The Development of Computational Thinking: A Constructionist School Computer Programming Initiative in a Girls' Primary School

Claire Carroll

Bachelor of Science in Physical Education (Hons)

Master of Arts in Education (Hons)

Submitted to: Mary Immaculate College, University of Limerick in fulfilment of the requirements for the Structured PhD in Education

January 2025

Supervisor: Prof. Aisling Leavy, Department of STEM Education, Mary Immaculate College, University of Limerick

Abstract

The Development of Computational Thinking: A Constructionist School Computer Programming Initiative in a Girls' Primary School

Claire Carroll

In September 2025, the newly established Science, Technology, Engineering and Mathematics (STEM) curriculum was rolled out in primary schools across Ireland with the intent to provide opportunities for students to learn the fundamentals of programming and build on their computational thinking skills. To date, much of the research on the learning of programming and computational thinking has been conducted with older students or in informal settings. Therefore, this study was conducted in a primary school context to investigate if primary school students could develop their computational thinking through engaging in programming activities. Sixty-seven students from third, fifth and sixth classes in an Irish primary school took part in a ten-week Scratch programming initiative designed to foster their computational thinking. Various data collection methods were adopted to capture the multidimensional nature of computational thinking, including artefact analysis, artefact-based interviews, questionnaires and an observation diary. Brennan and Resnick's (2012) three computational thinking dimensions (concepts, practices and perspectives) were used to interpret the data, before deductive and inductive analyses were adopted to conceptualise the nature of the dimensions. Findings indicate that the primary school students in this study excelled at both synchronisation and parallelism, with many students illustrating an understanding of more challenging concepts such as state synchronisation and parallel launching of multiple scripts. However, conditional loops and variable initialisation caused significant difficulty, with further analysis revealing that students would require further scaffolding to acquire these concepts. This research gave insights into the poorly defined computational practices and perspectives, providing teachers, researchers and policymakers with a more comprehensive picture of how these dimensions can manifest in a primary school classroom. In presenting thick descriptions of these students' experience, this research also highlights pedagogical factors that shaped and supported their computational thinking development, including programming language, project type, level of scaffolding and peer engagement. Finally, a reconfigured computational thinking framework is proposed, identifying aspects of computational thinking that were not sufficiently captured by the Brennan and Resnick (2012) framework.

Declaration

I hereby declare that this thesis represents my own work and has not been submitted in whole or in part, by me or another for the purpose of obtaining any other qualification.

Signed: Claire Carroll **Date:** 25/04/2025

Acknowledgements

I would like to express my deepest gratitude to those who supported and guided me in the completion of this thesis.

Firstly, thank you to my fabulous supervisor Professor Aisling Leavy whose invaluable advice, continuous support and patience helped me to navigate this research journey. Your breadth and depth of expertise, generosity of time and spirit, and of course good humour, enabled me to explore and develop my researcher voice.

To my research participants, I greatly appreciate the time and effort that you contributed to this study. I hope that your enthusiasm during this programming initiative will be an inspiration for students, teachers and parents who are embarking on their own programming journeys.

To my friends and colleagues who supported me throughout the research process, thank you for the personal encouragement, moments of inspiration and professional support. In particular, thank you to my writing buddy Breed Murphy who provided friendship, encouragement and accountability during the writing process.

Finally, I would like to acknowledge my family for their encouragement and support. To Paddy thank you for all your love and support, especially for your assurances that I would get there, your good humour, and providing a necessary distraction from the mental endeavours. To my parents Ann and Richard, you instilled in me the value of learning and encouraged me to pursue my dreams, thank you.

"The more that you read, the more things you will know. The more that you learn, the more places you'll go." (Dr. Seuss)

Table of Contents

ABSTRACT ..... **I**

DECLARATION ..... **II**

ACKNOWLEDGEMENTS ..... **III**

TABLE OF CONTENTS ..... **IV**

LIST OF FIGURES ..... **XII**

LIST OF TABLES ..... **XIV**

LIST OF ABBREVIATIONS ..... **XVI**

CHAPTER 1: INTRODUCTION **1**

1.1 Introduction 1

1.2 The Technology Revolution 1

1.3 Context of the Research Study 2

1.4 Computational Thinking: The Educational Concept 6

1.5 Rationale for and Original Contribution of the Research 11

1.6 Research Questions 14

1.7 Structure of Dissertation 14

CHAPTER 2: TECHNOLOGY IN IRISH SCHOOLS **16**

2.1 Introduction 16

2.2 Technology in Irish Society 16

2.3 Education for the 21ST Century: Incorporating Technology into Education 19

2.4 Policy on Technology in Education 21

2.4.1 Policy Development in the IT Schools 2000 Era (1997-2009) 21

2.4.2 Impact of Policy Development on Technology Integration in Education 29

2.4.3 Policy Development in the Digital Strategy for Schools era (2010-2020) 34

2.4.4 Policy Development post Digital Strategy for Schools 2015-2020 46

2.5 Policy Development in STEM Education 70

2.6 Gender Disparity in Technology Education 73

2.6.1 Factors Influencing Girls' Participation in Computing 74

2.6.2 Gender Equity in Computing – What Would That Look Like and How Can We Get There? 76

2.7 Computing in the Classroom 79

2.8 Computational Thinking in the Curriculum 79

2.8.1 Computational Thinking in Irish Curricula 80

2.8.2 Computational Thinking in International Curricula 81

2.8.3 Computational Thinking in the Context of this Study 88

2.9 Approaches to Developing Computational Thinking 100

2.9.1 Developing Computational Thinking through Unplugged Activities 100

2.9.2 Developing Computational Thinking through Plugged Activities 109

2.9.3 Motivation to Engage with Technology 140

2.10 Learning from Lessons of the Past 145

2.11 Conclusion 146

CHAPTER 3: LEARNING AND TEACHING PROGRAMMING.....147

3.1. Introduction.....147

3.2 Learning.....147

3.2.1 Approaches to Learning.....148

3.3 Teaching.....167

3.3.1 Pedagogy for Teaching Computational Thinking through Programming.....167

3.3.2 Programming Language.....171

3.4 Conclusion.....178

CHAPTER 4: RESEARCH DESIGN AND METHODOLOGY.....180

4.1 Introduction.....180

4.2 Purpose of this Research.....180

4.3 Research Questions.....181

4.4 Philosophical Assumptions and Interpretive Framework.....181

4.5 Case Study Research.....182

4.6 Purposive Sampling.....185

4.7 Research Participants.....186

4.8 The Programming Initiative.....188

4.8.1 Programming Language.....191

4.9 Conceptual Framework.....193

4.10 Data Collection Methods.....194

4.10.1 Artefact Analysis *196*

4.10.2 Artefact-Based Interviews *201*

4.10.3 Design Scenarios *204*

4.10.4 Building on the Assessment Approaches of Brennan and Resnick (2012) *206*

4.10.5 Questionnaires *206*

4.10.6 Observation Diary *209*

4.11 Data Analysis 211

4.11.1 Organising and Preparing the Data for Analysis *211*

4.11.2 Reading or Looking at all the Data *211*

4.11.3 Start Coding all the Data *212*

4.11.4 Generate a Description and Themes *217*

4.11.5 Represent the Description and Themes *218*

4.12 Validation Strategies 218

4.12.1 Corroborating Evidence through Triangulation of Multiple Data Sources *220*

4.12.2 Clarifying Researcher Bias or Engaging in Reflexivity *221*

4.12.3 Member Checking or Seeking Participant Feedback *225*

4.12.4 Prolonged Engagement and Persistent Observation in the Field *226*

4.12.5 Generating a Rich, Thick Description *226*

4.13 Ethical Considerations 227

4.13.1 Minimising Risk of Harm *228*

4.13.2 Informed Consent and the Right to Withdraw *229*

4.13.3 Fair Representation and Reporting Honestly *230*

4.13.4 Confidentiality and Anonymity *231*

4.14 Conclusion 231

CHAPTER 5: FINDINGS AND DISCUSSION 232

5.1 Introduction 232

5.2 Computational Concepts 232

5.2.1 Computational Concepts - Introduction *232*

5.2.2 Synchronisation – Making things happen as we want *237*

5.2.3 Parallelism - Doing Two Things at the One Time *242*

5.2.4 Data - Failing to Initialise *248*

5.2.5 Thinking Logically *253*

5.2.6 User Interactivity *257*

5.2.7 Flow Control *261*

5.2.8 Abstraction and Problem Decomposition *267*

5.2.9 Computational Concepts - Conclusion *269*

5.3 Computational Practices 270

5.3.1 Computational Practices - Introduction *270*

5.3.2 Abstracting and Modularising *270*

5.3.3 Experimenting and Iterating *275*

5.3.4 Testing and Debugging *280*

5.3.5 Reusing and Remixing *288*

5.3.6 Computational Practices – Conclusion *294*

5.4 Computational Perspectives 295

5.4.1 Computational Perspectives - Introduction *295*

5.4.2 Expressing - "It kind of helps children become a bit more creative!!!!!" *296*

5.4.3 Connecting *301*

5.4.4 Questioning *314*

5.4.5 Computational Identity *319*

5.4.6 Computational Perspectives – Conclusion *345*

5.5 Conclusion 347

CHAPTER 6: OUTCOMES, IMPLICATIONS AND CONCLUSIONS 348

6.1 Introduction 348

6.2 Context of the Research 348

6.3 Summary of Key Findings 351

6.4 Contribution to the Literature 355

6.5 Implications for Theory 357

6.6 Implications for Practice 360

6.6.1 Balancing Direct Instruction and Self-Discovery 360

6.6.2 Ensuring Children are Appropriately Challenged 363

6.6.3 Appropriate Embedding of Computational Thinking into the Initiative 363

6.6.4 Building the Requisite Programming Knowledge 364

6.6.5 Recognising the Strengths and Weaknesses of the Programming Languages 365

6.7 Implications for Policy 366

6.7.1 Professional Development Policy 366

6.7.2 Ensuring Digital Equity 367

6.7.3 Ensuring Cohesion between Curricular Areas 369

6.7.4 Ensuring Consistency in Defining Computational Thinking 371

6.8 Recommendations for Further Research 371

6.9 Conclusion 373

REFERENCES 375

APPENDICES 437

Appendix A – Previous Research on Programming Initiatives 437

Appendix B - Student Interview Protocol (Brennan et al. 2014) 438

Appendix C - Rubric for Assessing Evolving Computational Practices (Brennan et al. 2014) 441

Appendix D - Transcript of Artefact-based Interview with Isabela and Katya 443

Appendix E – Student Pre-Initiative Questionnaire 447

Appendix F – Student Post-Initiative Questionnaire *449*

Appendix G – Parent Questionnaire *451*

Appendix H – Participating Class Teacher Questionnaire *453*

Appendix I – Additional Teacher Questionnaire *456*

Appendix J - Diary Entry from Week 8 of the Programming Initiative *458*

Appendix K - An Example of the Coding Process Adopted in this Study *460*

Appendix L - Permission Request to the Board of Management *464*

Appendix M – Participant Information Sheets *468*

Appendix N – Participant Consent Forms *474*

List of Figures

Figure 1.1: *Teaching Approaches to Foster Computational Thinking (NCCA 2024a, p.23-24)* 11

Figure 2.1: *Digital Competence Learner Outcomes identified in the Digital Learning Framework for Primary Schools (DES 2017b, p. 5)* 55

Figure 2.2: *Digital Competence Learner Experiences identified in the Digital Learning Framework for Primary Schools (DES 2017b, p.6)* 56

Figure 2.3: *Agreement and disagreement around what Computational Thinking should be (Curzon et al. 2019, p.515)* 60

Figure 2.4: *Concepts and Processes Model (C - Concepts, P - Processes and Production Skills) (Waite and Quille 2022a, p.14)* 66

Figure 2.5: *Examples of how Being a digital learner can be developed through mathematics learning experiences (DES 2023b, p.6)* 68

Figure 2.6: *Learning experiences included in the NCCA (2023b) progression continua that incorporate digital technology* 69

Figure 2.7: *Computational Concepts and Approaches (Csizmadia et al. 2015, p.8)* 84

Figure 2.8*: Definition of Computational Thinking (CSTA and ISTA 2011, p.13)* 86

Figure 2.9: *Sequence of Instructions* 89

Figure 2.10: *A loop inducing multiple executions of the enclosed commands* 90

Figure 2.11: *Conditional within a catching game* 90

Figure 2.12: *Operators commonly used in game programming* 91

Figure 2.13: *Using variables to store numbers and strings* 92

Figure 2.14: *Actions triggered by events* 92

Figure 2.15: *Parallelism within one sprite* 93

Figure 2.16: *The My Block function allows us to create more efficient programs* 93

Figure 2.17: *When the green flag is clicked the sprite returns to the centre of the screen (0,0)* 94

Figure 2.18*: A script using broadcast to activate scripts with the matching when I receive block* 95

Figure 3.1: *The processes of assimilation and accommodation (Gould 2012, p.45)* 159

Figure 3.2*: Use-Modify-Create Model (Lee et al. 2011, p.35)* 169

***Figure 3.3**: Continuum of Approaches for Teaching Programming (Waite 2019)* _____ *171*

***Figure 4.1:** Storyboard for Project Development* _____ *191*

***Figure 4.2:** The offline Scratch user interface* _____ *193*

***Figure 4.3:** Dr. Scratch feedback on a level 2 developing project (8-14 points)* _____ *199*

***Figure 4.4:** Inductive Codes describing Computational Practices*_____ *216*

***Figure 4.5:** Inductive Codes describing Computational Perspectives* _____ *217*

***Figure 4.6:** Mapping of Emergent Codes and Data Sources*_____ *218*

***Figure 4.7:** Strategies for Validation in Qualitative Research (Creswell and Poth 2018, p. 260)* ___ *220*

***Figure 5.1:** Code illustrating misunderstandings relating to time synchronisation* _____ *239*

***Figure 5.2:** Broadcast and Receive Parallelism in Alice in Wonderland* _____ *245*

***Figure 5.3:** Design Scenario to assess understanding of parallelism* _____ *245*

***Figure 5.4:** Design Scenario to assess understanding of the If-else Logic concept*_____ *254*

***Figure 5.5**: Stack of blocks using the when this sprite clicked block*_____ *258*

***Figure 5.6:** Design Scenario to assess understanding of User Interactivity* _____ *259*

***Figure 5.7:** Stack of blocks using the forever block*_____ *264*

***Figure 5.8:** Anna's modified apple sprite* _____ *290*

***Figure 5.9:** Students' Enjoyment of the Programming Initiative* _____ *329*

***Figure 5.10:** 3rd Class Students' Enjoyment of the Programming Initiative* _____ *330*

***Figure 5.11**: 5th Class Students' Enjoyment of the Programming Initiative* _____ *330*

***Figure 5.12:** 6th Class Students' Enjoyment of the Programming Initiative* _____ *330*

***Figure 5.13:** Students' Views on Learning Programming in School* _____ *331*

List of Tables

Table 2.1: *ICT policy reports and initiatives published in Ireland (1997-2009)* 22

Table 2.2: *National (central) investment in ICT in primary and post-primary schools (1998 - 2017)* 23

Table 2.3: *Student-Computer ratio* in primary schools (1998-2013)* 33

Table 2.4: *International reports on ICT in education published 2005-2017* 36

Table 2.5: *The key priorities of the Digital Strategy for Schools 2015-2020* 37

Table 2.6: *The UNESCO ICT Competency Framework for Teachers (UNESCO 2011a, p.3)* 39

Table 2.7: *ICT policy reports and initiatives published in Ireland (2009-2020)* 40

Table 2.8: *Key findings from 'Digital Learning 2020: Reporting on Practice in Early Learning and Care, Primary and Post-Primary Contexts' (adapted from Butler and Leahy 2022a)* 46

Table 2.9: *ICT policy reports and initiatives published in Ireland (2021-2024)* 47

Table 2.10: *The objectives of the Digital Strategy for Schools to 2027 (DES 2022b, pp.21,43,52)* 52

Table 2.11: *The four pillars of policy development and action* 71

Table 2.12: *Actions necessary to continue the implementation of STEM across the education system (Government of Ireland 2023, p.7)* 73

Table 2.13: *Proposed Shifts in Emphasis to Achieve Gender Balance in STEM Education (Goos et al. 2020, p.61)* 77

Table 4.1: *Number of participants per class* 186

Table 4.2*: Weekly schedule of activities and the targeted computational thinking skills* 188

Table 4.3: *Daily structure of activities* 190

Table 4.4: *Synchronisation of the data collection methods with the initiative phase* 196

Table 4.5: *Competence Levels for Computational Thinking Concepts (Moreno-León et al. 2015, p.6)* 199

Table 4.6: *Final Projects Analysed by Dr. Scratch* 200

Table 4.7: *Artefact-Based Interview Participants* 203

Table 4.8: *Connecting the Assessment Approaches to the Dimensions of Computational Thinking (adapted from Brennan and Resnick 2012, p.22)* 206

Table 4.9: *Codes adopted in the deductive coding process* 213

***Table 4.10**: Moreno-León et al. (2015) Computational Thinking Components mapped onto those of Brennan and Resnick (2012)* _____ *214*

***Table 5.1:** Dr. Scratch Scoring Assignment for Computational Concept Development*_____ *233*

***Table 5.2:** The Dr. Scratch scores for individual Computational Concepts* _____ *234*

***Table 5.3:** Project Description and Scratch Scores for the Analysed Final Projects* _____ *235*

***Table 5.4:** Dr. Scratch scores for the synchronisation concept* _____ *238*

***Table 5.5:** Dr Scratch scores for the Flow Control concept* _____ *262*

***Table 6.1:** Reconfigured Computational Thinking Framework (adapted from Brennan and Resnick 2012)* _____ *359*

List of Abbreviations

STEM:	Science, Technology, Engineering, and Mathematics
STEMerg:	Science, Technology, Engineering, and Mathematics Education Review Group
NCCA:	National Council for Curriculum and Assessment
IDA:	Industrial Development Authority
DES:	Department of Education*
ICT:	Information and Communications Technology
CESI:	Computer Education Society of Ireland
ASTI:	Association of Secondary Teachers of Ireland
CEB:	Curriculum and Examinations Board
INTO:	Irish National Teachers' Organisation
NCTE:	National Centre for Technology in Education
UNESCO:	United Nations Educational, Scientific and Cultural Organisation
OECD:	Organisation for Economic Cooperation and Development
DCNER:	Department of Communications, Energy & Natural Resources**
EGFSN:	Expert Group on Future Skills Needs
LOGO:	Language of Graphics Oriented
NRC:	National Research Council (US)
ISTE:	International Society for Technology in Education
CSTA:	Computer Science Teachers Association
DJEI:	Department of Jobs, Enterprise and Innovation
NCC:	National Competitiveness Council
PDST-TIE	Professional Support Service for Teachers – Technology in Education
DCYA	Department of Children and Youth Affairs

DEIS	Delivering Equality of Opportunity In Schools
DLF	Digital Learning Framework
CAS	Computers at School
CPD	Continuing Professional Development
ACARA	Australian Curriculum, Assessment and Reporting Authority
ACM	Association for Computing Machinery
CS4HS	Computer Science for High Schools
NCWIT	National Center for Women and Information Technology
ICS	Irish Computer Society
ITEST	Innovative Technology Experiences for Students and Teachers
PRIMM	Predict Run Investigate Modify Make
EAL	English as an Additional Language
ENL	English as a Native Language
SERA	Scottish Educational Research Association
BERA	British Educational Research Association
DCYA	Department of Children and Youth Affairs***

*Department of Education (1924-1997), Department of Education and skills (1924-2010), Department of Education and Skills (2010-2020) and Department of Education (2020-present).
** Since renamed Department of the Environment, Climate and Communications
*** Since renamed Department of Children, Equality, Disability, Integration and Youth

Chapter 1: Introduction

1.1 Introduction

This study embraces the potential of the primary school to nurture and develop students' computational thinking skills. Adopting a pragmatic approach, this research seeks to explore what computational concepts, practices and perspectives Irish primary school students can develop through carefully designed programming learning experiences. In this first chapter, this research is contextualised by providing a review of the pertinent national policy and educational research, before outlining the background and rationale for this study. The chapter begins by acknowledging the growing importance of the Science, Technology, Engineering and Mathematics (STEM) education both nationally and internationally. The author recognises the opportunity that the consequential resurgence of teaching programming in schools provides for developing students' computational thinking skills. Following this the aims and objectives of this research study are presented, alongside the researcher's personal motivations for embarking on this research journey. The chapter concludes with an overview of the structural layout of the thesis, identifying the central components of each chapter.

1.2 The Technology Revolution

> Global society has entered the Fourth Industrial Revolution, in which artificial intelligence (AI), big data, robotics, the Internet of Things (IoT), and other emerging technologies will forever change the way humans live and work.
>
> (Leavy et al. 2023)

Technology is prolific in modern society, reshaping nearly every facet of our daily lives. The transformative impact of technology extends from the convenience of online shopping and the efficiency of digital banking to the vast array of

entertainment options available at our fingertips and the new avenues for communication and connection. Moreover, productivity has soared thanks to automation and cloud computing, allowing businesses to operate more efficiently and adapt to changing market demands. Artificial intelligence is further revolutionising industries by streamlining processes and providing insights that were previously unimaginable. Ireland has emerged as a prominent player in this fast evolving global technology landscape, fostering a thriving ecosystem of innovators and entrepreneurs. As of 2024, there are approximately 2,200 indigenous technology start-ups and scale-ups operating in the country, according to Bank of Ireland (2024). The impact of this innovation extends beyond traditional sectors, reaching into education through the use of gamification, assistive technologies, and immersive learning experiences. These tools are not only enhancing engagement but also expanding the boundaries of educational imagination. As technology continues to evolve, it is evident that its influence will only deepen. Thus, the challenge for educational practitioners and policymakers is to navigate this exciting yet complex landscape ensuring that their practices are aligned with the future demands of society.

1.3 Context of the Research Study

This research was originally motivated by several significant developments in the field of STEM education, which have reshaped the landscape of teaching and learning in STEM. The rise of STEM education in the 21st century is a direct response to the rapidly evolving technological landscape and the increasing demand for a workforce equipped with the skills and knowledge to thrive in a technology-driven economy. According to STEMerg (2016), expertise in STEM “is necessary to drive our economic ambitions, support innovation and provide the foundations for

future prosperity" (p.3). The increased recognition of STEM education has impelled policymakers and educators to rethink curricula and pedagogical approaches to encourage more students, particularly girls, to pursue STEM subjects. Research highlighting the impact of early exposure to STEM concepts on female participation further emphasised the need for strategic interventions in primary schools. In response to these insights, the Irish government initiated a review of existing STEM education practices seeking to identify strategies to enhance uptake of 'gateway' STEM subjects, including coding and computer science (DES 2016b). This review aimed not only to assess the existing landscape but also to explore innovative approaches that could attract a more diverse cohort of students into the STEM pipeline. The findings of the review were published in a report titled, 'STEM Education in the Irish School System'.

The report highlighted three critical observations which were relevant to this study. Firstly, it identified a pronounced gender imbalance in the number of students taking Leaving Certificate Technology subjects, with boys greatly outnumbering girls. This disparity not only raises concerns about equity and representation in technology fields, but also indicates a need for targeted interventions to encourage greater female participation in these subjects. Secondly, the report drew attention to the underrepresentation of women in the STEM workforce, emphasising that despite advances in gender equality in many sectors, significant gaps remain in STEM-related professions. This underrepresentation reduces diversity in perspectives and creativity, supports unhelpful stereotypes, and denies girls of appropriate female role models in STEM. Thirdly, the report acknowledged the vibrant informal STEM sector in Ireland, characterised by initiatives such as CoderDojo, which aim to inspire and engage young people in technology. However, the report advised that the

full potential of these initiatives were not being realised due to their disconnect from the formal education curriculum. By bridging the gap between informal and formal STEM education, there is an opportunity to foster greater engagement with STEM among students, particularly girls, and to create a more inclusive and representative workforce in the long term. Overall, the observations presented in the report advocate for a multifaceted approach to enhance gender equity in STEM fields, promote the integration of informal learning opportunities, and harness the full potential of Ireland's STEM sector. The publication of this report prompted a series of comprehensive reviews and reports aimed at tackling the persistent underrepresentation of women, and other marginalised groups in STEM disciplines, and increasing the teaching of programming in schools. Several important findings have emerged from these reports that informed this study.

Goos *et al.* (2020) conducted a comprehensive study examining the various factors that contribute to the underrepresentation of girls in STEM disciplines. Their research illuminated the barriers that hinder girls' participation, including societal stereotypes, lack of role models, and insufficient encouragement in educational settings. They advocated for targeted interventions aimed at enhancing girls' interest, motivation, and enjoyment in STEM subjects, building positive STEM identities for girls and dismantling negative stereotypes (Goos *et al*. 2020). The Department of Education identified four important areas for action based on the findings of the report (DES 2022c). They recognised the need to improve equity of access and inclusion across all STEM disciplines by effecting whole-school cultural change. The Department acknowledged the importance of providing robust support for STEM educators, recognising the pivotal role of educators in providing engaging and impactful experiences to inspire and equip the next generation of STEM learners.

The report highlighted the need to ensure equitable access to inspiring STEM experiences which are crucial in promoting future engagement and success. Finally, the Department pledged to support a societal shift to tackle existing barriers to gender equity in STEM.

While this research on gender imbalance in STEM fields was being conducted, another significant research initiative focused on how to effectively integrate programming into the primary school curriculum. Initial efforts concentrated on embedding programming within the primary mathematics curriculum, drawing inspiration from successful models in Finland and Northern Ireland (NCCA 2018). This approach sought to leverage the logical and computational thinking skills that are fundamental to both domains. However, while numerous reports recognised the intrinsic connections between mathematics and programming, they also raised concerns about framing programming as merely a mathematical discipline (NCCA 2019a; Millwood *et al*. 2018; NCCA 2018). Consequently, despite the inclusion of programming foundations in the 'Primary Mathematics Curriculum: Draft Specification Junior Infants to Second Class for Consultation' (NCCA 2017a), explicit references to programming were notably absent in the updated primary mathematics curriculum (DES 2023b). Instead, the National Council for Curriculum and Assessment (NCCA) advocated for the creation of a dedicated curriculum space for digital learning, facilitating a more comprehensive treatment of programming concepts, computational thinking methodologies, and the creative and practical applications of computing (NCCA 2019a). This shift reflects a growing recognition of the importance of digital literacy in education, emphasising the need to prepare young learners for a future increasingly shaped by technology.

Following this development, the draft science, technology, and engineering education specification was published in March 2024, providing a detailed roadmap of how programming skills will be progressively developed across the four stages of primary education (NCCA 2024a). The relationship between programming and computational thinking was established early in the curriculum development process. The draft mathematics curriculum positioned computational thinking as the cornerstone of programming, highlighting its importance in equipping students with the problem-solving skills necessary to thrive in the digital age. This connection was further reinforced in the 'Primary Developments Final Report on the Coding in Primary Schools Initiative' (NCCA 2019a), which stated that computational thinking lays the foundations for effective coding practices. Consequently, it is not surprising that the draft specifications for science, technology, and engineering explicitly recognise computational thinking as a key pedagogical practice. Given this context, this study focusses on how the interplay between programming and computational thinking can prepare learners for the challenges and opportunities that lie ahead in an increasingly technology-driven world.

1.4 Computational Thinking: The Educational Concept

> We are currently preparing students for jobs that don't yet exist…using technologies that haven't yet been invented…in order to solve problems we don't even know are problems yet.
>
> (Richard Riley, Secretary of Education under Clinton)

In many industries and countries, some of the most desirable jobs didn't exist ten or even five years ago. With rapid changes in technology and globalisation, the pace of change is expected to accelerate. According to a report published by the World Economic Forum (2016), 65% of children entering our primary schools today will end up working in jobs that don't even exist. So, how do we prepare our children for

such a rapidly changing employment landscape? The evolving construct, 'computational thinking', has been at the centre of recent discussions to capture and define new ways of thinking that are increasingly important in this digital age (Allan *et al.* 2010). In their guide to computing for primary school teachers, the NAACE (National Association of Advisors for Computers in Education, UK) state that "Computational thinking is a skill children must be taught if they are to be ready for the workplace and able to participate effectively in this digital world" (2013, p.2).

Seymour Papert is credited with pioneering the idea of children developing computational thinking through his work on LOGO programming in the 1980s. In 2006, Jeannette Wing wrote an influential article on computational thinking, giving a 21st century perspective to the concept. In her seminal paper, Wing described computational thinking as "solving problems, designing systems, and understanding human behaviour, by drawing on the concepts fundamental to computer science" (2006, p.33). She advocated for adding this new competency to every child's analytical ability, describing it as a vital ingredient of science, technology, engineering and mathematics (STEM) learning:

> "To reading, writing and arithmetic, we should add computational thinking to every child's analytical ability. It represents a universally applicable attitude and skill set everyone, not just computer scientists would be eager to learn and use."
>
> (Wing 2006, p.33)

Wing's arguments captured the attention of a broad academic community. Prompted by her article, a growing community of researchers, educators, industry and policymakers began to reference the concept of 'computational thinking'. The Royal Society offered a definition of computational thinking that emphasised its connection to computer science:

> "the process of recognizing aspects of computation in the world that surrounds us, and applying tools and techniques from computer science to understand and reason about both natural and artificial systems and processes."
>
> (Furber 2012, p.29)

However, despite much international debate on the scope and nature of computational thinking, there is still a lack of consensus on the definition of computational thinking. In an effort to bring some clarity to the discussions, Wing provided a revised definition:

> "Computational thinking is the thought processes involved in formulating problems and their solutions so that the solutions are represented in a form that can be effectively carried out by an information-processing agent."
>
> (Cansu and Cansu 2016, p.18)

Wing's definition has become a reference point for conceptualising computational thinking in the field of education (Yadav *et al.* 2017; European Commission 2016b). The definition adopted by the NCCA in their 'Background Paper for the Development of a New Primary Mathematics Curriculum' echoes Wing's emphasis on analysing a problem and creating a computer-friendly solution. They define computational thinking as:

> "taking a complex problem, understanding what the problem is and developing possible solutions which can be presented in a way that a computer or a human can understand."
>
> (NCCA 2016a, p.26)

More recently, in the draft STE Education specification, the NCCA proposed a definition for computational thinking that highlights the important role of computers in the problem-solving process.

> "Computational Thinking (CT) draws on the principles of computing to think about and solve problems. It involves the processes that we use when considering and solving problems in a way that computers can assist us."
>
> (NCCA 2024a, p.23)

Barr and Stephenson (2011) suggest that for a definition to be useful, it must be coupled with examples of what to measure. To this end, several educational bodies have attempted to create operational definitions by listing fundamental computational thinking skills. In 2010, the US National Research Council (NRC) convened key international researchers from the fields of education, computer science, and learning sciences to explore "the nature of computational thinking and its cognitive and educational implications" (NRC 2010, p.viii). The NRC (2010) report on this workshop listed over twenty concepts and capabilities that computational thinking might include: abstraction, decomposition, heuristic thinking, recursion and algorithms. The International Society for Technology in Education (ISTE) and the Computer Science Teachers Association (CSTA) collaborated to compile a list of characteristics and dispositions of computational thinking, such as representing data through abstractions, algorithmic thinking, comfortable with ambiguity and persistence (ISTE and CSTA 2011). However, Lye and Koh (2014) contend that general definitions of computational thinking, such as those proposed by the NRC and ISTE and CSTA, may not be suitable for use in the development and implementation of programming initiatives as they don't necessitate the use of technological tools (Lye and Koh 2014).

Similarly, Kafai and Burke (2013) and Resnick *et al*. (2009) suggest that such definitions do not appropriately capture the computational thinking developed by constructing digital artefacts. Brennan and Resnick (2012) developed their computational thinking framework in the context of programming workshops. Their framework examines three key dimensions of computational thinking: computational concepts (the concepts programmers engage with as they program, such as iteration and synchronisation), computational practices (the practices programmers develop as

they engage with these concepts, such as remixing and debugging), and computational perspectives (the perspectives programmers develop about the world and about themselves).

The lack of a widely agreed upon definition has made integrating computational thinking into school curricula challenging. Furthermore, there is debate on whether computational thinking should be integrated into education as a general subject, a multidisciplinary topic or as a discipline-specific topic (Sterling 2016). These issues are further exacerbated as there is disagreement on how best to assess the development of computational thinking in young people (Brennan and Resnick 2012). Therefore, there is still much to be done.

> "The ideas behind computational thinking, the development of a set of good practices and educational paradigms are still in their infancy. Its formal integration in the classroom is being made in the form of individual projects and experiments. Currently there are more challenges than proven results."
>
> (Olabe *et al.* 2011, p.357)

Despite these definition issues, computational thinking is included as both a key pedagogical practice and a key concept in the STE Education specifications (NCCA 2024a). The NCCA (2024a) suggest that initially computational thinking should be fostered through unplugged approaches, before students are given opportunities to apply and build on their computational thinking within a programming context. The draft STE Education specifications (NCCA 2024a) outline several teaching approaches that teachers can adopt to foster computational thinking across science, technology, engineering and beyond (Figure 1.1).

- Broadening children's understanding of CT as a problem-solving approach across Science, Technology and Engineering Education and other curricular areas
- Assisting children in breaking down problems into a series of manageable steps (decomposition) and applying this skill in both digital and non-digital contexts
- Modelling the recognition of patterns and encouraging children to identify patterns that they have previously encountered (pattern recognition), when completing both unplugged and plugged activities
- Providing opportunities for children to investigate and engage as both a 'user' and 'creator' of digital and non-digital technology
- Facilitating children to plan, design, test and run algorithms (set of instructions) using both unplugged and plugged activities
- Challenging children to test and modify their creations in digital technology through collaboration (debugging and abstraction)
- Encouraging children to persevere in their explorations and designs and to take risks as part of the process
- Developing the children's awareness of the CT processes they are using and how they are working like a computer scientist
- Providing opportunities for children to reflect on the CT processes and to evaluate their successes and setbacks.

Figure 1.1: *Teaching Approaches to Foster Computational Thinking (NCCA 2024a, p.23-24)*

1.5 Rationale for and Original Contribution of the Research

The integration of programming and computational thinking into the primary school curriculum illustrates our nation's commitment to "ambitious and responsive curriculum redevelopment and enactment" (NCCA 2022b, p.3). The redeveloped curriculum recognises computational thinking as a fundamental literacy (Wing 2006) and acknowledges the imperative to prepare all students for a future where digital fluency is a prerequisite for success. This forward-thinking vision for primary education in Ireland also reflects a broader commitment to equitable and inclusive education espoused by the primary curriculum framework (NCCA 2023a). Exposing all students, irrespective of their background, to essential skills such as computational thinking and programming, is crucial for bridging persistent gender and socio-economic gaps in the STEM disciplines. Encouraging young girls and underrepresented groups to explore programming and computational thinking can ignite their interest in STEM fields, inspiring a new generation of innovators and leaders, thereby contributing to a more balanced and equitable workforce. These

early positive experiences can be instrumental in dismantling long-standing barriers, challenging negative stereotypes and encouraging girls to envision themselves as future technologists, programmers, and innovators. Ultimately, this redeveloped curriculum recognises the importance of creating an educational environment that values diversity and fosters creativity, ensuring that every child can contribute to and thrive in an increasingly digital world.

However, while the potential benefits are immense, it is important to acknowledge that curriculum reform is a multifaceted and often challenging process. The NCCA (2022c) advise that "curriculum reform of the scale envisioned will not develop organically, at least not in most schools" (p.2) and that successful implementation hinges on the active engagement and understanding of all stakeholders. They acknowledge that Initial Teacher Education (ITE) will play a pivotal role in raising awareness and deepening understanding of the reformed curriculum, ensuring that the next generation of teachers are sufficiently prepared to work with a curriculum which is significantly different from the 1999 curriculum (NCCA 2022c). To effectively prepare future teachers for this new curriculum, teacher educators must themselves possess a nuanced understanding of the philosophical underpinnings, pedagogical approaches, learning theories, and diverse educational practices that underpin this curriculum.

In 2016, the recommendations from STEMerg for enhancing STEM offerings in the early years and Richard Bruton's call for the NCCA to explore how programming and computational thinking could be integrated into the primary curriculum piqued my interest and highlighted the critical need for me to engage in professional development. As an educator responsible for preparing future teachers, I realised that

I must equip myself with the necessary skills and knowledge to effectively support my student teachers in this new landscape. Motivated by the need to stay abreast of evolving educational demands, I sought out opportunities for professional development that would allow me to explore the integration of programming in the classroom. I engaged in two online professional development courses, the Professional Development Service for Teachers (PDST) workshop 'Scratch for Learning' and the FutureLearn course 'Teaching Programming in Primary Schools'. Through these professional development experiences, I gained invaluable insights into the practical applications of programming in the classroom, and it became increasingly evident to me that programming could serve as a powerful medium for cultivating computational thinking among young learners. However, upon conducting a review of the existing literature on the interplay between programming and computational thinking, there was a dearth of research that explored this relationship in formal primary school settings in Ireland. Most national and international research on developing computational thinking through programming has been implemented in second-level or third-level institutions, or within informal settings. Additionally, the research conducted within the Irish primary school context has been more broadly focused. For example, the 'Coding in Primary Schools Initiative' (NCCA 2019a) provided a broad focus that incorporated ideas for curriculum integration, provision for digital technology and optimum conditions for the embedding of digital technology in schools.

The lack of targeted studies highlighted a need for further exploration and understanding of how computational thinking manifests in a primary school programming context and the pedagogical approaches that best support its development. Consequently, the impetus for this study was to address this apparent

gap by designing and implementing a programming initiative within a primary school in Ireland. Brennan and Resnick's (2012) framework was adopted as a lens to explore Irish primary school students' evolving dimensions of computational thinking as they engaged in a ten-week programming initiative. Through this initiative I hoped to gain insights that would illuminate how structured programming education could effectively enhance computational thinking in young learners. Ultimately, this research endeavour aimed to not only contribute to a more informed discourse but also to inspire primary school teachers to embrace programming and computational thinking as fundamental components of a redeveloped primary curriculum.

1.6 Research Questions

The central question guiding this research is:

> In what ways can a constructionist school computer programming initiative support the development of primary school students' computational thinking?

The associated sub-questions are:

- What computational concepts, practices and perspectives do primary school students develop as they engage in a ten-week programming initiative?
- What are the recommended pedagogic approaches for developing primary school students' computational thinking through programming?

1.7 Structure of Dissertation

Chapter one provides a brief introduction to the research. The purpose of the research is outlined, the critical policy that informed the research is examined and the research questions are identified. The literature review is split into two distinct chapters. The first chapter begins by examining the key national and international

policy on incorporating technology into education. The second section of this chapter explores how computational thinking has been integrated into education curricula to date. While the final section of this literature review chapter explores how computational thinking has been developed in children through both plugged and unplugged activities. The second literature review chapter explores the pedagogical approaches to teaching programming which informed the design of the learning experiences and materials. Chapter four describes the research design, including the methodological approach, data collection methods, data analysis and issues related to reliability, validity and ethics. Chapter five is broken into three distinct sections: concepts, practices and perspectives. Within each section the relevant findings are discussed with consideration to existing literature, to provide a comprehensive picture of the computational thinking development of the primary school pupils. The final chapter concludes the dissertation by contextualising the findings and discussing the implications of these findings for theory, policy and practice.

Chapter 2: Technology in Irish Schools

2.1 Introduction

This chapter describes the trajectory of computing in Irish education since the launch of the 'Schools IT 2000' policy to the current landscape, and a new primary curriculum which emphasises the importance of digital technology. The chapter is presented in three sections. The first section explores the relevant national and international policy to provide the reader with a roadmap to guide them through historical policy decisions that have shaped the current landscape. The second section examines how these policies have influenced education curricula, with computational thinking becoming increasingly present on primary school curricula both nationally and internationally. The final section considers how these policies have been translated into practice in the classroom, identifying both plugged and unplugged approaches which have been adopted to foster computational thinking. These approaches are presented and critiqued to determine the features of interventions that can positively impact the development of computational thinking.

Section 1: Computing in Educational Policy

2.2 Technology in Irish Society

The promotional programmes and financial incentives introduced by the IDA to encourage foreign investment in Ireland have enabled Ireland to become a digital hub. Established technology companies such as Apple, Dell, Dropbox, eBay, Facebook, Google, Microsoft, PayPal and Twitter have set up offices in Ireland. In 2015, Harvard Business Review developed a Digital Evolution Index to examine the

status of digital economies around the world. Ireland was named a 'stand out' country, meaning the economy has shown "high levels of digital development in the past and continues to remain on an upward trajectory" (Chakravorti *et al*. 2015, p.4). In 2016, the head of Google in Ireland, Ronan Harris, described Ireland as the data capital of Europe (Kennedy 2016). Harris said:

> "There is a great opportunity for Ireland to capitalise on the fact that it is at the intersection of data for Europe and we can build on that and turn it into a strength."
>
> (Kennedy 2016)

The proliferation of technology companies in Ireland and the continued growth in the industry has ensured that information and communications technology (ICT) is now ubiquitous in Irish society. So much so that the children born into this age of technology are often referred to as 'digital natives' (NCCA 2019b; Prensky 2001). The last ten years have seen unprecedented technological evolution, changing the way we communicate, the way we access, use and manage information and ultimately, the way we experience the world around us (NCCA 2016b). Growing up surrounded by technology means it has a huge effect on the development of modern economies and societies (BT and Ipsos MORI 2016; Wilson *et al.* 2010; OECD 2006). The increasing permeation of technology in all aspects of life has given rise to the concept of a "knowledge-based society" (DES 2008b). Observers and advocates of this knowledge society believe that our development and use of increasingly advanced technologies will continue and that digital literacy will become a basic functional requirement for our personal and professional lives (OECD 2015; DES 2008b; NCCA 2004b).

Researchers have consistently stressed that more needs to be done to realise technology's potential and prepare our students for life in a digital society. Canadian

researchers, Smith *et al*. (2000) describe the computer as the most powerful tool ever created for information processing, with the potential to enrich people's lives in a variety of ways. However, they believe much of this potential is unrealised, as most people only interact with computers through programmes designed by others (end-user computing). In 2002, the National Academy of Engineering (NAE) authors published a report entitled, 'Technically Speaking: Why All Americans Need to Know More about Technology'. In this report they observed what they termed, an unacknowledged paradox, at the heart of our digital society. They wrote:

> "Although the United States is increasingly defined by and dependent on technology and is adopting new technologies at a breathtaking pace, its citizens are not equipped to make well-considered decisions or to think critically about technology."
>
> (NRC 2002, p.1)

They also commented that neither the United States education system nor the policymakers recognised the importance of digital literacy. Several years later, a study conducted by the Association for Computing Machinery (ACM) and the Computer Science Teachers Association (CSTA) reported that the United States education system had 'fallen woefully behind' in equipping students with the basic computer science knowledge and skills they need to succeed in life and the workforce (Wilson *et al.* 2010). They also referenced the paradox: that even as the role and significance of technology have become increasingly important in society and the economy, quality computer science has receded from view. Findings from the PISA data published in the OECD (2015) report, 'Students, Computers and Learning: Making the Connection', indicate a similar state of affairs in OECD countries. The report suggests that despite the prevalence of ICT in our daily lives, "the real contributions ICT can make to teaching and learning have yet to be fully realised and exploited" (OECD 2015, p.15). So, with the increasing permeation of

technology, education policymakers must make tough decisions about the integration of technology in future educational policies and initiatives. How should technology be incorporated into our education systems?

2.3 Education for the 21st Century: Incorporating Technology into Education

> "Cultures deal with their environments by adapting to them and simultaneously changing them. This is particularly true for technological cultures, such as the dynamic culture of computer users."
>
> (Repenning 1993, p.i)

The NCCA (2016b), in their report, 'Proposals for structure and time allocation in a redeveloped primary curriculum', suggest that "education not only reflects a society but is an influence in shaping its development" (p.5). A similar view was expressed in the OECD (2005) report, 'Are Students Ready for a Technology-Rich World?', where they acknowledged that the implications for education in a digitised society were twofold: technology can enable new approaches to teaching and learning, but it has also become increasingly important for people to have digital literacy skills for their personal and professional lives.

Many recent education policy documents have acknowledged the importance of transforming our education system to meet the challenges and needs of the 21st century. The Literacy and Numeracy Strategy (DES 2011) highlights the importance of adjusting the curriculum over time to reflect changes in circumstances. In their consultative paper, 'Building Towards a Learning Society: A National Digital Strategy for Schools', Butler *et al*. (2013) discussed the need to develop a long-term vision for education to ensure that all students have the opportunity to develop the knowledge and skills to be successful citizens in the 21st century. In the 'Action Plan for Education: 2016-2019', the Department of Education and Skills (DES) recognise

that the education environment is both challenging and complex and must be responsive to economic, social and technological changes (DES 2016b). The NCCA, the body responsible for curriculum development in Ireland, published a new primary curriculum in 2023. Their consultation document highlights the dynamic and interactive relationship between education and society. They discuss the increasing demands on the curriculum being made by an evolving society and its expectations of the education system (NCCA 2016b). Globally, The World Bank Group stated that all countries now face the challenges of preparing young people for the world of work and the kind of employment available in 21st century society (2011). The eEurope 2002 Action Plan has proposed that curricula must evolve to meet the needs of the knowledge society (NCCA 2004b).

In 2002, in their report to the government, 'Building the Knowledge Society', the Information Society Commission stressed that the application of ICT at all school levels was of fundamental importance given the increasing need for digital literacy in our society. This view is shared by many of those working in the education system. According to Butler *et al*. (2013), technology is central in transforming our education system to prepare young people to learn, live and work in the 21st century. The World Bank Group believe that changing job profiles and skills are driving the need for governments to develop comprehensive strategies for integrating ICT into our education systems (2011). In the United States, the ACM (Association for Computing Machinery) and CSTA (Computer Science Teachers Association) claim that in this digital age, to be a well-educated citizen, students must develop a deeper understanding of the principles of computer science (ACM and CSTA 2010). The importance of embedding digital learning objectives in our school curricula is recognised in the Digital Strategy for Schools: 2015-2020.

> "The availability of abundant information, advanced technology, a rapidly changing society, greater convenience in daily lives and keener international competition are impacting on education systems and on how we educate our young people and learners of all ages to live and work in this digitally connected world."
>
> (DES 2015, p.15)

In recent years, there has been an unprecedented push by governments of advanced nations (including Ireland) to improve the quality of education, and revitalise interest, in STEM (STEMerg 2016; Sterling 2016). It is claimed that, in this knowledge-based economy, expertise in STEM fields is one of the most valuable assets for advancing our economic ambitions, stimulating innovation, and establishing foundations for future prosperity (DJEI 2015; STEMerg 2016). So, what action has come from the recognition that technology is of critical importance to society, and consequently education?

2.4 Policy on Technology in Education

2.4.1 Policy Development in the IT Schools 2000 Era (1997-2009)

According to Faherty *et al.* (2023), a 1996 International Data Corporation (IDC) report ranking Ireland in the third division with regard to its "preparedness for the Information Age" (p.9) sparked efforts to integrate technology into education in a more meaningful way. In the following year, the DES published the first policy framework document on ICT in Irish Education, Schools IT 2000 (DES 1997). In this policy document, the DES recognised that:

> "Ireland lags significantly behind its European partners in the integration of information and communication technologies (ICTs) into first and second-level education. The need to integrate technology into teaching and learning…must be met in the interests of Ireland's future economic wellbeing."
>
> (DES 1997, p.2)

The tardiness of government policy on ICT in education perhaps reflects the passive approach to policymaking highlighted by Galvin (2009). He claims that the DES

were “simply not proactive in generating or driving policy led change…nor confident in its policy making ability”; instead, they responded to externally seeded and orientated policy (p.277). Indeed, Conway and Brennan-Freeman (2015) assert that in the mid 1990’s numerous policy documents were responsible for convincing educational policymakers that ICT should be introduced to schools. Since the 1990s, numerous policy initiatives and reports on ICT in Education have been published (Table 2.1). The publication of the Schools IT 2000 document marked the beginning of a period of considerable activity of ICT in Education policymaking between 1997-2003. The years from 2003 to 2008 was a quieter period in terms of policy but a busy period in terms of implementing ICT initiatives. This was followed by a policy lull, perhaps due to the economic recession. However, in 2015 the DES launched the Digital Strategy for Schools, 2015-2020 (DES 2015), which indicated the government’s renewed interest in improving ICT in Education. During this period schools received a series of national ICT grants to support the integration of ICT in teaching and learning (Table 2.2).

Table 2.1: *ICT policy reports and initiatives published in Ireland (1997-2009)*

Year	**Initiative/Report**
1997	Publication of first government policy document on ICTs in schools: Schools IT 2000: A Policy Framework for the New Millennium (Department of Education and Science, 1997)
1998	Introduction of Schools IT 2000 Initiative. This contained three strands: the Technology Integration Initiative, the Teaching Skills Initiative, and the Schools Support Initiative, including the Schools Integration Project (SIP) and Scoilnet.
1998	Establishment of the National Centre for Technology in Education (NCTE), with an initial brief to implement the Schools IT 2000 initiative, to develop ICT policy proposals and to provide ICT policy advice to the (then) Department of Education and Science.
1999	Statistical report. The state of IT in Irish schools (NCTE, 1999).
2001	Report on the implementation of Schools IT 2000 (NPADC, 2001).
2001	ICT in education – A Blueprint for the Future of ICT In Irish Education 2001-2003 (Ireland, 2001). ICT 2000 survey: Statistical report (NCTE, 2001).

