309-332). Hong Kong: Comparative Education Research Centre, University of Hong Kong; Springer.

Alano, J., Babb, D., Bell, J., Booker-Dwyer, T., DeLyser, L.A., McMunn Dooley, C., & Phillips, R. (2017). The K-12 computer science framework. Retrieved from https://k12cs.org/

Anderson, L. W. & Krathwohl, D. R. (2001). A taxonomy for learning, teaching, and assessing: A revision of Bloom's taxonomy of educational objectives. New York: Addison Wesley Longman

Balanskat, A. & Englehart, K., (2015). Computing our future. Computer programming and coding. Priorities, school curricula and initiatives across Europe. European Schoolnet.

Bell, T., & Roberts, J. (2016). Computational thinking is more about humans than computers. *Set: Research Information for Teachers, 2016*(1), 3-7.

Bell, T., Andreae, P., & Robins, A. (2014). A case study of the introduction of computer science in NZ schools. *ACM Transactions on Computing Education, 14*(2). http://dx.doi.org/10.1145/2602485

Bernardo, M., & Morris, J. (1994). Transfer effects of a high school computer programming course on mathematical modeling, procedural comprehension, and verbal problem solution. *Journal of Research on Computing in Education, 26*(4), 523-536. http://dx.doi.org/10.1080/08886504.1994.10782108

Bishop-Clark, C., Courte, J., Evans, D. & Howard, E. V. (2007). A quantitative and qualitative investigation of using Alice programming to improve confidence, enjoyment and achievement among non-majors. *Journal of Educational Computing Research, 37*(2), 193-207.

Bolstad, R., Gilbert, J., McDowall, S., Bull, A., Boyd S., & Hipkins, R. (2012) *Supporting future-oriented learning & teaching a New Zealand perspective* (report to the Ministry of Education). New Zealand: Retrieved from

https://www.educationcounts.govt.nz/publications/schooling/supporting-future-oriented-learning-and-teaching-a-new-zealand-perspective

Bouwkamp, K. (2015). Want to change the world? Learn how to code. Retrieved from https://www.codingdojo.com/blog/coding-for-social-good/

Brooks, M. (1986). Curriculum Development from a Constructivist Perspective. *Educational Leadership, 44*(4), 63.

Brown, G.T.L. (2006). Conceptions of curriculum: a framework for understanding New Zealand's curriculum framework and teachers' opinions. *Curriculum Matters*, vol. 2, p. 164. Retrieved from http://link.galegroup.com/apps/doc/A180748239/AONE?u=vuw&sid=AONE&xid=b36d96a1

Buie, E., & Seith, E. (2012). Time for young scots to switch on to computer programming. *The Times Educational Supplement Scotland,* (2256), 12. Retrieved from https://search-proquest-om.helicon.vuw.ac.nz/docview/1010847713?accountid=14782

Bull, A. (2009). *Thinking together to become 21st Century teachers*. Wellington: NZCER. Retrieved from: www.nzcer.org.nz/system/files/21st-century-teachers-200906.pdf

Burke, Q., O'Byrne, W. I., & Kafai, Y.B. (2016). Computational Participation: Understanding Coding as an Extension of Literacy Instruction. *Journal of Adolescent & Adult Literacy, 59*(4), 371-375.

Burning Glass Technologies. (2016). Beyond point and click: The expanding demand for coding skills. Boston, MA: Retrieved from https://www.burning-glass.com/research-project/coding-skills/

Chamberlain, G. P. (2006). Researching strategy formation process: An abductive methodology. *Quality & Quantity, 40*, 289–301. https://doi.org/10.1007/s11135-005-8094-3

Chen, E. (2017). Martin Jenkins: Digital technologies and Hangarau Matihiko consultation (Final report). New Zealand: Retrieved from:

http://www.education.govt.nz/ministry-of-education/consultations-and-reviews/recent-consultations-and-reviews/digital-technology-consultation/