2002	ICT Planning and Advice for Schools (NCTE) – planning pack to support schools in developing ICT plans to meet infrastructural and other ICT-related needs.
2003	2002 ICT school census. Report. (NCTE, 2003)
2004	Information and Communications Technology (ICT) in the Primary School Curriculum. Guidelines for Teachers (NCCA, 2004).
2004	Curriculum, Assessment and ICT in the Irish Context: A Discussion Paper (NCCA)
2006	NCTE 2005 Census on ICT Infrastructure in Schools. Statistical Report (NCTE)
2007	ICT Framework: A Structured Approach to ICT in Curriculum and Assessment (NCCA)
2008	ICT in Schools (Inspectorate of the Department of Education and Science)
2008	Investing Effectively in Information and Communications Technology in Schools: Report of the Minister's Strategy Group. (Department of Education and Science)
2009	Smart Schools = Smart Economy. Report of the ICT in Schools Joint Advisory Group to the Minster for Education and Science.
2009	Planning and Implementing e-Learning in Your School: A Handbook for Principals and Coordinating Teachers (NCTE, 2009) (included E-learning Roadmap)

Table 2.2: *National (central) investment in ICT in primary and post-primary schools (1998 - 2017)*

Year Begun	**Initiative**	**Amount (Million €)**
1998	Schools IT 2000: A Policy Framework for the New Millennium	€52
2001	Blueprint for the Future of ICT in Irish Schools	€78
2004	Networking Schools	€23
2005	Schools Broadband Programme (2005-08)	€30
2009	100 Mbps Connectivity Demonstration Programme (78 post-primary schools)	
2009	ICT in Schools Programme (equipment grants) (2009-10)	€92
2012-2015	100 Mbps to Post-primary Schools: National Rollout (jointly funded by the Department of Communications, Energy and Natural Resources and the Department of Education and Skills, with annual recurring costs to be paid by the DES – circa €13m pa.)	€41
2016/2017	Grant Scheme for Infrastructure	€30
2017/2018	Grant Scheme for Infrastructure	€30
2018/2019	Grant Scheme for Infrastructure	€50
2019/2020	Grant Scheme for Infrastructure	€50
2020/2021	Grant Scheme for Infrastructure	€50
2021/2022	Grant Scheme for ICT Infrastructure	€50
2023/2024	Grant Scheme for ICT Infrastructure	€50

2.4.1.1 Schools IT 2000: A policy framework for the new millennium (DES 1997)

The launch of Schools IT 2000 resulted in the establishment of the National Centre for Technology Education (NCTE). The NCTE were a group of ICT advisors tasked with implementing Schools IT 2000, developing future ICT policy proposals and providing ICT policy advice to the DES (DES 2008a). The main aims of the Schools IT 2000 Initiative were to provide all students with the opportunity to become computer literate, and to prepare them for participation in the knowledge society and to support teachers in the development of the requisite skills which would enable them to integrate ICT into their teaching (Crotty and Farren 2013). The initiative adopted a three-thronged approach to ICT integration with three major strands: the Technology Integration Initiative, the Teaching Skills Initiative and the Schools Support Initiative (Conway and Brennan-Freeman 2009; DES 2008a). The purpose of the Technology Integration Initiative was to support schools in developing their ICT infrastructure. The particular focus of this initiative was to have equipped Irish schools with at least 60,000 computers by the end of 2001 (DES 1997). In 2002, the NCTE census revealed that this figure had been surpassed, as it reported some 84,000 computers in schools (DES 2008a). The Teaching Skills Initiative was responsible for the provision of teacher training, namely, ICT skills and awareness, professional skills development in ICT and pedagogical skills development (DES 1997). Finally, the Schools Support Initiative was responsible for supporting schools developing whole-school technology plans. This included Scoilnet, an online support service providing advice on general ICT and a database of ICT-related resources, and The School Integration Project, an initiative in which pilot schools sought to identify good practices in relation to the integration of ICT in teaching and learning (DES 1997).

2.4.1.2 Report on the implementation of Schools IT 2000 (NPADC 2001)

In 1998, a National Policy Advisory and Development Committee (NPADC) was established to advise the Minister for Education and Science on developing a national policy for ICT in schools. The following year, the NPADC carried out a national survey to determine the impact of some aspects of the Schools IT 2000 on both primary and secondary education. The results, published in the Report on the Implementation of Schools IT 2000, painted a mostly positive picture of the initiatives implemented since the launch of Schools IT 2000 (Conway and Brennan-Freeman 2009). Positive developments included a significant increase in the number of computers in schools, access to the Internet for the majority of students, an increase in the use of ICT by teachers and the establishment of many ICT support mechanisms (NPADC 2001). The report also identifies some fundamental issues that could hinder the implementation of the Schools IT Initiative. These were:

> "the need for more training, the need for more funding (equipment/computers, maintenance, support) and the need for more support (technical support, encouragement to use ICT)."
>
> (NPADC 2001, p.8)

The report also recommended that the Department of Education and Science should "state clearly its policy for Information and Communication Technologies (ICT) in primary and second-level education" (NPADC 2001, p.9).

2.4.1.3 Blueprint for the Future of ICT in Education (DES 2001)

Building on the perceived success of the initial policy framework, the Department published its second policy document on ICT in education: 'Blueprint for the Future of ICT in Irish Education'. This was a three-year strategic action plan, emphasising the same core policy objectives and supporting the implementation of the same

initiatives begun in School IT 2000, while at the same time recognising some necessary shifts in policy identified in the NPADC's implementation report. Consequently, the main emphases of the policy were: boosting ICT capital provision to schools, increasing access to the internet, further integrating ICT into teaching and learning and improving teacher professional development (DES 2001). Building on the progress of the Technology Integration Initiative, the NCTE published a planning pack for schools in 2002 entitled 'ICT Planning and Advice for Schools'. This pack was designed to advise schools on preparing and implementing a whole-school ICT plan and purchasing equipment.

2.4.1.4 Policy Lull (2003-2008)

In the following years, 2003-2008, there was what Conway and Brennan-Freeman (2009) referred to as a 'policy lull'. However, this did not signify inaction in relation to ICT in education, as several initiatives undertaken in this period provided important insights for future ICT integration. During this time, the NCCA published several reports and consultative documents regarding the integration of ICT in education. The NCCA commissioned an investigation into the possibility of introducing a new Leaving Certificate computer-based subject. The report 'Computers and Curriculum: Difficulties and Dichotomies' recommended rejecting the idea of creating a stand-alone computer-based subject, citing concerns that it "could exacerbate existing inequalities within and between schools." (O'Doherty *et al.* 2004, p.87). In 2004, a discussion paper, 'Curriculum Assessment and ICT in the Irish Context: A Discussion Paper', was also published by the NCCA. This document was developed to stimulate discussion around the integration of ICT in curriculum and assessment in Irish schools, while also outlining the direction of

future work of the NCCA. In 2007, the NCCA published the 'ICT Framework: A Structured Approach to ICT in Curriculum and Assessment Revised Framework'. This framework outlined how competence in ICT could be developed using a cross-curricular pedagogical approach (NCCA 2007a). In addition, the Schools Broadband Programme (2005-2009) was rolled out nationally and the NCTE's infrastructural census of ICT in schools (2005/2006) was conducted. The DES also confirmed their commitment to continue to devise policy on integrating ICT in education in their 'Statement of Strategy 2005-2007' (DES 2005b).

2.4.1.5 Statement of Strategy 2005-2007 (DES 2005b)

This strategic framework was developed against the backdrop of a growing recognition in Europe that education is critical to economic success, social progress and future prosperity (DES 2005b). The turn of the twentieth century saw "a new paradigm emerging, creating knowledge-based economies" (Rodrigues 2002, p.3). Within this new paradigm, knowledge became recognised as "the driver of productivity and economic growth" (OECD 1996, p.3), and the internet became the main infrastructure (Rodrigues 2002). The Lisbon Strategy was launched in 2000 with the strategic goal of making the European Union "the most competitive and dynamic knowledge-based economy in the world" (Rodrigues 2002, p.19). The 'Strategy of Statement 2005-2007' acknowledged the importance of education in the creation of a knowledge-based society. Whereas initial policy on ICT in education focused on infrastructure and equipping schools, this strategy represented a shift in focus to computer literacy and equipment usage (Crotty and Farren 2013). This emphasis is evident from the strategy objective:

> "We will promote the use of ICT in schools and encourage pupils to achieve computer literacy and acquire the necessary skills for

> participation in the Information Society."
>
> (DES 2005, p.37)

The National Development Plan 2007-2013 reaffirmed the importance of ICT in education to Ireland's economic and social progress. It promised an investment of €252 for ICT in schools over the 2007 to 2013 period. The need for such investment was emphasised by reports from the NCCA which suggested "insufficient, inadequate, outdated, faulty and inaccessible ICT equipment" (NCCA 2007b, p.57) and an "erratic provision nationwide for ICT in schools" (NCCA 2008, p.218). In 2008, the Irish economy entered a severe recession that slowed both policy development and investment in ICT in education.

2.4.1.6 Smart Schools = Smart Economy (DES 2009)

Despite the recession, there was a growing recognition that the knowledge economy would be critical to our economic recovery (NCC and Forfás 2009a). Research indicated that there was a relationship between ICT integration in our education systems and economic growth (DES 2009). The 'Statement on Education and Training', published by the National Competitiveness Council (NCC) and Forfás supported this view, stating:

> "Ireland's future competitive advantages are likely to be in internationally trading sectors (e.g. software, high-technology manufacturing, financial services, and other business services) that depend on advanced telecommunications infrastructure and an ICT literate population. ICT has the potential to enliven learning in science, engineering and technology subjects, which underpin the skills on which future competitiveness will be based."
>
> (NCC and Forfás 2009b, p.15)

Highlighting the importance of digital literacy to our future economic prosperity, they urged the government to continue implementing the Strategy for ICT in Schools, even in the face of the 'funding impasse' (Forfás & NCC 2009b). The

'Smart Schools = Smart Economy' report, jointly published by representatives of ICT Ireland (which has since merged with the Irish Software Association to become Technology Ireland) and the Department of Education and Science, acknowledged the importance of investing in ICT for the education system (DES 2009). The report identified five core investment areas to be prioritised in 2009-2012. These key areas were:

> - classroom and student infrastructure,
> - technical support and the virtual learning environment (VLE),
> - teacher professional development,
> - ICT planning and multi-annual budgeting,
> - digital content growth, and enhanced broadband for schools."
>
> (DES 2009, p.6)

Two of the key recommendations of the Smart Schools = Smart Economy report that came to be realised were the establishment of an ICT in Schools Steering Group and the provision of broadband connectivity to all schools. The Steering Group was set up to provide a stakeholder forum and to advise future policy direction. Since its establishment, it has been consulted on various policy initiatives and reports, including the 2013 ICT Census and the Digital Strategy for Schools 2015-2020. The national roll-out of 100 Mbps to Post-primary Schools commenced in 2012, after a successful 100 Mbps Connectivity Demonstration Programme (78 post-primary schools) in 2010.

2.4.2 Impact of Policy Development on Technology Integration in Education

During the years 1997-2008, several reports examined the impact of these policy reports and the significant investment in technology integration in education. Since 1997, the DES has invested significantly in the integration of ICT in Irish education (Table 2.2). Beginning in 1998, the NCTE has conducted intermittent countrywide censuses of ICT infrastructure in Irish schools, the most recent of which was

conducted in 2013. In February 2007, the then Minister for Education and Science, Mary Hanafin, announced the appointment of a strategy group to advise on priorities for investment. In 2008, the Department published the report produced by the strategy group: 'Investing Effectively in Information and Communications Technology in Schools 2008-2013 (DES 2008b). Also, in 2008, the Department of Education and Science's Inspectorate evaluated the extent to which ICT was used in Schools and its impact on teaching and learning (DES 2008a).

2.4.2.1 ICT in Schools: Inspectorate Evaluation Report (DES 2008a)

This report examined progress made with respect to the objectives of the two previous government policies on ICT in Education: 'Schools IT 2000' and the 'Blueprint for the Future of ICT in Irish Education'. The report concluded that progress had been made, however, it highlighted some key deficiencies and made several recommendations for ICT provision in schools. These included:

> The introduction of a national ICT technical support and maintenance system for schools.
>
> An increased emphasis on the application of ICT in teaching and learning in teacher education at the pre-service, induction and continuing professional development stages.
>
> Schools should exploit the benefits to be had from ICT in their assessment procedures and in their administrative practices.
>
> Support services should give priority to the integration of ICT in teaching and learning.
>
> (DES 2008a, pps.xviii-xix)

In the same year, the Department published the report 'Investing Effectively in Information and Communications Technology in Schools, 2008 – 2013' (DES 2008b).

2.4.2.2 Investing Effectively in Information and Communications Technology in Schools, 2008 – 2013 (DES 2008b)

Similar to the 'ICT in Schools: Inspectorate Evaluation Report', this report acknowledged the progress made in both the infrastructure and usage of ICT in schools. However, it also acknowledged that efforts were hindered due to "lack of sufficient and sustained investment" (DES 2008b, p.i). It acknowledged issues raised by studies reporting on the impact of the policies including, inadequate infrastructure, no provision for technical support and inadequate broadband levels. They highlighted seven key investment objectives:

- Continuing professional development
- Software and digital content for learning and teaching
- ICT equipment – additional and replacement
- Schools broadband and services
- Technical support and maintenance
- Implementation structures and supports
- Innovative practice and research

(DES 2008a, p.ii)

Both 2008 reports emphasised the importance of investment in the professional development of teachers to assist in greater integration of ICT in schools. In the context of these reports, the NCTE developed a framework to guide schools in the development of an e-Learning plan (Crotty and Farren 2013). They published a handbook for principals and ICT coordinators entitled, 'Planning and Implementing e-Learning in your school: Handbook for Principals & ICT Coordinating Teachers', which provides a step-by-step guide in developing an e-learning plan for a school. They also published an e-Learning roadmap designed to help schools identify their current position regarding ICT use and to prioritise e-learning areas that require further development. However, these two Department reports were published in 2008, at the beginning of the economic downturn (Cosgrove *et al.* 2014a). The

economic crisis had an immediate effect on ICT in education, with the disbandment of the ICT advisor network, which had been providing professional development and support for innovative ICT projects in schools (Conway and Brennan-Freeman 2009). According to Conway and Brennan-Freeman (2009),

> "the renewed policy impetus provided by the two key reports published in 2008 is likely to be dampened somewhat because of cutbacks in public sector finances."
>
> (p.397)

However, these two documents were followed a year later by the aforementioned 'Smart Schools = Smart Economy' report, which showed a greater recognition of Ireland's changing economic circumstances (Crotty and Farren 2013).

2.4.2.3 ICT in Schools Census

The most recent census, conducted in 2013, was the fifth implementation of the ICT Census. Within this timeframe, several metrics are adopted to evaluate the efficiency of their investment and to inform future investment. The initial focus of ICT policy in education was to equip schools, and the metric used to measure this was the student-computer ratio (Crotty and Farren 2013). According to NCTE (2004), the reports on the 1998, 2000 and 2002 ICT Censuses provided valuable insights into the infrastructure of schools, but due to the factual nature of the surveys, they did little to reveal the quality of ICT use. With respect to student-computer ratio, the general trend indicates a decrease in student-computer ratios across each census. However, our participation in international studies shows we lag behind the international student-computer ratio average. In 2002, a Eurostat survey showed that Irish schools had a higher student-computer ratio (10.3) than the European average (9.3) (NCTE 2004). However, more recent data shows that Ireland has made progress with regard to student-computer ratio. In 2012, a PISA survey found that on average, 0.64

computers are available to each student in Ireland, similar to the International Average of 0.68 (Cosgrove *et al*. 2014a). This progress is further evidenced by Table 2.3, which shows the student-computer ratio in primary schools from 1998-2013.

Table 2.3: *Student-Computer ratio* in primary schools (1998-2013)*

	**1998	**2000	**2002	***2005	***2013
Primary	37:1	18:1	11.8:1	9.8:1	4.6:1
Post-primary	16:1	13:1	9.4:1	8.2:1	3.7:1

*These figures do not distinguish between computers designated for teacher, school administration or student use.
**Source: NCTE (2004, p.6)
***Source: Cosgrove et al. (2014a, p.77)

Reports on the first four censuses (1998, 2000, 2002 and 2005) highlighted several issues pertaining to ICT in schools (NCTE 2004; Shiel and O'Flaherty 2006). These were: insufficient/unequally distributed equipment, inadequate technical support and maintenance, limited internet connectivity, and the need for continued ICT professional development and greater guidance for teachers regarding educational ideas and digital resources.

The 2013 Census was conducted during the preparation of the Digital Strategy for Schools 2015-2020. As part of the research process for the Digital Strategy, the DES commissioned the PDST-TIE (Professional Support Service for Teachers - Technology in Education, formerly the NCTE), to develop and conduct a census of ICT in all primary and secondary schools. More recent policies on ICT in education have been more focused on equipment usage, which has been reflected in the scope of the 2013 ICT Census (Crotty and Farren 2013). The 2013 ICT Census included a teacher questionnaire which provided detail which extended "beyond ICT infrastructure and broad usage patterns" to "capture actual usage patterns in classes" (Cosgrove *et al*. 2014, p.51). The themes included in the 2013 ICT Census were:

- ICT infrastructure in schools
- Access to and frequency of ICT usage in teaching and learning
- Perceived obstacles and priorities in ICT usage
- Continuing Professional Development in the area of ICT
- Perceived impact of ICT usage on students' engagement in learning
- School policies on Internet safety and sharing of digital teaching resources

(ERC 2017)

The findings of the 2013 ICT Census provided evidence to guide the Department in the development of its vision and plan of action for ICT in Irish schools. The research highlighted four key areas to be addressed in the development of the Digital Strategy (Cosgrove *et al*. 2014):

- **Teaching, Learning, and Assessment using ICT**, including using ICT to develop 21st century skills and ICT integration in teaching learning and assessment.
- **Teacher Professional Learning**, including establishing standards of teacher professional knowledge and integration of teacher professional development with future ICT initiatives or curriculum developments.
- **Leadership, Research, and Policy** including research and evidence-based innovations informing policy and practice and support for in-school leadership.
- **ICT infrastructure**, including broadband provision, student-computer ratios, technical support and maintenance and purchase of software.

2.4.3 Policy Development in the Digital Strategy for Schools era (2010-2020)

From these reports, it is clear that despite efforts to focus on computer literacy and usage, ICT infrastructure still remained a pertinent issue requiring large amounts of investment. Following publication of the 'Smart Schools = Smart Economy' report, there was a period of relatively little activity despite significant advancements in the

field of technology (Cosgrove *et al.* 2014a). However, the Irish government, like many of their international counterparts, recognised the need to adapt their education system to meet the perceived needs of the knowledge society. In order to best achieve this, the government recognised the need for a countrywide digital strategy for education. According to Cosgrove *et al.* (2014), impetus to develop a national digital strategy for schools came from other government policies dealing with digital technology in society (Doing More with Digital: National Digital Strategy for Ireland, DCENR 2013), digital technology in industry (ICT Skills Action Plan 2014-2018, DES & DJEI 2014; ICT Action Plan: Meeting the high-level skills needs of enterprise in Ireland, DES 2012a), and digital technology in education (National Literacy and Numeracy Strategy - Literacy and Numeracy for Learning and Life 2011-2020, DES 2011; A Framework for the Junior Cycle, DES 2012b). The Programme for Government 2011-2016 outlined the government's commitment to integrating ICT across all education policies (Department of the Taoiseach 2011).

International initiatives and publications also incentivised development of a national digital strategy (Table 2.4). These initiatives and publications, include 'Key Competences for Lifelong Learning' (European Commission 2007), 'ICT Competency Framework for Teachers' (UNESCO 2008 & 2011b), the 'Digital Agenda for Europe' (European Commission 2010), 'Transforming Education: The Power of ICT Policies' (UNESCO 2011b) and 'Opening up Education: Innovative Teaching and Learning for All through New Technologies and Open Educational Resources' (European Commission 2014). The 2013 ICT Census was one of several research initiatives ordered by the Department of Education and Skills to determine investment priorities, and to inform the development and adoption of an effective digital strategy (Cosgrove *et al.* 2014). In December 2013, Butler *et al.* (2013)

published the consultation paper, 'Building towards a Learning Society – A National Digital Strategy for Schools'. In the consultation paper, Butler *et al.* (2013) highlighted the need to develop a long-term vision for education to meet the challenges of a rapidly changing digital society in the 21st century. The publication of this paper marked the launch of public consultation period, in which the Department of Education and Skills sought submissions from the various stakeholders, including students, teachers and parents, on the role of ICT in education (Butler *et al.* 2013).

Table 2.4: *International reports on ICT in education published 2005-2017*

Year	Report
2005	*Are Students Ready for a Technology-rich World? What PISA Tells Us (OECD) (and ICT indicators based on subsequent PISA surveys in 2006, 2009 and 2012)*
2006	*EU Survey on Use of Computers and the Internet in Schools in Europe.*
2007	*European Reference Framework on Key Competencies for Lifelong Learning (including digital competence)*
2008, 2011	*UNESCO ICT Competency Framework for Teachers*
2011	*Key Data on Learning and Innovation through ICT at School in Europe (European Commission)*
2011	*Transforming Education: The Power of ICT Policies (UNESCO, 2011b)*
2012	*European Survey of Schools: ICT in Education (ESSIE). Country Profile: Ireland.*
2013	*European Survey of Schools: ICTs in Education (ESSIE). Benchmarking Access, Use and Attitudes to Technology in Europe's Schools.*
2014	*Computing our future – Priorities, school curricula and initiatives across Europe (European Schoolnet)*
2014	*UNESCO ICT in Primary Education: Analytical survey*
2015	*Students, Computers and Learning: Making the Connection (OECD) (Based on results from PISA 2012)*
2015	*Computing our future: Computer programming and coding - Priorities, school curricula and initiatives across Europe (European Schoolnet)*
2015	*Promoting Effective Digital-Age Learning: A European Framework for Digitally-Competent Educational Organisations (JRC)*
2017	*The Digital Competence Framework for Citizens with eight proficiency levels and examples of use (JRC)*

2.4.3.1 Digital Strategy for Schools 2015-2020 (DES 2015)

The resultant report, 'Digital Strategy for Schools 2015-2020', was launched by the Minister for Education and Skills, Jan O'Sullivan, in October 2015. The Digital

Strategy aimed to provide "a clear vision for the role of ICT in teaching, learning and assessment for schools in Ireland" (DES 2015, p.4). Similar to the Smart Schools = Smart Economy report, this Strategy acknowledged the continuing pressure on public finances while highlighting the need to develop knowledge and skills to contribute to economic growth (DES 2015). The Strategy focused on the themes from the 2013 ICT Census: Teaching, Learning and Assessment Using ICT, Teacher Professional Learning, Leadership, Research and Policy, and ICT Infrastructure. The Department recognised that equipping schools with ICT does not equate to the effective use of ICT in schools.

Consequently, ICT integration is central to the Department's vision which they believe can equip students to live in and contribute to modern society. As recommended by the consultation paper, the Strategy recognises the need to review and adapt the 'UNESCO ICT Competency Framework for Teachers' (UNESCO 2008, 2011a) to provide guidance for schools in integrating ICT into their practice within the Irish context. With the UNESCO ICT Competency Framework for Teachers (UNESCO, 2011a) providing direction for the transformation of ICT integration in schools, the Department identified several key priorities. These priorities are outlined in Table 2.5.

Table 2.5: *The key priorities of the Digital Strategy for Schools 2015-2020*

Theme	**Priorities**
Theme 1: Teaching, Learning and Assessment Using ICT	• Provision for in-depth study of ICT in Senior Cycle, • Embedding digital skills in the teaching, learning and assessment of other subjects, e.g. coding, e-portfolios, • Providing Enhanced Digital Content through Scoilnet, strategic partnerships etc., • Assist schools in whole-school planning and evaluation by adapting the UNESCO ICT Competency Framework for Teachers (UNESCO, 2011a) and updating the 'e-Learning in Your School (NCTE, 2009).

Theme 2: Teacher Professional Learning	• Embedding ICT skills in ITE (Initial Teacher Education), • Enhancing access to and impact of CPD (Continuing Professional Development), • Provide flexible models of CPD, e.g. online or blended learning programmes.
Theme 3: Leadership, Research and Policy	• Support ICT initiatives in schools, • Work with stakeholders to promote responsible and ethical internet and social media use, • Engagement with the higher and further education sectors.
Theme 4: ICT Infrastructure	• Multi-annual funding to address ICT infrastructural requirements, • Improved access to high-speed broadband for schools, • Provide guidance to schools on technical support solutions.

2.4.3.2 The UNESCO ICT Competency Framework for Teachers (UNESCO 2011a)

The UNESCO ICT Competency Framework for Teachers (UNESCO 2011a) identifies three complementary and overlapping stages in the use of ICT in teaching and learning: technology literacy, knowledge deepening, and knowledge creation (Table 2.6). The three stages are defined as follows:

> Increasing the extent to which new technology is used by students, citizens and the workforce by incorporating technology skills into the school curriculum - which might be termed the Technology Literacy approach.
>
> Increasing the ability of students, citizens, and the workforce to use knowledge to add value to society and the economy by applying it to solve complex, real-world problems - which could be called the Knowledge Deepening approach.
>
> Increasing the ability of students, citizens, and the workforce to innovate, produce new knowledge, and benefit from this new knowledge - the Knowledge Creation approach.
>
> (UNESCO 2011a, p.7)

The UNESCO ICT Competency Framework for Teachers (UNESCO 2011a), then classifies the role of a teacher into six key components (see Table 2.6): Understanding ICT in Education, Curriculum and Assessment, Pedagogy, ICT, Organisation and Administration, and Teacher Professional Learning (UNESCO

2011a). The Department believes that this framework will enable schools and professional development providers to assess progress made in ICT integration to date and provide direction for building the capacities and capabilities of teachers into the future (DES 2015).

Table 2.6: *The UNESCO ICT Competency Framework for Teachers (UNESCO 2011a, p.3)*

	Technology Literacy	**Knowledge Deepening**	**Knowledge Creation**
Understanding ICT in Education	Policy Awareness	Policy Understanding	Policy Innovation
Curriculum and Assessment	Basic Knowledge	Knowledge Application	Knowledge Society Skills
Pedagogy	Integrate Technology	Complex Problem Solving	Self-Management
ICT	Basic Tools	Complex Tools	Pervasive Tools
Organisation and Administration	Standard Classroom	Collaborative Groups	Learning Organisations
Teacher Professional Learning	Digital Literacy	Manage and Guide	Teacher as Model Learner

2.4.3.3 The Digital Learning Framework for Primary Schools (DES 2017b)

The Digital Strategy for Schools 2015-2020 sparked a flurry of further research, policy developments and initiatives (see Table 2.7). One of the key motivations of the Digital Strategy for Schools 2015-2020 was to review and adapt the UNESCO Competency Framework to make it relevant to the Irish context. In September 2017, the Minister for Education & Skills, Richard Bruton, launched the Digital Learning Framework to provide a roadmap for the effective integration of ICT in teaching and learning within the Irish context. The framework developed by the PDST (supported by external experts) was trialled in primary schools during the period: October 2017 – June 2018. The framework identifies Statements of Practice which illustrate 'effective' and 'highly effective' school practices which enhance teaching and

learning through the use of digital technology. These Statements of Practice are underpinned by the UNESCO ICT Competency Framework for Teachers (UNESCO 2011a) and are informed by two frameworks developed by the EU Joint Research Centre: 'Promoting Effective Digital-Age Learning: A European Framework for Digitally-Competent Educational Organisations' (Kampylis *et al.* 2015) and 'DigComp 2.1: The Digital Competence Framework for Citizens with eight proficiency levels and examples of use' (Carretero *et al.* 2017). Some of the key uses and benefits of the Digital Learning Framework for Primary Schools (DES 2017b) are:

- Supports school readiness for evolving curricula.
- Promotes the integration of digital technologies in teaching and learning using constructivist principles.
- Promotes collaboration and facilitates individual or groups of teachers in the planning of and reflecting on their teaching and learning practices.
- Helps teachers, schools and the Department identify and respond to school and teacher continuing professional development (CPD) needs.
- Provides guidance as to what constitutes 'effective' and 'highly effective' practices in the use of digital technology in teaching and learning.

Table 2.7: *ICT policy reports and initiatives published in Ireland (2009-2020)*

Year	Initiative/Report
2012	Integration of NCTE into the Professional Development Service for Teachers (PDST) to form PDST Technology in Education (PDST-TIE)
2012	Educational Impact Evaluation Report on the Provision of 100 Mbit/s Broadband to 78 Post-primary Schools (DES).
2012	ICT Action Plan – Meeting the High Level Skills Needs of Enterprise in Ireland (DES)
2013	Building Towards a Learning Society – A National Digital Strategy for Schools (ERC and St Patrick's College, Drumcondra).
2014	The 2013 ICT Census in Schools – Main Report (PDST-TIE)
2014	Report on the Consultation with Young People on the Digital Strategy for Schools (DES & DCYA)
2014	

	ICT Skills Action Plan, 2014-2018 – Government, Education and Industry working together to make Ireland a global leader in ICT talent (DES & DJEI)
2015	Digital Strategy for Schools 2015-2020 (DES)
2017	Digital Strategy for Schools 2015-2020 Action Plan 2017 (DES)
2017	Digital Learning Framework for Primary Schools (DES)
2018	Digital Strategy for Schools 2015-2020 Action Plan 2018 (DES)
2018	Digital Learning Framework Trial Evaluation: Final Report (ERC)
2019	Digital Strategy for Schools 2015-2020 Action Plan 2019 (DES)
2019	Primary Developments: Final report on the Coding in Primary Schools Initiative (NCCA)
2019	Digital Learning Framework (DLF) national evaluation: Starting off Baseline report (ERC)
2020	Digital Strategy for Schools 2015-2020 Action Plan 2020 (DES)
2020	Digital Learning 2020: Reporting on practice in Early Learning and Care, Primary and Post-Primary Contexts

2.4.3.4. Digital Strategy for Schools 2015-2020 Action Plans

The publication of yearly action plans designed to plan and review implementation of the Digital Strategy for Schools 2015-2020 provided further evidence of the government's commitment to achieving the targets set out in the Digital Strategy for Schools 2015-2020.

Digital Strategy for Schools 2015-2020 Action Plan 2017 (DES 2017c)

The first of these Action Plans was launched in June 2017, with the aim of giving new impetus to achieving the aims of the Digital Strategy (DES 2017c). The key focuses of the Action Plan were:

- A clustering programme to encourage interschool collaboration on ICT initiatives.
- A programme of curricular reforms, including a new Computer Science subject at leaving certification level and introducing coding as part of a new primary school mathematics curriculum.

- Development of a new digital learning framework, adapted from the UNESCO ICT Competency Framework for Teachers.
- Providing a range of professional development programmes
- Development of an online portal for sharing and accessing exemplars of good ICT practice.
- Continuing the rollout of the €210 million ICT Infrastructure Investment
- Improving high-speed broadband connectivity in schools.
- Seeking support from businesses and industry for the integration of ICT in schools.
- Reviewing policy on online safety and ethical internet use.

This Action Plan represented a key step in the government's plan "that the Irish Education and Training System should become the best in Europe over the next decade" (DES 2016, p.1).

Digital Strategy for Schools 2015-2020 Action Plan 2018 (DES 2018)

The following year the DES launched the Digital Strategy Action Plan 2018. This report reviewed the implementation of the 2017 Action Plan and outlined the targeted actions for 2018. In particular, the report identified targets relating to technical support and broadband connectivity for primary schools had not been delivered and would be carried forward to 2018 (DES 2018). During 2017, progress was made in several areas identified in the 2017 Action Plan. This included the plan to embed digital technologies into curricula under development. For example, the development of the new primary mathematics curriculum commenced and included computational and creative thinking skills (ibid). Phase 1 of the Coding in Primary Schools Initiative commenced in September 2017 (NCCA 2019a). The first €30m of

funding from the Infrastructural Grant Scheme was issued to schools in the 2016/2017 school year with the aim of helping schools to embed digital technology in their teaching and learning (DES 2018). Many key actions identified for delivery in 2018 build on those identified in the 2017 Action Plan. However, there was now emphasis on adapting and evaluating the newly developed Digital Learning Framework (ibid).

Digital Strategy for Schools 2015-2020 Action Plan 2019 (DES 2019)

The 2019 Action Plan was the second annual review of progress made on the Digital Strategy for Schools 2015-2020. It recognised that Phase 2 of the Coding in Primary Schools Initiative had commenced, and a report on the initiative would be published later that year (DES 2019). The second €30m funding from the Infrastructural Grant Scheme was issued in early 2018 (ibid). The School Excellence Fund (Digital and STEM) Programme was launched in 2018 with the aim of promoting innovative use of digital technologies and STEM in schools (ibid). In October 2018, the ERC published the 'Digital Learning Framework Trial Evaluation: Final Report'. The report acknowledged that the participants (Digital Learning Team leaders, teachers and PDST advisors) considered the Digital Learning Framework trial a success with evidence of enhanced embedding of digital technologies in teaching, learning and assessment during the trial period (Cosgrove *et al.* 2018). Several factors were identified as necessary for a successful national rollout:

> leadership from the DES, the support provided by the PDST advisors, the provision of technical support, the provision of training and adequate time for teachers to engage, and the acknowledgment of the variability of schools' digital contexts.
>
> (Cosgrove *et al.* 2018, p.23)

The 2019 Action Plan again looked to build on the work of previous years, while also considering policy direction for the future (DES 2019).

Digital Strategy for Schools 2015-2020 Action Plan 2020 (DES 2020a)

The final Action Plan for the Digital Strategy for Schools 2015-2020 was launched in 2020. This marked the final year of the strategy, but also a time of turbulence for schools, as March 2020 saw mandated school closures in an effort to contain the spread of Covid-19. In its review of progress made on the Digital Strategy, the 2020 Action Plan outlined that Phase 2 of the Coding in Primary Schools Initiative had been completed and a report on the findings from the initiative had been compiled (DES 2020a). The report found that parents, teachers and principals believed it necessary to build the digital competence of children living in a technology-immersed world (NCCA 2019). It identified "creating with technology, understanding technology, and using technology" as three fundamental aspects of digital competence to be included in any future digital curricula (NCCA 2019, p.90). However, the report also acknowledged the necessity to be cognisant of potential negative impacts of technology on children, including the invasive nature of social media and excessive screen-time (NCCA 2019). The 'Digital Learning Framework (DLF) national evaluation: Starting off Baseline report' was also published at the end of 2019. Some of the findings of this report were consistent with the Digital Literacy Framework trial findings. Both the need for technical support and acknowledgement of the variations in digital technology contexts were highlighted in both reports (Cosgrove *et al.* 2018; Cosgrove *et al.* 2019). Four further issues for consideration were identified in the 2019 report: the positive impact of opportunities to collaborate and share learning, the importance of equitable access to connectivity, the necessity

to communicate effectively about CPD events and the importance of developing a shared understanding of 'embedding' (Cosgrove *et al.* 2019). In 2019, the Inspectorate of the DES conducted an evaluation of digital learning in early learning and care (ELC) settings, primary schools and post-primary schools (DES 2020b). The report on the evaluation acknowledged that most schools had taken positive steps to integrate digital technology in teaching, learning and assessment, and the majority of schools had developed a digital learning plan (a requirement of the Infrastructural Grant Scheme (NCCA 2018)). However, the report also noted that despite some encouraging examples of good practice, "there is still some way to go towards realising the full potential of digital technologies to enhance teaching, learning and assessment" (DES 2020b, p.36). Alongside the underutilisation of technology (see Table 2.8), the report asserted that there was a disparity in technology use between schools, and it was recommended that more "guidance as to what constitutes good practice" was required (DES 2020b, p.35). This finding was echoed in the OECD (2020) series publication 'Education Policy Outlook Ireland'. This paper recognised that Ireland needed to "address the risk of digital divide, including in the education sector", as advised by the Council of the European Union (OECD 2020, p.8). Several innovation projects were adopted or continued during 2019, including a collaborative project between the PDST and Mary Immaculate College, which saw 11 DEIS schools participate in the first LEGO League Junior curriculum (DES 2020a). This project involved pre-service teachers from Mary Immaculate College and classroom teachers working with primary school pupils to build and programme an autonomous Lego robot to solve a series of curricular-based design challenges (ibid). The rollout of the Infrastructural Grant Scheme continued in 2019, with a further €50m issued to schools in 2019 (ibid). Several actions

identified in the Action Plan for delivery in 2020 focused on supporting schools to engage in remote learning, while most other actions looked to build on the work of previous years (ibid).

Table 2.8: *Key findings from 'Digital Learning 2020: Reporting on Practice in Early Learning and Care, Primary and Post-Primary Contexts' (adapted from Butler and Leahy 2022a)*

Key Findings	**Primary**
School had developed a Digital Learning Plan *Knowledge of the six-step School Self Evaluation process helping to improve digital learning in a manageable/incremental way *Making meaningful links between digital learning and priority areas identified for school improvement using SSE process	73%
Digital learning in part of the lesson observed * Used by teachers to creatively engage learners in most lessons but in some lessons only used by teachers and not learners	55%
Lessons with learner collaboration using digital technologies	41%
Use of digital technologies as part of the assessment process	60%
Creation of new knowledge/digital artefacts	not well-established practice
Experienced the same applications/activities in different classes or year groups without adjustment (ages/development stage)	common practice
Unaware of Digital Learning Framework & supporting resources	many teachers
Unaware of supports such as Scoilnet/Webwise	some schools
Unsure how to access external professional learning support	some schools

2.4.4 Policy Development post Digital Strategy for Schools 2015-2020

Ireland's first digital strategy expired at the end of the 2020/2021 school year. In April 2021, the DES initiated a consultation process to review the Digital Strategy for Schools 2015-2020 and to inform the development of a new strategy. At the same time, work was ongoing reviewing the Digital Learning Framework and developing the primary curriculum framework and the new primary mathematics curriculum. The resulting publications from this work (see Table 2.9) will be examined in the coming sections.

Table 2.9: *ICT policy reports and initiatives published in Ireland (2021-2024)*

Year	Initiative/Report
2021	Digital Learning Framework (DLF) national longitudinal evaluation: One year on – Wave 1 report (ERC)
2022	Summary Report: Open call for submissions on the development of a new Digital Strategy for Schools to 2027 (DES 2022a)
2022	Digital Strategy for Schools to 2027 (DES)
2022	Baseline report: Towards a successor digital strategy for schools to 2027 (DES)
2022	Technology Skills 2022: Ireland's Third ICT Skills Action Plan (DES)
2022	Digital Technology - Being a Digital Learner (Butler and Leahy)
2022	Digital Technology - Design Thinking (Kenna 2022)
2022	Digital Technology - Interacting Safely, Ethically and Responsibly (O'Neill)
2022	Digital Technology – Using, Understanding and Creating (Waite and Quille)
2022	Digital Technology – Using, Understanding and Creating - Technical Report (Waite and Quille)
2023	Primary Curriculum Framework for Primary and Special Schools (DES)
2023	Primary Mathematics Curriculum for Primary and Special Schools (DES)
2024	Draft Science, Technology and Engineering Education Specification: For Primary and Special Schools (NCCA)

2.4.4.1 Digital Learning Framework (DLF) national longitudinal evaluation: One year on – Wave 1 report (Feerick et al. 2021)

Following completion of the lifespan of the Digital Strategy for Schools 2015-2020, the third evaluation of the Digital Learning Framework (DLF) was published in June 2021. This report showed an increase in the level of engagement with digital technologies by primary school teachers and pupils since the previous report (Feerick *et al*. 2021). The findings of this report also identified key enablers and challenges to successful Digital Learning Framework implementation. There was considerable crossover between the enablers identified in this report and those of the previous Digital Learning Framework report, including the need for infrastructure, connectivity, collaboration and acknowledgement of diversity. At this stage of review, the key enablers were identified as:

- adequate levels of infrastructure and connectivity;
- effective technical support;
- consultative and collaborative leadership;
- high levels of collaboration among teachers;

- the active promotion of and advocacy for the DLP in the school, and
- CPD that is sustained and tailored to the particular needs of the school.

(Feerick et al. 2021, p.158)

The challenges to the Digital Learning Framework identified in the report were strongly related to the enablers. At this stage of the review, the key challenges identified were sub-optimal infrastructure, maintenance, connectivity and technical support; lack of tailored supports that recognise the diversity of contexts; insufficient awareness of resources; disparity in the measurement and monitoring of levels of practice; sources of funding for future ICT infrastructure following disbursement of the ICT infrastructure grant and the underutilisation of technology for assessment (Feerick *et al*. 2021). The enablers and challenges identified in these three recent ERC reports and the NCCA's report on coding in the primary school should be considered when deliberating the introduction of any initiative in primary school classrooms.

2.4.4.2 Summary Report: Open call for submissions on the development of a new Digital Strategy for Schools to 2027 (DES 2022a)

The DES published a report on the consultation process for the development of the new digital strategy in April 2022. Coming less than a year after the Digital Learning Framework review, many of the same enablers and challenges to the embedding of digital technology were recognised in this report, including infrastructure, technical support, funding, assessment, professional development and leadership (DES 2022a). Perhaps because the consultation was more wide-ranging than the previous school-based evaluations, some additional issues were raised. These included issues around the digital divide, inclusion, Irish medium and Gaeltacht schools, and the importance of links with the curriculum and industry. The report recognised that

while “principles of equity in the current digital strategy are strong”, a stronger message of “no one left behind” could be included in the new strategy (DES 2022a, p.15). While the Covid-19 pandemic highlighted issues for students with poor broadband connectivity and students from lower-than-average household incomes, this report also raised issues for students in Irish medium and Gaeltacht schools and students with special education needs (DES 2022a). The outgoing strategy contained no reference to Irish medium or Gaeltacht schools, and it was suggested this should be amended in the new strategy to ensure all students had access to the same opportunities (DES 2022a). In a similar vein, it was acknowledged that the new strategy should support those with additional needs by focusing on the potential of digital technology to provide personalised learning (DES 2022a). These three findings reinforce the findings of the Inspectorate report on the use of digital technology in early learning and school settings (DES 2020b) and emphasise the need to act on advice from the OECD to address the risk of a digital divide in education settings (OECD 2020). The report also acknowledged parents’ important role in supporting their children’s learning and recognised that they also need opportunities to develop their digital competence (NCCA 2020). Ireland’s ambition to become a “a knowledge-driven and digitally-enabled society” (DES 2022a, p.43) requires that all students develop digital skills for future employability and as a general life skill. However, it was emphasised that how students engage with technology is of critical importance (DES 2022a). The European e-Skills Manifesto declares that full involvement in today’s society requires not only access to technology, but also the necessary skills to take full advantage of the ICT tools available (McCormack 2014). With the increasing importance of critical thinking skills in modern society, the report suggested that digital technology can play a role

in the development of such skills (DES 2022a), something which the NCCA considered in the development of the revised primary mathematics curriculum (NCCA 2016a) and the primary curriculum framework (NCCA 2020). This is a positive step, as the absence of clear curriculum guidance was identified as a contributory factor in schools' lack of engagement to date (Butler and Leahy 2022a). The need for the new strategy "to connect and link in with current and newly developing curriculum" was also recognised during the consultation process (DES 2022a, p.27).

2.4.4.3 Digital Strategy for Schools to 2027 (DES 2022b)

For the last number of years, Irish educators have been endeavouring to develop the competencies young people require to succeed in today's technology-immersed world, while negotiating the aforementioned challenges (Mac Giolla Phádraig 2022). However, despite these efforts, there is still work to be done, as illustrated by the DES Inspectorate report on the utilisation of technology in schools (DES 2020b). According to an OECD (2020) report, 47% of Irish adults have a below-basic level of digital skills, higher than the European average of 42%. The 'Digital Strategy for Schools to 2027' aims to build on the achievements of the previous strategy while further supporting schools to ensure all learners are prepared to thrive in modern society (DES 2022b). This new strategy was designed to complement and reinforce policies such as the 'EU Digital Education Action Plan', the Department of Education Statement of Strategy2021-2023' and 'Harnessing Digital – The Digital Ireland Framework' (DES 2022b). Indeed, the implementation of the new digital strategy is to be informed by two high-priority areas identified in the 'EU Digital Action Plan':

> Strategic priority 1: Fostering the development of a high-performing digital education ecosystem
>
> Strategic priority 2: Enhancing digital skills and competences for the digital transformation
>
> (European Commission 2020, pp10,13)

Taking into consideration the aforementioned strategies, the results from the consultation process and the EU-devised strategic priorities, the digital strategy sets out its objectives (see Table 2.10) under three pillars:

> Pillar 1: Supporting the embedding of digital technologies in teaching, learning and assessment
>
> Pillar 2: Digital Technology Infrastructure
>
> Pillar 3: Looking to the future: policy, research and digital leadership
>
> (DES 2022b, p.16-17)

The first pillar focuses on ensuring that the required supports and resources are available to schools and school personnel to further harness the potential of digital technologies in their practice (DES 2022b). The second pillar addresses connectivity (to include both broadband and network infrastructure), technical support and the provision of guidance on digital infrastructure issues (ibid). The final pillar attends to the need for policy alignment across multiple areas, and the provision of support for school leadership and research to ensure the effectiveness and relevance of the policy throughout its lifespan (ibid). Alongside the objectives of the strategy, several emerging trends related to the effective use of digital technologies, which are also relevant to this study, were identified in the new digital strategy, including the gender gap in digital skills, coding and computational thinking, and digital technologies supporting creativity (DES 2022b). The strategy recognised that women are underrepresented in technology sector jobs (ibid). It outlined its intent to reinforce and complement the work of current and future policies (e.g. the STEM Education Policy and its accompanying implementation plans) to improve gender balance in

this area (ibid). It acknowledged the importance of developing coding skills in schools and endeavoured to keep abreast of ongoing research with respect to both coding and computational thinking at the EU level (ibid). Finally, the strategy identified the potential opportunity digital technology provides to support creativity across a broad spectrum of school activities (ibid). Overall, the main aim of the strategy is to:

> Empower schools to harness the opportunities of digital transformation to build digital competence and an effective digital education ecosystem so as to develop competent, critically engaged, active learners while supporting them to reach their potential and participate fully as global citizens in a digital world.
>
> (DES 2022b, p.11)

Table 2.10: *The objectives of the Digital Strategy for Schools to 2027 (DES 2022b, pp.21,43,52)*

Pilar 1 Objectives
• Empower learners to become confident and competent digital learners. • All new educational policies and curricula will continue to ensure that they have the appropriate and effective use of digital technologies embedded. • Enhance inclusion, equity, learner participation and personalisation through the use of digital technologies by providing clear guidance and support. • Embed the appropriate and effective use of digital technologies for teaching, learning and assessment at each stage of the continuum of teacher education, i.e. Initial Teacher Education, Induction and Continuing Professional Development. • Run an awareness programme to ensure that all teachers and school leaders are aware of supports and resources available relating to the use of digital technologies for teaching, learning and assessment both nationally and internationally. • Ensure provision of flexible, differentiated, needs based teacher professional learning so that there is clear guidance, support and professional learning for all teachers and school leaders in the planning and use of digital technologies in all aspects of teaching, learning and assessment. • Provide further supports to assist schools in their self-assessment of progress in embedding digital technologies in teaching, learning and assessment.
Pillar 2 Objectives
• Establish sustainable funding mechanisms for the purchase and maintenance of digital technology infrastructure in schools. • Meet the requirements for all primary, special schools and post-primary schools to embed digital technologies in their teaching, learning and assessment by the provision of appropriate broadband connectivity. • Provide guidance and advice on the purposeful and effective use of digital technology infrastructure to support the embedding of digital technologies in teaching, learning and assessment. • Research, establish and make available technical support solutions that are appropriate to and accessible for all schools.

• Provide procurement mechanisms for schools, by working with relevant stakeholders, with comprehensive and easy to navigate guidance and support on procurement including how to access those mechanisms, which will assist schools in the purchase of equipment and services aligned to digital learning planning.
Pillar 3 Objectives
• Ensure the Digital Strategy for Schools to 2027 and its associated Implementation Plan supports and complements other relevant strategies and polices both nationally and at EU level. • Provide assistance and guidance to schools to understand how relevant strategies related to the use of digital technologies in teaching, learning and assessment can be implemented. • Continue to engage with and monitor relevant international and EU initiatives including the EU DEAP so that research and supports can be considered and adopted for the Irish context and disseminated to the system. • Promote participation and engagement in EU and national research. • Undertake research in key areas to inform the development of relevant supports and resources for schools. • Continue to support the development and dissemination of high quality resources to promote the safe, responsible and ethical use of the internet and digital technology, informed by national and international policy and best practice. • Promote the use of digital technologies to facilitate communication within the wider school community and education ecosystem particularly involving parents and learners.

2.4.4.4 NCCA commissioned research papers on Digital Literacy (NCCA 2022)

The draft primary curriculum framework identified seven competencies that would enable children to navigate a wide range of situations, contexts and challenges (NCCA 2020). *Being a digital learner* was one of the seven competencies proposed in the draft framework. The draft framework outlined how this competency would "support children to become curious, creative, confident and critical users of digital technology" (NCCA 2020, p.8). To support "the development of associated curriculum provision", the NCCA commissioned a series of papers on digital technology (Butler and Leahy 2022b; Kenna 2022; O'Neill 2022; Waite and Quille 2022a; 2022b) (see Table 2.9). These papers synthesised national and international research on digital technology and digital technology in education to provide a vision for digital technology in the new primary curriculum.

Digital Technology – Being a digital learner (Butler and Leahy 2022b)

The first of these papers sought to examine what *Being a digital learner* meant, the key ideas associated with *Being a digital learner*, and its relevance to children's learning in the Irish context. In their paper Butler and Leahy assert that *Being a digital learner* goes beyond digital competence to include both digital literacy and computational thinking (Butler and Leahy 2022b). In the coming sections, each one of these will be explored, with particular emphasis on computational thinking due to its centrality to this study.

Being a digital learner – Digital Competency

According to Butler and Leahy (2022b), the first of these 'digital competency', despite being a critical component of *Being a digital learner*, is not yet well defined. A review of the literature reinforces this view, as varied definitions can be found, even within the Department of Education's own documents. However, its importance is evident from its prominence in policy documents, including being identified as a priority in the new digital strategy (DES 2022b). In this document, a basic definition of digital competence is provided, "a good foundation in basic digital skills for all learners and a good understanding of the digital world" (DES 2022b, p.37). However, this document directs you to the 'Digital Learning Framework for Primary Schools' for a shared reference including descriptors on digital competence (DES 2022b). The Digital Learning Framework defines digital competence as:

> The set of skills, knowledge and attitudes that enable the confident, creative and critical use of digital technologies to enhance teaching, learning and assessment.
>
> (DES 2017b, p.14)

Alongside this definition, the Digital Learning Framework provides further detail on digital competence learning outcomes and learning experiences (see Figures 2.1 and 2.2). Despite the lack of a uniform definition, digital competence has been identified as a priority in Ireland (DES 2022b), Europe (European Commission 2020) and beyond (UNESCO 2018; Australian Government 2022).

STANDARDS	STATEMENTS OF EFFECTIVE PRACTICE	STATEMENTS OF HIGHLY EFFECTIVE PRACTICE
Pupils enjoy their learning, are motivated to learn and expect to achieve as learners	Pupils use appropriate digital technologies to foster active engagement in attaining appropriate learning outcomes. Pupils use digital technologies to collect evidence and record progress.	Pupils use appropriate digital technologies to foster their active, creative and critical engagement in attaining challenging learning outcomes. Pupils use digital technologies to collect evidence, record progress, evaluate and reflect, and to create new solutions and/or products.
Pupils have the necessary knowledge, skills and attitudes required to understand themselves and their relationships	Pupils have a positive attitude towards the use of digital technologies and are aware of possible risks and limitations. Pupils understand the potential risks and threats in digital environments.	Pupils have a positive attitude towards the use of digital technologies, being aware of possible risks and limitations, and have the confidence and skills to realise the benefits. Pupils can confidently protect their digital identity and manage their digital footprint.
Pupils demonstrate the knowledge, skills and understanding required by the primary curriculum	Pupils can use a range of digital technologies to demonstrate the knowledge, skills and understanding required by the Primary School Curriculum. Pupils use digital technologies effectively to develop their knowledge, skills and understanding in accordance with the content objectives, learning outcomes, skills and concepts of the Primary School Curriculum.	Pupils, in collaboration with their teacher and/or parents, follow their individual learning needs and preferences, with the aid of appropriate digital technologies. Pupils use digital technologies in highly effective ways to develop their knowledge, skills and understanding in accordance with the content objectives, learning outcomes, skills and concepts of the Primary School Curriculum.
Pupils achieve the stated learning objectives for the term and year	Pupils are provided with personal feedback and differentiated support based on evidence gathered using a range of methods including digital technologies. Pupils and/or parents use digital technologies to access information on learners' performance, in a safe and ethical way.	Pupils use evidence gathered by a range of methods including digital technologies to record progress and identify areas for improvement, and have opportunities to address these with their teacher. Pupils and/or parents use digital technologies to access, evaluate and interpret the results of formative, summative, self- and peer-assessments.

Figure 2.1: *Digital Competence Learner Outcomes identified in the Digital Learning Framework for Primary Schools (DES 2017b, p. 5)*

STANDARDS	STATEMENTS OF EFFECTIVE PRACTICE	STATEMENTS OF HIGHLY EFFECTIVE PRACTICE
Pupils engage purposefully in meaningful learning activities	Pupils use digital technologies for sourcing, exchanging of information to develop understanding and support basic knowledge creation.	Pupils use a variety of digital technologies for knowledge creation to source, critique, and manage information and to reflect on their learning.
Pupils grow as learners through respectful interactions and experiences that are challenging and supportive	Digital interactions, among pupils and between pupils and teachers, are respectful and positive, and conducive to well-being. Pupils use digital technologies confidently to deepen their knowledge by engaging in appropriate public discourse and civic participation.	Digital interactions, among pupils and between pupils and teachers, are respectful, challenging and support the well-being of all pupils. Pupils use digital technologies to respectfully communicate, collaborate, and co-create knowledge through active engagement in appropriate public discourse and civic participation.
Pupils reflect on their progress as learners and develop a sense of ownership of and responsibility for their learning	Pupils use digital technologies to collect evidence, record and reflect on their progress, and develop their competence as self-directed learners.	Pupils use digital technologies to creatively and critically develop their competence as autonomous, self-directed learners and are able to set meaningful personal goals for future learning.
Pupils experience opportunities to develop the skills and attitudes necessary for lifelong learning	Pupils have opportunities to apply their digital competence in new situations or contexts and have an age appropriate understanding of how digital technology can support lifelong learning.	Pupils apply their digital competence in innovative ways to new situations or contexts, creatively develop new solutions and/or products, and see themselves engaging in continuing education and training.

Figure 2.2: *Digital Competence Learner Experiences identified in the Digital Learning Framework for Primary Schools (DES 2017b, p.6)*

Being a digital learner – Digital Literacy

The second aspect of *Being a digital learner* identified by Butler and Leahy (2022) is digital literacy. Both the Primary Language Curriculum (NCCA 2019) and the Interim Review of the Literacy and Numeracy Strategy (DES 2017d) identified enhancing the digital literacy skills as a key priority for the future. Indeed, the Department of Education are currently developing a follow-on literacy and numeracy strategy which will include digital literacy. As part of that work, the Department of Education commissioned a literature review which included an undertaking to define each of the three literacies. The final report recognises that digital literacy is more complex than literacy in the traditional sense (Kennedy *et al.* 2023). According to

Kennedy *et al.* (2023), the greater volume of information, coupled with the variety of modalities (text, audio, image, video) accessible through digital technology, means that reader choice and control are amplified, requiring more complex processing and navigation strategies. They view becoming digitally literate as a life-long process that,

> "involves our students in constantly evolving socially situated practices supported by skills, strategies, stances, and dispositions that enable knowledge development and learning, and the generation, representation and understanding of ideas using digital tools not just as consumers, but also as creators and producers."
>
> (Kennedy *et al.* 2023, p.22)

The OECD (2021, p.4) suggest that "as international debate focuses on foreign trolls and conspiracy theorists", the time is right to integrate digital literacy into teaching and learning.

Being a digital learner – Computational Thinking

Computational thinking has increasingly been prioritised as a competence that can be developed in primary schools across the world (European Commission 2016b; European Schoolnet 2015). Wing (2006) popularised the term, however many researchers consider Seymour Papert in the 1980s (Caeli and Yadav 2020; Fessakis *et al.* 2018) or even Alan Perlis in the 1960s (Tedre and Denning 2016; Grover and Pea 2013) to be the first proponents of computational thinking. Papert's original conceptualisation of computational thinking focussed on combining critical thinking skills with the power of the computer to create innovative solutions to real life problems (Butler and Leahy 2022b). However, Wing's (2006) conceptualisation decoupled computational thinking from programming, instead defining it as a fundamental thinking skill which draws on "concepts fundamental to computer science" (p.33). Within this broader definition, computational thinking includes

algorithms, decomposition, abstraction and automation, critical components of problem solving in many disciplines and contexts (Butler and Leahy 2022b). As the concept of computational thinking began to gain traction with the education community, operational definitions began to emerge. Many of these definitions were reflective of Wing's view and included references to both 'thinking' and computer science concepts. For example, in the UK, Selby and Woollard (2013) developed a definition for computational thinking by reviewing multiple previous proposed definitions. This definition has been drawn on widely in UK educational documents, including in 'Computational Thinking: A Guide for Teachers' published by Computers At School (CAS) (Csizmadia *et al.* 2015). Selby and Woollard (2013, p.5) defined computational thinking as:

> an activity, often product oriented, associated with, but not limited to, problem solving. It is a cognitive or thought process that reflects
>
> - the ability to think in abstractions,
> - the ability to think in terms of decomposition,
> - the ability to think algorithmically,
> - the ability to think in terms of evaluations, and
> - the ability to think in generalizations.

Alternative definitions emerged in the literature, perhaps due to the widespread uptake of computational thinking in education systems globally, and the differing socio-political environments within which these education systems are based (Curzon *et al.* 2019). In the US, CSTA and ISTE (2011) developed a definition which emphasised the problem solving rather than the thinking aspect of computational thinking:

> Computational thinking (CT) is a problem-solving process that includes (but is not limited to) the following characteristics:
>
> - Formulating problems in a way that enables us to use a computer and other tools to help solve them.
> - Logically organizing and analyzing data

> - Representing data through abstractions such as models and simulations
> - Automating solutions through algorithmic thinking (a series of ordered steps)
> - Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
> - Generalizing and transferring this problem solving process to a wide variety of problems
>
> These skills are supported and enhanced by a number of dispositions or attitudes that are essential dimensions of CT. These dispositions or attitudes include:
>
> - Confidence in dealing with complexity
> - Persistence in working with difficult problems
> - Tolerance for ambiguity
> - The ability to deal with open ended problems
> - The ability to communicate and work with others to achieve a common goal or solution.
>
> (CSTA and ISTA 2011, p.13)

Like Wing's (2006) definition, computer science concepts such as abstraction and automation were also considered important (CSTA and ISTA 2011). However, perhaps diverging from Wing's principally cognitive focused definition, CSTA and ISTA (2011) incorporated dispositions and attitudes into their definition. According to Tedre and Denning (2016), this ambiguity has led to disagreement on what computational thinking should be taught and how it should be assessed. Curzon *et al.* (2019) suggest that rather than focusing on ambiguity, educators should instead focus on agreement across the various definitions of computational thinking (see Figure 2.3).

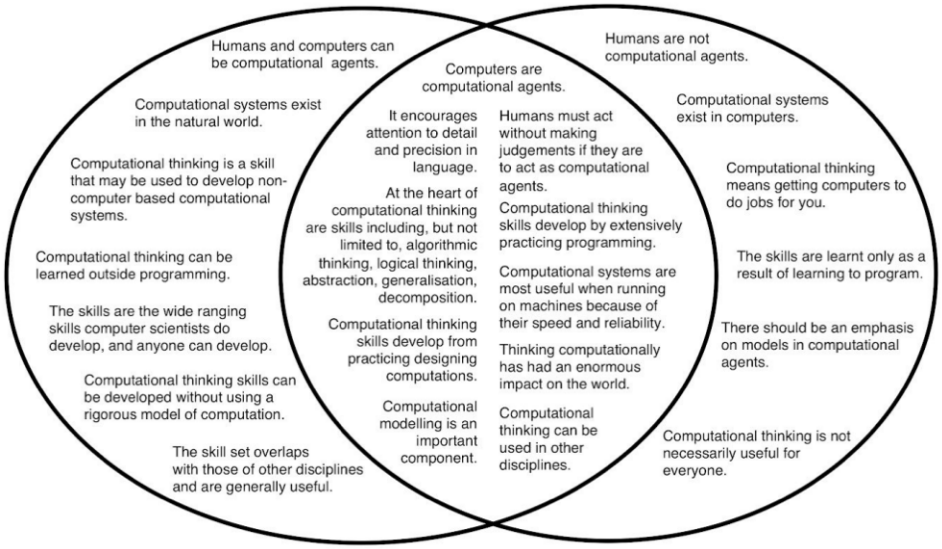


Figure 2.3: *Agreement and disagreement around what Computational Thinking should be (Curzon et al. 2019, p.515)*

In the Irish context, research on the integration of computational thinking into the curriculum commenced in 2016 following a request from the then Minister for Education and Skills, Richard Bruton. Since then, the NCCA has conducted several investigations and published three reports on the possible inclusion of computational thinking in the primary school curriculum (NCCA 2018; Millwood *et al.* 2018; NCCA 2019a). The NCCA (2018) report, 'Investigation of curriculum policy on coding in six jurisdictions' identified how computational thinking was integrated into the curricula of six jurisdictions. This investigation found that across the six jurisdictions, computational thinking is being implemented from children's first year of primary school, integrating coding across several curricular subjects is recommended, CPD for teachers is a necessity and using, understanding and creating with technology are considered key digital competencies (NCCA 2018).

Millwood *et al.* (2018) sought to define computational thinking for the Irish primary school context and consider what learning outcomes might be appropriate. They defined computational thinking as "competence in problem solving & design to create useful solutions, informed by the possibilities that Computing offers"

(Millwood *et al.* 2018, p.8). However, echoing the views of the NCCA (2018), Millwood *et al.* (2018) advised against narrowing the focus to developing computational thinking solely through coding or programming. They also recommended adopting a playful approach to the integration of technology. With regard to the development of learning outcomes, Millwood *et al.* (2018) were keen to include dispositional elements such as perseverance and openness, reaffirming the recommendations of CSTA and ISTE (2011). They proposed adopting the metaphor of 'head, hands and heart' to represent the competences of computational thinking (Millwood *et al.* 2018, p 7). Within this framing of computational thinking, head represents the knowledge or mental models and strategies, hands, the craft or skills using tools and media, and heart, the character or dispositional aspects that contribute to successful computational thinking (Millwood *et al.* 2018). The final report, 'Primary Developments: Final Report on the Coding in Primary Schools Initiative', drew on findings from primary schools on the integration of both coding and computational thinking in the primary curriculum (NCCA 2019). Many of the findings echoed those of Millwood *et al.* and the NCCA (2018), including the need to support children becoming digitally competent from an early age, adopting a playful approach, simultaneously developing soft skills such as collaboration and creativity, and the need for CPD for teachers (NCCA 2019). Interestingly, while teachers supported the view that coding and computational thinking should be developed through cross-curricular integration, they believed that the essential skills needed to be taught explicitly first (NCCA 2019). Consequently, it was recommended that an explicit space in the curriculum was required "to enable teaching and learning of the fundamental skills and concepts of coding and computational thinking" (NCCA 2019, p.92). STEM, one of five curricular areas

included in the primary curriculum framework could provide the necessary 'explicit learning space' (NCCA 2019).

The 'Being' in Being a Digital Learner

Alongside the three aforementioned components of *Being a digital learner,* Butler and Leahy (2022) are also keen to stress the importance of the 'being' element of the competence. They view the 'being' as emphasising the person rather than particular digital technologies. Therefore, they propose that implementation of this competence would require flexibility and interpretation at the school level (Butler and Leahy 2022). This is perhaps a caution from Butler and Leahy, against seeking a static definition of *Being a digital learner* within a constantly evolving technological and societal landscape (Butler and Leahy 2022).

Digital Technology – Design Thinking' (Kenna 2022)

In the research paper 'Digital Technology – Design Thinking', Dr Kenna sought to explore how design thinking could support the development of the competency, *Being a digital learner*. In particular, she highlighted the relevance of design thinking to computational thinking (ibid). This view is shared by Mitch Resnick, who suggests that computational thinking is best developed through the creation of personal projects, which help children develop their voice and their identity (Resnick 2017). While computational thinking and design thinking share many characteristics, it is this human aspect that distinguishes designing thinking from the pure logic of computational thinking, according to Dr Kenna (2022). Maeda (2019) proposes that in the future, this human aspect will be critical in controlling machines whose artificial intelligence will eventually surpass our own. Developing a curriculum to

cultivate this type of thinking, particularly in an ever-evolving technology world, presents a challenge for schools (Kenna 2022). However, Kenna (2022) posits that "design thinking offers a multi-faceted approach suited to addressing complex challenges such as the requirements of becoming a digital learner" (p.4). Design thinking can be used to solve complex problems by assuming a mindset that is unperturbed by uncertainty, while engaging collaboratively and iteratively in a range of practical activities (e.g. research, ideation, design making and testing) (Kenna 2022). Kenna identified project-based coding, using Scratch, as an example of how the design thinking process can be applied in the classroom. In particular, she highlighted two pedagogical approaches advocated by Resnick (2017), incorporating the 'four Ps' (projects, passion, peers and play) and adopting an iterative creative process (imagine, create, play, share, reflect) (Kenna 2022).

Digital Technology – Interacting Safely, Ethically and Responsibly (O'Neill 2022)

In this paper, Professor O'Neill endorses the decision to include a digital technology component in the new primary curriculum, citing the ample literature and policy debate regarding children's engagement with digital technology (O'Neill 2022). This paper explores the importance of supporting children in becoming safe, ethical and responsible users of technology in such circumstances. O'Neill (2022) highlights four curriculum design elements that he feels will be central to achieving this aim. Similar to the ideas set forth by Kenna (2022) and Resnick (2017), O'Neill (2022) highlights the importance of play in children's digital technology experiences. According to O'Neill (2022), fun and free play activities support children's curiosity, creativity and inventiveness in their use of technology. Again, echoing the views of Kenna (2022), O'Neill (2022) highlights the importance of empowering children to

contribute actively and positively in an increasingly data-driven society. He relates this idea to the broader concept of digital citizenship, encompassing notions of responsible technology use with the closely-linked notions of 'global citizenship', 'global competence', 'digital competence', 'digital literacy' and 'media and information literacy' (O'Neill 2022). The OECD (2015), EC (2016c) and UNESCO (2022) have all recognised the potential afforded by digital technology to open up new opportunities for civic participation. The third aspect of curriculum design deemed by O'Neill (2022) as critical to the inclusion of digital technology in the curriculum was the positioning of digital technologies within the curriculum. He advocated for a 'transversal curriculum treatment' of the proficiencies associated with *Being a digital learner* (O'Neill 2022, p. 22). In particular he highlighted the relevance of these proficiencies to being mathematical, creative, and communicating. The latter two resonating with the pedagogical approaches of Resnick (2017), which included collaborating, creating and sharing with peers. The final aspect of curriculum design highlighted by O'Neill (2022) was the necessity to develop age-appropriate and culturally-relevant learning resources to support the development of the appropriate digital technology knowledge, skills, values, attitudes and understandings. The necessity of developing age-appropriate outcomes to guide children's technology experiences was also a recommendation arising from the NCCA's investigation into coding in the curriculum (NCCA 2018).

Digital Technology – Using, Understanding and Creating and Digital Technology – Using, Understanding and Creating: Technical Report (Waite and Quille 2022a; 2022b)

The final two papers in the research series look to further explicate the digital technology aspect of the Mathematics, Science and Technology curricular area (NCCA 2020) from an 'using, understanding and creating' perspective. The first paper is a technical report that synthesises previous research in digital technology and presents a Concept and Processes Model (CP Model) developed to support learning in digital technology (Waite and Quille 2022a). The second paper is an abridged version of the full report (Waite and Quille 2022b). The CP Model they present was developed by integrating curricular statements arising from school pilots with theoretical competency model statements (Waite and Quille 2022b). The CP model identifies six concepts and eight processes (see Figure 2.4) that represent the key ideas and skills Waite and Quille believe should inform development of the new curriculum learning outcomes and progressions (Waite and Quille 2022b). Waite and Quille (2022a) specify that it is not expected that each concept and process would be addressed by every child in every class. Rather, teachers would make decisions based on their specific context, taking into consideration things like learning preferences, misconceptions, pedagogy and teacher knowledge and confidence (Waite and Quille 2022a). While the report does not include a review of digital technology pedagogies, it does provide examples of specific pedagogies and how they might fit with the concepts and processes included in their model. Some of the pedagogies mentioned are the Use, Modify, Create (Lee *et al.* 2011), PRIMM (Predict, Run, Investigate, Modify, Make) (Sentance *et al.* 2019) and sequences of targeted tasks such as debugging activities and Parson's problems (Raspberry Pi

2021), all of which will be discussed further in the next chapter (Waite and Quille 2022a). Waite and Quille (2022b) hope that this work will support policymakers, researchers and practitioners in the formation of learning outcomes and progression maps for the new curriculum.

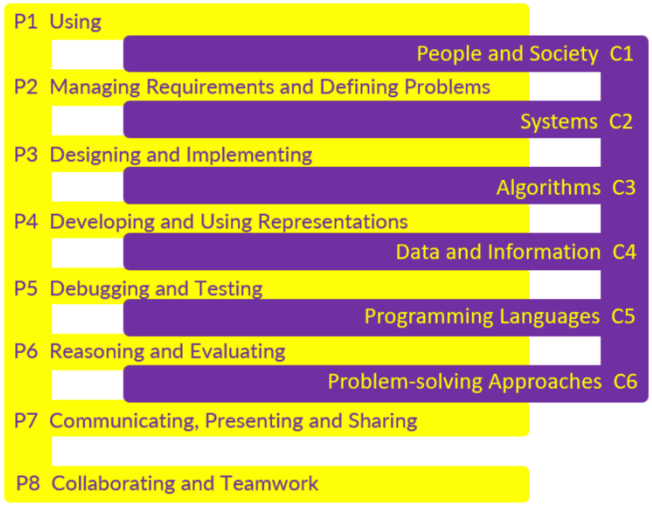


Figure 2.4: *Concepts and Processes Model (C - Concepts, P - Processes and Production Skills) (Waite and Quille 2022a, p.14)*

2.4.4.5 Primary Curriculum Framework for Primary and Special Schools (DES 2023a)

The 'Primary Curriculum Framework for Primary and Special Schools' was launched by the Minister for Education, Norma Foley, in March 2023. The framework outlines the vision, principles and components for a modern, redeveloped primary curriculum (DES 2023a). Teachers and school leaders will use this curriculum vision, principles and components, alongside the learning outcomes developed for the respective curricular areas to create a curriculum tailored to the needs of the children in their school (NCCA 2020). The rationale for reviewing the primary curriculum recognised the need to respond to national priorities and take account of new aspects of learning that have emerged in Ireland today (NCCA 2020). The new primary curriculum framework (DES 2023a) recognises the proliferation of

technology in modern society and the necessity to prepare children to thrive in such a society. In the revised primary curriculum, STEM is now one of five curriculum areas. Outside of the explicit technology curriculum space, *being a digital learner*, will be integrated across all curriculum subjects from junior infants to sixth class (DES 2023a). The primary curriculum framework explicates that this competency "supports children to become curious, creative, confident, and critical users of digital technology" (NCCA 2023a, p.10).

2.4.4.6 Primary Mathematics Curriculum for Primary and Special Schools (DES 2023b)

The new primary mathematics curriculum, launched in September 2023, is fully aligned with the vision, principles and components of the new primary curriculum framework (DES 2023b). Situated in the STEM curriculum area, mathematics provides an essential foundation upon which children's STEM learning can be developed and refined (DES 2023b). The curriculum proposes that

> as children work towards the Learning Outcomes in the Mathematics curriculum and engage in rich mathematical learning experiences, they simultaneously build and develop the key competencies.
>
> (DES 2023b, p.5)

The curriculum later recognises the role technology can play in supporting mathematical thinking, developing conceptual understanding, reducing procedural load and representing complex ideas (DES 2023b). Subsequently, the curriculum provides two examples of how *being a digital learner* can be developed through mathematics learning (Figure 2.5). The curriculum is supported by a mathematical toolkit that provides a range of support for implementing the curriculum (DES 2023b). This toolkit comprises of mathematical concepts, progression continua, support materials and examples of children's mathematical learning (DES 2023b).

Being a digital learner	• Using a range of digital technologies to expand how to engage with, express and represent complex mathematical ideas • Reducing complexity and allowing for the development of higher-order thinking

Figure 2.5: *Examples of how Being a digital learner can be developed through mathematics learning experiences (DES 2023b, p.6)*

Work on the development of support materials and examples of children's learning has commenced, while the progression continua have already been developed. The progression continua provide a sample learning trajectory to support teachers in designing rich mathematical experiences for their pupils. Within these progression continua there are numerous examples of how digital technology can be incorporated into children's mathematical work (Figure 2.6). Further examination of the examples provided reveals that many of them are examples of what Waite and Quille (2022b) describe as, 'using' technology. The children use digital technologies to achieve a mathematical purpose, for example, "conducts chance experiments with both small and large numbers of trials using appropriate digital technologies." (NCCA 2023b). However, there are fewer examples that require the 'creation of technology', which is identified as a key feature of *being a digital learner* in the primary curriculum framework (DES 2023a). The necessity to engage children as 'active producers' of technology, not just 'passive consumers', has been strongly espoused in the literature (Altun Yalcin *et al.* 2020; Resnick 2017; Kafai and Peppler 2011; Papert 1980). Parents involved in the 'Coding in Primary Schools Initiative' expressed a similar view, believing that children's technology experience should not be solely consumer-based but should focus on productive-based technology, promoting critical thinking and creativity (NCCA 2019a).

Chance	▪ Uses technology, to rapidly replicate random events (For example: toss coins, spinners or roll dice) for efficient investigations. ▪ Uses technology to rapidly identify the set of all possible outcomes in an investigation. ▪ Conducts chance experiments with both small and large numbers of trials using appropriate digital technologies.
Measures	▪ Uses technology to input and represent measurements for a range of purposes. ▪ Inputs appropriate measurements into computer programmes for the purposes of generating simulations and models.
Shape and Space	▪ Discusses mathematical features [scale, relative distance between locations] of conventional maps and digital maps or route-planning tools. ▪ Programmes simple digital devices to navigate appropriate maps or grids. ▪ Uses manipulatives or programmable devices to explore position, movement and direction. ▪ Visualises and checks with suitable materials or technology whether given nets will make specified 3-D shapes. ▪ Selects materials and/or uses technology to construct or represent shapes. ▪ Solves tasks and problems involving technology/ virtual manipulatives. ▪ Selects appropriate materials/ digital tools to explore and represent shape. ▪ Selects appropriate materials/digital tools to create precise diagrams of shapes using units of measurement. ▪ Uses simple programming to construct models. ▪ Selects appropriate materials/ digital tools to explore and represent shape movements. ▪ Selects appropriate materials/ digital tools to draw and label shape movements. ▪ Selects appropriate materials/digital tools, or creates labelled drawings to investigate, record or explain geometrical ideas [may include detail of angle or other measures]. ▪ Selects appropriate materials/digital tools, or creates precise labelled drawings, to investigate and/or justify conjectures. ▪ Instructs technology to perform a range of transformations. ▪ Uses software /technology to solve transformation based problems in meaningful contexts.

Figure 2.6: *Learning experiences included in the NCCA (2023b) progression continua that incorporate digital technology*

2.4.4.6 Draft Science, Technology and Engineering Education Specification: For Primary and Special Schools (NCCA 2024a)

The 'Draft Science, Technology and Engineering Education Specification: For Primary and Special Schools' (NCCA 2024a) was launched in March and according to the NCCA's website, the specifications for STEM will be finalised in early 2025 (NCCA 2024b). This specification was developed to complement the primary mathematics curriculum and together they comprise the new primary STEM curriculum. Within the specification technology is identified as a strand (a central category of learning). The learning outcomes for the 'Technology' strand enable

children to develop awareness of the different types of technology and appreciate how they help us to live and work, to explore how these technologies operate, and to create programs to achieve a desired outcome (NCCA 2024b). The curriculum recommends supporting children in achieving these aims by providing children with learning experiences in computational thinking, initially through unplugged activities before progressing to plugged activities. A sample of learning experiences that foster computational thinking are included in the curriculum document. These can be seen in Figure 1.1.

2.5 Policy Development in STEM Education

A significant backdrop to the integration of technology in the primary curriculum has been increasing recognition of "the critical importance of STEM disciplines for modern society" (STEMerg 2016, p.3). In acknowledgment of this, in 2013, the then Minister for Research and Innovation, Seán Sherlock requested a review of STEM education in Irish schools. The STEM Education Review Group (STEMerg) prepared and published a report, 'STEM Education in the Irish School System'. Among the findings of the review relevant to this study, was the significant gender imbalance in STEM disciplines, a strong informal STEM education sector that is being under leveraged, and the lack of a national policy on STEM education (STEMerg 2016). In response to these findings, 'The STEM Education Policy Statement 2017-2026' was published in 2017. Within this document technology was defined as

> a range of fields which involve the application of knowledge, skills and computational thinking to extend human capabilities and to help satisfy human needs and wants, operating at the interface of science and society.
> (DES 2017e, p.6)

The inclusion of computational thinking in the definition of technology perhaps illustrates its importance in the minds of policymakers. This policy document identified four areas of policy development and action that can be seen in Table 2.11.

Table 2.11: *The four pillars of policy development and action*

Pillar 1. Nurture learner engagement and participation
Pillar 2. Enhance early years practitioner and teacher capacity
Pillar 3. Support STEM education practice
Pillar 4. Use evidence to support STEM education

Some of the proposed outcomes relevant to technology in education were DES (2017e):

- develop an increased awareness of the importance of STEM education among all early year settings and school communities
- increase the uptake of leaving certificate Technology, particularly among girls
- provide quality STEM career information to school communities
- increase partnerships between schools and the wider STEM community
- provide quality CPD to all relevant teachers and early year practitioners
- all teachers and early year practitioners to adopt an inquiry-oriented approach in their teaching of STEM
- all teachers and early year practitioners to adopt a cross-curricular approach to their teaching of STEM
- access to high-quality curricula and highly effective practice exemplars for all STEM subjects at all levels of education
- increased access to extra-curricular STEM opportunities in schools
- availability of STEM awards at primary and post-primary levels
- increased research across all facets of STEM education
- review mechanisms for established STEM education initiatives and activities, alongside reports from the Department's Inspectorate

This plan was set to be achieved in three phases: enhancing (2017-2019), embedding (2020-2022) and realising (2023-2026). However, the embedding phase was delayed, and the enhancing phase was extended to 2022, due to the impact of Covid-19 (Government of Ireland 2023). Two key reports have reviewed progress on the

implementation of the plan to date: ‘STEM Education 2020: Reporting on Practice in Early Learning and Care, Primary and Post-Primary Contexts’ (DES 2020c) and ‘STEM Education Implementation Plan – Phase 1 Enhancing Progress Report’ (DES 2023d). These reports highlight the achievements of the implementation plan to date, as well as some areas where the implementation plan has been less impactful. The DES (2020c) report identified the lack of a national programme with clear links to the STEM policy statement, difficulties relating to the positionality of STEM within the curriculum, the need for additional support in terms of exemplars of best practice, STEM resources and CPD, as barriers to successful implementation. The DES (2023d) report recognised that while progress has been made on the uptake of STEM subjects, the difference is small. The persistence of gender inequality in STEM education was highlighted by both reports as an issue which needs to be addressed, and this will be explored further in the next section (DES 2023d; DES2020c). These findings were used to inform the ‘STEM Education Implementation Plan to 2026’ (Government of Ireland 2023) launched in March 2023. This plan sets out the actions necessary to continue the implementation of STEM across the education system, which are summarised in Table 2.12. So, while it can be seen that progress has been made in the integration of STEM in education, similar issues emerged to those identified in the technology in education research and policy documents. Lack of exemplars of good practice, lack of CPD and lack of an explicit place in the curriculum have all hindered the progress of STEM in education.

Table 2.12: *Actions necessary to continue the implementation of STEM across the education system (Government of Ireland 2023, p.7)*

- *Provision of examples of what STEM education can look like from early learning and care to post primary level*
- *Supports on what integrated STEM looks like at primary and post primary school level*
- *A range of quality professional learning experiences for early years educators and teachers across primary and post primary schools to support staff with STEM content knowledge, in planning and implementing integrated STEM activities across all three levels*
- *Ensuring that student teachers in initial teacher education have opportunities to engage in and teach STEM lessons*
- *Enhancing the partnership between schools and business/industry and the research community*
- *Provision of information on STEM careers and courses and equitable access to STEM/STEM and the Arts role models*
- *Provision of a central repository of resources and exemplars of STEM/STEM and the Arts learning opportunities*
- *Continued review of STEM curriculum and assessment across all levels*
- *Provision of funding to support projects that engage children and young people in STEM in primary and/or post primary schools.*

2.6 Gender Disparity in Technology Education

Gender inequality has been identified as a perennial issue in the STEM disciplines (Goos *et al*. 2020). Women continue to be greatly underrepresented in numerous STEM fields in Ireland (STEMerg 2016). According to the 'STEM education in the Irish school system' report, fewer than 25% of the STEM workforce in Ireland are women (STEMerg 2016). This figure is similar to those reported in the UK, where only 17% of the technology sector workforce are women (Millen *et al*. 2019) and the US, where only 25% of the computing workforce are women (Ashcraft *et al.* 2016). The existence of this gender gap means women and girls are failing to reach their full potential and, it contributes to the skills gap in the technology workforce (Gill 2014).

2.6.1 Factors Influencing Girls' Participation in Computing

This persistent gender gap is not correlated with access as Irish girls and boys have equal access to computers at school and, outside of school and the home, girls have greater access to computers than boys, according to the OECD (2015). Instead, intangible factors (e.g. students' interests, parental influence and cultural norms) have been identified as barriers to girls' engagement with technology (OECD 2015). Several researchers have found that gender stereotypes regarding STEM negatively impact girls' STEM career interest and consequently their career decisions (Capobianco *et al*. 2017; Starr 2018; Luo *et al*. 2021; O'Brien 2022;). Margolis and Fisher (2002) also recognised the presence of negative stereotypes of those associated with the computer science field. They found that the negative connotations of the 'geek myth' were felt more acutely by young female students than their male counterparts (Margolis and Fisher 2002). The Accenture report, 'Powering economic growth: Attracting more Young Women into Science and Technology' identified four key barriers operating in the Irish education system: negative stereotypes, negative perceptions regarding subject difficulty, lack of information about career options and a disconnect between students' subject leaving certificate subject choices and industry's skills needs (STEMerg 2016). In 2020, a 'Review of Literature to Identify a Set of Effective Interventions for Addressing Gender Balance in STEM in Early Years, Primary and Post-Primary Education Settings' was undertaken by Goos *et al.* (2020) with the aim of addressing gender disparity. The review noted that of the STEM disciplines, technology and engineering education were under researched compared with mathematics and science, possibly due to their respective statuses in school curricula (Goos *et al.* 2020). However, from the available literature, Goos *et al.* (2020) identified similar

factors to those highlighted in the OECD (2015) report. Learner, family and school level factors were all found to influence girls' participation in the field of technology (Goos *et al.* 2020). The learner-level factors included fear, lack of prior experience, low confidence levels, gender stereotypes, prejudices, social isolation and lack of role models. These factors interacted with family and school level factors, such as uninformed parents and guardians, lack of parental support, inadequately prepared or unenthusiastic teachers, curriculum design and "uninviting, uncomfortable, and often overtly 'masculine'" learning environments (Goos *et al.* 2020, p.31). According to Goos *et al.* (2020), these factors are all reflective of the beliefs and attitudes of wider society about the suitability of the suitability of technology as a female pursuit. Curricula relevance has been identified by many as a barrier to girls' engagement with technology (Goos *et al.* 2020; Gill 2014; AAUW 2000b). Sherry Turkle, co-chair of the AAUW Educational Foundation Commission on Technology, Gender, and Teacher Education, proposes that:

> Girls are critical of the computer culture, not computer phobic. Instead of trying to make girls fit into the existing computer culture, the computer culture must become more inviting for girls.
>
> (AAUW 2000b, para.3)

Similarly, Crick (2017) says that instead of asking "what is wrong with women" we should be asking "what is wrong with computing" (p16). Van Eck (2006) believes that if girls are provided with positive technology experiences, their attitudes toward technology can improve. According to the OECD (2005) report 'Are Students Ready for a Technology-Rich World?', boys are more likely to engage in programming activities than girls, whereas there is little disparity between genders in other computer literacy skills such as word processing or email. Morris and Trushell (2014), who examined the role of gender on attitudes towards programming, also found that programming is predominantly a male domain. However, Burnett *et al.*

(2010) found that the programming language used can significantly impact girls' willingness to engage in programming activities. These are all important considerations for the NCCA as they consider the inclusion of programming in the primary school curriculum.

2.6.2 Gender Equity in Computing – What Would That Look Like and How Can We Get There?

The literature reviewed in the previous section highlights the need to remove barriers to achieving gender equity in computing education and beyond. AAUW (2000a) identifies two contrasting definitions of 'gender equity' in computing. The first views gender equity as getting more girls to participate in computer-related disciplines, while the second emphasises the integration of girls' insights into computer culture (AAUW 2000a). However, the AAUW Educational Foundation Commission on Technology, Gender, and Teacher Education does not see these perspectives as competing, rather it views them as mutually reinforcing:

> One of the values in getting more girls and women in the computer pipeline is that their greater presence may transform the computer culture overall; by the same token, changes in the e-culture itself—the ways technology is discussed, valued, and applied—would invite more girls and women to participate fully in that culture.
>
> (AAUW 2000a, p.x)

The AAUW (2000a) recommend the following immediate actions to promote gender equity:

- Integrate computing across the curriculum
- Redefine Computer literacy as a life skill
- Respect and encourage multiple entry points to computing
- Challenge the current stereotype of computer professionals
- Develop tech-savvy teachers
- Raise awareness of the need for gender equity in computing
- Raise awareness of career prospects in computing
- Ensure girls can identify themselves in the culture of computing
- Support initiatives that encourage participation of girls in computing

Despite being published twenty years later, the Goos *et al.* (2020) report identifies similar actions are needed to remove barriers for girls in computing, perhaps suggesting that little progress has been made in the intermittent years (see Table 2.13).

Table 2.13: *Proposed Shifts in Emphasis to Achieve Gender Balance in STEM Education (Goos et al. 2020, p.61)*

Shift away from…	Shift towards…
STEM education interventions mainly aimed at post-primary school students	Inclusive STEM education programmes that start in early childhood and continue through adolescence into adulthood, in both formal and informal educational settings
One-off or short-term interventions that lack connections to a broader purpose or programme	Long term, coordinated, multi-agency partnerships between families, schools, higher education, government departments, learned societies, business and industry, community organisations
Seeking to change girls' attitudes, beliefs, and behaviours	Seeking to change STEM education structures, practices, and the representation of STEM in wider society, that create barriers to gender-balanced participation

Goos *et al.* (2020) recommend intervention strategies be targeted at learners, family, school and early years, and society. According to Goo *et al.* (2020), productive STEM identities need to be fostered in young girls while simultaneously addressing negative stereotypes. Family and school influences should be addressed by building the beliefs, attitudes and knowledge of parents and teachers, and developing gender-inclusive curricula and learning environments. They also acknowledge the need to address the wider societal issues affecting female participation and recommend the "communication of gender-inclusive social and cultural norms; promotion of approachable role models for girls and boys" (Goos *et al.* 2020, p.32). Indeed, Markus and Nurius (1986, p.954) assert that the "pool of possible selves" arise from

an “individual’s sociocultural and historical context”, from the models provided by the media and from “the individual's immediate social experiences”.

The work of the Gender Balance in STEM Advisory Group (Goos *et* al. 2020) resulted in the publication of the DES (2022c) ‘Recommendations on Gender Balance in STEM Education’ report. This report identified four key areas of action, distinct from the actions outlined in the ‘STEM Education Implementation Plan 2017-2019’ (DES 2022c). These four actions are:

1. Improve equity of participation across all STEM curriculum areas/subject choices by instilling whole school culture change, to include early years leaders and educators, school leaders, teachers, learners and parents/guardians,
2. Provide effective support in relation to practice in STEM for early years educators and teachers,
3. Support equitable learner access to, and experiences of, STEM to inspire learning, foster creativity and prepare for later engagement and success,
4. Support a societal and cultural shift to address current barriers to gender balance in STEM.

(DES 2022c, p.6)

Both Goos *et al*. (2020) and the AAUW (2000a) recommend that actions to promote gender equity need to be delivered from an early age, as they found that girls’ views on technology tend to be formed when they are young. This provides further motivation for the Irish government to include technology experiences in the primary school curriculum.

Section 2: Computational Thinking in Primary School Curricula

2.7 Computing in the Classroom

The previous section focussed on key policies relating to computing in education. Whereas many of the early policies on ICT in Education were focused on equipping schools with technology, the 'Statement of Strategy 2005-2007' policy document represented a shift in emphasis to digital literacy. Since the turn of the century, several reports and policy documents have suggested that we need to do more to realise the potential of technology and to prepare our students for life in a digital society. Despite the publication of numerous reports and policies aimed to realise the potential of computing in education, progress has been slow. However, governments continue to recognise the importance of technology in teaching and learning, as evidenced by the number of countries opting to introduce computer science and coding to their school curricula. The next section moves to the classroom and explores how the aforementioned policies have been implemented in schools, focusing particularly on coding and computational, the two key concepts relevant to this study.

2.8 Computational Thinking in the Curriculum

Introducing Computer Science subjects to the school curriculum at primary and post-primary levels is an emerging international trend. With the introduction of computer science as a core curriculum subject there has been a plethora of research studies, initiatives and policy reports on the topic. Inevitably, the content of curricula in each country will be different. However, most of these curricula, including the Irish curricula, aim to develop the programming skills and computational thinking of their students (García-Peñalvo *et al.* 2016; European Schoolnet 2014; DES 2018). Since

Jeanette Wing's seminal paper in 2006, efforts have been made to include the development of computational thinking at all levels of education across the world. In her paper, Wing (2006) described computational thinking as a fundamental skill for everyone, equating its importance with reading, writing and arithmetic. Programming, which educators Perlis (Katz 1960) and Papert (1980) had previously advocated as a tool that could be leveraged to develop higher order thinking, has begun to make a revival in schools.

2.8.1 Computational Thinking in Irish Curricula

In Ireland, the development of computational thinking is central to the short course in Coding introduced at Junior Cycle level in 2016 and the Computer Science subject at Leaving Certificate level first offered by forty schools in September 2018. The NCCA offer two definitions of computational thinking in the draft curriculum for Leaving Certificate Computer Science. Firstly, they describe computational thinking as "a thought process (or a human thinking skill) that uses analytic and algorithmic approaches to formulate, analyse and solve problems" (NCCA 2017b, p10). However, although they recognise that computational thinking shares some characteristics with problem solving, they clarify that computational thinking is "a problem-solving process that involves designing solutions that exploit the power of computers" (NCCA 2017b, p. 20). Hence, this definition highlights that what distinguishes computational thinking from problem solving is that the solutions to the problems must involve programming, automation and computer systems. This is reflected in Strand 2 of the Leaving Certificate Computer Science course, where students learn how to design solutions that exploit the power of computers (NCCA 2017b). It is also evident in Strand 3, where students develop a computational artefact, which can be a game, web page, digital animation, app, etc., which solves a

specific problem (NCCA 2017b). Although no specific definition of computational thinking was offered in the specification for the Junior Certificate short course, it states that students will develop computational thinking through the building and creating of software projects (NCCA 2016d). As the title of the course suggests, many of the activities designed to develop computational thinking are programming based. For example, the Junior Certificate Specification states that to develop concepts of computational thinking students should write their own code for programming tasks and they must also have the opportunity to analyse code and determine its purpose and identify any errors (NCCA 2016d). At primary level, it was proposed that opportunities to develop computational thinking could be included as part of the new primary mathematics curriculum. Within this context, Millwood *et al.* (2018) defined Computational Thinking as a "competence in problem solving & design to create useful solutions, informed by the possibilities that Computing offers" (p.8). This definition also suggests that students would exhibit competence in computational thinking by creating using technology. While teaching everyone the skill of computational thinking is a worthy goal, there are several pedagogical challenges. It is not simply a case of repackaging current curricula and introducing them at an earlier stage of the education cycle as has been done in some contexts (Lu and Fletcher 2009). The following section explores the pedagogical considerations in the teaching and learning of computational thinking at the primary school level.

2.8.2 Computational Thinking in International Curricula

In their survey of Ministries for Education, European Schoolnet (2014) found that six of the twenty-one countries referred to computational thinking in their curricula at either primary or secondary level. These were Ireland, Belgium (Flanders), the Czech Republic, the Netherlands, and Poland. However, the definition of the term

differed in each country. For example, Belgium (Flanders) uses computational thinking interchangeably with programming, emphasising the digital aspect. On the other hand, Poland views computational thinking as "a collection of mental tools", emphasising the thinking aspect (European Schoolnet 2015, p.28). In recent years, additional countries have introduced computational thinking to their primary school curricula. Three English-speaking countries that have introduced new primary school computer science curricula, that emphasise computational thinking in the past decade are England, Australia, and the US. Computational thinking has been integrated into a Computing curriculum in these three cases. However, some countries, like Ireland, do not have a specific computing subject at the primary level. These countries often adopt a cross-curricular approach, integrating computing into other subjects, often mathematics (European Schoolnet 2015). These contextual differences are important to consider when examining the practices of other jurisdictions.

2.8.2.1 Computational Thinking in the English Curriculum

England introduced a new National Curriculum for Computing in 2014. This curriculum was strongly influenced by a report published by the Royal Society, 'Shut down or restart?' (Furber 2012). This report was critical of computing education in England, complaining that it was too focused on digital literacy (ability to use computers) to the detriment of other aspects of computing (Furber 2012). They argued that computing should be divided into three distinct categories: Computer Science (academic discipline including programming languages, algorithms etc.), Digital Literacy, and Information Technology (use of computers in society). They argued for greater emphasis to be afforded to important aspects of computer science

such as computational thinking and programming skills. Computational thinking was defined as:

> "the process of recognising aspects of computation in the world that surrounds us, and applying tools and techniques from Computer Science to understand and reason about both natural and artificial systems and processes".
>
> (Furber 2012 p.29)

Following on from this report, computational thinking was identified as a core component of the National Computing curriculum in England (Department for Education 2013). In the computational thinking guide for teachers developed by Csizmadia *et al.* (2015), computational thinking is defined in terms of six concepts (logic, algorithms, decomposition, patterns, abstraction and evaluation) and five approaches (tinkering, creating, debugging, persevering and collaborating) (see Figure 2.7). Within the English curriculum, it is specified that computational thinking should be developed through a variety of programming activities (Department for Education 2013).

- In Key Stage 1 (5-7 year olds) students learn about algorithms by creating and debugging simple programs and use logic to predict the behaviour of programs.

- In Key Stage 2 (7-11 year olds) students use decomposition to design and write code for programs and use logic to explain how programs work and to detect errors in programs.

- In Key Stage 3 (12-14 year olds) students must use at least two programming languages (including one textual language) to solve computational problems and students should become familiar with the application of Boolean Logic in programming.

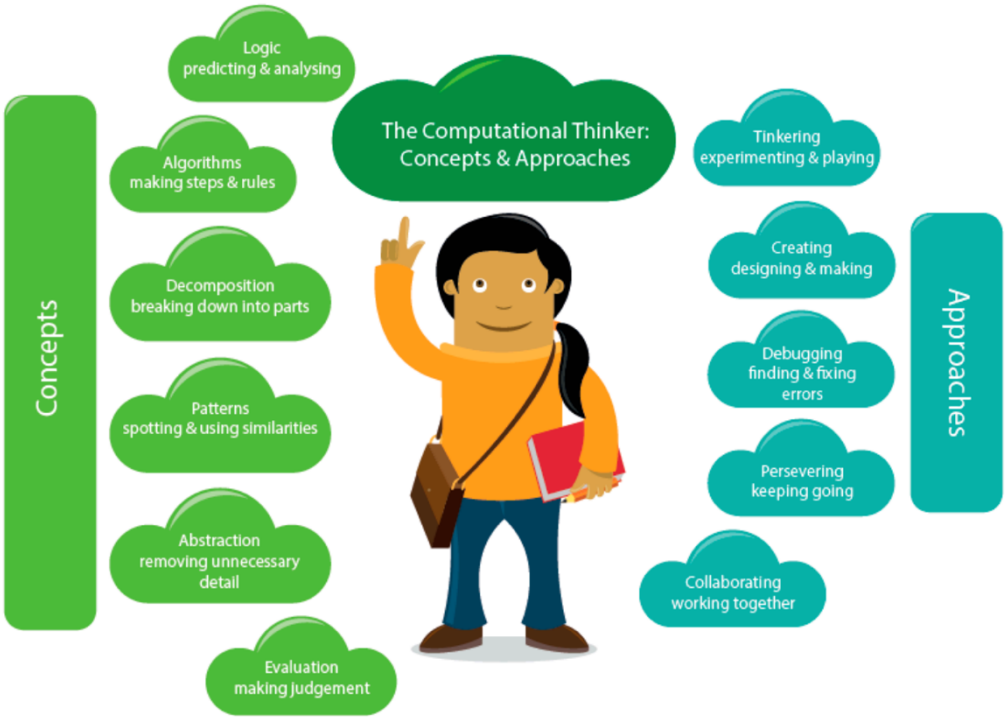


Figure 2.7: *Computational Concepts and Approaches (Csizmadia et al. 2015, p.8)*

2.8.2.2 Computational Thinking in the Australian Curriculum

Australia introduced a primary school Digital Technologies Curriculum in 2014. The Australian curriculum uses a very similar definition of computational thinking as the English curriculum, highlighting the need to embrace the potential of computation by applying techniques from computer science, "computational thinking is a process of recognising aspects of computation in the world and being able to think logically, algorithmically, recursively and abstractly" (ACARA 2016, section 4). Like it's English counterpart, the Australian curriculum identifies several key concepts of computational thinking. Although the terms given to these concepts are slightly different (abstraction, data collection, representation and interpretation, specification, algorithms and implementation), there are similarities, the overlapping concepts being abstraction and algorithms (ACARA 2023). Specification in the Australian curriculum draws on logic, which is one of the concepts of the English curriculum.

Data interpretation in the Australian curriculum involves the identification of patterns in data, so it aligns somewhat with the patterns concept in the English curriculum. Implementation of solutions in the Australian curriculum includes the evaluation of these solutions, and so has some similarity with evaluation in the English curriculum. However, unlike the English curriculum, decomposition is not included as a key concept on the Australian curriculum. The Australian curriculum is designed to provide students with opportunities to develop increasingly complex computational thinking skills (ACARA 2016). It is also expected that these skills will be developed through the use of a variety of programming activities (ACARA 2016):

- In Foundation to Year 2 (5-8 year olds) students should have opportunities to programme robotic toys using step-by-step instructions.
- In Years 3 and 4 (8-10 year olds) students should use a visual programming language to create an interactive application.
- In Years 5 and 6 (10-12 year olds) students should demonstrate an understanding of repetition, branching and interactivity by creating programs using a visual programming language.

2.8.2.3 Computational Thinking in the US Curriculum

The K-12 Computer Science Framework was developed to inform curriculum design and the development of curriculum standards for computing in primary education in the US. The K-12 Computer Science Framework identified computational thinking as one of four key elements fundamental to computer science education. In 2011, ISTE and CSTA collaborated with key educational stakeholders to develop an

operational definition of computational thinking which would be suitable for K-12 education. This definition was referred to in a previous section (2.4.4.4.1.3) but is included here again for ease of comparison. As can be seen in Figure 2.8, the definition of CSTA and ISTE (2011) includes six characteristics and five dispositions that are considered essential to computational thinking. There is some overlap between the characteristics identified by CSTA and ISTE (2011) and the concepts included in the aforementioned curricula (logic, algorithms, abstraction and analysing solutions). Two additional computational concepts (generalising and formulating problems so they can be solved by a computer) are also included. The K-12 Computer Science Framework (K-12 Computer Science Framework Steering Committee 2016) identifies 'algorithms and programming' as one of five areas "of disciplinary importance" in the field of computer science (K-12 Computer Science Framework Steering Committee 2016, p.59).

Computational thinking (CT) is a problem-solving process that includes (but is not limited to) the following characteristics:

- Formulating problems in a way that enables us to use a computer and other tools to help solve them.
- Logically organizing and analyzing data
- Representing data through abstractions such as models and simulations
- Automating solutions through algorithmic thinking (a series of ordered steps)
- Identifying, analyzing, and implementing possible solutions with the goal of achieving the most efficient and effective combination of steps and resources
- Generalizing and transferring this problem solving process to a wide variety of problems

These skills are supported and enhanced by a number of dispositions or attitudes that are essential dimensions of CT. These dispositions or attitudes include:

- Confidence in dealing with complexity
- Persistence in working with difficult problems
- Tolerance for ambiguity
- The ability to deal with open ended problems
- The ability to communicate and work with others to achieve a common goal or solution.

***Figure 2.8**: Definition of Computational Thinking (CSTA and ISTA 2011, p.13)*

With this in mind, students are expected to demonstrate an increasingly sophisticated understanding of programming across the K-12 years (K-12 Computer Science Framework Steering Committee 2016):

- In Grade 2 (7-8 year olds) students should have the opportunity to collaboratively develop programs and debug simple programs.
- In Grade 5 (10-11 year olds) students should demonstrate an understanding of variables and use decomposition in program development.
- In Grade 8 (13-14 year olds) students should demonstrate deeper understanding of variables, including the use of identifiers and memory location and they should create programs that provide meaningful solutions to problems.

However, the K–12 Computer Science Framework proposes that "computational thinking is fundamentally a human ability" (K-12 Computer Science Framework Steering Committee 2016, p.69). This nuance de-emphasises the role of the computer, thus supporting the use of unplugged approaches to developing computational thinking. The dispositions, considered an essential feature of computational thinking in the CSTA and ISTE (2011) definition, are less prominent in the other two curricula. While the English definition includes both persevering and collaborating, the Australian curriculum doesn't make any reference to a dispositional dimension of computational thinking.