Cuny, J., Snyder, L., & Wing, J. M., (2010). Demystifying computational thinking for non-computer scientists. Unpublished manuscript in progress, referenced in http://www.cs.cmu.edu/~CompThink/resources/TheLinkWing.pdf

Curran, C. (2018a). Boosting game development, interactive media. Retrieved from https://www.beehive.govt.nz/release/boosting-game-development-interactive-media

Curran, C. (2018b). We need more women in ICT; Minister Clare Curran. Retrieved from: https://www.cio.co.nz/article/641394/we-need-more-young-women-ict-minister-clare-curran/

Curran J. (2017). A guide to programming languages for coding in class. Retrieved from: https://www.teachermagazine.com.au/articles/a-guide-to-programming-languages-for-coding-in-class

Darder, A. (2015). *Friere and education*. New York: Routledge.

Dewey, J. (1916). *Democracy and education: An introduction to the philosophy of education*. New York: Macmillan.

Drucker, P. F. (1989). *The new realities: In government and politics, in economy and business, in society, and in world view*. Oxford: Heinemann Professional Publishing.

Eisner, E. W., & Vallance, E. (1974). *Conflicting conceptions of curriculum*. Berkeley, CA: McCutchan.

Evans, T. (2014). *Teaching and Learning with Digital Technologies in the Intermediate School Classroom: An Activity Theory Analysis of Classroom Interactions* (Doctoral Thesis, Victoria University of Wellington). Retrieved from https://viewer.waireto.victoria.ac.nz/client/viewer/IE961126/rep/REP961134/FL961135?dps_dvs=1532899686615~207

Falloon, G. (2016). An analysis of young students' thinking when completing basic coding tasks using Scratch Jnr. On the iPad. *Journal of Computer Assisted Learning*, *32*(6), 576-593. http://dx.doi.org/10.1111/jcal.12155

Falloon, G., Hale, P., & Fenemor, T. (2016). Planning and implementing coding in the junior classroom for competency and thinking-skill development. *Set: Research Information for Teachers, 2016*(1), 8-16. http://dx.doi.org/10.18296/set.0031

Fessakis, G., Gouli, E., & Mavroudi, E. (2013). Problem solving by 5-6 years old kindergarten children in a computer programming environment: A case study. *Computers & Education, 63*, 87-97. https://doi.org/10.1016/j.compedu.2012.11.016

Fields, D., Vasudevan, V. & Kafai, V.B. (2010). The programmers' collective: Fostering participatory culture by making music videos in a high school Scratch coding workshop. *Interactive Learning Environments, 23*(5), 613-633. https://doi.org/10.1080/10494820.2015.1065892

Flick, U. (2013). *The SAGE handbook of qualitative data analysis.* Germany: SAGE.

Friesen, N., Henriksson, C. & Saevi, T. (2012). *Hermeneutic phenomenology in education*. Rotterdam, The Netherlands: Sense Publishers.

Gardiner, B. (2014). Adding coding to the curriculum. The New York Times, 23rd March. Retrieved from: http://nyti.ms/1ril4PT.

Gesthuizen, R. & Chandler, P.D. (2014). Students who are new to programming: What ideas do they have? *The Journal of Digital Learning and Teaching Victoria, 1(2),* 28-34.

Gilbert, J. (2005). *Catching the knowledge wave?: The knowledge society and the future of education*. Wellington: NZCER Press.

Haden, P. (2006). *The incredible rainbow spitting chicken: Teaching traditional programming skills through games programming*. Paper presented at the Australasian Conference on Computing Education, Hobart, Australia. Retrieved from https://www.researchgate.net/publication/234779342_The_incredible_rainbow_spitting_chicken_Teaching_traditional_programming_skills_through_games_programming

Hagge, J. (2017). Scratching beyond the surface of literacy: Programming for early adolescent gifted students. *Gifted Child Today*, *40*(3), 154-162. https://doi.org/10.1177/1076217517707233

Hakala, J.T., Uusikylä, K, & Järvinen, E. (2015). Neoliberalism, Curriculum Development and Manifestations of "Creativity". *Improving Schools, 18*(3), 250-262.