2.8.3 Computational Thinking in the Context of this Study

These three cases illustrate that no common consensus has been reached on a definition of computational thinking. The definitions adopted often depend on contextual factors, with different groups defining computational thinking to suit their own agendas and priorities (Curzon *et al.* 2019). This research was conducted at the same time that the NCCA was considering how coding and computational thinking can be developed in primary school. This study sought to explore the development of computational thinking through programming in Irish primary schools; therefore, a definition suited to this context was necessary for this study. In their development of a definition of computational thinking for the Irish education context, Millwood *et al.* (2018) started by exploring "what is it that we expect of learners as an outcome" (p.6). They subsequently defined competence in terms of knowledge (cognitive), craft (kinaesthetic) and character (affective). These three competence domains are captured succinctly in Brennan and Resnick's (2012) framework for studying and assessing the development of computational thinking. Computational concepts represent the cognitive domain. These are "the concepts designers engage with as they program" (Brennan and Resnick 2012, p.1). Computational practices represent the kinaesthetic domain. These are the design practices learners engage in during the creation of their projects (Brennan and Resnick 2012). Finally, computational perspectives represent the affective domain. These are the "evolving understandings of themselves, their relationships to others, and the technological world around them" (Brennan and Resnick 2012, p.10).

2.8.3.1 Computational Concepts: Brennan and Resnick (2012) Framework

Brennan and Resnick (2012) identified seven key concepts that learners employ as they programme with Scratch: sequences, loops, conditionals, operators, data, events and parallelism. In the following section, each concept is explained in greater detail and a concrete example from a Scratch project is provided.

Sequences are a set of programming instructions to be executed by a computer to produce a particular behaviour or action. As computers can only execute these instructions one at a time, the order of these instructions can be important. For example, to get the pen sprite to draw a square, the 'pen down' block must precede the instructions to draw the square, as shown in Figure 2.9.

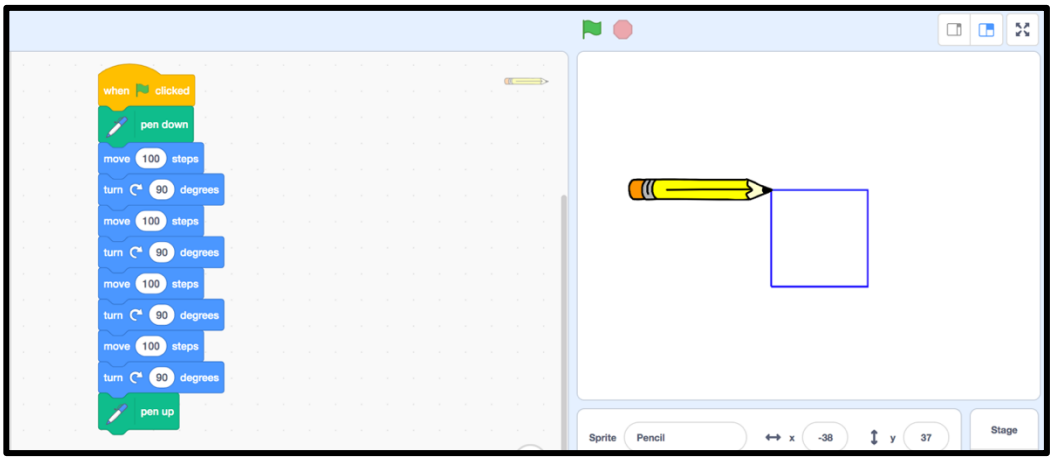


Figure 2.9: *Sequence of Instructions*

Loops allow for the repetition of a sequence of commands multiple times. They allow us to design more concise sequences of instructions. In the previous example, the pen sprite needed to move 50 steps and turn 90 degrees four times to draw the four sides of the square. Rather than moving and turning using eight consecutive blocks, we can use three blocks: 'move 100 steps', 'turn 90 degrees' enclosed by the 'repeat 4 times' block (Figure 2.10). Often times, sequences and loops are examined together under the broader concept of flow control (Zhang and Nouri 2019). Flow

control refers to the use of sequencing and iteration blocks to control the actions of the characters (Ch'ng *et al*. 2019).

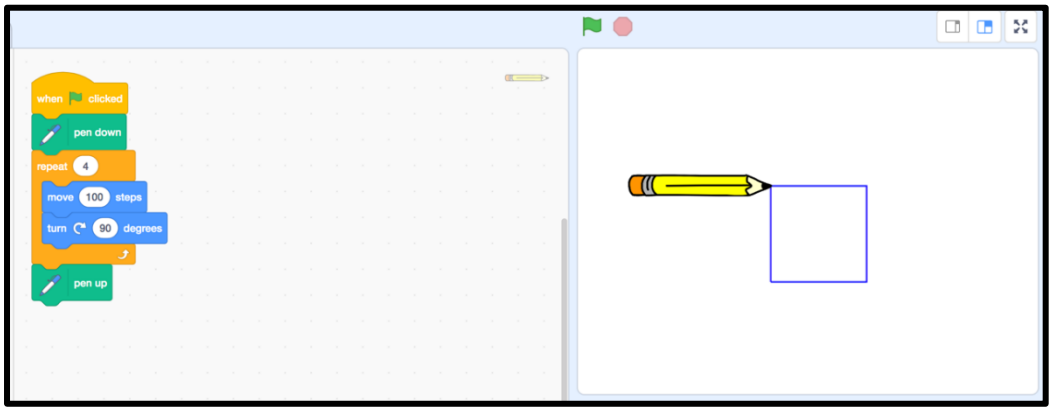


Figure 2.10: *A loop inducing multiple executions of the enclosed commands*

Conditionals allow us to create dynamic programs which perform different actions depending on certain conditions. In stories, which tend to have a linear structure, this is not important, but conditionals are integral to games in which there is no fixed trajectory. "In Scratch, any block whose label says "if," "when," or "until" is a sort of conditional construct" (Malan n.d.). In Figure 2.11, every time the fairy touches a crystal the magic spell sound plays, and the score increases by one.

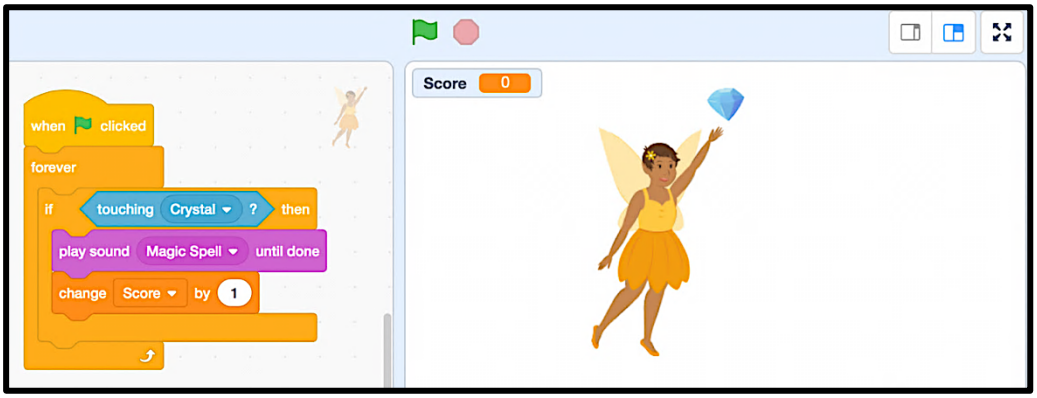


Figure 2.11: *Conditional within a catching game*

Operators allow for the use of mathematical operations (e.g. addition, subtraction etc.), Boolean logic (*and*, *or*, and *not*) and string operations (e.g. concatenation) in programs. In some programs it may be necessary to evaluate more than one

condition at a time. Operators allow programmers to do this. Figure 2.12 shows two types of situations where operators are commonly used in game programming.

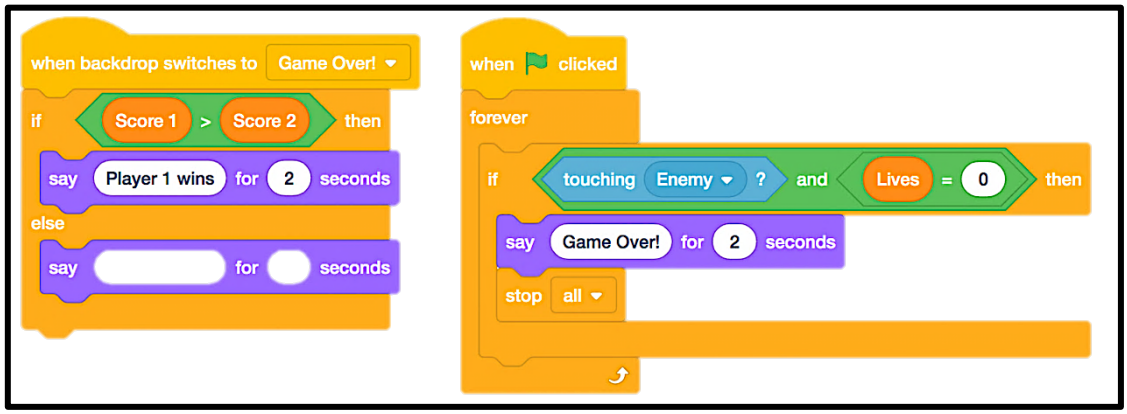


Figure 2.12: *Operators commonly used in game programming*

In studies using Dr Scratch to analyse programs, conditionals and operators are categorised together as logic or logical thinking (Zhang and Nouri 2019). Within this categorisation, logic is defined as the use of instructions to create programs that behave differently depending on the situation (Ch'ng *et al*. 2019).

Data supports the storing, retrieval and updating of values we might need throughout a program, such as level, score or lives. Scratch supports the use of two types of data: variables and lists. Variables can store a single number or string (sequence of typed characters). Lists can store multiple numbers and strings. In Figure 2.13, a variable has been created to store the number of bears and the refrain of the counting rhyme 'Five Little Bears'. According to Zhang and Nouri (2019) these four computational concepts, sequences, loops, conditionals and variables, are "the basic structures and components in any programming language" (p.9) and therefore are the most commonly assessed in research studies.

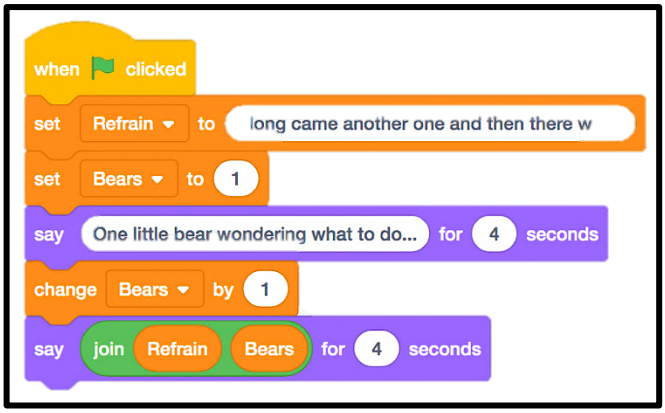


Figure 2.13: *Using variables to store numbers and strings*

Events are actions (key presses, mouse clicks) or occurrences (messages from other scripts) recognised by the program, that trigger the scripts below it to run. Studies using Dr Scratch categorise user actions under 'user interactivity', described by Ch'ng *et al*. (2019) as player actions that elicit "new situations in the project through the use of input peripherals such as keyboard, mouse or webcam" (p.252). In Figure 2.14, you can see two examples of events that trigger an action.

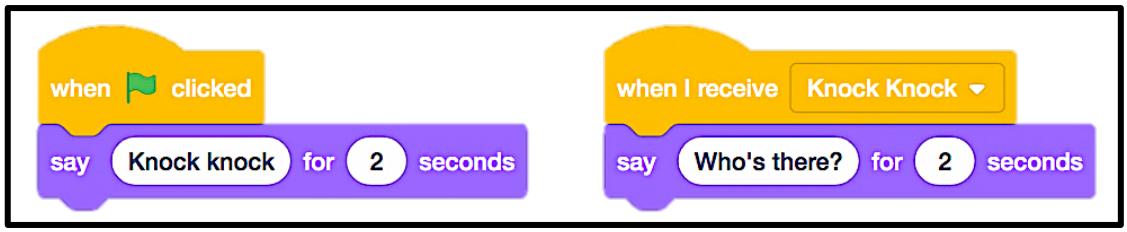


Figure 2.14: *Actions triggered by events*

Parallelism allows us to create programs in which several things are happening simultaneously. In Scratch, this can mean several scripts running at the same time within one sprite or scripts running simultaneously for different sprites. In Figure 2.15, the horse has been programmed to continuously play the 'scrambling feet' sound and to repeatedly change to 'next costume' (to mimic the running movement) in response to the green flag being clicked.

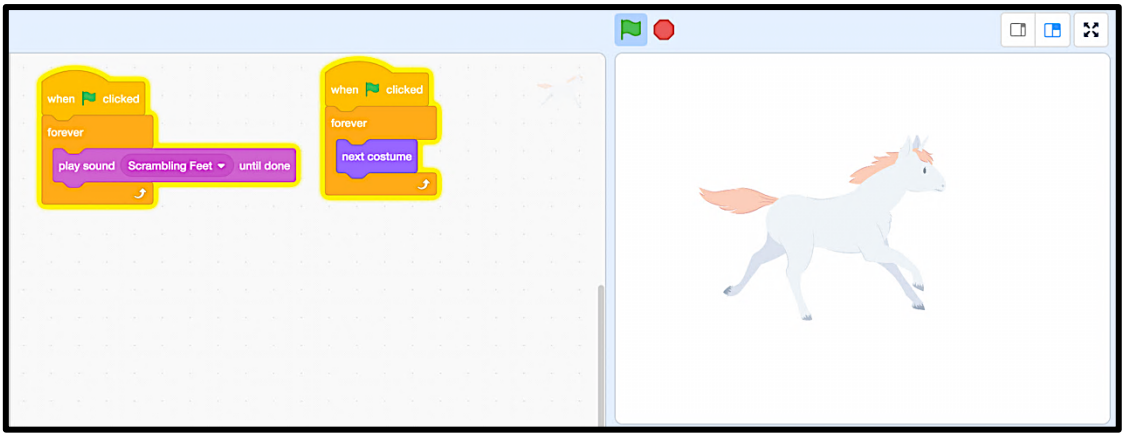


Figure 2.15: *Parallelism within one sprite*

Kong (2019) completed a review of the literature to identify the elements of computational thinking that had been evaluated by researchers between 2010 and 2019. Along with the seven concepts identified by Brennan and Resnick (2012) they included two further computational concepts: procedures and initialisation. These two further concepts are explained in the following section, and a concrete example from Scratch is provided.

Procedures are small snippets of code that perform particular actions. If there is a sequence of programming instructions that we want to include in different parts of our program, we can store these as a procedure. This avoids duplication of commands and allows us to create more efficient programs (Kong 2019). In Scratch, the 'My Blocks' feature supports the creation of user-defined procedures (Figure 2.16).

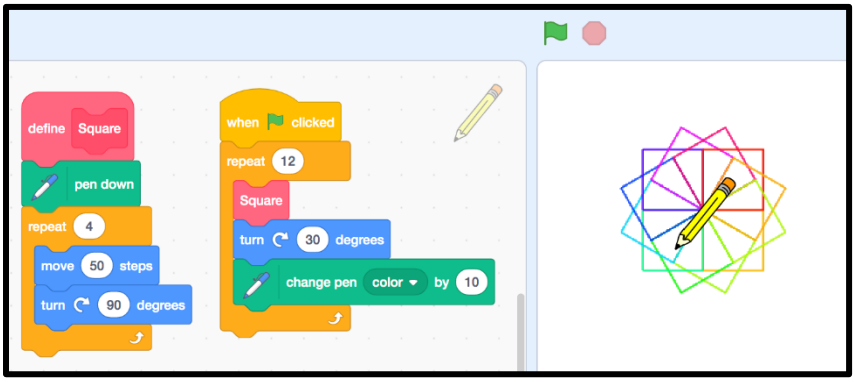


Figure 2.16: *The My Block function allows us to create more efficient programs*

In studies utilising Dr. Scratch, procedures are categorised as abstractions since they support the creation of programs that are easier to read, program and debug (Ch'ng *et al*. (2019).

Initialisation is the assigning of an initial value to variables or objects in our programs. Franklin *et al*. (2017) highlight the importance of initialisation, asserting that all variables should be initialised prior to their use. Initialisation is not exclusively related to variables we create ourselves e.g. setting the score to 0. Meerbaum-Salant *et al*. (2013) identified initialisation as distinct from variables, recognising that several attributes of sprites are variable (e.g. their position, their colour and their size) and these also need to be assigned an initial value if they are going to be modified during the program. If we create a chase game where we use the left and right arrows to move a sprite, we should include a script to return the sprite to its starting position on reset as shown in Figure 2.17.

Figure 2.17: *When the green flag is clicked the sprite returns to the centre of the screen (0,0)*

Like Kong (2019), Zhang and Nouri (2019) undertook a systematic review to identify and define the computational thinking skills that have emerged in studies of computational thinking education. Zhang and Nouri (2019) identified that several studies had examined coordination and synchronisation, which had not been included in Brennan and Resnick's (2012) conceptualisation of computational thinking. Synchronisation is the use of instructions to ensure the different scripts in the program happen in the desired order (Ch'ng *et al*. 2019). Zhang and Nouri (2019) characterised coordination and synchronisation as a computational practice,

however, many of the researchers who have studied it consider it to be a computational concept (Burke 2012; Fields *et al*. 2015; Moreno-León *et al*. 2015; Park and Shin 2019). In Scratch, the following blocks are all modes of synchronisation: *wait*, *broadcast*, *when I receive*, *wait until*, *when backdrop changes* (Zhang and Nouri 2019). Figure 2.18 illustrates a program script that incorporates synchronisation.

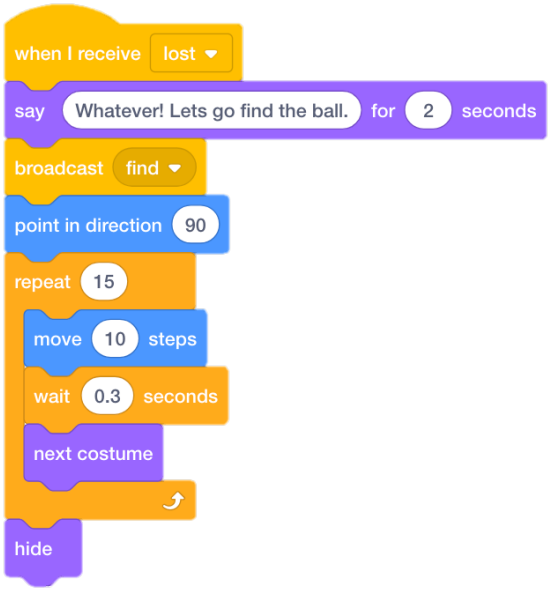


***Figure 2.18**: A script using broadcast to activate scripts with the matching when I receive block*

2.8.3.2 Computational Practices: Brennan and Resnick (2012) Framework

Brennan and Resnick (2012) identified four key problem-solving practices that programmers engage in while creating their programs: experimenting and iterating, testing and debugging, reusing and remixing, and abstracting and modularizing. The descriptions of each of these practices are summarised in the following section.

Experimenting and iterating are described by Brennan and Resnick (2012) as "iterative cycles of imagining and building" (p.7). This involves creating a little bit of code, running it, and then developing it further based on observations and new

ideas. Testing and debugging involves making sure the program runs as intended, and developing strategies for solving problems as they arise. Examples of debugging strategies include reading through your scripts or experimenting with scripts. Reusing and Remixing describe the practice of creating a project by building on the work of others. Dasgupta *et al*. (2016) define remixing as "the reworking and combination of existing creative artifacts" (p.1438). Reusing and remixing practices can range from a basic level, such as incorporating sprites, scripts, costumes and sounds from an existing project, to more advanced remixing, such as making decisions about how to modify existing segments of code or where to add procedures within a program (Kafai and Burke 2016). Abstracting and modularising is the practice of building something large by creating and combining smaller parts (Brennan and Resnick 2012; Martin *et al*. 2014). According to Brennan and Resnick, (2012) abstracting and modularizing are employed at multiple levels "from the initial work of conceptualizing the problem to translating the concept into individual sprites and stacks of code" (p.9). This practice of separating the functionality of a program into independent scripts, each responsible for executing a single behaviour or action, makes the code easier to read, debug and maintain.

Kong (2019) argued for the inclusion of two further computational practices, algorithmic thinking and problem decomposition. The first of these, algorithmic thinking, has been defined as the thought process involved in breaking down a complicated problem into a series of steps that lead to the desired outcome (Doleck *et al*. 2017; Lockwood *et al*. 2016). However, the inclusion of algorithmic thinking as an element of computational thinking is questionable. Lockwood *et al*. (2016) acknowledge the similarities between both definitions, while Doleck *et al*. (2017) note that these terms have been used interchangeably in the past. According to Kong

(2019) problem decomposition refers to "breaking down problems into smaller, more manageable tasks" (p.130). However, the similarities between Kong's definition of decomposition and Brennan and Resnick's (2012) definition of *being incremental and iterative* are notable. Brennan and Resnick (2012) describe *being incremental and iterative* as "approaching a solution in small steps" (p.7).

2.8.3.3 Computational Perspectives: Brennan and Resnick (2012) Framework

The final dimension included in Brennan and Resnick's (2012) framework is computational perspectives: expressing, connecting and questioning. Kong (2019) defined computational perspectives as the interpersonal and intrapersonal domains of computational thinking. Falloon (2015) suggests that they are the "views formed when and from building programmes such as self-expression, collaborating and connecting with others, and questioning" (p.883). Each of the three key dimensions of computational perspectives identified by Brennan and Resnick (2012) is discussed further in the following section. Expressing refers to the view that technology is something that we can use to create rather than just something we can consume. It is widely recognised that most of our interactions with technology are as passive consumers rather than active producers. However, Brennan and Resnick (2012) believe that computational thinkers view technology as a medium for design and self-expression, and they view coding as a way for them to express themselves. Connecting refers to a recognition that creativity can be enhanced by collaborating and engaging with others. Brennan and Resnick (2012) view computational thinkers as people who appreciate both the value of creating for others and the benefit of creating with others. They identified four distinct dimensions of this perspective:

- entertaining others (such as building up an audience of followers for a series of soap opera-esque projects),

- engaging others (such as designing a survey for other community members to respond to),
- equipping others (such as developing assets for other Scratchers to use in their own projects), or
- educating others (such as making tutorial projects that help other Scratchers learn something about Scratch, like how to use trigonometry in physics simulations or how to make popular projects).

Brennan and Resnick (2012, p.11)

The final computational perspective identified by Brennan and Resnick (2012) was questioning. Bandura (2001) suggested that

> everyday life is increasingly regulated by complex technologies that most people neither understand nor believe they can do much to influence.
>
> (p.17)

The computational perspective of questioning challenges Bandura's view on our relationship with technology. Instead, computational thinkers demonstrate a willingness to ask questions, critique technologies and consider alternative designs (Brennan and Resnick 2012).

Kong (2019) suggested adding computational identity and programming empowerment to the three computational thinking perspectives characterised by Brenan and Resnick (2012). The concept of computational identity evolved from the field of STEM, where research showed the importance of developing learners' scientific identities to nurture interest in STEM (Patton *et al*. 2019). In a similar way, computational identity acknowledges the need for young learners to feel they belong to the computing community. Kong (2019) proposes that there are four distinct components of computational identity: depth of engagement in programming activities, sense of affiliation, career orientation and self-actualisation ("whether learners feel competent after learning programming" (p.135)). According to Kong (2019) programming empowerment refers to peoples' experiences designing and

creating programmes that enable them to solve real-life problems and become confident participants in the digital world. Digital empowerment may be considered more of a long term goal, as Patton *et al.* (2019) suggest that digital empowerment

> allows students to shift their sense of themselves from individuals who "know how to code" to members of a community empowered to have a real impact in their lives and those of others.
>
> (p.41)

This implies that students would first need to learn how to code before being encouraged to consider how their coding can be impactful in their everyday lives and communities.

Kong and Lai (2022) proposed that affiliation was a key component of computational identity, a computational dimension. However, the affiliation component identified by Kong and Lai (2022) shares similarities with Brennan and Resnick's (2012) connecting component. Both constructs focus on children's developing perspectives on collaboration. Kong and Lai's definition extends the idea of creating with others by focussing on students' feeling of belonging with respect to their interest in programming. Goodenow (1993) defined sense of belonging in a classroom context as

> students' sense of being accepted, valued, included, and encouraged by others (teachers and peers) in the academic classroom setting and of feeling oneself to be an important part of the life and activity of the class.
>
> (p.25)

Good *et al*. (2012) developed and validated a 'sense of belonging' scale for Maths that included five factors, membership, acceptance, affect, desire to fade and trust (Good *et al*. 2012). This scale has subsequently been used by Mooney and Becker (2020) to examine sense of belonging in a computer science setting. They reported that learning experiences and the learning environment can impact students' sense of belonging, suggesting that educators can play a central role in fostering

belongingness. Nurturing students' feeling of belonging in educational contexts is important. Rainey *et al*. (2018) reported that students' sense of belonging can be a necessary condition for success in STEM disciplines. While Rich (2020) found that creating a feeling of belonging motivated students to persist in the face of difficulty.

Section 3: Developing Computational Thinking in Practice

2.9 Approaches to Developing Computational Thinking

Introducing computational thinking to the primary school curriculum, as with the introduction of any new content, brings with it both challenges and possibilities. The previous sections explored the rise to prominence of computational thinking in primary school curricula across the globe and how it has been conceptualised in different contexts. This section explores the approaches to the development of computational thinking that have been adopted in schools. Teachers are required to equip themselves with both subject knowledge and pedagogical knowledge. A review of computational thinking initiatives and studies reveals that educators and researchers have adopted two main approaches to teaching computational thinking in schools: plugged activities (computer programming) and unplugged activities (those that do not require the use of technology).

2.9.1 Developing Computational Thinking through Unplugged Activities

While the concept of computational thinking emerged from computer science, researchers propose that students can exhibit computational thinking without using technology tools (Allan *et al*. 2010; Caeli and Yadav 2019; Chen *et al*. 2023; Lye and Koh 2014). According to Kafai and Burke (2013):

> Computational thinking — while often strictly associated with computer science — actually is better understood as extending computer science principles to other disciplines in order to help break down the elements of any problem, determine their relationship to each other and the greater whole, and then devise algorithms to arrive at an automated solution.
>
> (p.62)

The use of unplugged activities to develop high order thinking skills predates Jeanette Wing's (2006) seminal paper on computational thinking. Seymour Papert, best known for his work using the programming language Logo, used unplugged activities to help students develop their computer programming skills. He encouraged children to make connections between the way the Logo turtle moved on-screen and the way they moved their bodies. In his book 'Mindstorms', he explained how they could physically act out the problems they were solving on the computer (Papert 1980). Abelson and diSessa (1986) believed that the turtle, as an animal, supported children's understanding of motion in a plane.

> Expressing motions in terms of FORWARDs and RIGHTs is a much more direct way of dealing with an animal's behavior than, say, describing movements in response to stimuli as changes in x and y coordinates.
>
> (p.55)

The first use of the term 'unplugged' to describe this approach was the Computer Science Unplugged project initiated by Mike Fellows, and Tim Bell in the 1990's. The term 'unplugged' was borrowed from a musical style of that era. 'Unplugged' in the musical context meant "avoiding having music cluttered by technology, returning to the essence of the music" (Bell *et al*. 2012, p.416). This reflected what Mike and Tim were trying to do with the Computer Science Unplugged project. They wanted to strip back the technology to eliminate distractions and focus on the essence of the topic. Along with Ian Witten, they developed a series of activities, that could be used to introduce fundamental concepts of computer science in classrooms and outreach settings without the use of computers (Bell *et al*. 1998). These activities involved

using your body or simple props (such as playing cards, marbles or string) to illustrate important computational thinking concepts. The CS unplugged project gained momentum in 2003 due to two changes in computer science circumstances; it became apparent that interest in studying computer science was in decline, and the Association for Computing Machinery (ACM) released a model curriculum for K-12 computer science, which referenced several of the project's activities (Bell *et al*. 2012). The unplugged activities were viewed as an accessible and engaging way to introduce children to the fundamentals of computer science. Having gained visibility through its prominence in the ACM K-12 curriculum, it was also promoted in the US by the Computer Science for High Schools (CS4HS) program, CSTA and the National Center for Women and Information Technology (NCWIT). An initiative at Korea University aimed to introduce computer science to the South Korean school curriculum, led to the first translation of the Computer Science Unplugged material. Sponsorship by Google in 2007 led to an enhanced web presence, which helped to increase international uptake. In 2011, the Computer Science Unplugged website (csunplugged.org) was accessed by people from 140 different countries (Bell *et al.* 2012). Although initially intended for outreach settings, unplugged activities are now present in many formal curricula. The Computer Science Inside project, initiated in 2006, developed a collection of unplugged activities designed specifically for the Scottish curriculum (Cutts *et al.* 2007). In 2011, Thinkersmith created a set of unplugged activities designed specifically for K-8 students (from age 5 up to age 14). This has since been expanded by code.org and now includes over 55 lessons (Angeli *et al*. 2016). In the UK, the Barefoot Computing project is an initiative that has developed a collection of resources designed to teach computer science components of the new primary computing programme. Most of the resources in the Barefoot

Computing collection are unplugged activities to make them as accessible as possible (The Royal Society 2017).

2.9.1.1 Unplugged Activities and Teachers

Although many unplugged initiatives predate the rise to prominence of computational thinking, it has become apparent that the unplugged approach emphasises computational thinking (Bell *et al*. 2012). In recent years, several studies have examined the use of unplugged activities to introduce teachers to computational thinking (Blum and Cortina 2007; Cho *et al*. 2014; Curzon *et al*. 2014; Morreale *et al*. 2012). These studies all reported positive results, stating that the teachers responded enthusiastically and felt that the workshops had provided them with useful ideas for the classroom. However, these results should be interpreted with caution, as in most cases, the lasting impact of these workshops on the teachers' practice is unknown. In their study, Cho et al (2014) found that initial enthusiasm shown by the teachers directly after the workshops quickly waned. Indeed, Bort and Brylow (2013) suggest that while teacher enthusiasm is often high directly after professional development workshops, very few teachers are able to transform the content of these workshops into viable classroom practice.

2.9.1.2 Unplugged Activities and Students

Developing Computational Concepts through Unplugged Activities

In recent years, a small number of studies have reported on the effect of unplugged activities on the development of computational thinking in the classroom. These have mostly focused on the computational concepts dimension of computational thinking. Folk *et al.* (2015) report on a project intended to introduce middle school

and high school students to computational thinking. K-12 teachers and graduate fellows worked together to create a series of lessons to teach computational concepts by integrating them into their existing curricula. They reported an increase in the computational thinking knowledge of the students following their participation in the lessons. Li *et al*. (2016) present similar positive findings from their study. They designed three unplugged activities to develop their students' computational thinking, with a particular focus on decomposition. They report that their approach had a 'significant impact' on developing their students' computational thinking ability. However, there is little detail provided, making it difficult to ascertain how the authors reached this conclusion. It is also worth noting that this study looked at high school students who were enrolled on an information technology course. This suggests that these students already had an interest in information technology so it is difficult to make inferences about the wider population (such as those you would find in a primary school classroom). Saxena *et al.* (2020) report on a study conducted with 11 pre-schoolers in Hong Kong. They designed and implemented a voluntary one-week (five two-hour lessons) summer computational thinking enrichment course. The course involved both unplugged and plugged activities. The teachers involved in implementing felt the unplugged activities provided concrete experiences to introduce students to computational thinking, which they were then able to apply to the plugged activity. However, it was noted that while students demonstrated mastery of sequencing and pattern recognition, not all students were successful in mastering algorithmic design (Saxena *et al.* 2020). Again, it is important to note that it was a voluntary summer course with few participants conducted over a very short time frame. There is a dearth of research on the use of unplugged activities to develop the computational thinking of primary school students. Four studies that

have explored this approach are Brackmann *et al*. (2017), Dag *et al*. (2023), Dwyer *et al*. (2013) and Faber *et al*. (2017). Jiang and Wong (2019) also conducted a study with primary school students, but they explored both plugged and unplugged approaches. Brackmann *et al*. (2017) report on a quasi-experiment conducted in two primary schools in Spain. In each of the schools, the students were divided into an experimental group who participated in the unplugged lessons and a control group who did not. They assessed the computational thinking skills of both groups of students with a pre-test and a post-test. They found that the computational thinking skills of the students who had participated in the unplugged lessons improved significantly, whereas the skills of those in the control group did not. Dwyer *et al*. (2013) explored the integration of computational thinking in a fourth grade (9-10 year olds) physics lesson. They found that the students struggled to develop specific instructions for a drawing task. They failed to include important information such as positioning, direction and size. This suggests that these students were struggling with the computational concept of data representation. The research of Dag *et al*. (2023) was based on a quasi-experimental study conducted with third and fourth grade students in a Turkish primary school. The students participated in a fourteen-week unplugged coding course. Their computational thinking skills were assessed before, immediately after and ten weeks after that again. The instrument assessed five dimensions of computational thinking: algorithmic design, abstraction, evaluation, decomposition and generalization. The results show a statistically significant increase in computational thinking skills across all dimensions from the pre-test to the post-test but no statistically significant difference between the post-test and the follow-up test ten weeks later. This suggests that the improvements in students' computational thinking skills were lasting, at least for this period of time. Faber *et al*.

(2017) conducted an exploratory study in twenty-six schools in the Netherlands. They designed a series of six ninety-minute unplugged lessons, which were subsequently taught to students in their final year of primary school. They reported that the lessons elicited a positive reaction from both the teachers and students and suggested that the unplugged approach offered a viable alternative to programming. The authors of this study did not refer to the development of a particular dimension of computational thinking. Jiang and Wong (2019) report on a six-week course of programming and unplugged activities conducted in five Hong Kong primary schools. The participants were 400 fourth graders between the ages of nine and eleven. They found that the students were positive about both approaches to developing computational thinking. They used a survey to determine their intrinsic motivation in both activities. Four dimensions of intrinsic motivation were examined, interest/enjoyment, value/usefulness, perceived competence and relatedness. While the mean scores for all dimensions were higher for programming than the unplugged activities, the only significantly significant difference between the approaches related to perceived competence. No research study that specifically examined the development of computational practices through unplugged activities was identified in a review of the literature.

Developing Computational Perspectives through Unplugged Activities

The main aim of most early unplugged initiatives was to promote interest in Computer Science (Brackmann *et al*. 2017). Consequently, much of the early research into unplugged activities explored the use of unplugged activities to foster an interest in computer science. Developing an interest in computing aligns most closely with the computational perspectives dimension of computational thinking. In

the US, several research studies have examined the ways unplugged activities impact students. Much of the research is descriptive rather than rigorous, and the results of these research studies varied greatly. Taub *et al*. (2012) examined the effect of unplugged activities on middle-school students' ideas about Computer Science. They found that although they enjoyed the unplugged activities, their attitude towards Computer Science didn't change. They felt that perhaps the students didn't see the connection between the unplugged activities and computer science. Feaster *et al*. (2011) also investigated the impact of a program of CS Unplugged activities on high school students' attitudes to computer science. The researchers reported no significant improvement in their students' attitudes towards or understanding of computer science. Mano *et al*. (2010) introduced an outreach program in a middle school seeking to improve students' attitudes towards Computer Science. Their program included both plugged and unplugged activities. However, in contrast to Feaster *et al*. (2011), they reported an improvement in their student's interest in Computer Science. Lambert and Guiffre (2009) delivered three computer science sessions (only one was an unplugged session) in three fourth grade classes in an elementary school. Their results also indicated an improvement in their students' confidence and interest in Computer Science. Despite the differing results of this research, and the anecdotal nature of much of it, it shows that unplugged activities have been adopted extensively in schools in the US in the last ten years. Research on the use of unplugged activities in classrooms in the UK or Ireland is less plentiful. However, a survey of 300 computing teachers in the UK found that a significant proportion of teachers identified unplugged activities as one of the most used and successful strategies for teaching computing in the curriculum (Sentance and Csizmadia 2016).

2.9.1.3 Perceived Benefits of the Unplugged Approach

Proponents of the unplugged approach have suggested several benefits of using unplugged activities rather than plugged activities. Firstly, they argue that in a time when we are trying to reduce the amount of screen time for children, unplugged activities offer a valuable alternative to the use of digital devices (García-Peñalvo *et al.* 2016). They cite several studies, which have shown that children who have higher levels of screen time and lower levels of physical activity are more likely to have psychological difficulties. A second reason why the unplugged approach has gained support is because it is easy to implement and accessible to all. Whereas the plugged approach requires technology, which not all schools can afford, and time must be spent becoming familiar with the programming language. Programming

> can require a significant commitment from the start in time and/or resources, whereas it is possible to have a significant impact with 'Unplugged' on a single one-hour visit to a school.
>
> (Bell *et al*. 2009, p.21)

Others suggest that programming is an unnecessary distraction from the real learning. Wing (2008) warns that we do not want the device to get in the way of the students grasping the concepts. As Caldwell and Smith (2017) observed:

> It is much more difficult to learn a new skill such as computational thinking while dealing with unfamiliar and recalcitrant technology. We have to expend mental effort to practise the new skill while simultaneously trying to operate a device.
>
> (p.4)

Bell *et al*. (2008) suggest that students see the computer as a toy, which can sometimes stop them from focusing on the concepts they are learning. Similarly, Gardeli and Vosinakis (2017) suggest that when students programme, they often use trial-and-error rather than thinking about what they are doing. Finally, some researchers argue that if we teach computational thinking through programming, students could get a narrow view of computational thinking. Wing (2008) cautioned

that we want to avoid a situation where students think they understand the concepts because they are proficient at using the device. She used the analogy of being able to use a calculator does not equate with understanding arithmetic. It is clear that unplugged activities can be an effective approach to introducing students to computational thinking. However, Shelton (2016) cautions that how these activities are implemented will determine their effectiveness. He refers to an example of an unplugged activity outlined by Papert in his book 'Mindstorms' as an example of good practice. In this unplugged activity children learned how to program the Logo turtle by 'playing Turtle', moving their body as the on-screen turtle must move (Papert 1980). In this example, the unplugged activity is not an alternative to the plugged approach; rather, it makes the learning accessible to the children by making the abstract concrete. Shelton (2016) contrasts this to the practice of using unplugged activities to introduce computational concepts without presenting a context, thus further suggesting that often the logical context in which to explore these concepts is through children creating programs (Shelton 2016).

2.9.2 Developing Computational Thinking through Plugged Activities

Alongside the emergence of computational thinking as a key competence for the twenty-first century is the growing trend of introducing programming to school curricula. Computer programming is considered a valuable means of developing higher-order thinking and problem solving skills (Fessakis 2013) and consequently many education systems are integrating programming into their curricula. Although it is widely recognised that programming is just one of the approaches which provides students with the opportunity to develop computational thinking, many educators and researchers suggest it is the best approach (Brennan and Resnick 2012;

Grover and Pea 2013; Shelton 2016; Webb *et al*. 2016). Webb *et al*. (2016) suggest that

> representing a solution to a problem as a program provides a perfect way to evaluate the solution, as the computer will execute the instructions to the letter, forcing the student to refine their solution so that it is very precise.
>
> (p.5)

2.9.2.1 But what exactly is programming?

Repenning (1993) describes a program as "a sequence of coded instructions for insertion into a mechanism (as in a computer)" (p.9). This set of instructions for completing a task is called an algorithm. Algorithms are the foundation of computer science: they are how we get computers to solve problems (Bell *et al*. 1998). Computers operate by following the algorithms that have been written to carry out a particular task—whether it be word processing, accounting, playing a game, or controlling a robot. Programs are written in languages that have been specially designed to control the operation of computers. Hundreds of different languages have been invented, such as Ada, BASIC, C, C++, COBOL, FORTRAN, LOGO, Pascal, RPG and Scratch. Each language has a limited set of instructions that provide the vocabulary with which the programmer specifies what they want the computer to do (Bell *et al*. 1998). Irrespective of what language programmers use, they must become adept at specifying precisely what they want the computer to do. Getting the instructions wrong will lead to an error in the programme. Errors like this are commonly called 'bugs' (Bell *et al*. 1998). The creation of a meaningful sequence of instructions can be extremely complex depending on the programming language and the technologies. This is one of the reasons that computer programming can often be viewed as a narrow, technical, specialised activity (Resnick *et al*. 2009b). However, once a program is working, the instructions can be executed by the mechanism with

high precision at very high speeds, which is what makes computer programmes so efficient (Repenning 1993). The 'Draft Science, Technology and Engineering Education Specification: For primary and special schools' distinguishes between programming and coding, stating that "programming is broader than coding in that it involves designing and testing, as well as writing instructions" (NCCA 2024a).

2.9.2.2 Programming and the Development of Thinking Skills

Several researchers suggest that efforts to develop computational thinking through programming are not new, and that we should learn from these previous attempts (Grover and Pea 2013; Lye and Koh 2014; Voogt *et al*. 2015). When computers first became available in schools, there was great enthusiasm for teaching children to program. The computer was viewed as an instructional tool that could be used to develop higher-order thinking skills. Grover and Pea (2013) cite attempts by Alan Perlis in the 1960s to introduce all college students to programming and the theory of computation, as the earliest example of efforts to incorporate computational thinking into education curricula. Perlis claimed that learning to program would provide students with an opportunity to develop a type of thinking which could be used to understand and solve any type of problem (Tedre and Denning 2016). In the 1980s Seymour Papert extended this vision to primary level education. His research on computers and education was based on the belief that "children can learn to use computers in a masterful way, and that learning to use computers can change the way they learn everything else" (Papert 1980, p.8). Papert, along with Feurzeig and Solomon, developed the programming language Logo with exactly that goal in mind (McDougall *et al*. 2014). Papert originally proposed that Logo could be used

effectively to teach concepts in mathematics but soon recognised its value in the development of higher order thinking skills (Tedre and Denning 2016).

> Computer presence could contribute to mental processes not only instrumentally but in more essential, conceptual ways, influencing how people think even when they are far removed from physical contact with a computer.
>
> (Papert 1980, p.4)

As claims about the effects of learning to program on thinking proliferated, researchers began to investigate these claims. Several of these researchers reported findings that were contrary (Pea and Kurland 1984; Simon 1987). Pea and Kurland (1984) are most often credited with revealing that Logo did not always benefit learning as it had been claimed. However, this study itself came in for some criticism, with researchers claiming the sample size was too small and variables were not adequately controlled (Guzdial 2004). In a later paper, the authors clarified that their study did not show Logo was ineffective; rather, that very few students actually learned to program (Kurland *et al*. 1986). Palumbo (1990) claims that developing higher order thinking through programming activities requires the development of two distinct types of knowledge, declarative knowledge and procedural knowledge. Firstly, the children needed to learn the command structure of the programming language they would use (declarative knowledge). Then, they needed to use this declarative knowledge in strategically different ways in order to solve problems (procedural knowledge). So, one of the important findings from these early studies is that most students struggled to gain the declarative knowledge they needed to access the procedural knowledge. Several other studies reported similar findings to Pea and Kurland's (1984) study that programming syntax was just too difficult for children to master (Guzdial 2015). However, despite the lack of conclusive evidence, many advocates of programming still argue that programming can enable the development of higher order thinking, given the right circumstances. They have put forward

several reasons for the inconsistent findings, claiming that much of the early research had been undertaken in educational environments much different from those advocated by Papert and others. A review of the literature revealed several key enabling factors that facilitate the development of higher order thinking skills through programming. These include:

- Appropriateness of the programming language (Fay and Mayer 1987; Mayer *et al*. 1986; Palumbo 1990).
- Pedagogy employed in the programming initiative (Au 1992; Oakley and McDougall 1997; Papert 1980).
- Relevance of the programming initiative (Oakley and McDougall 1997; Papert 1980; Resnick *et al*. 2009a).
- Appropriate student age (Johanson 1988; Palumbo 1990).
- Duration of programming initiative (Johanson 1988; Oakley and McDougall 1997; Palumbo 1990).
- Role of the teacher (Benton *et al*. 2017; Oakley and McDougall 1997; Resnick *et al*. 2009a).

Although some studies reported gains (Miller *et al*. 1988; Rieber 1987) without firm evidence supporting the link between programming and cognitive development, Logo had all but disappeared from schools by the end of the 1980s. With the decline of Logo, efforts to develop thinking skills through the use of computers were not widely reported (Lye and Koh 2014). Several reasons have been put forward for this period of inactivity. Gander *et al*. (2013) suggest that this lull was due to a lack of awareness of the potential of computing to develop powerful thinking skills and a narrow view of the computing subject. Kafai and Burke (2013) also cite a lack of understanding of the purpose of teaching programming. Why should students spend

hours creating programmes when ready-made multimedia packages are widely available on CD-ROMs? Özçinar (2018) believes the emergence of the personal computer caused a shift in focus away from programming towards computer literacy. Mayer (1988) claims that by the mid-1980s educators had come to accept that the difficulties children faced in learning even the basics of Logo would prevent positive learning outcomes. However, there has been a resurgence in efforts to develop computational thinking through programming in recent years. This resurgence has also been facilitated by the availability of numerous 'low floor' (easy to learn) programming languages that have been developed specifically for novice programmers, e.g., Scratch, Alice, Kodu, Toontalk and Stagecast Creator. Informal learning environments such as computer programming clubs and outreach settings are leading this revival (Kafai and Burke 2013). CoderDojo, which was launched in Cork in the National Software Centre in 2011, is one such example. Since then, CoderDojo has expanded rapidly and has built up a global network of community-based programming, volunteer-led clubs where young people "can learn to code, build a website, create an app or a game, and explore technology in an informal, creative, and social environment" (CoderDojo Foundation 2022, para. 1). Kafai and Burke (2013) believe that the success of these movements lies in the way these children are learning to programme. Rather than learning to programme through exercises designed to introduce them to the underlying concepts, they are learning to programme by creating applications like interactive stories and video games, which are relevant and meaningful to them. Internationally and nationally, many organisations have supported the decision of governments and policy makers to develop computing in schools. Global organisations such as CSTA, NCWIT and Code.org have created large repositories of resources, provided professional

development opportunities for teachers, and organised events to promote increased participation in computing. Global movements such as CoderDojo, cs4fn, CS Unplugged and Hour of Code have created resources for teachers with the aim of broadening participation in computer science. Global Initiatives such as TACCLE3 Coding and Google's CS First and Exploring Computational Thinking were created to provide teachers with the knowledge and materials they would need to integrate computational thinking into their teaching. Nationally, the Irish Computer Society (ICS) and LERO (the Irish Software Research Centre) have been involved in several initiatives to promote computing in education. Both organisations were involved in the development of the website scratch.ie. The website was created to support teachers, students and parents in Ireland, using the Scratch programming software. LERO work with the PDST to deliver Scratch training to teachers and founded the scratch programming competition in Ireland. The ICS now runs this programming competition open to students attending Irish primary and secondary schools. They also run the 'Bebras Challenge' in Ireland, an initiative whose goal is to promote computational thinking and informatics to teachers and students. They developed a digital skills programme, CLISTE (Computer Literacy and Internet Skills Through Education), which was created specifically for use in Irish classrooms.

2.9.2.3 An International Perspective on Programming in the Curriculum

Since 2012, the drive for coding in schools has continually gained momentum. In 2012, New York City Mayor Michael Bloomberg announced that his New Year's resolution was to learn to code. In the same year it was announced that all first graders in Estonia would learn to code. This triggered debate in many countries, including the UK and Ireland, about whether all children should be given the

opportunity to learn to code (Resnick 2012). In June 2014, the European Coding Initiative was created under the auspices of the European Commission "to promote coding and computational thinking at all levels of education" (All you need is code 2017, para. 1). Their aim is to establish coding as a key competence within every education system in Europe. In October 2014, the European Commission launched CodeWeek which held events all across Europe. These events brought coding to the attention of the media and contributed to rising political momentum for greater integration of coding in primary and post-primary education. Since then, coding in education has continued to be an increasing international trend. In 2015, the US president stated that all children should learn to code from an early age (European Schoolnet 2015). Schools in Australia have been pushing for a greater focus on STEM education at the primary level. In their National STEM School Education Strategy 2016-2026, the Education Ministers advocated for the inclusion of coding as a mandatory part of the national curriculum. In October 2014, European Schoolnet, a network of thirty ministries of education (including the DES), conducted a survey among twenty one member Ministries of Education to establish to what degree was coding already in national curricula (European Schoolnet 2014). The survey found that 16 countries integrate coding in the curriculum at the national, regional or local level: Austria, Bulgaria, the Czech Republic, Denmark, Estonia, France, Hungary, Ireland, Israel, Lithuania, Malta, Spain, Poland, Portugal, Slovakia and the UK (England) (European Schoolnet 2014). However, only Estonia, France, Israel, Spain, Slovakia, and the UK integrate or will integrate (Belgium Flanders, Finland, Poland, Portugal) coding at primary level (European Schoolnet 2014). Estonia has been a frontrunner in the integration of coding into the national curriculum. They introduced coding to primary schools in 2012 and have now

integrated coding at all levels in school education (European Schoolnet 2014). In September 2014, schools across England and Wales replaced ICT with a new 'computing' curriculum, with one of the core components of this being the teaching of coding. In Ireland, coding was introduced as a short course to the post-primary curriculum in 2014. However, it is not compulsory and depends on school curricula (European Schoolnet 2014). There is currently no national programme at primary level, although some primary school teachers may use Scratch programming in the instruction of shape and space, as this was an approach advocated in the 'Information and Communications Technology (ICT) in the Primary School Curriculum: Guidelines for Teachers' (NCCA 2004a). However, there has been growing support for further integration of coding in the primary school curriculum, as outlined in previous sections (1.5 and 2.2). With the introduction of STEM as a curricular area in the primary curriculum framework (DES 2023a), it now seems that the integration of coding to the primary school curriculum could become a reality.

Approaches to Integrating Programming into School Curricula

Several studies have explored how programming can be integrated into education curricula. Some of these studies have focused on outcomes other than computational thinking, such as the development of language or mathematical skills. Robertson (2012) examined how the literacy skills of 11-12 year olds could be developed using the media storytelling software Adventure Author. She found that game making had the potential to develop a range of storytelling skills including creative narrative, compelling dialogue and attractive visual design. Burke (2012) who worked with 12-14 year olds, suggested that the Scratch programming environment offered a new medium through which students can practice the composition skills they have learnt

in their literacy classes. Calder (2010) explored the development of mathematical concepts through the use of Scratch with Year 6 students (10-11 year olds) in New Zealand. He found that Scratch provided these students with an opportunity to develop their understanding of angles, coordinates and measurement. He believed this understanding was facilitated by the 'tinkerability' of Scratch, which allowed the students to actively experiment with positioning and angle size. Paparistodemou *et al*. (2017) also used the programming language Scratch to explore probability concepts with 8-13 year old students in Cyprus. They found that game making provided these students with opportunities to explore key concepts of probability such as randomness, sample spaces and fairness. Fessakis *et al*. (2013) worked with kindergarten children (5 year olds) using two Logo-based environments: Ladybug Leaf and Ladybug Maze. They found that these programming environments had the potential to improve children's mathematical skills in a variety of areas including, 1-1 correspondence, counting, number comparison and angle/turn concepts.

2.9.2.4 Evaluation of Programming Initiatives

Despite strong support for the introduction of programming to school curricula, few evaluations of programming initiatives in schools have been conducted (European Schoolnet 2015). Evidence supporting or refuting the impact of programming on student learning has been mostly anecdotal in nature, and very few studies have adopted an experimental research design. Indeed, as the introduction of computational thinking to primary school curricula is a relatively new idea, many of the research papers which examine the use of programming to develop computational thinking are conceptual papers (Brennan and Resnick 2012; Lockwood and Mooney 2018; Moreno-León *et al*. 2015; NCCA 2018). A review of

the literature also revealed a dearth of research on the development of computational thinking through programming in primary education, with many of the studies conducted in higher education or informal learning contexts. As can be seen in Appendix A, only four studies that explored the development of computational thinking concepts through programming in formal primary education settings were identified (Wilson *et al.* 2012; Sáez-López *et al*. 2016; Lee *et al*. 2017; Sáez-López and Sevillano-García 2017). There were a larger number of studies conducted with middle school or high school students (Meerbaum-Salant *et al*. 2010; Baytak and Land 2011; Burke 2012; Grover *et al*. 2014; Fields *et al*. 2015; Grover and Basu 2017). Studies conducted in informal settings were also included (Maloney *et al*. 2008; Denner *et al*. 2012; Seiter and Foreman 2013; Funke *et al*. 2017). While the difference between these settings should be acknowledged (e.g., students may have self-selected to participate perhaps signifying greater interest and higher self-efficacy (Newton *et al*. 2020)), these studies can still provide valuable insight to inform research at primary level. These studies adopted a variety of different approaches (see Appendix A) to developing and evaluating computational thinking, making it difficult to make comparisons or draw general conclusions. For example, some studies reported on interventions that were implemented over a period of 18 months (Maloney *et al*. 2008), whereas other interventions involved 8 or 10 programming sessions (Meerbaum-Salant *et al*. 2010; Wilson *et al*. 2012; Fields *et al*. 2015). In some studies the programming sessions were highly structured and involved analysis of a final project (Burke 2012), while other researchers adopted a less structured approach, favouring a learning on demand model in the sessions, and collecting projects weekly (Maloney *et al*. 2008). Therefore, the findings explored in the following sections must be considered with this in mind.

Flow Control: Sequencing and Loops

Maloney *et al*. (2008) analysed projects created by over eighty youths (8-18 year olds) who attended an after-school Computer Clubhouse. They reported that although 111 of these projects contained no code, the remaining 425 projects all incorporated sequential execution. Burke (2012) conducted eleven sessions with ten middle school boys (aged 12-14). These sessions were part of a school elective course which aimed to develop students' English composition and programming skills. Eight of the ten participants felt that storytelling was a suitable medium for learning about sequencing in Scratch. Funke *et al*. (2017) described the computational thinking concepts developed by fourth grade students, aged 9-10, who participated in a three-day introductory programming course. The majority of fourth grade students had little difficulty with the concept of sequencing, with 90% (of 127) projects arranged in accurate systematic order. Wilson *et al*. (2012) described similarly high levels of understanding of this concept among students from Primary 4 to Primary 7 (8-11 year olds) in a Scottish primary school, following an eight-week game design intervention. Sáez-López *et al*. (2016) adopted a quasi-experimental approach to assess the use of Scratch in the development of computational concepts among 5th and 6th grade students in Spain. This research adopted a somewhat longer term approach than many of the other reported initiatives, consisting of twenty one-hour sessions, spanning two academic years. They reported positive findings in relation to the learning of both sequences and loops. In their study, Funke *et al*. (2017) identified heavy usage of the *forever* block in the coding of animations,

perhaps suggesting the creation of animations would be a good medium for the teaching of infinite loops.

Maloney *et al*. (2008) also noticed that the youths frequently included loops in their projects, despite receiving no formal instruction. They posited that the prevalence of loops in the analysed projects was due to the popularity of animations and games, which they believe support the development of this concept. Similar to the findings of Maloney *et al*. (2008), Burke (2012), Grover *et al*. (2014) and Meerbaum-Salant *et al*. (2010) found that most students in their respective studies demonstrated an understanding of loops. Meerbaum-Salant *et al*. (2013) who investigated the use of the Scratch programming environment to teach computer science (CS) concepts to 14-15 year olds, attributed students' comprehension of loops to early introduction to infinite loops. Sáez-López and Sevillano-García (2017) conducted design-based research to ascertain the value of using technological resources (Scratch and Raspberry Pi) in developing computational thinking. The participants were 144 sixth grade students from four primary schools, including a control group of 35 students who completed the same unit of study without the technological resources. They reported very high levels of understanding in terms of sequences and loops from the experimental group. In contrast, the control group achieved low values regarding both sequences and loops, illustrating the value of using programming to develop these concepts. Park and Shin (2019) analysed 524 "high-quality Scratch projects on the Scratch website" (p.1274). They found that the Scratch scores were high for flow control and suggested that Scratch is a 'highly appropriate' language for learning flow control (sequencing and loops).

However, not all studies reported such positive findings with respect to children's understanding of loops. In their study, Baytak and Land (2011) found that students' use of loops were limited to use of the *forever* block, suggesting they failed to reach a proficient level (Moreno-León *et al*. 2015) for this concept. Aivaloglou and Hermans (2016) who analysed 250,000 projects from the Scratch website reported similar findings. Their research revealed abundant use of the *forever* block but limited use of the *repeat until* block. Grover and Basu (2017) found that the middle school students in their study struggled to fully comprehend loops that involved variables. However, it was acknowledged that they had received very little instruction on this concept. From the studies, it seems that sequencing is an accessible concept for most primary school students. Similarly, it seems that a basic level of understanding of loops is achievable, whereas acquiring the more complex aspects of loops may be more challenging (Pea 1986). The research recommends the use of programming games (Hainey *et al*. 2020; Maloney *et al*. 2008) and animations (Maloney *et al*. 2008) in the development of both these concepts.

Conditionals, Operators and Boolean Logic

Although some researchers present positive findings, the results pertaining to conditional statements and operators, in particular Boolean logic (*and*, *or*, and *not*), were not quite as optimistic. Grover *et al*. (2014) found that the twenty six middle school students, enrolled in an elective course in computing, performed well on questions that assessed their understanding of conditionals, achieving a mean score of 85%. Maloney *et al*. (2008) reported that the youths in the computer clubhouse frequently used conditionals in the development of their projects. Wilson *et al*. (2012) reported that conditional statements were among the most used programming

concepts by the primary school children in their study, perhaps suggesting that games are a good context for introducing children to this concept. Denner *et al*. (2012) examined the development of CS concepts by 11-12 year old girls participating in a voluntary after-school programming initiative using the Stagecast Creator Software. Like Wilson *et al*. (2012) their study employed a game design approach. They also reported that most of the students' games included conditionals, however they observed that some (18%) of these were not functional. The two most common uses of conditionals were to initiate a scene or stage change or to create a response to character interaction. Seiter and Foreman (2013) report that conditionals were under-utilised by students from the lower grades in their study. However, this finding must be interpreted with caution as these projects were acquired from different sources on the Scratch website, so it is unknown if students had any prior exposure to this concept. Maloney *et al*. (2008) suggest that Boolean logic is a concept not easily discovered without assistance. In their study, they reported limited use of Boolean operators but found that the number of projects utilising this programming concept did increase over time. Burke (2012) found that conditionals were used infrequently by the boys in their study, with conditional statements found in only 30% of projects and Boolean logic only utilised in 20% of projects. However, Burke (2012) suggested the storytelling medium limited the opportunity for the use of conditional statements and Boolean Logic as stories tend to have a fixed course and ending. Baytak and Land (2011) found that the middle grade students in their study demonstrated only a basic understanding of conditional statements, rarely using more complex conditional statements than the *if* block. They also reported that their students had difficulty creating their own Boolean expressions without teacher guidance. Grover and Basu (2017) found that half the middle school students in their

study struggled with Boolean logic, with many students illustrating misconceptions relating to the *or* operator. They suggest that block based programming languages such as Scratch may leave students with an incomplete understanding of operators. However, Šerbec *et al*. (2018) who analysed the projects of primary school students and Sophomore students found that the older students demonstrated greater abstraction proficiency, suggesting that these differences could be explained by the different reasoning abilities of older and younger students.

Data Representation: Variables

According to Maloney *et al*. (2008), variables, like Boolean logic, is a concept that is more likely to be understood with formal instruction. They found that variables were underutilised by the computer clubhouse users until "a timely visit by a knowledgeable mentor" (p.371). The findings of Meerbaum-Salant *et al*. (2013) support the views of Maloney *et al*. (2018). They reported that their ninth grade students encountered difficulties with the variable concept. However, they believed that these difficulties related to the students' limited exposure to the concept, and they felt their students' understanding of variables could be improved through explicit instruction. Several other researchers portray limited variable use in programmes. Seiter and Foreman (2013), who analysed 150 projects of students form grade one through six, found that data representation was used quite sparsely in these projects, in particular among those created by students in grades one and two. They suggested that this was one of the more challenging programming concepts for children to grasp. Variables were only used in 10% of the Scratch projects by the middle school participants enrolled in Burke's (2012) 'Programming-as-Writing' course. However, Burke (2012) proposed that the storytelling medium does not

naturally encourage the inclusion of variables due to its linear nature. While Denner *et al*. (2012) expected that game making might encourage the use of variables, they found limited use of variables. They noted that when variables were included, they tended to be character variables, and the inclusion of global variables (a variable accessible by all sprites) was very infrequent. Contrastingly, Baytak and Land (2011) and Wilson *et al*. (2012) reported positive findings with respect to the use of variables in game making. Baytak and Land (2011) found that the middle grade students in their study were able to use complex commands like variables in creating their Scratch games. Wilson *et al*. (2012) reported that variables were one of the most used programming concepts in their study and suggested game making encouraged the use of variables. These contrasting findings highlight that there are many factors that can facilitate or impede the development of these computational concepts.

Event Handling and User Interactivity

The findings in relation to event handling were predominantly positive with several researchers reporting effective comprehension of this computational concept. Seiter and Foreman (2013) identified event handling as one of the concepts that students in the younger grades were able to comprehend, with children from second grade upwards implementing this concept, albeit at a basic level. Sáez-López *et al*. (2016) reported positive findings in relation to event handling during their two-year study of Scratch programming in five Spanish primary schools. Wilson *et al.* (2012) found high levels of event handling in the games created by the primary school children in their study, with 90% of games requiring user interaction. Similarly, user interactivity was high in the projects made by the computer clubhouse users in

Maloney *et al*.'s (2008) study. Maloney *et al*. (2008) proposed that this was due to the popularity of game creation, a project type which they believe promotes the use of user interactivity. However, Burke (2012) reported that event handling was used in all projects of the ten middle school boys enrolled in his storytelling through programming course, suggesting that story creation also supports event triggers. Given that Burke (2012) described stories as 'straight-linear narratives' with 'fixed endings' it is possibly surprising that such high levels of event handling were reported. Perhaps though this finding can be explained by considering the findings of Funke *et al*. (2017). They reported that event blocks were regularly used by the fourth graders in their study. However, they found limited use of interactive blocks, with nearly two-thirds of projects incorporating no interactive elements for the user. In the explanation of their findings, they distinguish between types of events, viewing the *ask and wait* block and resultant keyboard input, as distinct from other event triggers such as *when green flag clicked*. This suggests that different project types (stories, games and animations) may support the discovery and application of different types of event triggers. Park and Shin (2019) propose that the choice of programming language is also a factor in computational concept development. They found that App Inventor was more effective than Scratch in supporting the development of both variables and user interactivity.

Parallelism

Parallelism was another programming concept that was found to be accessible to young learners in the majority of research studies. Burke (2012) reported that the middle school boys in his study were proficient at incorporating parallelism in their programming, with parallelism included in all of their Scratch stories. The primary

school students in Sáez-López et al.'s (2016) study also successfully implemented parallel execution in their Scratch projects. Similarly, Funke and Geldreich (2017), Maloney *et al*. (2008), Troiano *et al*. (2019) and Wilson *et al*. (2012) all reported high usage of parallelism in their respective studies. Sáez-López and Sevillano-García (2017), who examined the development of programming concepts with and without the use of technology, reported significantly higher levels of understanding with respect to parallelism among students who engaged in programming. Meerbaum-Salant *et al*. (2013) identified two distinct types of concurrency (parallelism):

> Type I concurrency occurs when several sprites are executing scripts simultaneously, for example, sprites representing two dancers.
>
> Type II concurrency occurs when a single sprite executes more than one script simultaneously, for example, a Pac-Man sprite moving through a maze while opening and closing its mouth.
>
> (p.248)

They found that students' understanding of type one was much stronger than type two, and recommended explicit teaching was necessary to address the difficulties the students encountered with type two concurrency. Seiter and Foreman (2013) believe that parallelism is beyond the comprehension of younger students and suggest it is a concept more comprehensible to later primary grades. Indeed, both von Wangenheim *et al.* (2017) and Weng and Wong (2017) reported low levels of understanding in relation to this concept among the children in their studies.

Initialisation and Synchronisation

Seiter and Foreman (2013) make similar recommendations about the teaching and learning of initialisation. They propose that initialisation requires sophisticated understandings of design and is therefore more commonly understood by those in grade three or above. Interestingly the other research studies, that were identified in a

search for studies on initialisation, were conducted with older students. Meerbaum-Salant *et al*. (2013) reported that the students in their study had difficulty defining the concept of initialisation, however they acknowledged that in-class observations illustrated greater understanding than the definitions suggested. They acknowledged that like variables, initialisation was an abstract concept and therefore more explicit instruction would be beneficial. Prior to their study, Fields *et al*. (2015) identified initialisation and synchronisation as challenging concepts for novice programmers and consequently designed their workshop to support the learning of these concepts. The participants in the workshop were high school students, aged 14-15, attending a school recognised for its focus on STEM. They found that the students were successful in implementing a range of initialisation strategies in the design of their Scratch music videos. Fields *et al*. (2015) also found that their high school students successfully applied a variety of synchronisation strategies, ranging from simple to complex, to collaboratively create a cohesive Scratch music video. Hoover *et al*. (2016), albeit with a very small sample (five participants), reported similar findings with middle school students. Other researchers had previously found students had failed to incorporate more complex synchronisation in their programmes. Wilson *et al*. (2012) reported that while many of the games created by the children in their study included synchronisation, it was predominantly at a basic level, i.e. use of the *wait* block. Similarly, Seiter and Foreman (2013) found that very few students in any grade used synchronisation at the highest level i.e. use of the *wait until* block. Baytak and Land (2011) found that while boys rarely implemented the broadcast feature in their programs, most girls used many broadcasting features in their games, illustrating more advanced levels of synchronisation. Despite the difficulties encountered in these studies, other studies have reported more positive outcomes.

Maloney *et al*. (2008) observed that synchronisation commands were heavily used by the computer clubhouse users, although they noticed "a significant reduction in the number of projects utilizing this particular concept" as time went on (p.370). Burke (2012) proposed synchronisation is an essential component of digital storytelling, reporting that his study participants used it repeatedly in all of their Scratch stories.

Procedures or Abstraction

As already outlined, in studies utilising Dr. Scratch as an assessment tool, procedures are termed abstraction and problem decomposition (Ch'ng *et al*. (2019) and the lowest level focuses on problem decomposition, whereas levels two and three require the use of abstraction constructs 'My Blocks' and cloning respectively. Previous research has shown high levels of understanding, among students of varying ages, with respect to problem decomposition (Hoover *et al*. 2016; Šerbec *et al*. 2018; Setyawan 2020; Troiano *et al*. 2019). However, contrasting findings have been reported with respect to abstraction concepts, with several studies identifying abstraction as the most challenging computational concept (Hoover *et al*. 2016; Park and Shin 2019; Troiano *et al*. 2019). These studies found that very few students achieved higher than a basic proficiency for this concept. Troiano *et al*. (2019) who evaluated the projects of 317 eighth grade students found that only 20% of them illustrated an understanding of abstraction. Lawanto's (2016) research involving students of a similar age (seventh and eighth grade), also identified abstraction as a 'relative weakness' of his participants. These students, which included students who had created projects categorised as 'proficient', achieved an average score of less than 1.1. Seiter and Foreman (2013) suggested that grade level might be an

important factor in the successful acquisition of abstraction concepts. However, both Šerbec *et al*. (2018) and Setyawan (2020) found that older students also struggle with these concepts. Šerbec *et al*. (2016) predicted that the older Sophomore students would outperform the primary school students with respect to abstraction scoring. However, this did not happen, instead the majority of both groups achieved a score of one. Setwayan (2020) who assessed the computational concept development of pre-service teachers also reported very low abstraction scores, with the majority (77%) of pre-service teachers scoring just one for this concept. Werner *et al*. (2014) were one of the few researchers who reported some success with respect to abstraction proficiency. They attributed the success of their students to their previous exposure to the concepts.

> Youth in the programming classes are able to directly correlate their experience designing a game with the classic Snake mechanic, a maze game, and a Flappy Bird clone to more advanced steps for designing their own self-determined games.
>
> (Werner *et al*. 2014, p.49)

Computational Concept Development – Influencing Factors

Research on the development of computational concepts has revealed computational concepts that novice programmers have successfully implemented in their programming and those they have found more challenging. Although there is not complete consensus on this, many studies have identified flow control, parallelism, and synchronisation as more accessible concepts for novice programmers, whereas abstraction, data representation and logic are deemed more challenging. Further examination of the literature reveals that contextual factors such as programming language (Franklin *et al*. 2016), project type (Adams and Webster 2012; Burke 2012; Moreno-León *et al*. 2020), exposure to concepts (Meerbaum-Salant *et al*. 2013), age (Lawanto 2016; Seiter and Foreman 2013; Šerbec *et al*. 2018) and length of initiative

(Baytak and Land 2011), can all impact students' understanding of computational concepts. At the same time, mathematical concepts such as coordinates (Germia and Panorkou 2020) and inequalities (Turner *et al*. 2016) might provide additional challenges to the development of computational concepts. Any programming initiative attempting to develop computational concepts would require careful consideration of these factors.

Developing Computational Practices through Programming

Although there are fewer studies examining the computational practices dimension of computational thinking (Vourletsis *et al*. 2021), there were many studies that explored the development of problem solving through programming. As problem solving is considered to be a key element of the computational practices dimension, some of these studies are explored in this section. Lai and Yang (2011) examined the use of a visual programming language (Scratch) to facilitate the development of Taiwanese sixth grade students' (11-12 years old) problem-solving skills. The intervention took place over the course of one semester and was part of a computer literacy course. They found a significant difference between the experiment group and control group in terms of problem-solving ability, particularly in relation to predicting. Also, in Taiwan, Lin and Liu (2012) conducted their study of parent-child pair programming in the context of a five-day computer camp. They used the Logo-based programming environment MSWLogo. They found that the parents encouraged the children to analyse the task and plan how they would solve the problem before writing any code. Consequently, the children "wrote programs in a more systematic and disciplined manner instead of resorting to trial-and-error and tinkering" (p.162). Falloon (2015) reports on a programming initiative undertaken

with 9-10 year olds in New Zealand using the programming environments Pyonkee and Cargobot. This initiative took place outside the classroom in a modern learning environment (MLE), where multiple class groups work together in a common space. He found the students rarely used guesswork or trial-and-error in their problem solving, instead employing higher order thinking skills such as reflection, analysis and evaluation. Paparistodemou *et al*. (2017), who delivered a series of voluntary workshops for 8-13 year olds, also report that the process of designing, coding, revising and debugging facilitated the development of children's higher order problem-solving skills, for example, deductive reasoning. Each of these aforementioned studies took place in an informal learning environment except for Lai and Yang's (2011) study, and their study was conducted with students who had selected to participate in the computer literacy course.

Several studies conducted in a classroom setting have also described the problem solving skills employed by their students while programming. Fessakis *et al*. (2013) found that the kindergarten children in her study preferred to use a more primitive 'stepwise refinement' approach to problem solving. These students tended to execute commands one by one and receive feedback. Calder (2010) also reports that the students in his study used this 'predict and check' problem-solving approach. These findings are consistent with the findings of Robins *et al*. (2003), who found that novice programmers tend to write their code line-by-line rather than programming in chunks. Daily *et al*. (2014) also reported that their students used similar strategies. They propose that although this trial and error approach is not an efficient way of coding and debugging, it provides an opportunity to introduce alternative strategies. However, they did observe some of their students using the computational practices of abstraction and modularization. When creating their dance projects, the students

employed abstraction and modularization, breaking the dance into parts and creating scripts for each part before putting them all together. They also observed the students testing and debugging. This testing and debugging often involved them performing the moves themselves before refining their code. Kalelioglu and Gülbahar (2014), investigated the effect of Scratch programming on Turkish fifth grade students' problem solving skills, and reported findings to the contrary. They employed a quasi-experimental design in their study and found that programming in Scratch did not have a significant impact on the students' problem solving skills. However, they found that the programming intervention improved students' confidence in their problem solving ability. Webb and Rosson (2013) used several programming technologies in their study including Alice, Scratch and Mindstorms. They also reported improvement in the self-efficacy for computational problem solving of their students (10-13 year olds). Calder (2010) reported that the students in his study found Scratch to be an intrinsically motivating context for mathematical exploration. Wilson and Moffat (2010) also reported that the children (7-9 year olds) in their study found Scratch to be an enjoyable tool for learning about programming.

Developing Computational Perspectives through Programming

Computational perspectives relates to "students' understanding of themselves, their relationships to others, and the technological world around them" (Lye and Koh 2014, p.53). A review of the literature revealed few studies, which explored the computational perspective dimension of computational thinking. Of these studies, only one was conducted with primary level students. Carjaval (2017) examined the affective responses of children to Scratch programming in the context of mathematics, found that positive responses were more likely to occur when the

activities were exploratory in nature rather than having a mathematical focus. Fronza *et al*. (2015) report on a one-week Summer School facilitated by a University in Italy. The Summer School aimed at senior high school students (16-18 year olds) was designed to develop students' computational thinking skills through programming using App Inventor. They suggested that the students' enthusiasm to show friends and family their creations was an informal indicator of their positive reaction to the programming experience. Burke (2012) also described the eagerness of his students to share their projects online and with their classroom peers. 90% of participants (nine out of ten students) agreed or strongly agreed that the programming experience had been enjoyable. The majority of participants also reported they learned more about computing and felt more skilled at computing after the Scratch workshops. Burke (2012) also reported that several of the students chose to base their stories on real-life experiences. Daily *et al*. (2014) suggest that viewing technology, as a medium for self-expression is illustrative/a key element of computational perspective formation. They reported that the students in their study enjoyed the opportunity the programming experience afforded them to express themselves and be creative. Similarly, a study carried out by Kafai *et al*. (2014) also revealed that opportunities for personal expression through computing gave students a greater appreciation of the relevance of computing to them. These students (16-18 year olds) engaged in programming activities using the Arduino environment during an e-textiles module, a component of their computer science course. Alongside expanding views about the relevance of computing, the students also reported a growing understanding of the discipline of computing and an increase in their self-perception as computer scientists (Kafai *et al*. 2014). Like Carjaval (2017), the

authors propose that the openness of the task had a major impact on the success of the module.

2.9.2.5 Perceived Benefits of the Plugged Approach

Advocates of programming argue that teaching children to program has a wide range of benefits, from digital literacy and computational thinking to creativity and problem solving. One of the arguments against using the plugged approach is the idea that children are spending enough time in front of screens without encouraging this practice further. However, a paper published by the NAEYC and the Fred Rogers Center for Early Learning and Children's Media (2012) states that "all screens are not created equal" (p.3). In this paper, the authors suggest that digital technology has advanced beyond the passive and non-interactive uses to include more interactive choices, which necessitates the development of new criteria around usage. Howland *et al*. (2013) describe this as a shift "from technology-as-teacher to technology-as-partner" (p.7). In their vision, they see technology as a tool which can support meaning making and higher order thinking, rather than a tool which students learn from. Smith *et al*. (2000) were also critical of the passive use of computers in schools, claiming that such an approach prevented students from exploiting the power of computers. As Resnick *et al*. (2009a) remarked, "it's as if they can 'read' but not 'write'" (p.62). So, it seems that as digital technology expands in scope, so too must the concept of digital literacy. The National Research Council (1999) define digital literacy or fluency as

> the ability to reformulate knowledge, to express oneself creatively and appropriately, and to produce and generate information (rather than simply to comprehend it).
>
> (p.viii)

This definition emphasises the importance of creativity, expressing oneself and creating content. Several other researchers have highlighted these as key elements of digital literacy (Resnick *et al*. 2009a; Lye and Koh 2014; Kafai and Burke 2016; Weber and Greiff 2023). Programming is an example of interactive technology use, which encourages the creation of content rather than just the consumption of content (Maloney *et al*. 2008; Monroy-Hernández and Resnick 2008). It enables the creation of new types of content: interactive stories, animations and games (Resnick 2013). Such programming activities provide students with the opportunity to express their ideas and create digital artifacts that are personally meaningful to them (Kafai and Burke 2016; Lye and Koh 2014). According to Kafai and Burke (2016), this emphasis on designing content fits well with Wing's (2006) definition of computational thinking: "solving problems, designing systems, and understanding human behaviour" (p.33). As the content is designed for use by others, the creators must be cognisant of human behaviour as they create, and they must problem solve as they write their ideas as code (Kafai and Burke 2016). Several researchers also draw parallels between creative programming activities and the three dimensions of computational thinking, as defined by Brennan and Resnick (2012): computational concepts, computational practices, and computational perspectives.

Developing Computational Concepts

The complexity of learning programming syntax was identified as one of the main reasons why students in early programming initiatives failed to acquire computational concepts. However, in recent years more developmentally appropriate technologies have emerged. Several of these are visual programming languages with commands that are close to spoken English and without unnecessary syntax (Lye and

Koh 2014). In block-based programming, the blocks only fit together in specific ways, further reducing the chance of syntax error (Resnick 2007). According to Kelleher and Pausch (2014) this allows "students to focus on the logic…rather than worrying about the mechanics of writing programs" (p.131). So, for example, instead of concentrating on the command structure of the programming language for creating a variable, they can focus on how they can use a variable (such as score) to enhance their game. Both Shelton (2016) and Kafai and Burke (2016) suggest that allowing children to create their own programs is the best approach to introduce them to computational concepts. Similarly, Grover and Pea (2013) state:

> Noteworthy efforts like CS Unplugged (http://csunplugged .org/) that introduce computing concepts without the use of a computer, while providing valuable introductory activities for exposing children to the nature of CS, may be keeping learners from the crucial computational experiences involved in CT's common practice.
>
> (p.40)

These arguments all support the use of plugged approaches in the development of computational concepts.

Developing Computational Practices

Brennan & Resnick (2012) identified four key computational practices (testing and debugging, being incremental and iterative, abstracting and modularizing, and reusing and remixing). Several aspects of programming languages, like Scratch, make them useful tools in the development of computational practices. Lye and Koh (2014) and Maloney *et al*. (2010) suggest that because the effects of their commands are immediately observable as animated objects, this facilitates practices such as testing and debugging. Maloney *et al*. (2010) believe that debugging is further facilitated by Scratch, as Scratch does not require a script to be fully complete before it can be run. This feature of Scratch allows users to break longer scripts into smaller

pieces of code which can then each be tested individually to identify which part of the code contains the bug. Being able to develop and run small fragments of code also encourages the practice of being incremental and iterative. Scratch also allows for the continual addition of new blocks as the program runs, further supporting an incremental and iterative approach (Resnick 2007). Each sprite's behaviour or actions are controlled by its own set of commands in Scratch. Brennan & Resnick (2012) and Busuttil (2014) both found that this feature encouraged users to develop the practice of abstracting and modularizing. They reported that when children were creating projects, they broke down their tasks by separating the desired behaviours or actions of individual sprites and then coding them independently. According to Brennan and Resnick (2012) this practice of modularizing also aided in the debugging of projects. Scratch was designed to help young people learn how to program through both exploration and peer sharing (Maloney *et al*. 2010). To this end, reusing and remixing is a practice both encouraged and facilitated through the Scratch website. The website allows users to create and share their own projects as well as view and remix projects designed by others. This social aspect of Scratch has many benefits including: receiving feedback from others on your projects (Maloney *et al*. 2010), getting ideas for new projects (Busuttil 2014), and learning new concepts from projects of others (Busuttil 2014; Maloney *et al*. 2010) thus allowing users to create more complex programs than they could have created by themselves.

Developing Computational Perspectives

In their definition Brennan and Resnick (2012) identified three behaviours that illustrate the development of computational perspectives: expressing (recognising computing as a medium for creation), questioning (evolving perspectives on

technology) and connecting (collaborating and sharing). Several researchers have recognised the potential of programming activities as a medium for students to create and express themselves (Brennan and Resnick 2012; Kafai and Burke 2016; Resnick *et al*. 2009b; Romero 2010; de Carvalho *et al*. 2020). Resnick (2007) suggests that programming not only helps develop students Creativity with a big c, but it can also help students develop 'creativity' with a small c. By this Resnick (2007) means that programming can help students "become more creative in the ways they deal with everyday problems" (p.2). Myers (2008) makes similar claims, suggesting that the trial-and-error approach adopted in programming encourages the development of creativity that carries over to other domains. Lye and Koh (2014) suggest such opportunities for self-expression can impact students' computational perspectives on technology. Similarly, Kafai (1995) suggests that opportunities for creative and personal expression can make technology more meaningful. Kafai and Burke (2016) also suggest that teaching students to program could help change current perspectives of computer science as a 'geek' culture. Kafai and Burke (2016) describe learning to program as a 'social enterprise' and see it as an opportunity for social participation. Several researchers have highlighted the social aspect of programming. Papert (1984) promoted the 'ask three before you ask me' strategy to encourage social learning among his students. Resnick (2007) suggests that after kindergarten less emphasis is put on sharing and collaborating, but that this is something which schools are trying to change. He suggests that programming activities would provide such opportunities for sharing and collaborating, as they are essential elements of the programming process. Robertson (2012) reported on a study, which examined the development of audience awareness skills. The students designed, shared and reviewed games they programmed using Adventure Author

software. She found that engaging in peer reviewing helped students to both improve their work and develop their evaluative skills. Several researchers, including Jeanette Wing herself, have expressed concerns that adopting the plugged approach to develop computational thinking might lead students to develop a narrow view of computational thinking as programming. However, Kafai and Burke (2016) believe that using computers to develop computational thinking does not necessarily mean equating computational thinking with programming. Instead, they propose that the focus of any plugged approach should be on designing digital artefacts that are personally meaningful to the creator. Hence, the focus of the activity is not on the development of programming skills but on the development of key aspects of computational thinking: "solving problems, designing systems, and understanding human behaviour" (Wing 2006, p.33).

2.9.3 Motivation to Engage with Technology

Within the literature on developing computational thinking through unplugged and plugged activities there were several references to students' developing motivation to engage with technology (Calder 2010; Wilson and Moffat 2010; Feaster *et al*. 2011; Burke 2012; Taub *et al*. 2012; Jiang and Wong 2019). Kong (2019) recognised the importance of developing learners' computational identity to nurture interest in computing. He proposed that the Brennan and Resnick (2012) framework neglected to capture learners' motivation to engage in computing and contended that computational identity should be included as a computational perspective. Therefore, this section will explore engagement and the motivational aspects that impact students' motivation to engage in programming.

Kong (2019) referenced Wenger's (1998) definition of engagement in his description of engagement:

> the most immediate relation to a practice-engaging in activities, doing things, working alone or together, talking, using and producing artifacts.
> (Wenger 1998, p.4)

Fredricks *et al*. (2004) identified three distinct aspects of engagement: behavioural, cognitive and emotional. Behavioural engagement relates to student involvement in activities and on-task behaviour. Cognitive engagement reflects student investment and captures their willingness to apply themselves when faced when challenged by high levels of complexity. Lastly, emotional engagement concerns students' affective responses in the classroom, including interests, values and emotions.

2.9.3.1 Behavioural Engagement

Fredricks *et al*. (2004) identified three definitions for behavioural engagement, the first two of which were relative to the context of this study. The first definition focussed on positive conduct and the absence of disruptive behaviours (Fredricks *et al*. 2004). It included behaviours such as following the class rules and not getting into trouble. The second definition related to students' involvement in learning academic activities and included behaviours such as attention, concentration, effort, persistence, asking questions, and contributing to discussions. The final definition concerned participation in school-related activities including school governance and was deemed inappropriate for this short-term initiative. Hjorth (2017) reported that Scratch positively impacted the behavioural engagement of the students in her study. She described them as both highly curious and highly focused, illustrating a willingness to ask questions, while also attentive to their tutors' explanations. Ching *et al*. (2019) also reported that the students in their study were highly engaged and

motivated by the programming tasks they undertook, with students requesting more time to work on their projects and expressing their excitement for the next activity. However, they did note that sometimes the students showed signs of boredom, particularly during activities that required turn taking. Relatedly, Brennan (2013) warned that large class sizes can lead to difficulties monitoring behaviour and providing support, negatively impacting students' behavioural engagement.

2.9.3.2 Cognitive Engagement

According to Fredricks *et al*. (2004) cognitive engagement emphasises psychological investment in learning, self-regulation and being strategic. While Fredricks *et al*. (2004) note there are similarities between the constructs included in the definitions of behavioural and cognitive engagement, they stress that cognitive engagement is focussed on achieving mastery rather than simply fulfilling behavioural expectations. Therefore, cognitive engagement includes constructs relating to intrinsic motivation such as a preference for challenge and a willingness to go beyond requirements (Fredricks *et al*. 2004). While Seymour Papert recognised the importance of challenge in computing tasks, he also recognised that these tasks must provide opportunity for the development of knowledge and skills.

> These rapidly changing times challenge educators to find areas of work that are hard in the right way: they must connect with the kids and also with the areas of knowledge, skills and (don't let us forget) ethic adults will need for the future world.
>
> (Papert 2002, para. 3)

During data analysis, one such skill that the students in this study were developing was perseverance when faced with difficulty. Although perseverance is not represented in Brennan & Resnick's (2012) framework, several researchers have identified it as an important dimension of computational thinking (Barr *et al*. 2011;

Millwood *et al*. 2018; Weintrop *et al*. 2016; Woollard, 2016). However, while these research papers promoted the inclusion of perseverance in their defining of computational thinking, they didn't reveal what it might look like or how it might be developed. Both Papert (2002) and Resnick (2006) suggested that perseverance can be developed by providing students with challenging, but not overwhelming, activities that are linked to their interests and passions. Similarly, Brennan (2013) found that the students in her study were more motivated to persevere through challenge when they felt a deep personal investment. However, Blikstein and Worsley (2016) stress the importance of balancing challenge and achievability to avoid exposing students to excessive levels of frustration. Abdunabi *et al*. (2019) posits that students' self-efficacy (their belief in their programming competence) impacts both their choice of activity and their willingness to persevere in the face of difficulty, providing further justification for balancing challenge and achievability.

2.9.3.3 Emotional Engagement

Conceptions of students' emotional engagement incorporate the constructs of emotions, interest and value (Fredricks *et al*. 2004). Emotions includes the range of positive and negative feelings, including happy, sad, interested, bored (Fredricks *et al*. (2004). Several researchers have reported on students' emotional response to their Scratch programming experiences. Wilson and Moffat (2012) found that Scratch made learning to programme a positive experience. Calder (2010) and Burke (2012) reported that Scratch was a motivating medium to teach Maths and English respectively. These positive experiences are important as Lusa Krug *et al*. (2023) asserts that they can impact students' intention to persist with computing. Studies on student interest have distinguished between situational and personal interest

(Fredricks *et al*. 2004). Situational interest is transient and can be aroused by specific aspects of tasks for example novelty, whereas personal interest suggests a more enduring interest revealed by consistently choosing to pursue an activity (ibid). According to Wigfield and Eccles (2000) there are three different components of values: intrinsic value, attainment value and utility value.

> Attainment value is…the importance of doing well on a given task. Intrinsic value is the enjoyment one gains from doing the task…Utility value or usefulness refers to how a task fits into an individual's future plans".
>
> (Wigfield and Eccles 2000, p.72)

Utility Value as defined by Wigfield and Eccles (2000), bears resemblance to the construct of imagination as defined by Kong and Lai (2022). According to Kong and Lai (2022) imagination refers to children's perspectives on their future involvement in programming activities, which includes both potential career aspirations and future commitments (Kong and Lai 2022). Many researchers emphasise the need for all students to develop computational thinking, including those who allege no interest in pursuing careers in computing (Grover and Pea 2013). Therefore, the inclusion of career aspirations or future commitments in computational thinking is disputed in the literature. Consequently, for the purpose of this research these constructs will be considered as one and defined as utility value. Abdunabi *et al*. (2019) stresses the importance of developing students' value of programming, as they found that it impacts their willingness to persist in solving challenging problems.

2.10 Learning from Lessons of the Past

> The phrase 'technology and education' usually means inventing new gadgets to teach the same old stuff in a thinly disguised version of the same old way. Moreover, if the gadgets are computers, the same old teaching becomes incredibly more expensive and biased towards its dullest parts, namely the kind of rote learning in which measurable results can be obtained by treating the children like pigeons in a Skinner box.
>
> (Papert 2005, p.353)

While there are those who believe that technology can advance and improve education, others disagree citing a long history of postulations regarding the potential of technology which have failed to materialise (Crosier and Simeoni 2019). Indeed, despite rapid technological advances in recent years relatively little has changed in terms of how students learn (Crosier and Simeoni 2019). However, those who still believe in the educational potential of technology suggest this lack of progress stems from the way technology is being utilised rather than the technology itself. Back in 1980, Seymour Papert complained that most educators were using computers to dispense information, provide feedback and for drill-and-practice rather than to develop their students' thinking skills. He referred to this practice as "the computer programming the child" (Papert 1980, p.19). Wyeth and Purchase (2000) says this "mechanistic style of computer use" has caused educators to question what technology can offer that is significantly different from other means (p.141). Kafai and Burke (2016) assert that greater efforts are needed to integrate technology into the teaching of subjects. Both Wilson *et al*. (2012) and Kafai and Burke (2016) highlight the influential role of the teacher in exploiting the full potential of technology. Wilson *et al*. (2012) suggest that many teachers are not comfortable using the technology. Kafai and Burke (2016) identify a dearth of suitably qualified teachers. Both these issues were highlighted in the '2013 ICT Census in Schools', which reported low levels of teacher knowledge and confidence in the use of ICT in

teaching and learning (Cosgrove *et al*. 2014a). This reported lack of teacher knowledge and confidence should perhaps not be surprising as the European Commission (2011) report, 'Key Data on Learning and Innovation through ICT at School in Europe 2011' revealed that very few primary school teachers engage with continuous professional development relating to the integration of ICT.

2.11 Conclusion

This chapter explored how computer use in schools has evolved, and why many governments are now pushing to incorporate both programming and computational thinking into their school curricula. In reviewing research on the implementation of technological initiatives spanning several decades, it became evident that the teacher and the pedagogical decisions they make play a significant role in the successful implementation of these initiatives (Pea 1983; Oakley and McDougall 1997; AAUW 2000a; NCCA 2019a; Butler and Leahy 2022b). Therefore, in the next chapter the teaching approaches, programming languages and assessment tools best suited to developing computational thinking in primary school students will be examined.

Chapter 3: Learning and Teaching Programming

3.1. Introduction

In this chapter, the researcher will outline the theoretical framework adopted in this research study, including the principal concepts, their definitions, and reference to existing scholarly theory relevant to this research study. According to Fox and Bayat (2007)

> concepts are collectives used to label certain bits of experience. In other words they are elementary constructs by which reality is classified and categorised.
>
> (p.6)

Monette *et al*. (2011) describe concepts as "mental constructs or images developed to symbolize ideas, persons, things or events" (p.30). Explanation of the relevant concepts provides a clear statement of the theoretical assumptions underpinning this particular research study. Some of the key concepts in this research study include learning, teaching, instructional design, constructivism, and constructionism.

3.2 Learning

There are several definitions of learning. Two definitions commonly used in the literature to describe learning are those of Hilgard and Mitchell. Hilgard *et al*. (1975) describe learning as a "relatively permanent change in behaviour that occurs as a result of prior experience" (p.194). Mitchell (1978) defines learning as:

> the process by which new behaviours are acquired. It is generally agreed that learning involves changes in behaviour practising new behaviours and establishing permanency in the change.
>
> (p.113)

Chitale *et al*. (2012) suggest that this change in behaviour is accompanied by knowledge, skill or expertise acquisition that is relatively permanent. Contrary to the beliefs of those who may have experienced 'traditional' teaching approaches, learning is not acquiring knowledge or skill by mere mechanical repetitions or rote

memorizations. Rather, it is “a process in which the learner organises different elements and experiences to reach a particular goal” (Kumari *et al*. 2004, p.209). From examining the various definitions of learning, the researcher has identified three elements that are central to the concept of learning:

- The notion of change underlies most definitions of learning, be it change in one’s understanding or behaviours.
- This change occurs as a direct result of their experience or their reflections on their experience rather than through a process of maturation.
- This change must be relatively permanent.

Taking these three elements into consideration, the researcher feels that Merriam *et al*. (2012) offers a comprehensive definition of learning for the purpose of this research study:

> Learning is a process that brings together cognitive, emotional, and environmental influences and experiences for acquiring, enhancing, or making changes in one's knowledge, skills, values, and worldviews.
>
> (p.277)

3.2.1 Approaches to Learning

This view of learning focuses on learning as a process rather than learning as an end product. In educational psychology, there are many different perspectives on how we learn. Many psychologists and behavioural scientists have developed different learning theories. Learning theories develop hypotheses that describe how the learning process takes place. Hill (1977) observed that learning theories have two principal values:

> One is in providing us with a vocabulary and a conceptual framework for interpreting the examples of learning that we observe.
>
> The other, closely related, is in suggesting where to look for solutions to practical problems.
>
> (p.261)

There is little consensus on how many learning theories there actually are or how these learning theories should be grouped for discussion. The most widely accepted learning theories are the behaviourist, cognitivist and constructivist theories. Therefore, the researcher will provide an overview of these different learning theories in the following sections.

The Behaviourist Learning Theory

The behaviourist perspectives on learning originated in the 1900s and became one of the most influential theories in education in the 20th century. In the early twentieth century Watson championed the popular behaviourist movement (Weibell 2011). The major contributors to the behaviourist orientation were Thorndike, Tolman, Guthrie, Bandura, Hull, Pavlov and Skinner. However, not all of these had a direct influence on education. Skinner, in particular, applied his theories to the field of education. According to Merriam *et al*. (2012) the behaviourist learning theory operates on a principle of 'stimulus-response'. The observable behaviour rather than any internal thought processes is the focus of study. "Of primary concern is how the association between the stimulus and response is made, strengthened, and maintained" (Ertmer and Newby 2013, p.48). Therefore, behaviourism disregards any notion that there may be any mental component to learning (Pritchard 2013). This perspective on learning assumes the learner is essentially passive (Callison 2015). The learner is characterised as reactive rather than having an active role in their learning (Etmer and Newby 2013). The learner is considered to start off as a 'tabula rasa' or a clean slate and their behaviour is then shaped by either positive or negative reinforcement. All behaviour is evoked in response to stimuli in the external environment. This acquisition of a new behaviour in response to a stimulus is called conditioning. There

are two types of conditioning: classical conditioning and operant conditioning. In classical conditioning the response to the stimulus is a natural reaction to the environment. However, in operant conditioning, the new behaviour is not a reaction to the environment rather an action upon its environment (Gould 2012). According to Grippin and Peters (1984), the principles of contiguity and reinforcement are the conditions necessary for a certain behaviour or response to be evoked by a stimulus. Contiguity is understood as "how close in time two events must be" to stimulate behavioural change, while reinforcement refers to "any means of increasing the likelihood that an event will be repeated" (Merriam *et al*. 2012, p.278). Finally, behaviourists consider the environment to be a critical factor in student learning (Ertmer and Newby, 2013). Behaviourists believe that environmental factors, rather than individual learner characteristics, determine what is learnt (Merriam *et al*. (2012).

Behaviourism in the Classroom

The teaching styles adopted by any teacher in the classroom are inevitably based on their views on learning (Gould 2012). Therefore, teaching styles adopted by those with a behaviourist perspective will primarily be concerned with delivery methods which bring about a change in behaviour. The key assumptions underpinning the behaviourist perspective are:

- An emphasis on producing observable and measurable outcomes in students [behavioural objectives, task analysis, criterion-referenced assessment]
- Pre-assessment of students to determine where instruction should begin [learner analysis]
- Emphasis on mastering early steps before progressing to more complex levels of performance [sequencing of instructional presentation, mastery learning]

- Use of reinforcement to impact performance [tangible rewards, informative feedback]
- Use of cues, shaping and practice to ensure a strong stimulus-response association [simple to complex sequencing of practice, use of prompts]

(Ertmer and Newby 2013, pp.49-50)

In behaviourism, instruction is centred on the presentation of the stimulus and providing the learners with opportunities to make the appropriate response (Ertmer and Newby 2013). Some of the teaching strategies most commonly associated with the behaviourist perspective are drill and rote learning and programmed learning (Gould 2012). Programmed learning is the breaking down of the learning into smaller chunks and reinforcing these sequentially until the desired outcome is achieved. Therefore, the setting of measurable learning objectives is an important aspect of the behaviourist perspective. Indeed, accountability at all levels of education is closely linked to the behaviourist perspective (Merriam *et al*. 2012). The *No Child Left Behind* legislation and the push from policymakers for evidence-based practices are all indicative of a behaviourist perspective. The behaviourist teaching strategies have generally proven reliable in facilitating learning that involves recall of facts, defining and illustrating concepts, applying explanations and performing a specified procedure (Ertmer and Newby 2013). However, behaviourism has been the subject of considerable criticism partly due to the teaching styles it encouraged and partly due to its perceived limitations in the development of particular aspects of learning. According to Gould (2012) the fundamental objection to behaviourism lies in its attempt to "reduce a complex process such as learning into a relatively simplistic framework" (p.43). In the context of educational technology, behaviourism has been criticised for its view of the learner as a "passive bystander, required only to press the RETURN key", and its emphasis on the development of low-level skills rather than more complex thinking skills (Cooper 1993, p.13). Opportunities for

transfer or generalisation are limited unless the two situations are very similar (Gould 2012). The autocratic, teacher-led, transmission-based style of teaching proposes an identical outcome for all learners and denies these learners their right to self-determination and personal decision making (Vialle *et al*. 2005). Gould (2012) is also critical of this adoption of a one-size-fits-all approach, deeming it unsuitable for providing learning experiences for all children. Gould (2012) and Vialle *et al*. (2005) are also critical of the emphasis behaviourism places on reinforcement, believing that motivation controlled extrinsically by reinforcement can lead to reduced motivation to engage with the learning.

3.2.1.2 The Cognitivist Learning Theory

During the late 1950s and early 1960s, cognitivism replaced behaviourism as the dominant paradigm. This revolution of learning theories was initiated by Noam Chomsky's argument that language acquisition could not be explained purely in terms of conditioning (Erneling 2010). Researchers argued that behaviourism's rigid emphasis on the link between stimulus and response was too simplistic (Harasim 2017). The major proponents of the cognitivist perspective were Gagne, Bruner, Ausubel, Koffka, Kohler, Lewin and Piaget. The complex mental phenomena, deemed unimportant in the behaviourist perspective, are considered critical to the acquisition of knowledge in the cognitivist perspective. Cognitivism is less concerned with the stimulus or resulting response but more concerned with the processing of this stimulus (Gould 2012). Cognitivists share a similar epistemology to behaviourists, objectivism. They, too, believe that knowledge exists independently of the learner. Internal structures store knowledge gained from past experiences. The learner then uses this stored knowledge to interpret new

information that they receive. Learning is therefore conceptualised as "modification of the learner's internal structure, such that a new experience alters the internal structure" (Gayle *et al*. 2009, p.34). Internal processes such as thinking, memory, knowing and problem solving lie at the heart of the cognitive perspective. Cognitivism is underpinned by a number of common assumptions:

- current learning builds upon prior knowledge,
- learning involves processing information to derive understanding and meaning (Cognitive Information Processing),
- understanding is dependent upon establishing relationships between new and existing knowledge,
- relationships between these mental constructs are stored internally as cognitive structures (Schema Theory).

(Harasim 2017; Gould 2012; Merriam *et al*. 2012)

Cognitivism in the Classroom

The cognitive perspective believes that permanent storage of new knowledge

> requires careful organization of data and correlation of new information with existing knowledge so that information to [can] be shifted from short-term to long-term memory.
>
> (Dimitropoulos and Manitsaris 2010, p.167)

The primary objective of cognitivism is to discover and model how a learner manipulates the information they receive to arrive at a specific learning objective (Gayle *et al*. 2009). Some theorists link the ability of learners to develop particular mental constructs with human cognitive developmental stages, for example Bruner and Piaget (Gayle *et al*. 2009). Ausubel and Gagne are two theorists who linked the understanding of mental processes to instruction. Ausubel (1960) suggested that new information needed to be delivered in an orderly manner. He developed the concept

of the ‘advanced organiser’ to prepare learners for the new information. An advanced organiser is an overview or conceptual model of the new information a learner is about to receive. According to Reynolds and Fletcher-Janzen (2007), an advanced organiser

> may be either verbal or graphic, and can take a variety of formats, including overviews, outlines, analogies, examples, thought-provoking questions, concrete models, and figures such as cognitive maps.
>
> (p.63)

Gagne developed a nine-step framework for an effective learning process. These nine steps were: gaining attention, informing the learner of the objective, stimulating recall of prerequisite learning, presenting the stimulus material, providing learning guidance, eliciting the performance, providing feedback, assessing the performance and enhancing retention and transfer (Gagné 1985).

3.2.1.3 The Constructivist Learning Theory

Constructivism emerged in the 1970s and 1980s as educational practitioners and researchers “began to reject the notion that humans could be programmed like robots”, always responding in the same manner to a stimulus (Harasim 2017, p.62). The major contributors to the constructivist theory were Dewey, Piaget, Bruner and Vygotsky. The constructivist theory posits that learning is an active process in which the learners construct their own meaning from experiencing the world and reflecting on their experiences. Knowledge cannot be simply transmitted, rather the student must construct their own meaning from their experiences (Larochelle and Bednarz 1998). New knowledge is shaped by both past experiences and cultural factors. Constructivism differs from the traditional views of education, which consider knowledge construction to be an independent, objective process. Constructivists do not believe that knowledge is mind-independent and that knowledge can be

'mapped' onto the learner (Ertmer and Newby 2013). Instead, they view learning as subjective, internally constructed and socially and culturally mediated (Gayle *et al*. 2009). This view of learning represents a shift from the 'knowledge-acquisition' to 'knowledge-construction' metaphor (Alosaimi 2016). Learners do not transfer knowledge into their memories but build their own interpretations of the world based on their own experiences and interactions (Ertmer and Newby 2013). Therefore, the interactions between the learner and their environment are critical to the constructivist theory. Although constructivism as a learning theory has many different variations, the concept of 'learner-centredness' is fundamental to this learning theory (Alosaimi 2016). In the coming sections the researcher will examine the key theories of constructivism.