Hayes, J., & Stewart, I. (2016). Comparing the effects of derived relational training and computer coding on intellectual potential in school-age children. *British Journal of Educational Psychology*, *86*(3), 397-411. http://dx.doi.org/10.1111/bjep.12114

Johnson, M., Wood, A., & Sutton, P. (2014). *Digital technology in school.* (2014 report). New Zealand: 2020 Communications Trust

Kalelioğlu, F. (2015). A new way of teaching programming skills to K-12 students: Code.org. *Computers in Human Behavior*, *52*, 200. https://doi.org/10.1016/j.chb.2015.05.047

Kafai, Y.B. (2016). From computational thinking to computational participation in k-12 education. *Communications of the ACM*. 59(8), 26-27. https://doi.org/10.1145/2955114

Kan, A. & Ongchoco, D. (2017). 8 student startups using tech for social good. Retrieved from https://blog.codingitforward.com/8-student-startups-using-tech-for-social-good-762e5d155381

Ke, F. (2014). An implementation of design-based learning through creating educational computer games: A case study on mathematics learning during design and computing. *Computers & Education, 73*, 26-39.

Kea (2017). Interview with World Class New Zealander Maru Nihoniho. Retrieved from https://www.keanewzealand.com/our-stories/interview-with-world-class-new-zealander-maru-nihoniho-2/

Kules, B. (2016). *Computational thinking is critical thinking: Connecting to university discourse, goals, and learning outcomes.* Proceedings of the Association for

Information Science and Technology, Copenhagen, Denmark. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/pra2.2016.14505301092

Lambic, D. (2010). Presenting practical application of Mathematics by the use of programming software with easily available visual components. *Teaching Mathematics and Its Applications, 30*, 10-18. https://doi.org/10.1093/teamat/hrq014

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

McDowall, S. & Hipkins, R. (2018). How the key competencies evolved over time: Insights from the research. New Zealand Council for Educational Research. Retrieved from: http://www.nzcer.org.nz/research/publications/key-competencies-insights

Marsh, C. J. & Willis, G. (1995). *Curriculum: Alternative approaches, ongoing issues*. Englewood Cliffs, N.J: Merrill.

Mayer, R. E. (1988). *Teaching and learning computer programming: Multiple research perspectives*. Mahwah, NJ: Erlbaum.

McGuiness Institute. (2016). *History of education in New Zealand: Prepared to accompany the infographic: Timeline of significant events in the history of education in New Zealand, 1867–2014*. Wellington, New Zealand: McGuiness Institute.

McPhail, G. & Rata, E., (2016). Comparing curriculum types: 'Powerful knowledge' and 21st century learning'. *New Zealand Journal of Educational Studies.* 51(1), 53–68. https://doi.org/10.1007/s40841-015-0025-9

Miles, M. B., Huberman, A. M., & Saldaña, J. (2014). *Qualitative data analysis: A methods sourcebook* (Third edition.). California: SAGE.

Ministry of Education (2007). *The New Zealand Curriculum.* Wellington, New Zealand: Ministry of Education.

Ministry of Education (2014). *Future-focused learning in connected communities.* A report by the 21st Century Learning Reference Group. New Zealand: Retrieved from https://www.education.govt.nz/assets/.../FutureFocusedLearning30May2014.pdf

Ministry of Education. (2017a). *Digital Technologies Hangarau Matihiko*. Wellington, New Zealand: Ministry of Education.

Ministry of Education. (2017b). Key competencies. Retrieved from http://nzcurriculum.tki.org.nz/Key-competencies#collapsible2

Ministry of Education (2018). New digital readiness professional support programme launched for educators. Retrieved from: https://www.education.govt.nz/news/new-digital-readiness-professional-support-programme-launched-for-educators/

Ministry of Youth Affairs (2009) *Youth participation: Benefits for your organisation*. Wellington, New Zealand: Ministry of Youth Affairs. Retrieved from: http://www.myd.govt.nz/resources-and-reports/publications/youth-participation-benefits.html

Moreno-León, J., Robles, G., & Román-González, M. (2016). Code to learn: Where does it belong in the k-12 curriculum? *Journal of Information Technology Education, 15,* 283-303. Retrieved from: http://www.informingscience.org/Publications/3521

Mow, I.T.C. (2008). Issues and difficulties in teaching novice computer programming. Innovative Techniques Technology. *E-learning, E-assessment and Education* 199–204.