Constructivism: Learner-centredness

Learner-centredness demands that the learner rather than the content should be at the focal-point of education. O'Neill and McMahon (2005) suggest that there are three dimensions to the learner-centred construct. The first dimension is associated with learner choice. According to Booyse (2010) learner choice encompasses "choices of teaching strategies, teaching activities, practical implementation or the choice of contexts to connect with the learner's experience" (p.12). The second dimension relates to the nature of learner activity. This dimension centres on the idea that the learner is not passive and must take an active role in their learning (Taylor 2013). The final dimension considers the power relations between the learner and the teacher. This dimension steers away from the didactic role of the teacher and focuses on learner responsibility (Taylor 2013). In their interpretation of learner-centredness Lea *et al*. (2003) suggested that seven principles underpinned learner-centredness:

- reliance upon active rather than passive learning,
- an emphasis on deep learning and understanding,
- increased responsibility and accountability on the part of the student,
- an increased sense of autonomy in the learner,
- an interdependence between teacher and learner,
- mutual respect within the learner–teacher relationship and
- a reflexive approach to the learning and teaching process on the part of both teacher and learner.

(p. 322)

The increased responsibility of the learner for their learning is designed to allow them to engage more deeply with their learning. However, Taylor (2013) warns that learner-centredness is often juxtaposed with teacher-centredness, that learner-centredness does not mean teacher passivity, and that learners should not be left to their own devices and never be taught directly.

Piaget's Theory of Constructivism

Piaget rejected the notion that acquiring knowledge was a passive process. Piaget believed that cognitive development was a progressive reorganisation of mental constructs resulting from both biological maturation and environmental experience. There were three basic components to Piaget's cognitive theory, schemata, adaptation processes and stages of development (McLeod 2024a). Piaget used the term schema to describe how we internally organise and store the new knowledge we acquire (Gould 2012). Piaget considered these schemata to be the building blocks of cognitive models, allowing us to form a mental representation of the world (McLeod 2024a). Piaget (1952) defined schemata as "a cohesive, repeatable action sequence possessing component actions that are tightly interconnected and governed by a core meaning" (p.7). Gould (2012) described schemata as a mental category or unit which holds all the knowledge we possess relating to a particular object or occurrence. McLeod (2024a) describes schemata as "a set of linked mental representations of the

world" (sect.10). These schemata evolve and develop as our cognitive abilities mature. Piaget believed that as our knowledge and experiences of the world increase so too does the number and complexity of our schemata. However, as was previously mentioned, Piaget was one of the theorists who linked the acquisition of knowledge with human cognitive developmental stages. He believed that for children to perceive reason, think abstractly and understand in rational terms, they needed to pass through a series of developmental stages (NCCA 2007c). Piaget believed that during a learner's maturation, they go through a series of cognitive changes. The four stages of this developmental process are:

- Sensorimotor stage: (birth - 2 years) children explore the world and through sensory and motor interaction develop an understanding of object permanence.
- Pre-operational stage: (2 to 7 years) this stage is characterised by a child's egocentric thinking Egocentrism refers to the child's inability to see a situation from another person's perspective.
- Concrete operational stage: (7 to 12 years) marks a major turning point as children begin to think logically, develops concrete reasoning and adopts alternative viewpoints.
- Formal operational stage: (12 – adulthood) development of abstract reasoning and logically test hypotheses.

Learning is an iterative process where the learner interacts with the world and experiences new constructs which produce a sense of mental dissonance. When a learner's existing schemas undergo the necessary cognitive changes and become capable of explaining what it perceives, the learner returns to a state of equilibrium. Gould (2012) describes equilibrium as

> a state of balance exists in which the child's internal cognitive structures (network of schemas) 'fit' with the external environment.
>
> (p.45)

Similarly, Payne (2009) describes the process as mental changes that involve

> the coordination of inner experiences with outer experiences, within the specific community, which would restore the individual to a state of equilibrium.
>
> (p.233)

Unlike the behaviourist theorists, Piaget did not believe that knowledge acquisition was a linear process. Knowledge is not simply absorbed into memory; rather, learners need to construct their own meaning and make sense of the information they receive in terms of their existing frame of reference (NCCA 2007c). In this way, Piaget believed knowledge construction was an internalised and personal process where the emphasis is on the individual (Ng 2015). For this reason, Piaget's theory of constructivism is also known as personal constructivism (Ng 2015). This view of knowledge construction highlights the importance of providing learners with real life examples and negates the use of abstract learning experiences which are removed from the learner's everyday experiences (NCCA 2007c). Piaget's constructivism is based on two fundamental principles of processing and building new knowledge, assimilation and accommodation. This process of adapting current schemas to create a place for new knowledge is illustrated in Figure 3.1. Assimilation is the process of incorporating new experiences into the existing schemas and this occurs when the learner's experiences are aligned with their internal representations of the world. In contrast, when a learner's experiences contradict their internal representations, this needs them to alter their existing perceptions of the world. Accommodation, therefore, involves reframing existing schema to integrate these new experiences into their existing frameworks. According to Piaget's theory, accommodation is the mechanism by which failure, followed by experimentation, leads to learning (Booyse

2010). Piaget's theory of constructivism is an example of 'cognitive constructivism' whereas Bruner and Vygotsky, who gave more emphasis to the social aspect of learning, were considered 'social constructivists'.

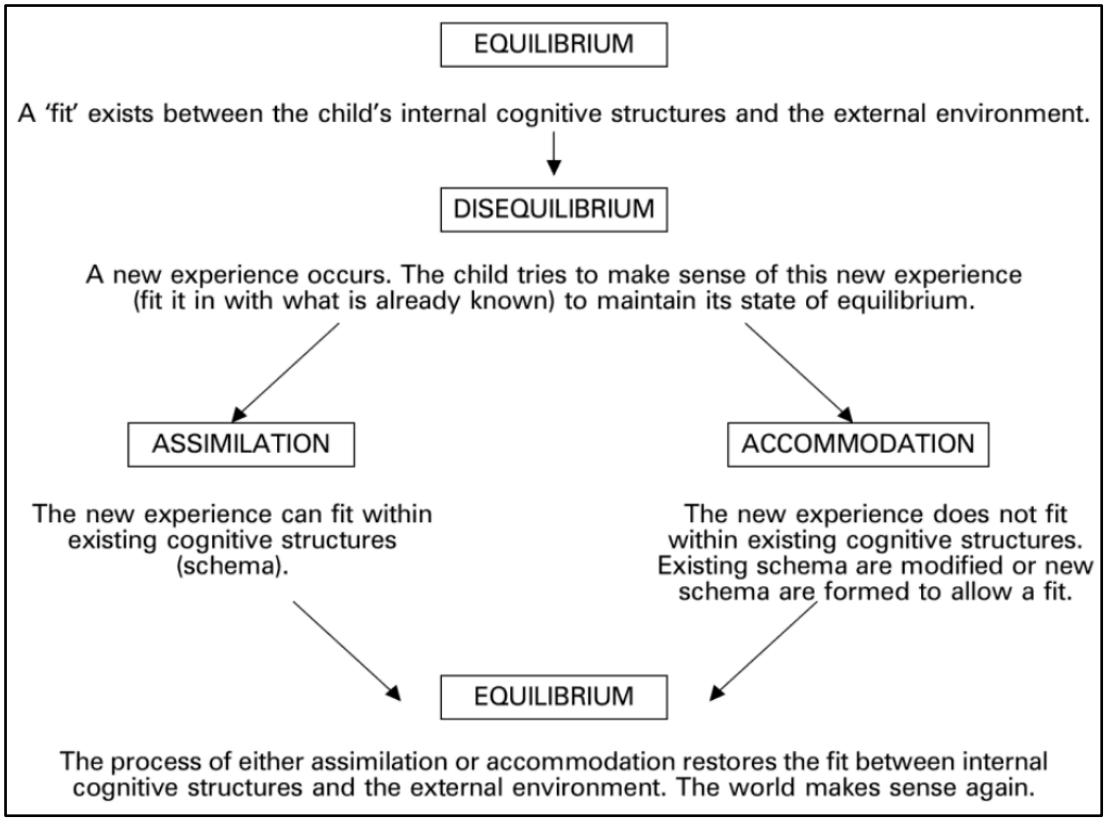


Figure 3.1: *The processes of assimilation and accommodation (Gould 2012, p.45)*

Bruner's Theory of Constructivism

Bruner, like Piaget, viewed learning as an active process in which learners construct their own knowledge from their experiences. Bruner also emphasised the importance of biological constraints on cognitive development (NCCA 2007c). Bruner's theory proposed that children construct knowledge using three modes of representation; enactive representation (action-based), iconic representation (image-based), symbolic representation (language-based) (Ng 2015; McLeod 2024b). According to Ng (2015), Bruner believed that children think through these modes because actions, pictures and words are the modes people around them use to perform tasks and

interact. “Each stage of the cognitive development is characterised by a different way of internalising the representation of the external environment” (Ng 2015, p.81).

In the enactive stage (0-1 years), a child is not able to internalise the external environment and relies on concrete manipulation of objects. McCleod (2024b) suggests that this is why we find it difficult to describe motor tasks in iconic or symbolic form. During the next stage of childhood (1-6 years) iconic representation becomes the more dominant representation. In the iconic stage information is stored visually and the child is able to internalise the external environment as images. McCleod (2024b) proposes that this is why learners often find it useful to have illustrations and diagrams to accompany verbal information. From 7 years onwards the symbolic mode of representation, also referred to as the abstract stage, becomes more dominant. In this stage information is stored in the form of symbol or word. Symbols and words are more flexible than other forms of representation; they can be manipulated, ordered and classified allowing the learner to organise their thinking by connecting concepts (Ng 2015 and McCleod 2024). Ng (2015) believes that, because of their multimodal affordances, digital technologies are useful for reinforcing iconic and symbolic representations of Bruner’s cognitive theory. In contrast to Piaget’s theory, Bruner’s stages are not linear but loosely sequential and interwoven. At different stages of a child’s development, different representations take a dominant role. A child at the iconic stage can use both iconic and enactive representation, and a child at the symbolic stage can function in all three ways. Bruner believed it to be advantageous to progress from enactive, to iconic, to symbolic representation when introducing new material. Following this progression, and with appropriate organisation of learning experiences, Bruner believed that even children of a very young age were capable of learning any material. Therefore Bruner was opposed to

Piaget's notion of readiness, instead believing that structuring learning experiences in a particular way would enable even young learners to grasp complex concepts. Bruner coined the term scaffolding to describe how children build on knowledge and experiences they have previously mastered. According to McKenzie (2000) scaffolding involves the

> continuous sorting and sifting as part of a 'puzzling' process—the combining of new information with previous understandings to construct new ones. Students are adding on, extending, refining and elaborating. It is almost as if they are building a bridge from their preconceptions to a deeper, wiser, more astute view of whatever truth matters for the question or issue at hand.
>
> (pp3-4)

Bruner advocated the use of a spiral curriculum as a means of scaffolding students' learning. According to Chambers *et al*. (2013) use of a spiral curriculum involves introducing ideas at a basic level first and then revisiting these ideas at intervals, expanding and building on them until they are understood at a deeper, more complex level. However, Bruner felt that presenting material in a highly structured manner caused learners to become over-reliant on others (Gould 2012). So, like Piaget, Bruner proposed that learners ultimately must construct their own knowledge by organising and categorising the information they receive using their own coding system (Ng 2015). He, therefore emphasised learning through discovery, describing discovery as "a matter of rearranging or transforming evidence in such a way that one is enabled to go beyond the evidence…to additional new insights" (Bruner 1961, p.22). He suggested that learning for oneself rather being told by a teacher enables "one to acquire information in a way that makes that information more readily viable in problem solving" (Bruner 1961, p.26). Bruner felt that by acquiring knowledge in this way, learners would be better able to apply, generalise and transfer their knowledge to unfamiliar contexts.

Like Piaget and Bruner, Vygotsky emphasised the role of concrete experiences in the learner's cognitive development. However, both Bruner and Vygotsky placed more emphasis than Piaget on the learner's social environment. According to Vygotsky (1978) all learning occurs through social interaction:

> Every function in the child's cultural development appears twice: first between people (interpsychological) and then inside the child (intrapsychological). This applies equally to voluntary attention, to logical memory, and to the formation of concepts. All the higher functions originate as actual relationships between human individuals.
> (p.57)

Bruner's concept of scaffolding also highlighted the social nature of learning and the role of teachers in facilitating and sequencing the learners' experiences. The two main components of Vygotsky's contribution to cognitive theory were, 'the more knowledgeable other' and the 'zone of proximal development'. Vygotsky focused on the development of knowledge through social interaction and emphasised the role played by 'mediating agents' such as teachers (Ng 2015). He defined intelligence as an individual's capacity to learn through instruction from a more knowledgeable other (Johnson 2014). The more knowledgeable other is

> someone who has a better understanding or a higher ability level than the learner, with respect to a particular task, process, or concept.
> (Agarwal and Nagar 2010, p.45)

Although the implication may be that the more knowledgeable other is an older adult or teacher, this is not necessarily the case. The more knowledgeable other can be any individual with more knowledge or experience, a child's peer, a sibling or even computer software. Cicconi (2013) suggests that the advent of the internet created additional means of accessing a knowledgeable other. The concept of the more knowledgeable other is integrally linked to Vygotsky's second important principle, the zone of proximal development. Vygotsky's zone of proximal development is

similar to Bruner's concept of scaffolding. The zone of proximal development refers to

> the distance between the child's actual developmental level as determined by independent problem solving and the child's level of potential development as determined through problem solving under adult guidance or in collaboration with more capable peers.
>
> (Vygotsky 1978, p.86)

In other words, the zone of proximal development is the difference between what a learner could actually achieve independently and what a learner could potentially achieve with support and guidance from more knowledgeable others. Adams and DeVaney (2010) describe it as the "zone of potential for cognitive development" (p. 18). When a learner is in the zone of proximal development providing the appropriate assistance or 'scaffolding' is crucial to push the child to a more sophisticated level of thinking. Once the learner masters the task or develops the relevant understanding the scaffolding can then be withdrawn. Agarwal and Nagar (2010) liken the process to that of an apprenticeship. The novice works with an expert, thereby enabling them to complete skills they wouldn't be able to capable of doing independently. This allows for the internalisation of the shared cognitive processes. Vygotsky's concepts of 'the more knowledgeable other' and 'the zone of proximal development' are important concepts in cooperative learning environments with groups of mixed ability. Vygotsky views interaction with peers as an effective strategy in assisting learners to their potential developmental level. Low achievers or less competent learners can reach higher levels of cognitive function with the help of high achievers in their group (Agarwal and Nagar 2010).

Constructivism in the Classroom

Ackerman (2001) argues that wherever diversity reigns centralised planning or mere transmission won't work. Instructional practices in a constructivist classroom

emphasise the constructivist concepts of active learning, discovery learning, learner-centred collaboration and building on prior knowledge. According to Gayle *et al*. (2009) engaging in learner-centred instruction requires employing a diagnostic teaching strategy to determine a learner's prior knowledge, constructing contextually meaningful knowledge, being aware of learner differences such as stages of development or learning styles. Ertmer and Newby (2013) highlight several assumptions underpinning the constructivist perspective that influence classroom practice:

> - An emphasis on the identification of the context in which the skills will be learned and subsequently applied [anchoring learning in meaningful contexts].
> - An emphasis on learner control and the capability of the learner to manipulate information [actively using what is learned].
> - The need for information to be presented in a variety of different ways [revisiting content at different times, in rearranged contexts, for different purposes, and from different conceptual perspectives].
> - Supporting the use of problem solving skills that allow learners to go "beyond the information given" [developing pattern-recognition skills, presenting alternative ways of representing problems].
> - Assessment focused on transfer of knowledge and skills [presenting new problems and situations that differ from the conditions of the initial instruction].
>
> (p.58)

Baek (2011) propose that there are three basic dimensions to constructivist learning, situated learning, cognitive challenge and collaboration.

3.2.1.4 The Constructionist Learning Theory

Building on Piaget's constructivist perspective, Seymour Papert developed the constructionist theory. In the 1960s Seymour Papert and his colleagues at the Massachusetts Institute of Technology initiated a research study looking at how technological tools could enhance children's thinking and learning. For Papert, digital technology served as a tangible means of constructing knowledge in forms

other than words (Kafai and Burke 2016). They developed a programming language, Logo, which has been foundational in the development of many educational technological tools since then. The theoretical underpinning of their research has become known as constructionism or learning-by-making. Although both Piaget and Papert view learners as constructors of their own knowledge, constructionism develops further on the ideas of constructivism suggesting that students are more likely to develop new ideas while actively engaged in developing some sort of artefact.

> Constructionism suggests that learners are particularly likely to make new ideas when they are actively engaged in making some type of external artifact - be it a robot, a poem, a sand castle, or a computer program - which they can reflect upon and share with others.
>
> (Kafai and Resnick 1996, p.1)

Papert (1990) describes the difference between the two as:

> The word with the v expresses the theory that knowledge is built by the learner, not supplied by the teacher. The word with the n expresses the further idea that happens especially felicitously when the learner is engaged in the construction of something external or at least sharable.
>
> (p.3)

Therefore, the constructionism perspective has two separate dimensions: the construction of an artefact and the social aspect. Ackerman (2001) identifies that three main differences between constructionism and constructivism are:

- the role of external aids at the higher levels of a learner's development,
- the types of external aids studied (Papert focuses on digital technologies), and
- the type of initiative the learner takes in the design of her own artefacts of learning.

Kafai (1995) proposes that in the process of designing an artefact, learners take responsibility for their own learning, through gathering information, asking questions and problem solving. The emphasis on learning-by-making suggests a strong connection between design and learning. According to Kafai and Resnick

(1996), design and learning are not always viewed as related concepts. Traditionally, design theorists were primarily concerned with the final product whereas learning theorists were primarily concerned with the process. However, in recent times the theories of design and learning have begun to converge. Both theorists now emphasise 'construction of meaning' as a key process (Kafai and Resnick 1996). The second dimension of the constructionist theory is the social aspect. According to Kafai and Resnick (2012), Papert, like Bruner, was at variance with Piaget's theory of readiness. He believed that cognitive development was hindered by cultural inadequacies rather than biological maturity. He proposed that the sociocultural context was crucial in the development of internal cognitive structures, particularly when learners were in the zone of proximal development. Resnick (1996) argues that in constructionist theory, collaboration involves more than just the exchange of information. He identified three separate dimensions of collaboration that construction activities supported: discussing constructions, sharing constructions and collaborating on constructions. Learners can discuss their ideas during the design, implementation and evaluation stages. Upon completion learners can share their construction, moving beyond simple discussion and becoming part of the shared knowledge, available for others to copy, edit or develop further. Finally, construction activities support direct collaboration on the design and development of knowledge artefacts. Through these interactions, the learners communicate, share ideas, reflect on their own thinking, clarify differences, try to understand various viewpoints by negotiating a shared meaning and construct new understandings (Bantwini 2015). Papert suggests that alongside sharing our ideas, sharing our inner feelings is also critical to learning. Papert, therefore emphasises the role of affect in constructionist theory (Kafai and Resnick 1996). According to Harel and Papert (1991) learners are

more engaged and motivated in learning if they are constructing an artefact for others to see, critique working on tasks and projects that are meaningful for them.

3.3 Teaching

3.3.1 Pedagogy for Teaching Computational Thinking through Programming

Ball *et al*. (2008) stress the importance of pedagogical knowledge for teaching, asserting that "teaching demands a simultaneous integration of key ideas in the content with ways in which students apprehend them". Up to recently the emphasis of computer education in Ireland was on computer literacy rather than on computer programming. Consequently, there has been little research done on pedagogy for teaching programming in the context of Irish primary schools. In their review of national and international research on computing education, Waite and Quille (2022b) presented some pedagogical guidance for consideration in the redeveloped primary curriculum. Within the literature, they identified three broad approaches to teaching digital technology: direct instruction, exploration, and problem-solving and making. They reported a trend in research and practice towards a blended approach, that incorporates "some direct teaching to introduce concepts and vocabulary along with exploration, problem-solving and making to put concepts into practice" (Waite and Quille 2022b, p.17). With increased emphasis on programming in school curricula, several approaches to teaching programming have gained increased recognition. The Use-Modify-Create model was developed in an out-of-school context by the Innovative Technology Experiences for Students and Teachers (ITEST) working group on Computational Thinking (Lee *et al*. 2011). However, it has been successfully adopted in school contexts (Lytle *et al*. 2019; Boulden *et al*. 2021). Within the context of the computing curriculum in England, two further

approaches which have been developed specifically to help teachers structure their programming lessons are PRIMM (Sentance and Waite 2017) and a continuum of approaches for teaching programming developed by Waite and Liebe (2021). All three approaches recognise the need to employ a diverse range of teaching strategies in order to meet the educational needs of all learners. According to Dede (2008) this understanding is important if researchers hope to develop effective pedagogy for teaching programming.

> To progress, the field of instructional design must recognize that learning is a human activity quite diverse in its manifestations from person to person, and…provide many alternative ways of teaching, which learners select as they engage in their educational experiences.
>
> (p.59)

3.3.1.1 Use-Modify-Create

The Use-Modify-Create (UMC) model is a three-stage model (Figure 3.2), designed to scaffold learners as they gradually progress to increasingly complex interactions that promote the development of computational thinking (Lee *et al*. 2011). In this model the students move from the *Use* stage where they use programs created by somebody else, through the *Modify* stage where they modify (or remix) an existing program so it behaves in a different way, to the final stage, *Create* where they are encouraged to create programs of their own design (Lee *et al*. 2011). The idea behind the UMC model is that students move through the progressively challenging stages as their capacities and skills increase (Lee *et al*. 2011). The developers of the UMC model recommend that students should be allowed to transition back and forth between stages rather than following a strictly unidirectional progression (Lee *et al*. 2011).

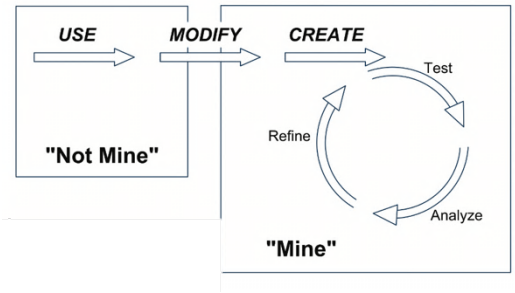


***Figure 3.2**: Use-Modify-Create Model (Lee et al. 2011, p.35)*

3.3.1.2 PRIMM

The PRIMM approach to teaching programming builds on the Use-Modify-Create model but emphasises the importance of language and dialogue to aid understanding (Sentence *et al*. 2019). The developers of the PRIMM model are proponents of Vygotsky's zone of proximal development and his emphasis on language (Sentance *et al*. 2019). Hence, their approach emphasises the role of the teacher in structuring lessons and facilitating discussion, to extend children's learning beyond what could be achieved through exploration (Sentance *et al*. 2019). This approach combines a range of diverse teaching strategies from 'copying code' to 'tinkering' and unguided exploration. PRIMM is an acronym for Predict, Run, Investigate, Modify and Make (Sentance and Waite 2017). The first stage *Predict* requires students to predict what they think a program will do. In the *Run* stage students run the program and test their predictions. In the *Investigate* stage the students examine the code in more detail, examining exactly what each line of code means. Activities include tracing code, annotation of code and identifying individual parts. These first three stages map to the *Use* stage of the UMC model (Sentence and Waite 2017). The fourth stage *Modify* is equivalent to stage two of Lee et al.'s. (2011) model and involves modifying the code to make the program do something different. The final stage *Make* requires the students to design and create their own program and is equivalent

to *Create* in the UMC model. These five steps are designed to facilitate discussion and scaffold learners as they develop their programming skills (Sentance *et al*. 2019).

3.3.1.3 Continuum of Approaches for Teaching Programming

Jane Waite developed the continuum of approaches for teaching programming through her work with primary school teachers (Waite 2019). Her model is a linear continuum of instructional techniques for teaching programming (Figure 3.3). The aim of the continuum is to show teachers the broad range of pedagogies available to them, from copy code right through to unguided tinkering. The six instructional techniques included in the continuum, outlined in Waite and Liebe (2021) are:

- Copy code – Students recreate programs which were designed and programmed by others by following a set of instructions.
- Targeted tasks – are activities which are designed to teach specific programming concepts. These include summarising activities, tracing activities, spot the difference activities and fixing buggy code.
- Shared programming – In this method the teachers (or students) demonstrate how to create a particular program.
- Guided exploration – The teacher provides the students with a number of commands and the students must discover their function through exploration.
- Projects – The students must create a programming project. This can be done in stages, beginning by examining similar projects, then recreating an exemplar and finally independently creating their own project.
- Tinkering – The students are given free rein to explore and play with the technology as they wish.

***Figure 3.3**: Continuum of Approaches for Teaching Programming (Waite 2019)*

The range of instructional techniques included in the continuum enables the blended approach which, according to Waite and Quille (2022b), many researchers and practitioners are adopting in the teaching of programming.

3.3.2 Programming Language

> The tools we use have a profound (and devious!) influence on our thinking habits, and, therefore, on our thinking abilities.
>
> (Dijkstra 1982, p.14)

According to Dijkstra (1982), the choice of programming language can impact the way programmers think. Hence, recent trends to introduce programming to primary school curricula has prompted debate regarding which programming language is best suited to develop the thinking skills of these young learners. According to Weintrop and Wilensky (2017), the programming language chosen can impact the teacher's experience, the class experience, the classroom culture and the pedagogical approaches adopted. The two main types of programming languages are visual and textual. Initial efforts to teach programming in the 70s and 80s relied on text-based programming languages such as BASIC, COMAL, and LOGO (Leahy and Dolan 2014). However, in recent years visual programming languages have been widely adopted in schools, fuelled by the greater availability of user-friendly visual programming languages such as Scratch, Alice and Toontalk.

3.3.2.1 Choosing a Programming Language

Researchers have proposed several factors which educators should consider when choosing a programming language:

- Age appropriateness – Clements (1999) and Benton *et al*. (2017) suggest that spatial, mathematical and problem-solving abilities of students should influence the choice of programming language.
- Support diversity – Resnick *et al*. (2009a) and Romero (2010) both recommend that the chosen programming language should have 'wide walls' i.e., they should appeal to learners with diverse interests, backgrounds and learning styles.
- Enjoyment – According to Merino-Armero *et al*. (2018) higher levels of motivation correlates with more time spent practicing which Cooper *et al*. (2000) found leads to the creation of more sophisticated programs.
- Syntax – Several researchers have identified difficulties in mastering the syntax of programming languages as the greatest obstacle to learning how to program (Pea and Kurland 1984a; Federici and Stern 2011; Kelleher and Pausch 2005).
- Suitability – Papert (1980) and Resnick *et al*. (2009a) both advocate for using programming languages that have 'low floor' and 'high ceilings' i.e., they should be easy to use but they should also enable the creation of complex programs.
- Support Personalization – Both Resnick *et al*. (2009a) and Romero (2010) believe that young learners prefer programming languages which support the use of a range of media types (e.g. photos, voice recordings, music clips, videos and graphics).

The majority of programming languages are textual, meaning their syntax combines words, numbers and punctuation signs, similar to written natural languages. Logo was one of the first programming languages designed specifically to allow children access to programming (Kölling and McKay 2016). Along with BASIC and Pascal, it was one of the most widely used programming languages in classrooms before the advent of visual programming languages. Logo is an example of a text-based programming language consisting of simple commands for movement and drawing. For example, the following command could be used to draw a square:

```
pendown repeat 4 [forward 100 left 90]
```

Logo is best known for its use in directing the motion of a 'turtle'. In the early versions of Logo, children programmed a physical robot (turtle), with a retractable pen, to move about a space drawing a pattern as it went. Later versions replaced the physical robot with a graphical representation of a turtle on a computer screen. However, following reports from several studies which found that the syntax was too difficult for children to master, the use of Logo, and other text-based programming languages, in classrooms declined rapidly. Hence, subsequent efforts to develop educational programming languages have prioritised "simplifying the mechanics of programming" to make them more accessible (Kelleher and Pausch 2005, p131). According to Kelleher and Pausch (2005), they have done so in a number of ways including removing unnecessary syntax, designing languages that are more similar to spoken English, introducing a visual element to languages, and developing alternatives to typed programs.

Programming Language: Visual Languages

Since the turn of the century, a large number of educational programming languages have been released, many of which have been visual programming languages (Kölling and McKay 2016). Visual programming languages allow users to create programs by manipulating visual elements rather than typing code. Several of these visual programming languages are block-based drag-and-drop style languages that have been developed specifically for novice programmers (Weintrop and Wilensky 2017). The explosive proliferation of these visual programming languages indicates a belief among developers that such languages can make programming more accessible (Frädrich *et al*. 2020; Kölling and McKay 2016; Stahlbauer *et al*. 2019). Initially, this belief was based on 'anecdotal folklore' and individual experiences rather than on scientific evidence (Kölling and McKay 2016). However, in recent years, several studies have evaluated visual programming languages against the more traditional textual alternatives (Kölling and McKay 2016; Homer and Noble 2017; Weintrop and Wilensky 2017). These studies identified several benefits of visual programming languages when compared with textual programming languages.

Benefits of Visual Programming Languages

Readability:

Kelleher and Pausch (2005) highlight the importance of allowing

> students to focus on the logic and structures involved in programming rather than worrying as much about the mechanics of writing programs. (p131)

Palumbo (1999), Wilson and Moffat (2012), and Federici and Stern (2011) all argue that difficulties with syntax hinder students from accessing the real learning opportunities which programming can provide. Wilson and Moffat (2012) say that

the lower priority of learning syntax takes precedence because the students have to master enough syntax in order to access the key concepts. Palumbo (1999) found that many students failed to gain the declarative knowledge (syntax) they needed to access the procedural knowledge (problem solving strategies). Therefore, readability should be an important aspect to those choosing a programming language for educational contexts. Weintrop and Wilensky (2017) suggest that block-based programming languages are more readable and reduce the necessity to memorise syntax or commands. Wilson and Moffat (2012) also refer to the cognitive benefit of visual languages which reduce the memory load. Götz *et al*. (2022) support this view in their description of Scratch, acknowledging that as all available coding blocks are accessible in the block palette, this means users "do not have to memorise all available commands" (p.412). Both Lye and Koh (2014) and García-Peñalvo *et* al. (2016) contend that visual programming language commands are more similar to spoken English than those of traditional textual programming languages. Hence, they are easier for novices to learn without becoming frustrated by syntax errors and bugs (Lai and Yang 2011).

Error Avoidance:

According to Kölling and McKay (2016) where possible

> preference should be given to preventing errors over reporting them. If the system can prevent, or work around an error, it should.
>
> (p.8)

Visual programming languages such as Scratch have several features which aim to prevent errors. Scratch will attempt to execute any program by skipping over any erroneous aspects (Homer and Noble 2017). Both Weintrop and Wilensky (2017) and Homer and Noble (2017) identify several visual cues employed by block-based programming languages to aid program composition. These include block colour,

block shape and block syntax, where only compatible blocks will fit together. Similarly, Götz *et al*. (2022) assert that the diverse block shapes in Scratch supports users by only allowing syntactically valid combinations.

Feedback:

> The system should provide timely and constructive feedback. The feedback should indicate the source of a problem and offer solutions.
> (Kölling and McKay 2016, p.8)

Homer and Noble (2017) identify the availability of instant feedback in its graphical microworld as a key benefit of visual programming languages such as Scratch. Users can develop fragments of code and see the effects of their programming unfold immediately. Stacks of blocks can also be modified as the program is running encouraging users to experiment and problem solve.

Engagement and Motivation:

> The system should engage and motivate the intended audience of learners. It should stimulate learners' interest or sense of fun.
> (Kölling and McKay 2016, p.8)

Koh *et al*. (2010) suggest that the most compelling argument for using visual programming languages in education settings is the motivational value. Indeed, many studies have reported high levels of motivation and enthusiasm among users of visual programming languages (Kelleher and Pausch 2005; Kordaki 2012; Maloney *et al*. 2008; Weintrop and Wilensky 2017; Wilson and Moffat 2012). Weintrop and Wilensky (2017) found that students who engaged with visual block-based programming languages showed greater interest in taking further computer science classes than those who engaged with textual programming language. Homer and Noble (2017) report that Scratch has had great success in driving engagement, particularly with younger users. Maloney *et al*. (2008) and Wilson and Moffat (2012)

also reported that Scratch generated great excitement and engagement among novice programmers in their respective studies. Kölling and McKay (2016) believe that the success of Scratch, and other visual programming languages, in generating excitement and engagement stems from its efforts to appeal to the creative interests of their users. Indeed, Maloney *et al*. (2008) found that many children associated Scratch with creative arts subjects rather than mathematics. Federici and Stern (2011) propose that the motivational value of these programming languages lies in "reducing the effort needed for them to achieve interesting results in a very short time" (p.1352). Similarly, Geist (2016) describes these programming languages as more accessible, more rewarding, and less frustrating than traditional programming languages.

Age Appropriateness

According to Jean Piaget's theory of cognitive development, learners aged between 7-11 are beginning to develop operational thinking. Children at this stage are considered capable of logical thought, but their logic is still reliant on physical (concrete) objects. Several research studies have suggested that formal operational reasoning is necessary to succeed in procedural text-based programming (White 2003). However, visual languages involve the manipulation of objects on a screen. White (2003) suggests that

> since this is a concrete component, it may be that those who are pre-formal operation thinkers would be able to handle this challenge of visual objects on a screen and be successful.
>
> (p.100)

Geist (2016) believes that the key to making programming accessible to young children is in ensuring that the programming process is both creative and constructive. She emphasises that graphical programming languages like Scratch

facilitate such a process and likens the drag and drop process to creatively constructing with blocks.

> Just as when playing with blocks and creating towers, children are creating when using codes. The medium may be different, but the creative process is the same.
>
> (Geist 2016, p.300)

All these arguments provide evidence that visual programming languages can be effective in teaching programming to primary school children. Scratch, in particular, has received growing recognition as a suitable programming language for young beginners (Kordaki 2012). Alongside the aforementioned benefits, Scratch is also accessible to all, as it is freely available and sustained by a community of users worldwide (Kordaki 2012). According to Federici and Stern (2011), Scratch is heavily based on the principles of the constructionist learning theory, further enhancing its usefulness as the programming language of choice for this programming initiative.

3.4 Conclusion

Within this chapter, theories of teaching and learning were analysed to determine the best approach to teaching programming in the classroom. Constructionism as a learning theory has a long tradition in computing education (Csizmadia *et al*. 2019), and the benefits of learning programming through constructionist approaches have been widely recognised (Federici and Stern 2011). However, a blended approach to pedagogy has become popular in research and practice (Waite and Quill 2022b), and this approach incorporates aspects from other learning theories, such as Vygotsky's theory of Social-Constructivism. For example, the PRIMM approach, developed by Jane Waite advocates Vygotsky's *Zone of Proximal Development* and his emphasis

on language and discussion. The literature suggests several aspects of teaching and learning that support the learning of programming. The learning context should provide an opportunity to create personally meaningful artefacts while engaging in deep meaningful dialogue, discussing, sharing and collaborating on constructions (Resnick 1996). In facilitating these learning opportunities, the teacher should adopt a broad variety of pedagogical approaches including direct instruction, exploration and problem solving and making (Waite and Quill 2022b). The choice of programming language is instrumental in the success of teaching and learning approaches (Dijkstra 1982). Age-appropriateness, error-avoidance, readability and motivational value were all identified as critical factors in choosing an appropriate programming language. Scratch was identified as a suitable programming language from both a pedagogical and contextual perspective.

Chapter 4: Research Design and Methodology

4.1 Introduction

In the previous chapter the researcher explained the key concepts in this research to provide a clear statement of the theoretical assumptions underpinning this particular research study. In this chapter, the researcher outlines the methodologies used in this study, provides a rationale for choosing those specific methodologies, and discusses their limitations. This chapter also provides details of the data collection methods used, describes how the data from these sources was triangulated and outlines the background information of the participants who took part in the study. Finally, any ethical considerations pertinent to this study are outlined.

4.2 Purpose of this Research

In her 2006 paper, Jeanette Wing proposed that "to reading, writing and arithmetic, we should add computational thinking to every child's analytical ability" (Wing 2006, p.33). Computational thinking has been promoted as a process for solving complex problems in all domains. Wing and other proponents of computational thinking have stressed that "computational thinking is a fundamental skill for everyone, not just for computer scientists" (Wing 2006, p.33). This view of computational thinking as a fundamental skill for all learners has led to increased efforts to include computational thinking at all levels of education worldwide. Ireland is no different. Computational thinking is a central concept in the Junior Certificate short course in coding and the Leaving Certificate Computer Science subject. The NCCA has conducted a review of how computational thinking can be integrated into the primary school curriculum. A review of current literature on the development of computational thinking skills reveals that many educators believe

programming to be a worthwhile approach. The goal of this research is to contribute to what is known about this phenomenon in the Irish context by exploring, critiquing and evaluating:

> In what ways can primary school students develop their computational thinking through a constructionist school computer programming initiative?

4.3 Research Questions

The associated sub-questions are:

- What computational concepts, practices and perspectives do primary school students develop as they engage in a ten-week programming initiative?
- What are the recommended pedagogic approaches for developing primary school students' computational thinking through programming?

4.4 Philosophical Assumptions and Interpretive Framework

Dewey (1938, p.25) believed that in searching for solutions to problems confronted by new educational practices, that "amid all uncertainties there is one permanent frame of reference: namely, the organic connection between education and personal experience". Adopting this philosophical perspective, the researcher was more concerned with understanding "the informal appreciation of experience" rather than "thinking of cause and effect" (Stake 2010, p.56). Thus, research focused on generating descriptions and situational interpretations of the phenomena, which others can then use to modify their understandings of the phenomena (Stake 2010). Therefore, this research relies on *microinterpretation*, that is "how a particular thing works in a particular situation" (Stake 2010, p.39). The researcher explores how

computational thinking can be fostered through a constructionist school computer programming initiative in an Irish primary school. This emphasis on "what works" is fundamental to the pragmatic philosophy (Creswell 2013). Pragmatism emphasises the centrality of the research question (Creswell and Poth 2018), embracing and promoting a flexible approach to research design (Greene 2007). The researcher therefore adopted an interpretive framework based on pragmatism in their approach to this research. This interpretive framework guided the researchers practice throughout the research process, as set forth by (Creswell and Poth 2018),

> the individual using this worldview will use multiple methods of data collection to best answer the research question, will employ multiple sources of data collection, will focus on the practical implications of the research, and will emphasize the importance of conducting research that best addresses the research problem.
>
> (p.27)

4.5 Case Study Research

In determining the most appropriate research design for this study, several factors were considered, namely, the research context (education), the type of research question (how, and in what ways) and the research goal (assess the impact of a programming initiative). According to Freebody (2003), educational research is more challenging than most comparable research designs. One of the reasons for this challenge is that education by its very nature is complex.

> Educational activities are inherently complex and dynamic, both in the local settings in which they occur and, beyond those sites, as part of a society's publicly co-ordinated activities. With changes in the socio-cultural make-up of the participants within the boundaries of an educational site (say, a family, school or state), we observe changes in the qualities of educational goals, outcomes and processes.
>
> (Freebody 2003, p.1)

This complexity makes it inherently difficult, if not impossible, for researchers to implement a true experimental design (Behar-Horenstein and Niu 2011). Indeed, researchers have questioned the suitability and applicability of experimental design

in educational settings (Behar-Horenstein and Niu 2011; Freebody 2003; Kember 2003). Freebody (2003, p.35) suggests that experimental research, "structured through the logics of quantification", overlooks "lots of interesting and potentially consequential things about the phenomenon". Therefore, a research approach that is sympathetic to the uniqueness of the context while providing an in-depth, experiential and naturalistic examination of the phenomenon was deemed most appropriate for this research study. According to Cohen *et al*. (2007, p.256) the strength of case studies "lies in their attention to the subtlety and complexity of the case in its own right", while Bassey (1999) and Merriam (1998) propose that case studies allow for the analysis of phenomena within their real life context. Yin (2009) suggests that the use of case study design is appropriate when investigating research questions which seek to explain an existing circumstance (for example, when a 'how' or 'why' question is being asked about a contemporary phenomenon). While other qualitative research approaches, such as narrative research, phenomenology, and grounded theory, focus on participants' perceptions of a phenomenon (Creswell and Poth 2018), a case study provides insights into process (Merriam 1998; Rose *et al*. 2023; Smith 2018). Merriam (1998) defines process as

> describing the context and population of the study, discovering the extent to which the treatment or program has been implemented, providing immediate feedback of a formative type, and the like.
>
> (Merriam 1998, p.33)

Therefore, in the context of this study, the case study approach facilitates evaluation of the programming initiative. Having considered the particulars of this research study, case study was selected as the most suitable research approach.

> Case study is the study of the particularity and complexity of a single case, coming to understand its activity within important circumstances.
>
> (Stake 1995, p. xi)

Stake (1995) claims that case studies can be further differentiated into intrinsic, instrumental, and collective case studies. Intrinsic case study refers to research which develops from one's intrinsic interest in a particular case. In instrumental case study research, "the researcher focuses on an issue or concern and then selects one bounded case to illustrate this issue" (Creswell and Poth 2018, p.98). The pertinent issue in this instance was the education policy trajectory towards embedding computational thinking and programming in the primary school curriculum in Ireland. The genesis of this research was a desire to investigate what repercussions this might have for girls, a group traditionally marginalised in the field of computing (Vitores and Gil-Juárez 2015). According to Stake (1995, p.3), instrumental case study is employed when we "have a research question, a puzzlement, a need for general understanding, and feel that we may get insight into the question by studying a particular case". Collective case study requires the study of several cases rather than exploring a single case. Wolcott (2008) suggests that single case studies are preferable to multiple cases, proposing that the use of multiple cases diminishes the time and attention that can be devoted to each case. Selection of a single case study has been criticised for being too 'narrow' and therefore, providing little basis for scientific generalisation (Yin 2009). However, the purpose of this research is not to understand other cases, but to gain an in-depth understanding of this particular case (Stake 1995). Patton (2002) acknowledges that while single cases do not yield generalisations, one can learn a great deal from them. Similarly, Stake (1995, p.8) argues that the real objective of case study is particularisation, not generalisation. However, Yin (2009) proposes that while single case studies are not generalisable to populations, they are generalisable to theoretical propositions. Therefore, a single,

instrumental case study was deemed the preferred approach to provide insights into how computational thinking can be developed in primary school students.

4.6 Purposive Sampling

In qualitative research, purposive sampling is commonly employed to select the participants and sites for study (Creswell and Poth 2018). In purposive sampling, the researcher selects participants and sites that provide insights into the research problem and phenomenon (Creswell and Poth 2018). Patton (2014) describes purposive sampling as the strategic selection of information-rich cases, "cases that by their nature and substance will illuminate the inquiry question being investigated" (p.265). Patton (2014) generated a list of different purposive sampling strategies and outlined the differing purposes of each strategy. Included in this list is criterion sampling, the purpose of which is to "seek cases that meet some criterion" (Creswell and Poth, p.159). STEMerg (2016) found that women were greatly underrepresented in the Irish STEM workforce, with less than 25% of jobs that require STEM skills held by women. According to a report produced by the ET 2020 Working Group on Digital Skills and Competences, early experiences, such as those provided in school, are crucial in attracting more women to the STEM workforce (European Commission 2016). Hence, the researcher was particularly interested in how female students would be engaged and motivated by this programming initiative. Therefore, access to female students was a criterion for site selection. As previous research has shown that teacher-student interactions can be impacted by gender stereotypes in co-education settings (Anderson 2023; Vekiri 2010), a single-gender setting was deemed preferable for this study. Once these criteria had been applied, convenience sampling was used to identify a suitable case. Convenience sampling involves choosing sites and individuals from which the researcher can easily access and

collect data (Creswell and Poth 2018). While selecting a case due to its convenience has the disadvantage of self-selection sampling bias (Lavrakas 2008), Stake (1995, p.4) highlights that "good instrumental case study does not depend on being able to defend the typicality of the case". Instead, Stake (1995) advocates for the selection of cases that are hospitable to our inquiry and easy to access.

4.7 Research Participants

This research takes the form of a single instrumental case study conducted in an all-girls primary school in the South of Ireland. Three classes were included in the study, third class, fifth class and sixth class. There were ninety children in the three classes and sixty-seven of them participated in the study (see Table 4.1). At the time of the study, the school had a high percentage of English as an Additional Language (EAL) learners (43% of the participants). The majority of students had little to no programming experience. They worked in pairs (or threes where pairs were not possible), sharing one computer. The groups or pairs were formed based on where the children were sitting the first day of the initiative and they remained stable throughout the initiative. Researchers have studied pair programming in a primary school setting (Denner *et al*. 2014; Gallardo-Virgen and DeVillar 2011) and a wide range of benefits have been reported. These include improved quality of code, decreased time to complete, improved understanding of the programming process, enhanced communication skills and enhanced learning (Preston 2005).

Table 4.1: *Number of participants per class*

Class	Third	Fifth	Sixth
Number of Pupils	23	21	23

The parents of these children were invited to contribute their views on the programming initiative, and the possible introduction of computational thinking and

programming to the primary school curriculum. A total of forty parents agreed to participate and responded to the parent questionnaire. The teachers in the three participating classes and six other teachers from the school also gave their views on computational thinking and programming in the primary school. All nine of these teacher participants were female. As the focus of this study was on the computational thinking development of primary school students, and hence the breadth of the study was already extensive, no further demographic data was collected on either the teachers or parents.

The number of school-owned devices available for use in the initiative was limited. Although the school had forty-four digital devices, many of them were unavailable for use in this initiative. Eighteen laptops and two iPads were designated for teacher or administration use. There were nine iPads designated for use by students, but Scratch was not compatible with iPads at the time of the study. Of the remaining fifteen devices, seven were desktop computers installed in a computer room which was too small to accommodate a full class. This left eight laptops available for use during the programming sessions. In order to facilitate pair programming, a Bring Your Own Device (BYOD) approach was adopted. This approach allowed students to use their own devices for the programming sessions. It is not an approach widely used in Irish primary schools. In the 2013 ICT Census, just 10% of principals at primary level indicated that students were allowed to use their own devices in class (Cosgrove *et al*. 2014a). However, in the Digital Strategy for Schools, BYOD was mentioned as a possible solution for the lack of devices in schools in the Digital Strategy for Schools (DES 2015). In this study, the BYOD approach ensured a student-to-computer ratio of at least 1:3.

4.8 The Programming Initiative

Following discussion with the school principal and teachers it was decided that the programming initiative would take place over a ten-week period, with a one-hour session scheduled during the school day for each class group each week. The initiative was limited to ten weeks due to curricular restraints identified by the principal and teachers involved. This timeframe was considered sufficient as several studies had reported that students could progress their computational thinking skills within an 8-10 week period (Baytak and Land 2011; Wilson *et al*. 2012). The first five sessions focussed on helping the children learn the basics of programming while creating their first animations (Table 4.2). In the following weeks, there was no explicit programming instruction, instead, the sessions were run based on a 'learning on demand' model (Kafai and Ching 2001) as pupils created their own animations. In this model, the students develop ideas of what they want to program, and either ask their partner, another pair or the instructor for assistance. In the final week, the students were presented with design scenarios which they had to investigate and debug.

Table 4.2*: Weekly schedule of activities and the targeted computational thinking skills*

Week	Learning Objective	Activity
1 (31st Jan/1st Feb)	Students will: - be introduced to the concept of **sequencing**. - experiment with a range of Scratch blocks in the Control, Motion (coordinates), and Sound categories.	Create an animation incorporating movement and images.
2 (7th/8th Feb)	Students will: - become more familiar with the concept of **sequencing**. - be introduced to the concepts of **events**, **synchronisation** and **data** (position, direction, size etc.). - practice **experimenting and iterating** while creating projects.	Create a knock knock animation using images, sound and movement.

3 (21st/22nd Feb)	Students will: - develop greater fluency with computational concepts (**events, sequencing** and **data**). - be introduced to the computational concepts of **conditionals** and **parallelism**. - gain familiarity in **reusing** and **remixing** while designing their game.	Create a race game which uses sensing to effect a change in the game.
4 (28th Feb/1st Mar)	Students will: - be introduced to the computational concepts of **looping (forever) and data (variables**). - practice **abstracting** and **modularising** while designing a game for their classmates. - demonstrate greater persistence and creativity in finding solutions to problems.	Create a hungry shark game with sounds, scoring and other effects.
5 (7th/8th Mar)	Students will: - be introduced to the computational concept of **operators**. - develop greater fluency with the computational concepts of **parallelism, conditionals** and **data (variables).** - become more familiar with the computational practices of **experimenting and iterating, testing and debugging, reusing and remixing,** and **abstracting and modularising**.	Build and extend a fruit drop game project using operators and sensors to track lives and keep the score.
6-9 (14th Mar-5th Apr)	Students will: - set out their own goals for a project. - build on all previously learned computational thinking skills.	Plan, create and edit a Scratch project of their choosing.
10 (25th/26th Apr)	Students will: - explore a range of computational concepts (**sequences, events, parallelism, looping, conditionals, operators** and **data**) through the practice of **testing and debugging**.	Investigate a series of design scenarios and develop suitable solutions

Each of the first five sessions followed a similar structure (Table 4.3), moving from exploring pre-existing projects to modifying and extending those projects, mirroring the first two stages of the Use- Modify-Create approach of Lee *et al*. (2011). Each session began with a brief introduction (5 minutes) outlining the new concepts that would be explored in that session. The introduction was followed by a 15-20 minute highly structured activity, either copying code or shared programming. The students then had five minutes to freely explore the newly introduced Scratch commands and twenty minutes to modify these projects During the last ten minutes of each programming session students shared their work. Sometimes this was done in a presentation style where students came to the top and demonstrated their program.