Nelson, M., Sahami, M., & Wilson, C. (2016). A New Framework to Define K-12 Computer Science Education. *Communications of the ACM*, *59*(4), 20.

Netsafe. (2015). Digital citizenship and digital literacy. Retrieved from: https://www.netsafe.org.nz/digital-citizenship-and-digital-literacy/

Palumbo, D., & Michael Reed, W. (1991). The effect of BASIC programming language instruction on high school students' problem solving ability and computer

anxiety. *Journal of Research on Computing in Education, 23*(3), 343-372. http://dx.doi.org/10.1080/08886504.1991.10781967

Pellas, N. & Peroutseas, E., (2017). Leveraging Scratch4SL and Second Life to motivate high school students' participation in introductory programming courses: Findings from a case study. *New Review of Hypermedia and Multimedia, 23*(1), 51-79. https://doi.org/10.1080/13614568.2016.1152314

Popat, S. & Starkey, L. (2017). *Learning to code, or coding to learn.* Manuscript submitted for publication.

Psycharis, S., & Kallia, M. (2017). The effects of computer programming on high school students' reasoning skills and mathematical self-efficacy and problem solving. *Instructional Science, 45*(5), 583-602.

Sáez-López, J-M., Román-González, M., & Vázquez-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

Schiro, M. (2008). *Curriculum theory*: *Conflicting visions and enduring concerns.* Los Angeles: Sage Publications.

Schoenfeld, A H. (2013). Reflections on Problem Solving Theory and Practice. *The Mathematics Enthusiast*: 10(1). Retrieved from: https://scholarworks.umt.edu/tme/vol10/iss1/3

Sharma, B. R. (2004). *Strategies of non-formal education*. New Delhi: Sarup & Sons.

Siemens, G. (2005). Connectivism: A learning theory for the digital age. *International Journal of Instructional Technology and Distance Learning.* Retrieved from: http://www.itdl.org/Journal/Jan_05/article01.htm

Starkey, L. (2010). Teachers' pedagogical reasoning and action in the digital age. *Teachers and Teaching,* 16(2), 233-244. http://dx.doi.org/10.1080/13540600903478433

Starkey, L. (2011). Evaluating learning in the 21st century: a digital age learning matrix. Technology, Pedagogy and Education, 20(1), 19-39. http://dx.doi.org/10.1080/1475939X.2011.55402

Starkey, L. (2012). Teaching and learning in the digital age. New York, NY: Routledge.

Sterling, L. (2016, August). Coding in the curriculum: Fad or foundational? Paper presented at the ACER research conference, Melbourne, Australia. Retrieved from: https://research.acer.edu.au/research_conference/RC2016/9august/4

Theodoraki, A. & Xinegalos, S. (2014). Studying students' attitudes on using examples of game source code for learning programming. *Informatics in Education, 13*(2), 265-277. http://dx.doi.org/10.15388/infedu.2014.07

Te Kete Ipurangi (2017). Technology in the New Zealand curriculum. Retrieved from: http://nzcurriculum.tki.org.nz/The-New-Zealand-Curriculum/Technology/Progress-outcomes#collapsible2

Tucker, A. (2003). *A Model Curriculum for K--12 Computer Science: Final Report of the ACM K-12 Task Force Curriculum Committee*. New York: Association for Computing Machinery.

Tuomi, P., Multisilta, J., Saarikoski, P., & Suominen, J. (2018). Coding skills as a success factor for a society. *Education and Information Technologies*, *23*(1), 419-434.

Von Glasersfeld, E. (1995). *Radical constructivism: A way of knowing and learning*. London: Falmer Press.