Table 4.3: *Daily structure of activities*

Week	Structure of Activities
1-5	• Introduction – Outlining new concepts (5 minutes) • Structured Activity – Copy Code or Shared Programming (20 minutes) • Free Exploration of new commands (5 minutes) • Modification of pre-existing project (20 minutes) • Sharing of Projects (presentation or gallery walk) (10 minutes)
6	• Introduction (5 minutes) • Storyboard creation (15 minutes) • Final Project Development (30 minutes) • Sharing of Storyboards (presentation) (10 minutes)
7-9	• Introduction (5 minutes) • Final Project Development (45 minutes) • Sharing of Projects (presentation or gallery walk) (10 minutes)
10	• Introduction (5 minutes) • Design Scenarios (50 minutes) • Discussion (5 minutes)

Other times it took the form of a 'walking gallery' where students moved around the classroom viewing their peers' projects. In the next four sessions the students worked on creating their own Scratch programmes, reflecting the 'Create' stage of the Use-Modify-Create approach of Lee *et al*. (2011). In week six after a brief introduction the students had fifteen minutes to plan their final project using the storyboard template. The storyboard structure was adapted from Burke (2012) and included space for both images and description (Figure 4.1). Then they began work creating their own programs. The last ten minutes of this session was set aside for students to present their storyboard to the class. In weeks seven to nine, the sessions consisted of a brief introduction, forty-five minutes of free programming and ten minutes of sharing their work either by group programme demonstrations or gallery walks. In the final week, the students were presented with the design scenarios and they worked on these in their groups for fifty minutes.

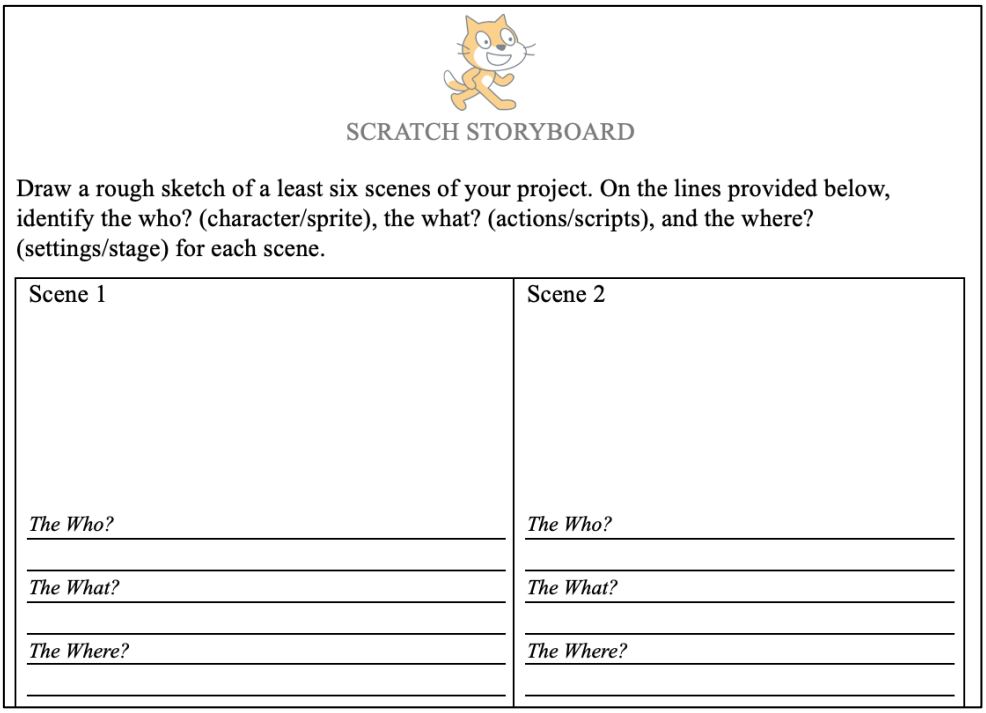


Figure 4.1: *Storyboard for Project Development*

4.8.1 Programming Language

The following criteria are deemed important in the selection of a programming language for beginners, simple and concise syntax and semantics (Brusilovsky *et al*. 1997; Mannila and de Raadt 2006; Parker *et al*. 2006), attractive and meaningful (Brusilovsky *et al*. 1997; Resnick *et al*. 2009a), offers powerful computing capacity (Mannila and de Raadt 2006; Parker *et al*. 2006; Resnick *et al*. 2009a), provides instant feedback (Brusilovsky *et al*. 1997; Homer and Noble 2017; Mannila and de Raadt 2006), contains features which support debugging (Mannila and de Raadt 2006; Parker *et al*. 2006) and has achievable system requirements (Parker *et al*. 2006).

4.8.1.1 Scratch Programming Language

Scratch, a visual programming language, was chosen as the programming tool for the initiative (Figure 4.2). According to Donnelly (2016) Scratch is the most widely

taught coding language in CoderDojos (coding clubs for children). Originally inspired by Papert's work (Papert 1980), Scratch was intended to support creative work with multimedia (Maloney *et al*. 2008) in 'computer clubhouses' or after-school learning centres for children from deprived communities and was first deployed in 2005 (Wilson and Moffat 2012). The focus of Scratch is on making multimedia products and sharing them in the large and active online community hosted by the project website. This is intended to enable and develop children's creativity, but also to introduce them to programming, in a fun way. In Scratch programmes are created by fitting 'blocks' together like Lego or pieces of a jigsaw (Wilson and Moffat 2012). In this regard, Scratch is considered a visual programming language (Green and Petre 1996).

4.8.1.2 Rationale for Choice of Programming Language

There are several features of Scratch that make it a suitable language for teaching programming to beginners. Firstly, Scratch has been described as a low-floor, high-ceiling and wide-wall programming language (Resnick *et al*. 2009a). This means it is easy to learn and operate, has powerful programming capacity and allows for the creation of attractive and meaningful projects for young users. Although van Zyl *et al*. (2016) question whether Scratch is indeed a high-ceiling programming language, they acknowledge it is suitable for teaching introductory programming concepts. In Scratch, the blocks will only connect in 'syntactically appropriate' ways (Kordaki 2012) and the code is highlighted as it is executed providing immediate visual feedback on script execution (Maloney *et al*. 2010). These features eliminate syntax errors and support programme debugging (Maloney *et al*. 2010). Scratch is an accessible programming language, it is freely available and an offline version can be

downloaded which enables users to work on projects without an internet connection. This was important in the context of this study as internet connectivity in the school was described as sluggish and unreliable by the principal and teachers.

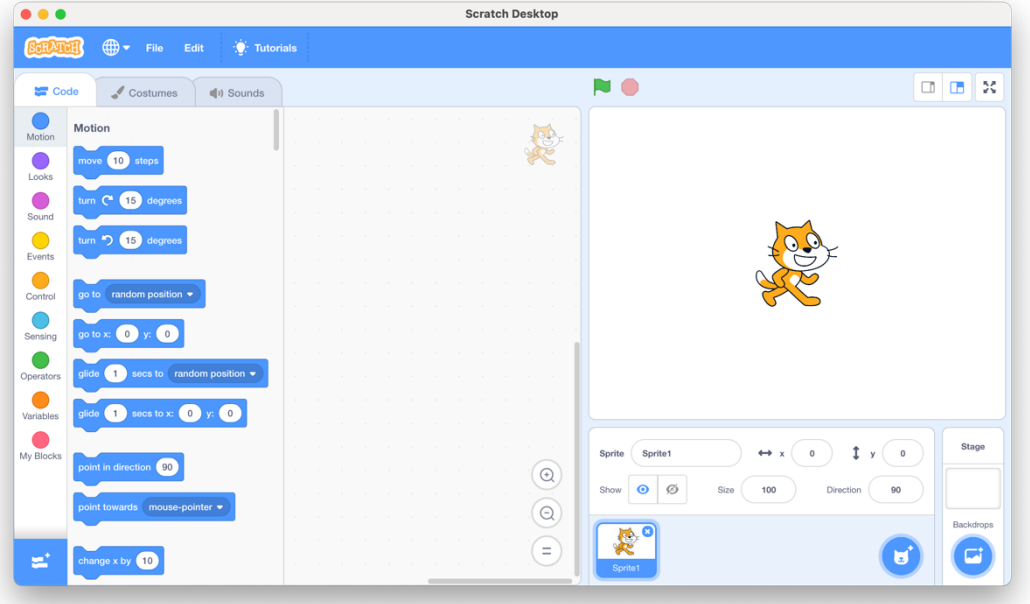


Figure 4.2: *The offline Scratch user interface*

4.9 Conceptual Framework

This study adopts Brennan and Resnick's (2012, p.2) framework as a means of conceptualising "the learning and development that take place with Scratch". Several researchers have proposed that Brennan and Resnick's framework is an appropriate framework for conceptualising the computational thinking developed through visual programming languages (Brennan and Resnick 2012; Falloon 2015; 2017; Kong 2019; Zhang and Nouri 2019). Firstly, this framework was developed in the context of visual programming languages. Brennan and Resnick developed their framework from their observations of, and interviews with, young learners creating with Scratch. Hence, the "framework concentrates primarily on the sort of knowledge used, and how it is used, when students create code" (Falloon 2016, p. 578). This link between coding, and more specifically Scratch, and Brennan and Resnick's framework provides further reason to adopt the framework for this study. Secondly,

the three competence domains outlined by Millwood *et al*. (2018) in their definition of computational thinking, knowledge, craft and character, are encapsulated in Brennan and Resnick's (2012) framework. As observed in the literature review, many studies focused on the development of computational concepts, whereas this framework helps to provide a more comprehensive understanding of computational thinking development. It is acknowledged that both Kong (2019) and Zhang and Nouri (2019) have identified additional computational thinking components, present in the literature, but not included in Brennan and Resnick's (2002) framework. In pursuit of a more comprehensive understanding of computational thinking development, this study adopted a pragmatic epistemological approach in their use of this framework. So, while Brennan and Resnick's three computational thinking dimensions were used to interpret the data, both deductive and inductive tools and analysis were used to determine the nature of the dimensions. Therefore, the findings of this study will be used to build upon and enhance Brennan and Resnick's framework and to evaluate the claims of Kong (2019) and Zhang and Nouri (2019) within the Irish context.

4.10 Data Collection Methods

Several different approaches for assessing the development of computational thinking have been employed including automated tools (Ke 2014; Moreno-León *et al*. 2015), pre-post testing (Jenson and Droumeva 2016; Pala and Türker 2021; Zapata-Cáceres 2024), Parson's problems (Denny *et al*. 2008; Ericson *et al*. 2018), portfolio assessments (Bers et al. 2014; Denner *et al*. 2012), micro-ethnographic methods such as video (Falloon 2016), rubrics (Bonner *et al*. 2021; Seiter and Foreman 2013), questionnaires (Liu *et al*. 2017; Sáez-López *et al*. 2016; Yadav *et al*. 2014) and interviews (Allsop 2019; Barron *et al*. 2002; Brennan and Resnick 2012).

Despite increased efforts in recent years to determine the best strategies to assess computational thinking, consensus has yet to be reached (Brennan and Resnick 2012; Grover *et al*. 2014; Neira *et al*. 2021; Tang *et al*. 2020). Both Lockwood and Mooney (2018) and Hsu *et al*. (2018) have urged the research community to address the uncertainty around the assessment of computational thinking. Several reasons for the diversity in assessment methods are proposed in the literature; the lack of clarity in defining computational thinking (Lyon and Magana 2020; Tang *et al*. 2020), the multidimensional nature of computational thinking (Barron *et al*. 2002; Weintrop *et al*. 2021), the diversity of contexts in which computational thinking is taught (Gane *et al*. 2021; Weintrop *et al*. 2021), and the variety of pedagogic approaches employed (Lyon and Magana 2020; Neira *et al*. 2021; Waite and Quille 2022a; Weintrop *et al*. 2021). Grover (2017), Tang *et al*. (2020) and Weintrop *et al*. (2021) have urged researchers to embrace this diversity and to adopt a multitude of assessment approaches to capture the multidimensional nature of computational thinking. This is in keeping with the pragmatic approach to research, which highlights the importance of combining methods to ensure the most appropriate methods are employed to best answer the research question (Brannen 2016).

> Dewey's pragmatic perspective does not offer a prescriptive mixed methods approach or design per se,…it does offer a description about how methods are to be considered. Pragmatically, mixed methods are used with the understanding that they are being utilized intelligently to attend to a specific problem.
>
> (Hall 2013, p.19)

Therefore, in accordance with the pragmatic approach, a variety of different data collection methods were employed to gain the most comprehensive picture possible of the phenomena being studied (Table 4.4). The data from these different sources were then triangulated to generate strongly supported findings. In the following sections, the assessment strategies adopted in this study are described, and the

rationale for adopting these strategies is discussed. Although micro-ethnographic methods such as video or audio recordings may have yielded further valuable insights, their implementation was deemed impractical due to the substantial time investment required for both data collection and analysis, particularly considering the already comprehensive range of methods employed.

Table 4.4: *Synchronisation of the data collection methods with the initiative phase*

Week	**0**	**1-10**	**11 →**
Phase	**Pre-Initiative**	**Teaching Phase**	**Post-Initiative**
Data Collection	• Pre-Initiative Questionnaire (student)	• Observation Diary (weekly) • Design Scenarios (week 10)	• Post-Initiative Questionnaire (student, teacher, parent) • Artefact Analysis (final projects) • Artefact-Based Interview (student)

4.10.1 Artefact Analysis

A pre-post testing design was not appropriate for assessing the participant's development of computational concepts, as the majority of participants in the study had no previous experience of coding using Scratch. In situations such as this,

> administration of pre-unit assessments for software-dependent instruction is problematic as student fluency requires some degree of familiarity with the software.
>
> (Webb 2010, p.904)

Therefore, data on the development of computational concepts was only collected after they had completed the ten-week initiative. Artefact analysis was the approach adopted to assess participant's development of computational concepts. According to Vivian *et al*. (2017, p.4),

> the production of a programming project can provide an engaging summative assessment activity for students to demonstrate their knowledge and application of skill.

These programming projects can be either structured programming assignments with a single correct solution or open-ended programming assignments. In constructionist-based learning environments, open-ended programming assignments are preferred as they allow for greater learner agency (Alves *et al.* 2019). The programming project is then analysed to assess for the use of specific computational concepts.

4.10.1.1 Artefact Analysis – Automated Tools

Artefact analysis can be a time-consuming and subjective process, particularly if there are a large number of projects involved (Alves *et al.* 2019; Grover 2017). Therefore, the use of automated assessment tools has been recommended (Alves *et al.* 2019; Grover 2017), allowing more time "to focus on the manual assessment of complex, subjective aspects such as creativity" (Alves *et al*. 2019, p.19). One such tool is Hairball, a static analysis instrument which analyses the code in a program without actually executing them. Hairball allows educators to detect programming errors and verify the presence of certain programming constructs (Boe *et al*. 2013). Dr. Scratch is a free web application powered by Hairball that allows educators to analyse scratch projects by assigning a computational thinking score and detecting bugs. Dr. Scratch was chosen as it is easy to use, it has been widely used in computational thinking research, facilitates analysis of open-ended programming projects, and it was found to have strong correlations with manual evaluations by computer science education experts (Moreno-León *et al*. 2017).

4.10.1.2 Artefact Analysis – Dr. Scratch

> Dr. Scratch analyses a Scratch project to assign a CT (Computational Thinking) score depending on the competence demonstrated by the developer on the following seven concepts: abstraction and problem

> decomposition, parallelism, logical thinking, synchronization, algorithmic notions of flow control, user interactivity and data representation.
>
> (Moreno-León and Robles 2015, p.2)

The competence demonstrated is determined by the presence of particular blocks in a project. Dr. Scratch uses static code analysis (examining code without executing it) to detect the presence of these particular coding blocks. Whereas some assessment tools simply identify the quantity of coding blocks present, Dr. Scratch recognises a hierarchy of coding blocks i.e., the use of certain blocks is considered to demonstrate higher levels of computational thinking. Each computational thinking concept is given a score between 0-3, depending on the competence demonstrated in the project. The evaluation of these competences is based on the rules in Table 4.5. The overall computational thinking score is calculated by adding up the partial scores of each computational thinking concept, resulting in the computational score ranging from 0 to 21 points. The final score identifies the project as being at a basic, developing, or mastery level (Moreno-Leon and Robles 2015). Projects with up to 7 points are considered to prove a basic level of computational thinking, while projects between 8 and 14 points are evaluated as developing, and projects with 15 or more points are identified as mastery (Figure 4.3). Moreno-León *et al.* (2017) compared the Dr. Scratch scores for over fifty projects with evaluations of 'Scratch experts' and found a positive correlation between these scores. Moreno-Leon *et al.* (2016) also found positive correlations between Dr. Scratch and both Halstead's metrics, and McCabe's Cyclomatic Complexity (two classic software metrics). These positive correlations validate the assessment process employed by the Dr. Scratch tool and make it a suitable artefact-analysis tool for use in this study.

Table 4.5: *Competence Levels for Computational Thinking Concepts (Moreno-León et al. 2015, p.6)*

Computational Thinking Concept	Competence Level			
	Null (0)	**Basic (1 point)**	**Developing (2 points)**	**Proficiency (3 points)**
Abstraction and problem decomposition	-	More than one script and more than one sprite	Definition of blocks	Use of clones
Logic Thinking	-	If	If else	Logic operations
Data Representation	-	Modifiers of sprites properties	Operations on variables	Operations on lists
Parallelism	-	Two scripts on green flag	Two scripts on key pressed, two scripts on sprite clicked on the same sprite	Two scripts on when I receive message, create clone, two scripts when %s is > %s, two scripts on when backdrop change to
Synchronisation	-	Wait	Broadcast, when I receive message, stop all, stop program, stop programs sprite	Wait until, when backdrop change to, broadcast and wait
Flow Control	-	Sequence of blocks	Repeat, forever	Repeat until
User Interactivity	-	Green flag	Key pressed, sprite clicked, ask and wait, mouse blocks	When %s is >%s, video, audio

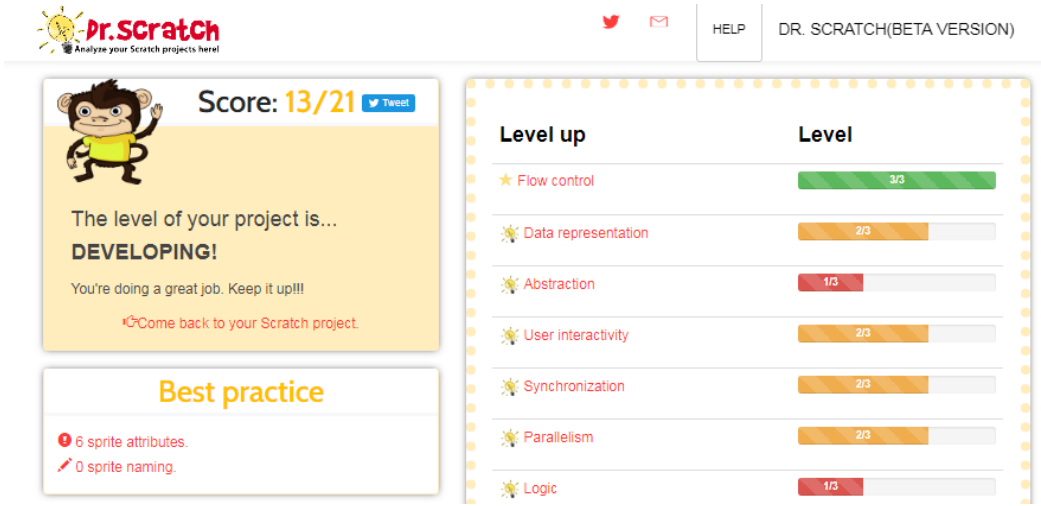


Figure 4.3: *Dr. Scratch feedback on a level 2 developing project (8-14 points)*

4.10.1.3 Artefact Analysis in this Study

In this study, the majority of pupils worked in pairs or threes on their projects. There was one sixth-class pupil who worked on her own. Across the three classes, there

were thirty-two groups altogether, and it was anticipated that each group would submit one final project. However, four of these groups failed to submit a completed final project (three third-class groups and a fifth-class group) due to absences and computer issues, and there were three projects which the Dr. Scratch application was unable to analyse (one third-class project and two sixth-class projects). Therefore, twenty-five final projects were analysed using the Dr. Scratch application (Table 4.6).

Table 4.6: *Final Projects Analysed by Dr. Scratch*

	3rd Class	**5th Class**	**6th Class**
Number of Students in Class	23	21	23
Number of Groups	10	10	12
Groups of three students	3	1	0
Groups of two students	7	9	11
Groups of one student	0	0	1
Incomplete Projects	3	1	0
Final Projects Completed	7	9	12
Final Projects Assessed by Dr. Scratch	6	9	10

4.10.1.4 Limitations of Automated Tools

Some limitations of these automated analysis tools have been acknowledged, including they are product-oriented (Brennan and Resnick 2012; Romero *et al*. 2017), the presence of a specific coding block does not necessarily illustrate understanding of the associated coding construct (Grover 2017; Tegon 2017), snippets of code could be obtained from another source without any consideration for the function of each coding block (Browning 2017), and they do not assess the non-cognitive aspects of computational thinking such as creativity or dispositions (Grover 2017; Tang *et al*. 2020). Grover (2017) and Brennan and Resnick (2012) suggest that these limitations can be addressed by adopting alternative assessment

strategies that can then be used to examine the process alongside the finished product.

4.10.2 Artefact-Based Interviews

The presence of a specific block is not sufficient to confirm understanding of computational concepts (Brennan and Resnick 2012; Vivian *et al*. 2017; Alves *et al*. 2019; Moreno-León *et al*. 2015). The use of a particular block could be due to 'exposure' to the block in previous programming tasks, it could have been incorporated from an existing project, or its use could have been suggested by another person (Brennan and Resnick 2012). Artefact-based interviews can support artefact-analysis by providing further insights into students' understanding of computational concepts (Brennan and Resnick 2012; Vivian *et al*. 2017). Examining the projects with the students provides an opportunity for them to explain how particular segments of code work (Vivian *et al*. 2017; Grover and Pea 2013). Artefact-analysis is also completely product oriented (Brennan and Resnick 2012), meaning it provides no insight into the computational practices employed during the project's development. Artefact-based interviews offer the opportunity to discuss the process of creating projects, hence providing insights into the computational practices that were employed (Barron *et al*. 2002; Vivian 2017). Several studies have proposed the use of artefact-based interviews to assess the development of computational practices (Brennan and Resnick 2012; Grover *et al*. 2015; Hennessey *et al*. 2017). Alves *et al*. (2019) and Barron *et al*. (2002) suggest that artefact-based interviews are useful in the assessment of non-cognitive aspects of computational thinking, such as the ability to collaborate, the tendency to persist when faced with difficulty, and the use of tools to create novel products.

> These are the kinds of learning processes that are not only knowledge-based but involve social competence and emotional resilience. To capture the development of these fluency-related processes we have created an approach to assessment called artifact-based interviewing. This novel method centers on eliciting learning narratives by cueing student memory and perspectives using students' project-based work.
>
> (Barron *et al*. 2002, p.669)

Tang *et al*. (2020) also advocate for greater use of interviews in the measurement of computational thinking, citing its important role in explicating student thinking during the design process. Grover (2017, p.282) found that while summative computational thinking assessments disadvantaged students who had low English proficiency, artefact-based interviews provided a more accurate picture of their developing computational thinking. This was important in the context of this study due to the high number of English as an Additional Language (EAL) learners.

4.10.2.1 Limitations of Artefact-based Interviews

Brennan and Resnick (2012) highlight that interviews can be time-consuming, rely on students' memory of the programming process and are a self-reporting measure. While both Grover (2017) and Tang *et al*. (2020) recommend the use of artefact-based interviews in conjunction with artefact analysis to provide a more comprehensive picture of computational thinking development, they recognise that this might not be practical for larger numbers of students. Lauterbach (2018) acknowledges that interviews have been criticised as they rely on participants' memories and their ability to describe their experiences retrospectively. Furthermore, Austin *et al*. (1998) propose that self-reporting measures can be inconsistent, particularly in the case of children.

4.10.2.2 Artefact-Based Interviews in this Study

Following completion of the ten-week programming initiative, fifteen artefact-based interviews were conducted. The artefact-based interviews were conducted with students in their programming pairs or groups. The interview participants represented a range of classes (third, fifth and sixth) and project types (game, story and animation), as can be seen in Table 4.7. Purposive sampling was used to ensure these criteria are included. The interviews ranged in duration from 15 to 20 minutes.

Table 4.7: *Artefact-Based Interview Participants*

Third Class		**Fifth Class**		**Sixth Class**	
Pseudonyms	***Project Type***	***Pseudonyms***	***Project Type***	***Pseudonyms***	***Project Type***
G1: Sophie & Bethany	Game	G5: Louise & Bernadette	Story	G10: Megan & Sylvia	Game
G2: Maisy, Shauna & Laila	Animation	G6: Eimear & Stephanie	Story	G11: Denise & Michelle (Unanalysed)	Game
G3: Chloe & Maja	Animation	G7: Isabela & Katya	Animation	G12: Anna, Zara & Fiona	Game
G4: Sarah & Tara (incomplete)	Story	G8: Eleanor & Charlie (incomplete)	Game	G13: Amber & Allanah (A)*	Animation
		G9: Katie, Georgina & Valerie	Animation	G14: Kirsten & Laura	Story
				G15: Lisa & Aisha	Story

The interviews were semi-structured and focused on understanding the computational thinking the students engaged in during their project creation. The interview questions were adapted from two instruments developed by Brennan *et al*. (2014), the Student Interview Protocol (Appendix B) and the Rubric for Assessing Evolving Fluency with Computational Practices (Appendix C).

The Student Interview Protocol developed by Brennan *et al*. (2014) was designed to engage students in conversation about their projects and processes at the beginning, middle and end of their Scratch experience. It contains several categories of

questions (defining Scratch, offering feedback, solving problems and the project development process) intended to provide insights into students' developing computational concepts, practices and perspectives. The Rubric for Assessing Evolving Fluency with Computational Practices focuses exclusively on students' developing computational practices. It includes questions that focus on each of the four computational practices (experimenting and iterating, testing and debugging, reusing and remixing, abstracting and modularising) and it provides possible indicators of low, medium and high proficiency levels (Brennan *et al*. 2014). An interview transcript for one of the interviews is included for reference in Appendix D.

4.10.3 Design Scenarios

Design scenarios are projects which have been specifically designed to provide students with the opportunity to critique, extend, debug and remix them (Brennan and Resnick 2012). The design scenarios allow the researcher to observe what computational concepts and practices the participants employed while engaging in programming tasks. This assessment type enables exploration of computational thinking development in real-time rather than relying on students to describe the process retrospectively. According to Lowe (2019) and Ahn *et al*. (2022), the design of these scenarios is critical to their usefulness. Lowe (2019) emphasises that the chosen projects should "represent exemplary code occasionally 'corrupted' with intentional bugs" and "drive authentic interest and experience in wanting to observe the execution of code" (p.8). Ahn *et al*. (2022) highlight the importance of choosing age-appropriate debugging tasks.

4.10.3.1 Design Scenarios in this Study

Design scenarios created by Karen Brennan and Mitch Resnick in collaboration with researchers from the Education Development Center were used in this study (Brennan *et al*. 2014). These design scenarios were chosen as they were readily available on the Scratch website, they had been tested with students across a number of grades, they varied in complexity with respect to the concepts assessed, and they offered choice with respect to program genre to appeal to different interests (Brennan and Resnick 2012). The design scenarios were introduced as projects designed by other students which contained a bug that students needed to identify and fix. The students were presented with a design scenario and given the following tasks:

> (1) explain what the selected project does,
> (2) describe how it could be extended,
> (3) fix a bug, and
> (4) remix the project by adding a feature.
>
> Brennan and Resnick (2012, p.19)

The students worked in their pairs and groups on the design scenarios in the final week of the ten-week initiative.

4.10.3.2 Limitations of Design Scenarios

The use of design scenarios as a computational thinking assessment tool has not been widely researched. Therefore, there is limited evidence available on either its strengths or limitations as an assessment tool. However, Brennan and Resnick (2012) propose several limitations of this approach, including the fact that implementation can be time-consuming and that the use of externally selected projects can negatively impact student interest and intrinsic motivation.

4.10.4 Building on the Assessment Approaches of Brennan and Resnick (2012)

Gibbs (2016) posits that because perspectives are not directly observable, they are more difficult for researchers to access than constructs such as characteristics or behaviours. Indeed, several researchers have highlighted that evaluating the development of computational perspectives is more difficult than evaluating the development of computational concepts and practices (Brennan and Resnick 2012; Lye and Koh 2014; Zhang and Nouri 2019). Table 4.8, adapted from Brennan and Resnick (2012), illustrates how the three aforementioned approaches support the assessment of computational concepts, computational practices and computational perspectives. As evidenced in the table, Brennan and Resnick (2012) found that none of the three aforementioned assessment approaches were particularly effective in understanding students' developing computational perspectives.

Table 4.8: *Connecting the Assessment Approaches to the Dimensions of Computational Thinking (adapted from Brennan and Resnick 2012, p.22)*

	Concepts	**Practices**	**Perspectives**
Artefact Analysis	presence of blocks indicates conceptual encounters	n/a	n/a
Artefact-Based Interviews	nuances of conceptual understanding, but with limited set of projects	yes, based on own authentic design experiences, but subject to limitations of memory	maybe, but hard to ask directly
Design Scenarios	nuances and range of conceptual understanding, but externally selected projects	yes, in real-time and in a novel situation, but externally selected projects	maybe, but hard to ask directly

4.10.5 Questionnaires

Gibbs (2016) reports that researchers commonly rely on participant self-report data, such as questionnaires, to capture perspectives that cannot be directly observed.

Questionnaires allow for the exploration of attitudes, beliefs, values and past behaviours (Menter *et al*. 2011) and can, therefore provide valuable insights into the development of perspectives. Indeed, Kong (2019) proposed that administering questionnaires at different time points can be an effective approach to evaluating the development of computational perspectives. According to Jebb *et al*. (2021), Likert scales are a convenient approach to measure unobservable constructs. Gibbs (2016) asserts that Likert scales are the most commonly used approach to attitude measurement. However, while closed questions such as these are more practical for researchers and more convenient and straightforward for respondents, open questions allow for more diverse responses, enabling more accurate descriptions of respondent perspectives (Gibbs 2016). Therefore, questionnaires containing a mix of open and closed questions were chosen to capture the developing computational perspectives of the students.

4.10.5.1 Limitations of Questionnaires

While Gibbs (2016) acknowledges that questionnaires offer a quick and easy way to gather information on respondent's perspectives, she cautions that self-report data can be notoriously unstable. Brennan and Resnick (2012) who also recognise that social and psychological insights can be gained from self-report data, suggest that additional and complementary insights can be gained from those close to the 'Scratcher', such as a teacher or a parent (Brennan and Resnick 2012). They propose that computational thinking assessment can be greatly enriched by embracing a multicity of viewpoints, including "self, peer, parent, teacher and researcher assessments" (Brennan and Resnick 2012, p.24). The gathering and blending of data

from a variety of sources were thus used to improve the veracity of the self-report data (Creswell and Creswell 2018).

4.10.5.2 Questionnaires in this Study

Prior to the commencement of the programming initiative, students were asked to complete a short questionnaire (Appendix E) to ascertain their prior computing experience and gain insight into their initial computational perspectives. The initial questions on the questionnaire were designed to ascertain the type of technology they used, the purpose of use, and their technological self-efficacy. The remaining questions sought their views on how technology could support creativity and collaboration, and whether they supported its inclusion in the primary school curriculum. Following completion of the programming initiative, students and their parents were asked to share their views on the programming initiative and the inclusion of programming in the primary school curriculum through the use of a post-initiative questionnaire (Appendix F and G). The post-initiative questionnaire revisited students' views on how technology could support creativity and collaboration, and whether they supported its inclusion in the primary school curriculum. Additionally, there were questions developed to garner students' emotional responses to the programming initiative, did they enjoy it, what aspects of the initiative were more/less enjoyable. The parent questionnaire was used to gather data about their child's computer use (device, function, frequency), their child's experience of programming, their own technological self-efficacy, their awareness and understanding of proposed changes to the curriculum (the introduction of programming and computational thinking) and their perception of what their child learned from the programming initiative. The teachers of the classes involved in the

programming initiative (Appendix H), as well as the other teachers in the school (Appendix I), were also invited to share their views on the possible introduction of computational thinking and programming in primary schools. Both teacher questionnaires were used to gather data about their own technology use (device, function, frequency), their technological self-efficacy and their awareness and understanding of proposed changes to the curriculum (the introduction of programming and computational thinking). Like the parent questionnaire, the class teacher questionnaire contained questions pertaining to how they perceived their students' enjoyment and performance in the programming initiative. Brennan and Resnick (2012) suggest that insights into how a programming experience impacts students' computational perspectives can often emerge from parent or teacher data. Hence, both questionnaires contained questions exploring their child/student's experience of the programming initiative.

4.10.6 Observation Diary

According to Stake (1995), observations provide researchers with greater understandings of their case. Creswell and Poth (2018) view observation as a key tool for the collection of data in qualitative research. Similarly, Stake (2010, p.32) emphasises the value of observation to qualitative researchers:

> They [qualitative researchers] favor a personal capture of the experience, so they can interpret it, recognize its contexts, puzzle the many meanings even while still there, and pass along an experiential, naturalistic account so that readers can participate in some of the same reflection.

Brennan and Resnick (2012) highlight the instrumental role that observation plays in providing insights into computational thinking development during the programming process (Brennan and Resnick 2012). As such, it was deemed important to include observation as an additional assessment approach. Observation is the act of "using

the five senses of the observer" to take field notes on a phenomenon at the research site (Creswell and Poth 2018, p.166). During observation, the researcher records activities and events that are relevant to the research question (Creswell and Creswell 2018). There are four observational roles that can be assumed by the observer, depending on the extent of their participation in the research: complete participant, participant as observer, nonparticipant or complete observer (Creswell and Poth 2018).

4.10.6.1 Observation in this Study

In this research study, the observation type employed was participant as an observer, as the researcher assumed the dual role of teacher and researcher (Creswell and Poth 2018). Assuming a participant as observer role familiarises the community with the researcher, reducing the problem of reactivity and enabling the researcher to gain insider views on what's happening in the context (Bernard 2011). The observations were recorded in an observation diary which contained the date, the week of the initiative, and a section for each dimension of computational thinking. The researcher completed a diary entry after each session. A sample diary entry is included in Appendix J.

4.10.6.2 Limitations of Observations

While there are strengths associated with assuming a participant as observer role, there are also limitations. Observation requires time to collect and record data; therefore researchers have to consider how this can be incorporated into their role (Menter *et al*. 2011). When observations are recorded retrospectively, they are not as reliable as if they were recorded in real-time (Menter *et al*. 2011). Additionally,

Stake (2010) highlights the role of interpretation in observation, this can increase the likelihood of researcher bias.

4.11 Data Analysis

Creswell and Poth (2018) describe data analysis as the process of preparing and organising data for analysis, reducing the data into themes and then representing the data through discussion, tables and figures. Padgett (2012) proposes that case study research requires a systematic process of data analysis. Creswell and Creswell (2018, pp.193-195) describe a systematic data analysis process consisting of five sequential steps:

1. organise and prepare the data for analysis,
2. read or look at all the data,
3. start coding all the data,
4. generate a description and themes, and
5. represent the description and themes.

This approach to data analysis adopted in this study as is described in greater detail in the coming sections.

4.11.1 Organising and Preparing the Data for Analysis

The initial phase of data analysis involved the transcription of interviews, the typing up of field notes obtained from design scenarios and observation diaries, and the data from questionnaires. These files were sorted and labelled by data form, participant and date collected, as recommended by Creswell and Poth (2018).

4.11.2 Reading or Looking at all the Data

This phase involved reading and re-reading the data in its entirety to get a general sense of the data before becoming caught up in the process of coding (Creswell and

Poth 2018). During this phase, the review feature of Microsoft Word was used to add notes to the transcriptions and field notes.

4.11.3 Start Coding all the Data

In the coding of the data, a deductive or theory-driven approach was adopted. In deductive coding, the analysis is shaped by existing theoretical or conceptual ideas (Braun and Clarke 2021). The existing theory then provides a lens through which the researcher can read and interpret the data (Braun and Clarke 2021). In deductive coding, the researcher examines the data to determine the extent to which the data fits within codes identified in the existing literature (Bingham and Witkowski 2021). Several researchers support the use of deductive coding in qualitative data analysis. According to Vaismoradi *et al*. (2013), deductive coding is useful when the aim of the research is to examine a previously studied phenomenon in a different context or situation. Bingham and Witkowski (2021) recommend the use of deductive approaches when conceptual frameworks are available to guide data analysis. Miles *et al*. (2020) recommend the use of a conceptual framework to support data analysis in studies where multiple data collection methods are employed, and there is a risk of data overload. Therefore, it was deemed appropriate to begin the coding process using deductive coding (see Table 4.9).

Table 4.9: *Codes adopted in the deductive coding process*

Themes	Sub-Themes	Literature Source
Concepts	• Synchronisation • Parallelism • Data • Thinking Logically • User Interactivity • Flow Control • Abstraction and Decomposition	Moreno-León *et al*. (2015)
Practices	• Abstracting and Modularising • Experimenting and Iterating • Testing and Debugging • Reusing and Remixing	Brennan and Resnick (2012)
Perspectives	• Expressing • Connecting • Questioning • Computational Identity	Brennan and Resnick (2012) Kong (2019)

4.11.3.1 The Deductive Coding Process

During this coding, the researcher drew on the framework of Brennan and Resnick (2012) by initially categorising the raw data (observation diary entries, questionnaires, artefact-based interviews) into the computational thinking dimensions: concepts, practices and perspectives. This data was then further categorised using components of each dimension, as described in the literature. Due to the multitude of computational thinking definitions available, there are subtle differences between the computational concepts identified by Brennan and Resnick and those included in the Dr. Scratch rubric. The components included by Moreno-León *et al*. (2015) in the Dr. Scratch rubric were utilised to further categorise the computational concepts data (synchronisation, parallelism, data, thinking logically, user interactivity, flow control and abstraction and debugging). The Moreno-León *et al.* (2015) components were adopted to support the process of combining this qualitative data with the quantitative results derived from the artefact analysis. While, the names of the constructs do not correlate directly with the computational concepts identified by Brennan and Resnick (2012), they reflect the same aspects of

computational thinking. In Table 4.10 the Moreno-León *et al.* (2015) competencies are mapped to the computational competencies identified by Brennan and Resnick (2012). Abstraction is not included in the Brennan and Resnick (2012) framework but has been included in several other conceptions of computational thinking (Kong 2019).

***Table 4.10**: Moreno-León et al. (2015) Computational Thinking Components mapped onto those of Brennan and Resnick (2012)*

Moreno-León *et al*. (2015) Dr. Scratch Competency	**Brennan and Resnick (2012) Computational Concepts**
Flow Control	Sequences, Loops, Conditionals
Data Representation	Data, Initialisation
Abstraction	*Not included*
User Interactivity	Events
Synchronisation	Conditional
Parallelism	Parallelism
Logic	Conditionals, Operators

The computational practices and computational perspectives data was further categorised using the components of each dimension as described by Brennan and Resnick (2012), augmented with the conceptualisations provided by Kong (2019). An example of the deductive coding process is provided in Appendix K. Once these initial iterations of analysis were completed, each of these units of data were reviewed. This review revealed that while the preexisting categories succinctly described the data relating to the computational concepts, the preexisting categories for both the computational practices and computational perspectives data were too broad and required further refinement. This was perhaps unsurprising given that computational concepts are the most-studied dimension of computational thinking, whereas only a limited number of studies have been conducted on the development of computational practices and perspectives (Vourletsis *et al*. 2021).

4.11.3.2 The Inductive Coding Process

Therefore, once saturation was reached in the deductive coding process, a cycle of inductive analysis was completed to further refine the categories relating to both the computational practices (Figure 4.4) and perspectives (Figure 4.5). Inductive coding was not deemed necessary for computational concepts, as they have been well researched in the past, and the preexisting categories adequately captured the computational concept data. The inductive analysis process involved re-examining the data within these categories, aggregating the data into smaller categories of information, and then assigning a name to the code. Three approaches to naming codes were employed. Codes that emerged that were expected based on the literature were assigned names drawn from the literature. For example, Brennan and Resnick (2012) identified two programming processes that require abstraction and modularising, conceptualising the problem and then translating this concept into code. These code names were adopted as they succinctly described the unit of data. In vivo codes were used when the exact words used by participants best described the emerging concepts. For example, "It kind of helps children become a bit more creative!!!!!" captured the excitement of children, parents and teachers at the creative opportunities provided by the programming initiative. Finally, when neither of these approaches were possible, the researcher composed a code name that best described the data. For example, 'Building on Existing Projects' and 'Idea Generation' described how students engaged in reusing and remixing when creating their Scratch projects.

Computational Thinking Dimension	Computational Thinking Dimension Components	Emergent Codes

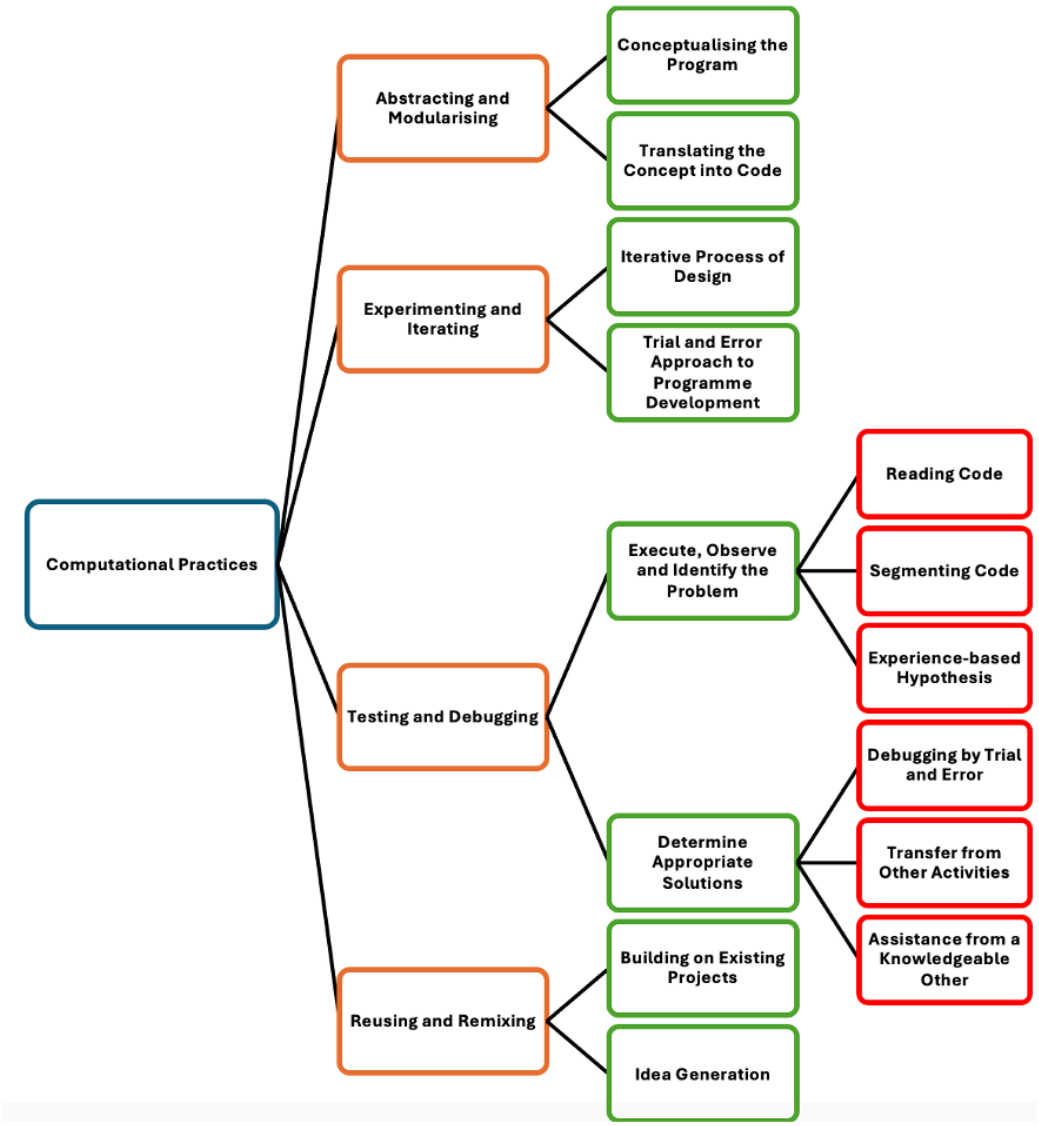


Figure 4.4: *Inductive Codes describing Computational Practices*

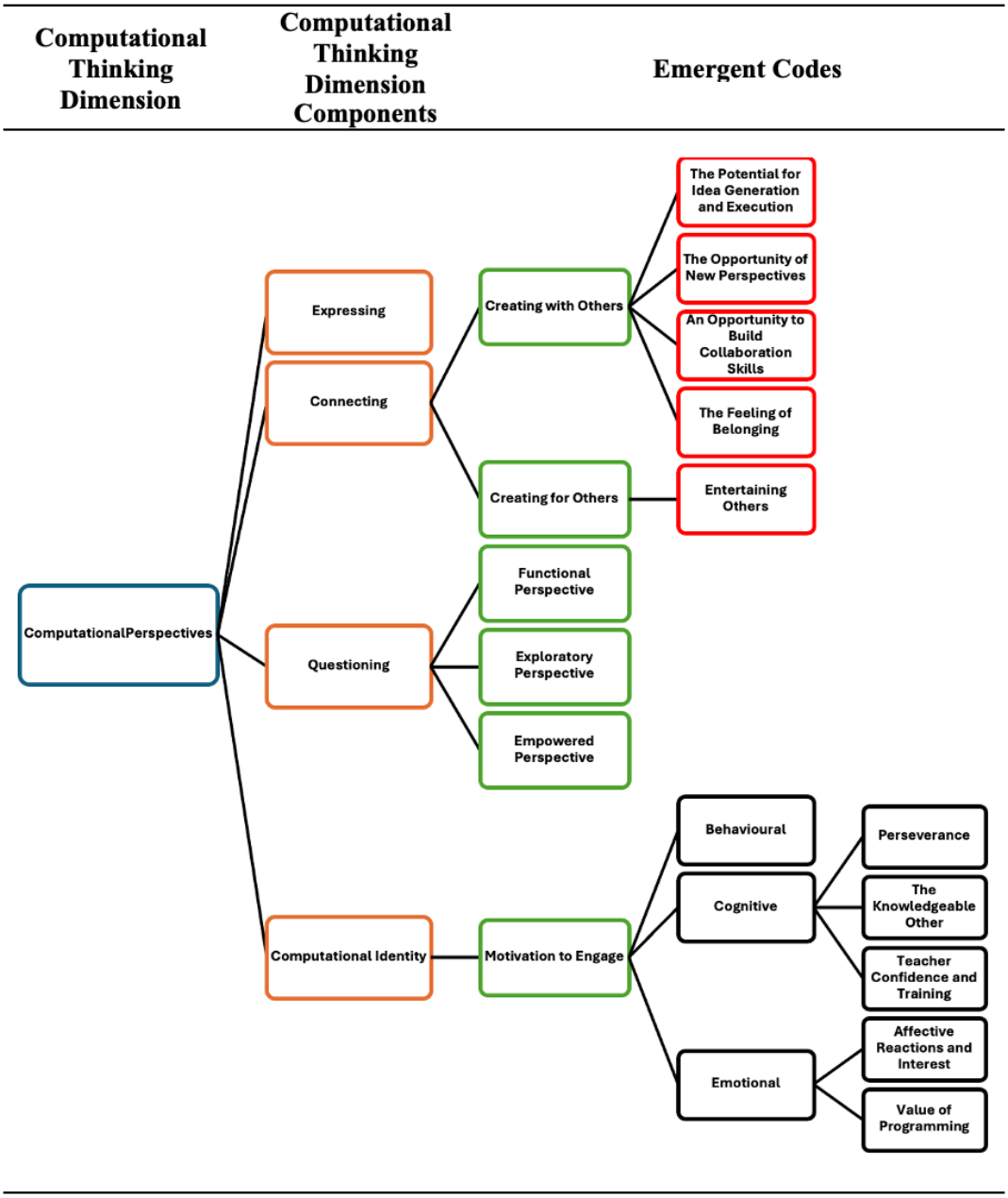


Figure 4.5: *Inductive Codes describing Computational Perspectives*

4.11.4 Generate a Description and Themes

Once the initial coding process was completed, interpretation was required to combine the inductive codes and make connections between the emerging concepts (Creswell and Poth 2018). During this phase, the codes were reviewed to ensure they were an accurate representation of the meanings evident in the data. The establishment of a code was dependent on its occurrence within and across

participants/groups and data sources. The verification of a code required that it be evidenced across five or more participants/groups and two or more data source. An example of the criteria application process is provided in Figure 4.6. Following this the deductive codes and inductive codes were combined to develop a revised conceptual framework (see Table 6.1).

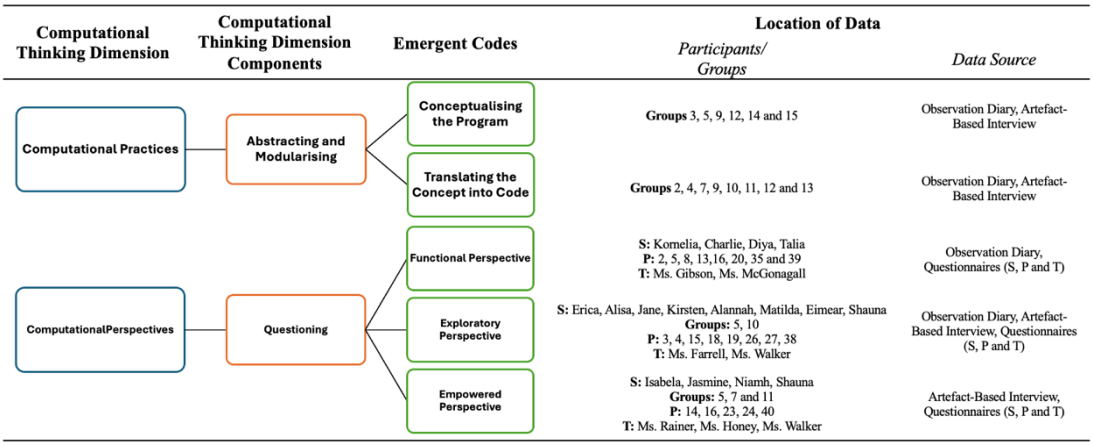


Figure 4.6: *Mapping of Emergent Codes and Data Sources*

4.11.5 Represent the Description and Themes

In this final phase of the Creswell and Creswell (2018) data analysis process the description and themes are represented as a qualitative analysis. In this phase, the inductive and deductive findings were described and interpreted with respect to existing research. This narrative was supported by figures and tables that illustrated the findings of the quantitative data obtained from the artefact analysis (described in section 4.10.1.2) and the Likert scale questions on the questionnaire.