Webb, M., Fluck, A., Cox, M., Angeli-Valanides, C., Malyn-Smith, J., Voogt, J., and Zagami. J. (2015). *Technology advanced quality learning for all: EDUsummit 2015 summary report.* New Zealand: Retrieved from www2.curtin.edu.au/edusummit/edusummit2015-ebook.pdf

Webb, M., Bell, T., Davis, N., Katz. Y.J., Reynolds, N., Chambers, D.P,…Mori, N. (2017a) Computer Science in the School Curriculum: Issues and Challenges. *IFIP Advances in Information and Communication Technology, 515*. https://doi-org.helicon.vuw.ac.nz/10.1007/978-3-319-74310-3_43

Webb, M., Davis, N., Bell, T., Katz, Y.J., Reynolds, N., Chambers, D.P., & Syslo, M.M. (2017b). Computer Science in K-12 School Curricula of the 2lst Century: Why, What and When? *Education and Information Technologies, 22*(2), 445-468. https://doi.org/10.1007/s10639-016-9493-x

Wilson, A., & Moffat, D. C. (2010). *Evaluating scratch to introduce younger schoolchildren to programming.* In the Proceedings of the Psychology of Programming Interest Group Workshop, September, Madrid, Spain.

World Economic Forum. (2018). *Towards a reskilling revolution: A future of jobs for all*. (Report No.1). Switzerland: Retrieved from: www.weforum.org/reports/towards-a-reskilling-revolution

World Economic Forum. (2016a). *The future of jobs: Employment, skills and workforce strategy for the fourth industrial revolution*. (Report No.1). Switzerland: Retrieved from: www3.weforum.org/docs/WEF_Future_of_Jobs.pdf

World Economic Forum. (2016b). *Mastering the fourth industrial revolution.* (Report No.1). Switzerland: Retrieved from: https://www.weforum.org/reports/world-economic-forum-annual-meeting-2016-mastering-the-fourth-industrial-revolution

Xia, B. S., (2017). An in-depth analysis of teaching themes and the quality of teaching in Higher Education: Evidence from the programming education environments. *International Journal of Teaching and Learning in Higher Education, 29*(2), 245-254.

Young, M., Lambert D., Roberts C., & Roberts, M. (2014). Knowledge and the future school: curriculum and social justice. London: Bloomsbury

Zuga, K.F. (1992). Social reconstruction curriculum and technology education. *Journal of Technology Education, 3(2),* 48-58.

Appendix

Appendix 1. Interview guide and questions

1. Contact principals regarding research and approval to work with a teacher of coding at Year 7/8. (Provide information sheet and consent form)
2. Discuss research with the teacher (provide information sheet and consent form) and collect relevant/available school documentation on coding. Teacher to identify a range of students to invite to participate
3. Teacher to issue information sheet and consent form to students to take home and discuss with parents. Student and parent signature required for consent
4. Once signed consent forms have been obtained, hold focus group at a venue at school as agreed by with the school/teacher with food and drink available.
5. Place voice recorders in a suitable place to record audio.
6. Introduction and confidentiality/voluntary aspect discussed.

The following is an outline of the focus group questions however there was further promoting and probing throughout to obtain the best data;

Whole group discussion focused on:

When did you first hear about or start learning coding at school?

What do you think coding is all about?

What types of activities do you work on in coding during class?

What do you enjoy about these activities?

How do you think completing these activities has been useful for you?/ How do you think coding helps you? (further probing)

- Students in pairs asked to brainstorm on paper (could draw these or take photos of their work if available): What have you made when learning coding at school that you couldn't before? Whole group discussion:

Sharing what has been put on their paper, students can add to theirs if others remind them of something they have done. How did you learn how to do this?

How have you used coding outside of class? If you haven't, why not? Or how do you think you could you use coding outside of class? How do you see yourself using coding outside of class? (further probing)

- Draw arrows from school activities that have/could inform outside school activities
- Discussion about what has been drawn:

What else do you want to learn about coding?

- Discuss summary of two or three key points before participants leave the focus group or the main points after each question to see if they agree