4.12 Validation Strategies

The term validity in qualitative research is contentious (Creswell and Poth 2018). Creswell and Poth (2018) posit that using quantitative terms, such as internal validation, external validation, objectivity and reliability, to describe validation in qualitative studies ‘muddies the water’ and that quantitative language is incongruent

with qualitative work. Several authors have suggested alternative qualitative descriptions of validation that are more compatible with naturalistic research (Lincoln and Guba 1985; Cohen *et al*. 2007; Creswell and Poth 2018). Lincoln and Guba (1985) assert that validation in qualitative research involves establishing credibility, authenticity, transferability, dependability and confirmability. Creswell and Poth (2018) propose nine validation strategies to operationalise these new terms (Figure 4.7). These strategies are organised in three categories according to the lens adopted by the researcher: "researcher's lens, participant's lens, and the reader's or reviewer's lens" (Creswell and Poth 2018, p.259). They recommend that researchers engage in a minimum of two of these validation strategies in any given qualitative study. Due to the qualitative nature of this study, various validation strategies were adopted in this study. These included, corroborating evidence through triangulation of multiple data sources, clarifying researcher bias or engaging in reflexivity, member checking or seeking participant feedback, prolonged engagement and persistent observation in the field and generating a rich, thick description (Creswell and Poth 2018).

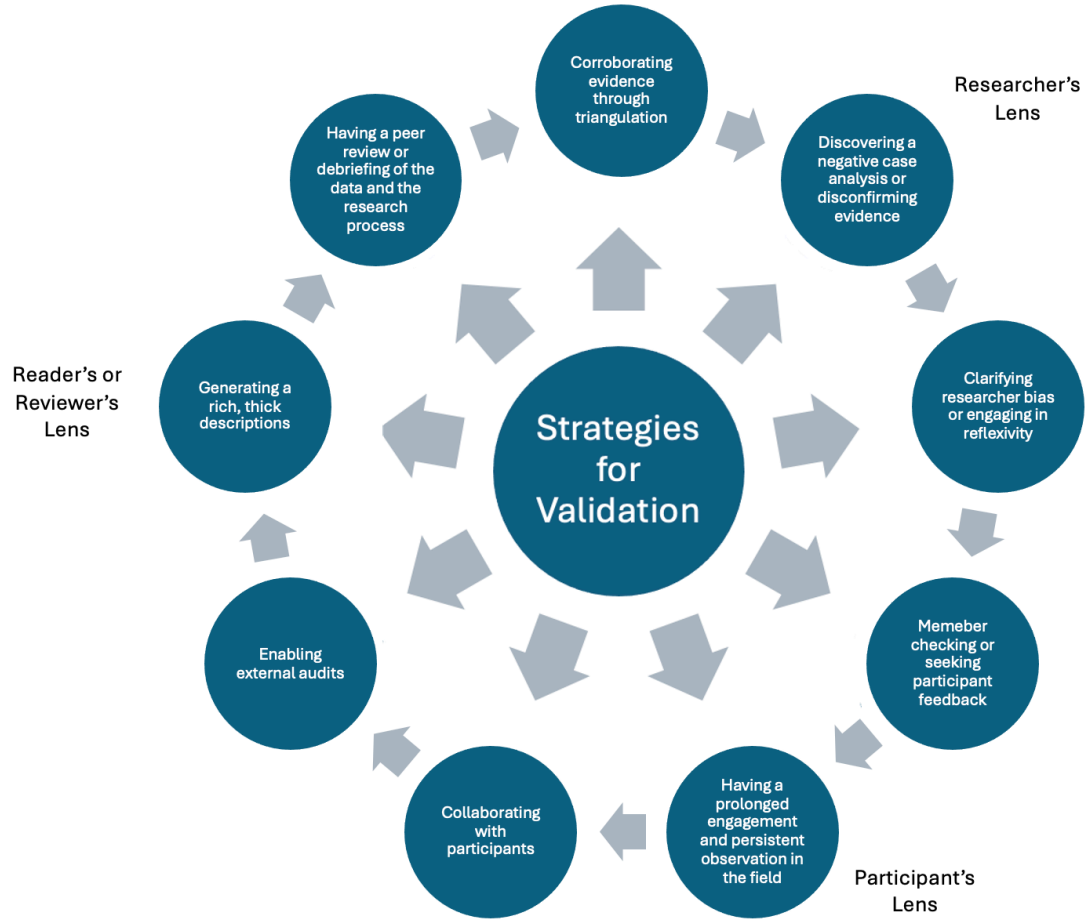


Figure 4.7: *Strategies for Validation in Qualitative Research (Creswell and Poth 2018, p. 260)*

4.12.1 Corroborating Evidence through Triangulation of Multiple Data Sources

According to Creswell and Poth (2018), the researcher ought to avail of multiple methods, sources, investigators, and theories to gather corroborating evidence. Stake (1995) highlights that by using "multiple approaches within a single study, we are likely to illuminate or nullify some extraneous influences" (p.114). Furthermore, Cohen *et al*. (2007) suggests that the more the methods differ from each other the more confident the researcher can be about the findings. During this study, the researcher adopted what Denzin (1978) described as 'between methods' triangulation, employing a variety of different data collection methods (artefact-analysis, artefact-based interviews, questionnaires and observation diary) as a vehicle for cross validation. For example, in determining students' understanding of

computational concepts the researcher cross-referenced the artefact-analysis data with data from the observation diary and the artefact-based interviews to ensure they yielded comparable data. Although, the researcher used the Brennan and Resnick (2012) framework as a lens for this research study, the researcher drew on multiple theories when defining the concepts, designing the initiative, developing instruments, gathering data and during analysis. For example, the researcher also drew on the conceptualisations of computational thinking of Kong (2019), Zhang and Nouri (2019), Moreno-León *et al*. (2015) and Martin *et al*. (2014) during the inductive coding process.

4.12.2 Clarifying Researcher Bias or Engaging in Reflexivity

Creswell and Poth (2018) highlight the importance of researchers disclosing from the outset their understandings about the experiences, values and biases that they bring to a research study so that the reader can understand the researcher's position in the inquiry.

4.12.2.1 Role of the Teacher-Researcher

During this study, the researcher assumed the dual role of teacher and researcher. The motivation for assuming the role of teacher/researcher was twofold. Firstly, it proved difficult to source teacher participants who felt they had the required expertise to teach programming. NCCA (2019a) reported that teachers were apprehensive toward programming and computational thinking and identified several challenges, including teacher mindset, lack of knowledge and professional development. Secondly, studying learning through the teacher/researcher position offered an opportunity to gain unique perspectives that might not have been accessible as researcher alone. Hoong *et al*. (2007) suggest that while researchers can

> "collect data from teaching practice via conventional methods as an outsider, not as an insider…it is doubtful if the kind of information gathered from the outside would be as 'experience-near' (Geertz 1983) as those obtained from insider's accounts.
>
> (p.4)

Wong (1995) suggests that the definition of teacher/researcher can be broad and diverse, therefore it is necessary to describe the specific nature of the teacher/researcher role as it pertains to this study.

In this study, the researcher took responsibility for planning and teaching the sessions as none of the class teachers had prior experience of teaching programming and therefore did not feel comfortable taking on this responsibility. However, the class teachers remained in the room for the sessions providing encouragement and positive reinforcement, thereby they undertook the role of 'champion' or 'supporter', rather than a pedagogic role during the sessions. In preparation for developing and teaching the programming initiative, the researcher completed two online professional development courses in teaching programming to primary school students: the Professional Development Service for Teachers (PDST) workshop 'Scratch for Learning' and the FutureLearn course 'Teaching Programming in Primary Schools'.

Reflexively managing the dual roles of teacher and researcher was a pertinent issue in this research study. The dual role of teacher-researcher is complex, being both causal and observational. Assuming the role of teacher-researcher involves managing the dual commitments associated with each role. As a teacher, one is motivated by one's goals as a teacher, in this case developing my student's computational thinking. However, as a researcher one is motivated to contribute to the discourse on the phenomenon under study, in this case can students develop computational thinking through programming. During the research, the researcher adopted the approach of

Tabach (2011) in balancing the motivating agendas of teacher and researcher. This approach advocates keeping a temporal separation between the two roles and having an awareness of the role adopted at any given time. Prior to teaching, the learning experiences and materials were designed from the researcher perspective, with due consideration to the practical demands of classroom life. During the initiative the classes were taught from the teacher's perspective, "while keeping an 'observer's eye' on things" (Tabach 2011, p.32) with the aid of an observation diary to record interesting occurrences or dialogue. As advised by Tabach (2011), during data analysis the main perspective was that of the researcher. However, it cannot be claimed that the researcher adopted an entirely objective perspective in this role. As Stake (2010, p.164) observes

> Becoming a researcher, especially for a person doing qualitative research, is partly a matter of learning how to deal with bias. All researchers have biases, all people have biases, all reports have biases, and most researchers work hard to recognize and constrain hurtful biases.

With these concerns in mind, particular attention was given to explicitly identifying reflexively the biases, values and personal experiences of the researcher that shape their interpretations during the study Creswell and Creswell (2018). According to Creswell and Creswell (2018) reflexivity requires researchers to attend to their past experiences and how their past experiences shape their interpretations.

4.12.2.2 Reflexivity

Prior to taking the decision to commence the research study, the researcher had no experience of programming or computational thinking, either as a learner or a teacher. The researcher's interest was piqued by the possibility that these concepts might soon be included in a redesigned primary school curriculum (NCCA 2016a). Having no previous involvement in programming or computational thinking meant

that the researcher began the research without any preconceived bias, adding to the rigour of the study. The two courses undertaken by the researcher were both accessible and feasible for primary school teachers in that they were freely available online (one of them was offered as a PDST summer course) and required eight and ten hours study respectively. It was hoped that this would dissuade the teachers from seeing the researcher as an 'expert' and counteract any preconceptions they may have regarding the feasibility of the proposed curricular changes impacting on their view of this study. The researcher adopted Brennan and Resnick's (2012) computational thinking framework as a lens to examine students' evolving computational thinking. This shaped the data collected and the researcher's interpretations of this data. However, during the coding process the researcher leaned on several other conceptualisations of computational thinking e.g., Moreno-León *et al*. (2015) Kong (2019), and Zhang and Nouri (2019), to provide a deeper understanding of the students' computational thinking development.

Throughout the study the researcher embedded opportunities to interrogate connections that emerged between assumptions and theoretical orientations and the researcher's interpretations and perspectives. Reflective memos were written to capture the researcher's personal experiences during these moments. These reflective memos included observations about the teaching, data collection, potential sources of conflict, concerns about reactions of participants and reflections on data analysis. The written memos allowed the researcher to reflect on how the researcher's past experiences impacted the research study. For example, how the researcher's inexperience as a teacher of programming impacted the students' learning:

> Many students experienced difficulty today with variable initialisation, rather than expected difficulty with creating/changing variables. Brennan and Resnick's conceptualisation of data didn't place much emphasis on

> variable initialisation, and I didn't have the teaching experience to predict these difficulties. If I had known this, I would have spent more time on the initialisation of variables in the first few weeks.
>
> (Reflective Memo, 22nd March 2017)

The continuous reflection also enabled the researcher to consider how their relationship with the participants might impact the research.

> Although I am not an experienced programming teacher, there have been some comments from the students and a teacher that suggest they consider me to be an 'expert' and think that Scratch can only be taught by an 'expert'. Perhaps this will impact the computational perspectives they develop in a different way than if the classes had been taught by their class teacher.
>
> (Reflective Memo, 4th Apr 2017)

Although every effort was made to ensure objectivity, as Stake (2010) states, all researchers have biases. However, as efforts were made to recognise and constrain these biases, it is hoped that the inevitable subjectivity can "add to the depth of perception" (Stake 2010, p166) and "contextualize and enrich the psychological research process and its products" (Gough and Madill 2012, p.374).

4.12.3 Member Checking or Seeking Participant Feedback

Creswell and Creswell (2018) recognise member checking as an important strategy in determining the accuracy of qualitative findings. The interview offered a further opportunity to improve the validity of the findings in the form of member checking (Cohen *et al*. 2007; Stake 1995). According to Creswell and Poth (2018), participants in qualitative research can play an important role in validation. The interview was an opportunity to check with the participants of the study the adequacy/accuracy of the analysis and interpretations the researcher had made from the data collected earlier. An example from this particular study would be asking questions about particular blocks of code found in projects. It enabled the researcher to verify if the presence of a particular block of code demonstrated understanding of computational concepts.

The students were also given an opportunity to review the transcripts of the interviews to appraise their meaning and accuracy. In this way participants were able to corroborate the accuracy of the data, analyses and interpretations and add credibility to the narrative account (Creswell and Poth 2018).

4.12.4 Prolonged Engagement and Persistent Observation in the Field

Lincoln and Guba (1985) highlight the importance of prolonged engagement in the field for building rapport with the participants, learning the culture and checking for misinformation arising from distortions formed by the researcher or the participants. Creswell and Poth (2018) suggest spending "as much time in the field as is feasible during the study and prior to beginning data collection" to become familiar with both the site and participants (p.262). Menter *et al*. (2011) suggest that the quantity and quality of data gathered can be enriched when there is a rapport between the participants and researchers. Therefore, several efforts were made to build rapport with participants, including making two classroom visits before data collection began, arriving early on teaching days and chatting informally with the students, and maximising the number of contact hours with the students, in line with what was feasible for the school.

4.12.5 Generating a Rich, Thick Description

The reader's or reviewer's lens is the final lens that Creswell and Poth (2018) suggest should be considered when engaging in the validation process. Cohen *et al*. (2007) say that in qualitative data, "validity might be addressed through the honesty, depth, richness and scope of the data achieved" (p.105). Creswell and Creswell (2018) propose that through detailed descriptions the findings become more realistic and richer, while Creswell and Poth (2018) assert that detailed descriptions allow the

reader to make decisions regarding transferability to other settings. The researcher sought to generate thick description by providing detailed descriptions of the setting (e.g., the computing facilities in the school, class sizes), the data collection methods used (e.g., their use in previous studies, how they were adopted to this study and their possible limitations), and the data analysis (e.g., the deductive and inductive coding processes). By providing such thick descriptions, the researcher puts the findings in context for the researcher to determine to what extent these findings can be transferred to settings, people and situations (Creswell and Poth 2018).

4.13 Ethical Considerations

It is widely agreed that research should improve how things work (Stake 2010). In working to make improvements in education, educational researchers aim to expand knowledge and understanding of all aspects of education and from a variety of perspectives including students, educators, policymakers and wider society (BERA 2011). However, educational research is a very complex and dynamic activity, making it difficult to ensure "that we are 'getting it right' or…that we are avoiding 'getting it wrong'" (Menter *et al*. 2011, p.47). Therefore, when undertaking educational research it is important that researchers "weigh up all aspects of the process…within any given context…to reach an ethically acceptable position in which their actions are considered justifiable and sound" (BERA 2011, p.4). Cohen and Poth (2018) stress that when planning and designing their studies, researchers must anticipate what ethical issues may arise during the study and plan how these issues will be addressed.

In considering the ethical issues that might arise in this study, the researcher considered their responsibility to the participants of the research, to other

stakeholders in educational research, to the field of educational research, and to the community of educational researchers (SERA 2005). During this research study several ethical guidelines were consulted as the researcher negotiated their various ethical responsibilities including, the 'Scottish Educational Research Association: Ethical Guidelines for Educational Research 2005' (SERA), the 'Ethical Guidelines for Educational Research' (BERA 2011) and the 'Guidance for Developing Ethical Research Projects involving Children' (DCYA 2012). The researcher ensured the research adhered to the Mary Immaculate College ethical guidelines, and before beginning the investigation the researcher sought and was granted ethical approval by the Mary Immaculate Research Ethics Committee (MIREC). The researcher recognised that while being granted approval by the ethics committee was important the 'bulwark of protection' must be provided by the researcher themselves (Stake 2010). Ethical procedures which the researcher undertook to protect the participants of the research included minimising risk of harm, informed consent and the right to withdraw, fair representation and confidentiality and anonymity.

4.13.1 Minimising Risk of Harm

"The need to protect children when they are involved in research is self-evident" and therefore, one of the foremost concerns in research ethics is protecting participants from the risk of harm (DCYA 2012, p.vi). While risk can refer to physical harm, Stake (2010) posits that in social research the risks are more often mental harm. These risks can include mental distress, embarrassment, humiliation, loss of standing in a group, loss of respect or self-respect or the stigmatising of particular social, cultural, religious or racial groups (Stake 2010; DYCA 2012). The DCYA (2012) guidelines advocate for a 'minimal risk' standard which specifies that "the anticipated probability and magnitude of harm or discomfort are not greater than

those ordinarily encountered in daily life…" (p.2). During this research study the recommendations of the DCYA (2012) were followed; specifically, evaluating for potential risk, and ensuring measures were in place to mitigate against potential harm. Measures taken to minimise harm were guided by the recommendations of Creswell and Poth (2018) and included, seeking ethical approval for the study, disclosing the purpose of the study, assuring participants that participation is voluntary, obtaining the appropriate consent and assent, having sensitivity to the needs of the children, respecting power imbalances, and adopting appropriate data management.

4.13.2 Informed Consent and the Right to Withdraw

Prior to the study, the researcher contacted the principal and board of management to gain approval to access the school and to study participants. An information letter (Appendix L) was sent to both parties specifying the aim of the research, the proposed data collection methods, the potential impact of the research, and how participants' rights would be protected throughout the research process. The school agreed to participate, and written consent was obtained from both the principal and board of management. Following this, the researcher made a presentation to all teachers in the school, informing them about the study and inviting them to participate. The teachers of the senior classes (3^{rd} to 6^{th}) in the school were approached and their classes were invited to participate. Three teachers agreed to their class's participation in the study. These three teachers and six other teachers also agreed to participate in the study. Information sheets and consent forms were provided and signed by all participating teachers (Appendix M and N).

As the research involved the participation of children, parental and/or guardian informed consent and the child's informed assent were required for their participation in the study. Information sheets informing them of the aims, methods and potential outcomes of the research, and consent forms seeking their permission for their children to participate in the study, were distributed to the parents (Appendix M and N). According to DYCA (2012), if information regarding a study "is presented in a child-appropriate manner and children are supported throughout the decision-making process, then many children will be competent to assent to participate" (p.2). Therefore, the aims, methods and purposes of the research were explained in child-accessible language and the children were asked if they would like to participate (Appendix M). The children were made aware that their participation in the study was voluntary, they could still participate in any activities without being part of the study, and that they had the right to withdraw from the study at any time. In accordance with the recommendations of DYCA (2012, p.3) the children were then "given time to assimilate the information, ask questions and consult with others", including their teacher, classmates and their parents and/or guardians. The children were then provided with consent forms where they recorded their decision to assent/dissent (Appendix N).

4.13.3 Fair Representation and Reporting Honestly

The researcher endeavoured to accurately represent the perspectives of participants in the study. Students were provided with an opportunity to review the transcripts of their artefact-based interviews to clarify meanings or identify inaccurate representations. Creswell and Creswell (2018) warn that "it is easy to support and embrace the perspectives of participants in a study" or "cast the results in a favourable light" (p.94-95). Therefore, the researcher ensured to report multiple

perspectives (see Figure 4.6) and was careful to report the full range of findings, including those that were contrary to the themes and/or hypotheses held by the researcher (Creswell and Creswell 2018).

4.13.4 Confidentiality and Anonymity

Confidentiality implies that data that includes "identifiable information on participants should not be disclosed to others without the explicit consent of the participants" (DYCA 2012, p.3). The researcher had sole access to the data, and all data were stored on a password-encrypted computer. "The principle of anonymity is that individual participants should not be identifiable in research documentation, unless agreed to by the participant" (DYCA 2012, p.3). Therefore, the identity of all participants were protected by assigning pseudonyms and the names of the participants was not stored with the data. In accordance with the Mary Immaculate College Record Retention Schedule the data will be stored for the duration of the project plus three years.

4.14 Conclusion

In conclusion, the research method chosen for this study was the use of a single instrumental case study and this proved to be an effective methodology. It provided in-depth insights into how and in what ways computational thinking can be fostered through a constructionist school computer programming initiative in an Irish primary school. Every effort was made to ensure the validity of the data collected particularly through triangulation of data sources and member checking in the collection of observation data. In the next chapter the researcher will examine the results obtained by this data collection process.

Chapter 5: Findings and Discussion

5.1 Introduction

In this chapter the findings of the programming initiative are outlined, using the Brennan and Resnick (2012) framework to describe the computational thinking skills demonstrated by the children. There were several tools employed to analyse the computational thinking development as outlined in the previous chapter. The findings from each of these tools were combined and are discussed in respect of the three computational thinking dimensions: concepts, practices and perspectives. This answers the first research question: what computational concepts, practices and perspectives do primary school students develop as they engage in a ten-week programming initiative? The emergent themes provided a window into the ways in which the different pedagogical strategies impacted computational thinking development thereby answering the second research question: what are the recommended pedagogic approaches for developing primary school students' computational thinking through programming? Therefore, within each section (concepts, practices and perspectives), consideration is given to the pedagogic approaches that supported or hindered students' computational thinking development.

5.2 Computational Concepts

5.2.1 Computational Concepts - Introduction

Computational concepts are the first dimension of computational thinking identified in the Brennan and Resnick (2012) framework. Computational concepts are the programming concepts that users employ when engaging in programming activities. Dr. Scratch, the artefact analysis tool used in this study, inspects the source code of

each project and assigns a score ranging from 0-3 based on the level of competency demonstrated for each computational thinking concept (see Table 5.1). The results from Dr. Scratch are then combined to provide an overarching view of the computational concepts that the children employed in their final projects, by assigning each project an overall computational thinking proficiency score, and as such provides a good starting point for analysing computational concept development.

Table 5.1: *Dr. Scratch Scoring Assignment for Computational Concept Development*

Computational Thinking Concept	**Competence Level**			
	Null (0)	**Basic (1 point)**	**Developing (2 points)**	**Proficiency (3 points)**

Twenty five projects final projects were analysed. The projects ranged from fruit drop and hide and seek games to animated stories of classics such as Alice in Wonderland, and original animations. All of the projects analysed were evaluated as developing, with scores ranging from 8-12 (see Table 5.2). The mean computational thinking score was 10.2 and the median score was 10. These findings compare favourably with findings from other studies. For example, Troiano *et al*. (2019) reported that the 8th grade students (aged 13-14) in their study also showed a developing proficiency, albeit with a higher median score of 14. As can be seen in Table 5.3, the Dr. Scratch scores enabled the researcher to identify strengths and weaknesses in students' understanding of particular computational concepts. The data indicated that the students excelled at synchronisation and parallelism, with many of them achieving the highest score (3) for this concept. The results also identified that students achieved the lowest mean scores for the concepts of logic and abstraction and problem decomposition (0.32 and 1.0 respectively).

Table 5.2: *The Dr. Scratch scores for individual Computational Concepts*

Concepts	All Classes Mean Score	3rd Class Mean Score	5th Class Mean Score	6th Class Mean Score
Synchronisation	2.8	2.3	3.0	2.9
Parallelism	2.6	1.7	3.0	2.8
Data Representation	1.04	1.0	1.0	1.1
Logic	0.32	0.7	0.2	0.2
User Interactivity	1.2	1.5	1.1	1.1
Flow Control	1.28	1.5	1.2	1.2
Abstraction and Problem Decomposition	1.0	1.0	1.0	1.0
Total	10.24	9.7	10.6	10.3

The sample size is too small to ascertain if factors such as age or project type had any impact on the computational thinking score (only six third class projects analysed by Dr. Scratch). However, while it cannot be determined if the differences in scores are statistically significant, it is interesting to note that while the third class group had the lowest mean score (9.7), the fifth class had a higher mean score (10.6) than the sixth class group (10.3). There was only a very slight difference in computational thinking scores between the project types, with the mean scores for games and stories (10.3) slightly higher than for animations (10.2). In the following sections, the scores for the individual computational concepts are presented and explored in relation to the relevant theory.

Table 5.3: *Project Description and Scratch Scores for the Analysed Final Projects*

Project Title	Pseudonyms	Type	Class	Computational Concept							
				Flow Control	Data Representation	Abstraction & Problem Decomposition	User Interactivity	Synchronisation	Parallelism	Logic	Total
Dunken Dounuts	Sophie & Bethany	Game	3rd	2	1	1	2	2	1	1	10
Hide and Seek	Kornelia & Anna	Game	3rd	2	1	1	2	2	1	1	10
Joke on a Bordwalk	Diya & Erica	Animation	3rd	1	1	1	1	3	3	0	10
Puppy Love	Maisy, Shauna & Laila	Animation	3rd	1	1	1	1	2	1	1	8
Name in Lights	Chloe & Maja	Animation	3rd	2	1	1	2	2	1	1	10
Hippo and Taco	Paula, Millie & Jasmine	Animation	3rd	1	1	1	1	3	3	0	10
Homless Dancers	Jane & Linda	Animation	5th	1	1	1	1	3	3	0	10
Halloween	Alisa & Joan	Animation	5th	1	1	1	1	3	3	0	10
Funky Road Ride	Suzy & Darina	Animation	5th	1	1	1	1	3	3	0	10
Get off the road	Isabela & Katya	Animation	5th	1	1	1	1	3	3	0	10
Fetch	Georgina, Valerie &Katie	Animation	5th	2	1	1	1	3	3	1	12
Monkey Competition	Matilda & Caroline	Animation	5th	1	1	1	1	3	3	0	10
Dino & Peggy	Miriam & Amy	Animation	5th	1	1	1	2	3	3	0	11
The Pig Chase	Stephanie & Eimear	Story	5th	1	1	1	1	3	3	0	10

Alice in Wonderland	Louise & Bernadette	Story	5th	2	1	1	1	3	3	1	12
Ding Dong Ping Pong	Megan & Sylvia	Game	6th	2	2	1	2	2	1	1	11
Dragon War	Kirsten & Laura	Animation	6th	1	1	1	1	3	3	0	10
Wizards and Witches	Evelyn & Talia	Animation	6th	1	1	1	1	3	3	0	10
Under da sea	Amber & Alannah	Animation	6th	1	1	1	1	3	3	0	10
Crazy Concert	Amelia & Niamh	Animation	6th	1	1	1	1	3	3	0	10
Ball Burst	Anna, Zara & Fiona	Animation	6th	2	1	1	1	3	3	1	12
Going to the woods	Nikole & Rachel	Story	6th	1	1	1	1	3	3	0	10
Perfect Night	Jane & Kelsey	Story	6th	1	1	1	1	3	3	0	10
Visit to Santa	Rhianna & Amie	Story	6th	1	1	1	1	3	3	0	10
Unicorn Tea	Lisa & Aisha	Story	6th	1	1	1	1	3	3	0	10

5.2.2 Synchronisation – Making things happen as we want

Synchronisation refers to "coordinating the actions of multiple sprites" (Funke *et al*. 2017, p.1232) to ensure flow in the execution of a program. The Dr. Scratch application assesses for synchronisation at three levels. Scoring for the synchronisation concept can be seen in Table 5.4. One point is assigned for including timing synchronisation which is assessed by checking for the presence of a *wait* block. The *wait* block is the easiest way to synchronise the actions of sprites. A *wait* block instructs the sprite to wait for a specified time while another action is occurring. Two points are assigned for either event synchronisation or for using a control block to stop scripts from executing. Event synchronisation involves the use of the *broadcast* and *when I receive* blocks. There are three control blocks for stopping scripts: *stop all*, *stop this script* and *stop other scripts in sprite*. Three points are assigned for state synchronisation which involves coordinating actions by waiting for a particular condition. The three blocks related to state synchronisation are *wait until*, *when backdrop switches to* and *broadcast and wait*. The highest score, three, was achieved in 80% of the projects (20 of the 25). The five remaining projects scored two for this concept. Manual examination of the projects revealed that all twenty five projects included at least one use of the *wait* block and the *broadcast* and *when I receive* blocks. This illustrated that all students demonstrated some understanding of both time synchronisation and event synchronisation. The twenty projects who scored three, also demonstrated an understanding of state synchronisation, through the inclusion of the *wait until* block (ten projects) and/or *when backdrop switches to* block (fifteen projects). None of the projects included the *broadcast and wait* block, the other block considered to illustrate an understanding of state synchronisation. Three of the five projects who received a lower score for

synchronisation were games. No game received a score of three, suggesting that animations and stories are more suited to developing and assessing students' understanding of synchronisation than games.

Table 5.4: *Dr. Scratch scores for the synchronisation concept*

Computational Thinking Concept	**Competence Level**		
	Basic (1 point)	**Developing (2 points)**	**Proficiency (3 points)**
Synchronisation	Wait	Broadcast, when I receive message, stop all, stop program, stop programs sprite	Wait until, when backdrop change to, broadcast and wait
Number of Projects	0	5	20

Observations recorded in the researcher diary indicated that the students required assistance in developing these understandings, initially struggling with time synchronisation, in particular. Observations from the classroom revealed that one of the difficulties the children had was understanding just how quickly the computer executed their code. Miriam, Anna and Zala (3rd class) assumed that their code was not working because they did not see their character's costume changing, but it was actually changing so fast that it was imperceptible. This illustrated how much the students were relying on the visual feedback Scratch provided and, in its absence, frailties in their understandings were revealed. In this particular instance their Scratch program didn't support them in identifying the omission of the *wait* block, and consequently they struggled to troubleshoot (Figure 5.1).

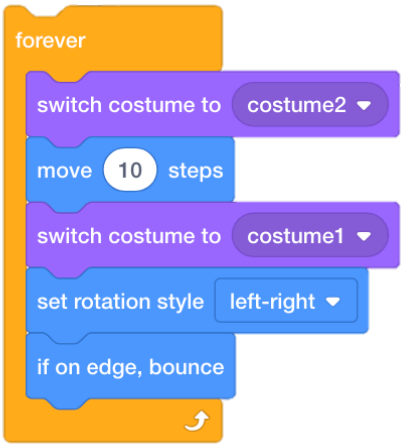


Figure 5.1: *Code illustrating misunderstandings relating to time synchronisation*

Data from the artefact-based interviews supported the Dr. Scratch findings that students had developed a good understanding of the synchronisation concept by the end of the programming initiative. Georgina, Valerie and Katie (5th class) created an animation which included a dog fetching a ball. The following extract from their artefact-based interview illustrates Katie's understanding of time synchronisation, which they used to create animation delays.

> Researcher: What does this block do?
>
> Katie: That makes the ball stop moving for a second…am…actually not even a second.
>
> Researcher: And why did you include that block?
>
> Katie: It's just cos…cos the computer does things really fast and we couldn't see it (the ball)…am…we could only see it when it got to the back. This (points to the wait block) slows down the computer so our eyes can keep up.

Third class students Sarah and Tara also discovered the importance of time synchronisation while creating a dancing ballerina project. Again, it took them some time to realise why it initially appeared like their code was not working.

> Tara: We made a ballerina project. At first we didn't know why the girl didn't move after like one step and then I just knew…because it happened before for our Knock Knock.

> Researcher: What did you know?
>
> Tara: She wasn't doing only one step…no she was doing all the steps together really fast. That's why we couldn't see her moving.
>
> Researcher: So what did you do?
>
> Tara: What you have to do is add a 'Wait 1 sec' or something in-between the steps. You see everything actually did work, it was only running too fast for us to see!

Students from both fifth and sixth class illustrated that they understood the most complex form of synchronisation. In her artefact-based interview, Aisha (6th class) explained how they used the *when backdrop switches to* block to trigger music at the beginning of their story.

> Researcher: How did you get the music to play at the start of your project?
>
> Aisha: It was easy. We wanted to have the disco backdrop at the start cos the two unicorns met at the disco and there had to be music at the disco. This block (points to *when backdrop switches to* block) allowed us to decide the music played from the start.
>
> Researcher: So how does this block work?
>
> Aisha: Whatever you put below it happens after the background changes…but you need this block too (points to *switch backdrop to* block). This one changes the backdrop to our starting one.

Louise and Bernadette (5th class) also used this block to switch between scenes in their Alice in Wonderland story. In her interview Louise explained how using this block ensured their story flowed as they wanted.

> Researcher: I am interested in why you included this block in your program. Can you tell me what it does?
>
> Louise: Am we had lots of different places where we wanted Alice to go to, like the…am…the garden and…am…down the rabbit hole. We put this (points to *when the backdrop switches to* block) here so that they all (points to sprites) knew when they had to like appear and start talking and stuff.

Park and Shin (2019) suggest that Scratch is a suitable programming language for the development of synchronisation skills. The findings of this study and prior

research on this programming concept seem to support their assertion. The high levels of understanding of this concept among the students in this study was similar to the findings of Hoover *et al*. (2016) and Fields *et al*. (2015) who reported that their students (middle school and high school) demonstrated complex synchronisation strategies. Moreno-León and Robles (2014) who had a larger sample size (100 randomly chosen projects downloaded from the Scratch repository) than both these studies, also found that Scratch users scored well for this concept. The mean score (> 2) was higher than that achieved in data representation, logic and user interactivity. Similarly, Lawanto (2016) who analysed 360 Scratch projects of seventh and eighth grade students, also found that synchronisation was a programming strength among their students. These students received an average score of 2.69. However, it is noteworthy that all of the previously cited studies which reported higher levels of understanding in respect of this concept were conducted among middle school students, high school students or users whose demographic is unknown. No studies conducted with primary school students reporting high levels of understanding of this concept were identified. Indeed research with younger participants reported fewer positive findings. Several researchers reported limited use of this computational concept in participants' projects. Maloney *et al.* (2008) who examined 536 projects created by youths (8-18) at a computer clubhouse in Los Angeles found that relatively few projects (24.7%) contained the synchronisation concept and that the use of this concept decreased from year one to year two of the project. Some studies suggest that while a score of 1 is achievable for students of primary school age, the understandings required for scores of 2 and 3 are beyond many of these students. Wilson *et al*. (2012) who assessed the projects of 60 students (aged 8-11) found that while 72% of students included synchronisation in their

project, the majority of usage was timing synchronisation, with only one student including event synchronisation. Interestingly, the students in their study were engaged in game design, and this type of project received lower scores than stories or animations in this study. Seiter and Foreman (2013) suggest that younger students struggle with state synchronisation, identifying the concept behind the *wait until* block as particularly difficult for primary students due to its abstract nature. However, it was present in 44% of the projects in this study.

5.2.3 Parallelism - Doing Two Things at the One Time

In programming, a thread is like a mini-program within a program, that can run at the same time as other mini-programs. A program with multiple threads can do multiple things at once (parallelism). When programming either games or stories, it is useful to separate threads for conceptually distinct tasks. In Dr. Scratch, projects achieve a score of one for parallelism when two or more scripts start when the green flag is clicked. Two points are assigned to a project when two or more scripts are started simultaneously by the user in the middle of program execution, for example starting when a key is pressed or when a sprite is clicked. A score of three points requires that one script impacts the behaviour of another script, while they are running in parallel. Two scripts running simultaneously on any of the event blocks – *when backdrop switches to*, *when loudness > when I receive*, or use of the *create clone* – block yields a score of three for this concept.

The students tended to use the computational concepts of synchronisation and parallelism together, as many of them wanted two or more events or actions to occur simultaneously. Consequently, in this study the high scoring of the students in synchronisation was closely related to high scoring in parallelism. The same twenty

projects which received the highest score for synchronisation also achieved the highest score for parallelism (see Table 5.3). The remaining five projects scored one for this concept. The games achieved lower scores in parallelism. Three out of the five projects which scored one for this concept were games. Manual examination of the project revealed which blocks the students were using in their projects. All the projects had two or more scripts starting on *when green flag clicked*. The projects that achieved a score of three included two scripts programmed to start on *when I receive* (fourteen projects) or *when backdrop switches* (ten projects). No groups used either the *when loudness* > block or the *create clone* block. However, neither of these two concepts were explored in the instructional sessions. While these particular projects had limited opportunity for the students to use the *when loudness* > block, there were situations when the creation of clones would have lessened the coding load for the children. This was similar to the findings of Hoover *et al*. (2016) and Moreno-Leon (2016) who also noted the absence of cloning, and suggested the exclusion of cloning blocks could be due to a lack of familiarity with the concept.

Similar to the findings on synchronisation, the project type seemed to influence the programming concepts included in the final projects. In the programming of games, the students incorporated background music and timers, which necessitated initiating two or more scripts simultaneously. For example, in the fruit drop game designed by Sophie and Bethany (3rd class), they programmed the fruit to start falling and the theme music to start simultaneously at the beginning of a game. In the game that Megan and Sylvia (6th class) designed, they spent some time getting the ball to start moving and the clock to begin counting down at the start of their ping pong game. In contrast, in the programming of stories and animations, students were more likely to include parallelism related to broadcasting and receiving or when the backdrop

switches. To make their stories and animations as real as possible, several groups wanted their programs to do two things at once. For example, in the Alice in Wonderland story animated by Louise and Bernadette (5th class), they wanted their character to hide and scream at the same time to create the impression that Alice was falling down the rabbit hole (Figure 5.2). Anna, Zara and Fiona (6th class) wanted to create the impression of a ball bursting in their story. When a ball touched the branch of a tree, they wanted a speech bubble saying 'pop' to appear at the same time as a pop sound played. It took them some time to work out that they needed to separate the two events into two separate threads, both initiated by an *if then* block. In the creation of their animation, Georgina, Valerie and Katie's (5th class) originally programmed the ball in their animation to move across the screen and then got that ball to hide and a smaller ball to appear in the background. But they weren't happy that this looked realistic. They wanted to create the perception of the ball getting smaller as it got further away. To achieve this visual effect, they needed to incorporate concepts of both parallelism and synchronisation. They spent nearly a full session working out how to get these two events to work simultaneously and look realistic. This was difficult because it involved using trial and error, in two separate threads, to work out how quickly they wanted the ball to move (this was controlled by the number of steps the character needed to take before a *wait* block was needed), how quickly they wanted the ball to get smaller (again, a *wait* block was needed) and what the size the final ball should be. The functionality of Scratch, most notably it's immediate visual feedback feature, effectively supported this trial and error approach. Including both parallelism and synchronisation in their programs required the students to use algorithmic thinking to design a series of instructions to complete a particular task.

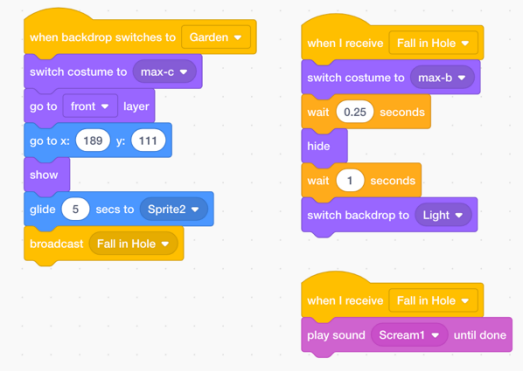


Figure 5.2: *Broadcast and Receive Parallelism in Alice in Wonderland*

The design scenario in Figure 5.3 was used to further assess the student's understanding of parallelism. The students were given the following instructions, adapted from the Scratch-Ed project page (Brennan *et al*. 2014):

> In the Performance project, Keely has designed an animated performance project. Keely wants the singer to sing while she is moving, not after. What is the bug? How do we fix the bug? Keely wants each drum to start only if it is clicked. How do we add this feature?

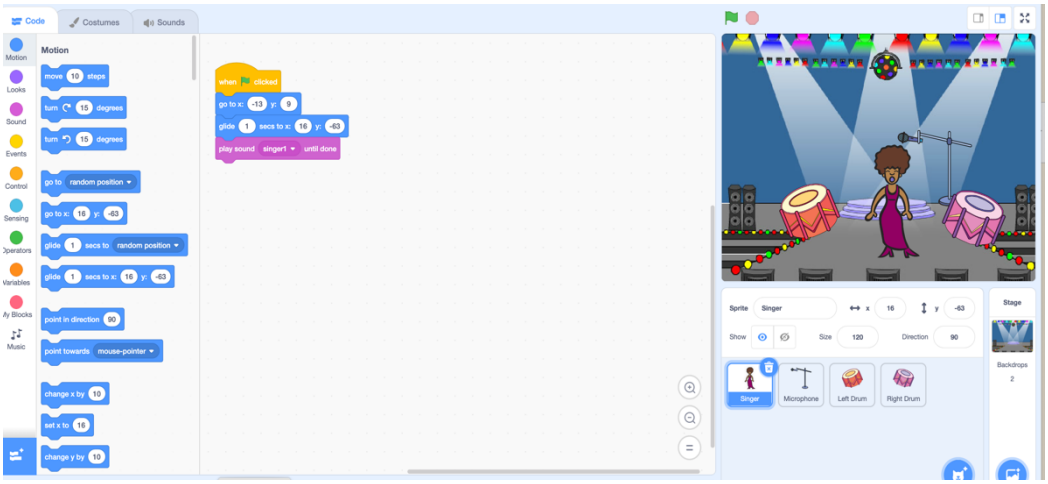


Figure 5.3: *Design Scenario to assess understanding of parallelism*

This design challenge required students to demonstrate their understanding of creating simultaneous events at the most basic level. All twenty seven groups who attempted this design challenge successfully completed it. The most common solution, employed by all but two of the groups, was to separate the commands and program the two scripts to begin when the green flag was pressed. While this solution only demonstrated an understanding of level one parallelism, it was a valid

solution to debug the project. The other two groups, both in fifth class, used synchronisation rather than parallelism to solve the problem. Instead of running the two scripts on the same events block (*when green flag clicked*), they included *broadcast* and *when I receive* blocks to synchronise the events. The positioning of the *broadcast* block within the script offered an alternative solution, as this block continues on to the next block without waiting for the triggered scripts to finish. This revealed an understanding of the functionality of this block, for these two groups, which hadn't been determined by examination of their final projects. When asked about the functioning of the block in her artefact-based interview, Zara clarified her understanding of the broadcast concept.

> Researcher: So, I see you solved the problem and now the singer is singing as she moves. How did ye do that?
>
> Zara: We used this block (points to the *broadcast* block) to tell all the other sprites what was happening. Well actually it's the same sprite but it lets this (points to the second script) know when it needs to go.
>
> Researcher: And can you show me (points to the scripts on screen), when does it need to go?
>
> Zara: Yeah, right here after this (points to the *broadcast* block). This one (points to the *glide* block) and this one (points to the *play sound* block) go together.

According to Zhang and Nouri (2019), parallelism which requires several sprites to execute scripts simultaneously, is more intuitive for students than parallelism which requires a single sprite to execute more than one script simultaneously. However, as illustrated by Zara, students in this study had grasped this more difficult concept. In their study, Funke and Geldreich (2017) also found 63% of the assessed projects supported the parallel launching of more than two scripts. The findings in relation to parallelism in this study, garnered and triangulated from a variety of different data sources, illustrate very high levels of understandings in relation to this concept when

compared to previous research. While studies on students' understanding of parallelism report mixed results, there were many studies that found this was a difficult concept for students to understand, particularly at the higher levels. Several of these studies employed Dr. Scratch in their assessment of parallelism, similar to this study. Šerbec *et al*. (2018), who studied the development of computational thinking of students aged between 8-12, found that their students performed poorly in both the parallelism and synchronisation categories. They attributed their poor performances to students' limited understanding of simultaneous events. The projects assessed in von Wangenheim *et al*.'s (2017) study achieved scores of less than two, while the majority of projects in Weng and Wong's (2017) study scored only one. Indeed, Weng and Wong (2017) reported that parallelism was used by relatively few of the primary school students in their study. In contrast to these results, Hoover *et al*.'s (2016) small scale study reported scores of either two or three, illustrating a proficient understanding of parallelism, similar to the findings in this study. Other studies have explored students' understandings of parallelism using alternative assessment measures. Meerbaum-Salant *et al*. (2013) reported that most students (92.5%) were unable to define parallelism, with some students referencing the unrelated mathematical concept of parallelograms. Sáez-López and Sevillano-García's (2017) reported more promising results with students who had completed twenty programming sessions scoring 3.79 out of a possible 5 (1 = 'Poor', 2 = 'Passable', 3 = 'Acceptable', 4 = 'Good', 5 = 'Excellent') on the parallelism items. This score was among the highest values for all computational concepts, illustrating it was among the best understood by their students, similar to the findings of this study.

5.2.4 Data - Failing to Initialise

Data representation refers to how information is stored, processed and transmitted in a program. There are two key mechanisms for storing data in Scratch: variables and lists. Within the Scratch environment two different types of variables exist: attribute variables (e.g. position, direction or size) and variables that are declared by the programmer (e.g. score, level or timer) (Franklin *et al*. 2016). Scratch also supports the use of lists, which differ from variables, in that they can store multiple pieces of data (e.g. high scores or backpacks in gameplay). The Dr. Scratch application assesses for data representation at three levels. At level one, it checks to see if any of the sprite's characteristics were modified, at level two it searches for variables that are declared by the programmer and at level three it searches for the presence of lists.

Data representation was underutilised in the games, animations and stories of the students in this study. All but one of the final projects scored one for this concept. This means that somewhere in their program the students included code that modified the characteristics of a sprite. Only one of the programs, a ping pong game, included a variable declared by the programmers and none of the programs contained lists. Hence, the data from the students' final projects offers limited insight into these students' understandings relating to attribute variables. This finding is consistent with the findings of Maloney *et al*. 2008, Baytak and Land (2011) and Kafai and Peppler (2011), who all reported limited use of variables in the projects they analysed (created by youths aged 8-18). Hoover *et al*. (2016) also found that the middle school game designers in their study omitted list data structures from their projects. However, as has been mentioned previously, the presence or absence of particular blocks is insufficient in assessing students' developing computational thinking. Several researchers have posited that particular programming activities can

promote or inhibit the use of specific programming concepts (Adams and Webster 2012; Kafai and Burke 2016). In particular, it has been suggested that creating stories does not necessitate the use of variables (Burke 2012). Kafai and Burke (2016) suggest that stories, "by their very nature as established narratives, allow for less variability and conditionality in coding sequences" (p.319). Whereas, Adams and Webster (2012) found that games encouraged the use of variables. Therefore, analysis of the final projects using Dr. Scratch did not paint the complete picture of students' understandings of this computational concept.

Qualitative data was used to further ascertain what the students knew or didn't know about data representation. The observation notes, in particular, provided interesting insights with regard to data initialisation, a computational concept that Kong (2019) highlighted was not included in Brennan and Resnick's (2012) framework. In Scratch, variables can be assigned a default value upon their creation, the user can assign a value to them at the beginning (explicit initialization) or the user can assign a value to them during processing (implicit initialization). Franklin *et al*. (2016) identified three critical aspects that students must learn about data initialisation:

- whether they need to perform initialisation,
- when this initialisation should occur and
- how they perform initialisation.

The researcher was mindful of all three when observing the students engaging in their programming tasks. The decision of whether or not initialisation is required in Scratch is dependent on the inclusion of blocks that modify attributes of a sprite or background. As most students scored one for data representation for their final projects, this meant that they all modified the characteristics of a sprite somewhere in the project. This showed that the students knew whether, when and how to perform 'implicit initialization'. It also meant that all programs required explicit initialization.

However, it was found that ‘explicit initialisation’ was not as intuitive for the students. While the Scratch drag-and-drop system avoids syntax errors by preventing users from connecting blocks in meaningless ways (Maloney *et al*. 2010), it does not prevent ‘functionally incorrect programs’ (Frädrich *et al*. 2020; Stahlbauer *et al*. 2019). Frädrich *et al*. (2020) identified incorrect initialisation of attributes and variables as a common ‘bug pattern’ (Frädrich *et al*. 2020, p.90), among novice programmers.

Other researchers have reported issues with this particular type of ‘bug pattern’. Meerbaum-Salant *et al*. (2013) also reported that students found it difficult to think about the initial state of sprite characteristics and variables. Franklin *et al*. (2017) reported that the 4th and 5th grade students in their study found initialisation difficult, whereas the 6th grade students performed well on this concept. However, they did note that while differences between these groups did emerge, they were not statistically significant. These results are in contrast to the findings of this study. Within the limited sample of this study, the class level was not found to impact students’ understanding of this concept. Students across all levels struggled with this concept.

Franklin *et al*. (2016, p.220) also noted the difficulties that students have in knowing ‘when’ to initialise.

> Although perhaps not obvious, it is actually critical in Scratch to initialize at the beginning rather than the end – execution can be stopped at any time, so only initialization at the beginning is guaranteed to occur.

Students in this study experienced similar issues as those described by Franklin *et al*. (2016). In twenty two of the final projects, the students failed to set an initial value for the attribute variables of their sprites, thereby activating the default value assigned to the sprite. Having no explicit initialisation caused difficulties with these

variables when students ran their program twice in a row or if they stopped execution in the middle of the program. These issues arose because Scratch has not got a 'definite beginning' of a program (Franklin *et al*. 2016). As the program has no 'definite beginning', in Scratch variables begin from their previously saved state. So, when a Scratch program is run twice, the starting values for the second execution are the finishing values from the preceding execution (Franklin *et al*. 2016). Louise and Bernadette experienced this difficulty, when Alice, a character who was hidden in the final scene of their story, failed to appear in the first scene of the story the second time they ran the program. In this case, explicit initialisation of the show/hide attribute was required to resolve this issue. However, without assistance, the girls were unable to determine what the issue with their program was, and consequently, how it could be resolved. This illustrated that they were unaware 'whether' they needed to perform explicit initialisation. This particular difficulty was confounded by the particular blocks they were using. Unlike variables declared by the programmer and some attribute variables (e.g. position), there is no way to read the value of the show and hide blocks (they can only be set) in Scratch 2.0 (Franklin *et al*. 2016). This makes it more difficult for students to identify their current state. Georgina, Valerie and Katie also had some issues with attribute initialisation during the programming of their final project. In their animation they reduced the size of the ball to make it appear like it was moving further away. However, when they ran their animation a second time the ball remained at the reduced size. In this case, while the girls were able to identify the problem, they were unable to determine a suitable solution. Rather than setting an initial value for the sprite size, they instead opted to manually reset the size. Their chosen strategy resulted in the problem reoccurring every time they ran their program twice in a row. This illustrated that the girls didn't

know ‘how’ to perform explicit initialisation for that attribute variable. A more appropriate solution would have been the use of the absolute block, a block which directly sets the attribute (Boe *et al.* 2013). Difficulties with initialisation like this occurred frequently across the ten weeks. Franklin *et al*. (2016) also found that students did not use the absolute blocks when performing initialisation.

From observing the students programming, it appeared that most of the problems related to explicit initialisation were due to the functionality of the Scratch 2.0 environment. In Scratch 2.0 when a new sprite is created, it is assigned a default set of values and while the variable values are not displayed directly, the visual effect is apparent (Franklin *et al*. 2016). This was particularly relevant in this study in relation to the positioning of sprites on the stage. As students could see where the Sprites were positioned on the stage, they didn’t consider initialising the sprites coordinates to be important. Scratch also allows users to manually change sprite attributes rather than assigning them with an initial value. For example, Scratch allows users to drag and drop their sprites into their initial positions rather than having to assign an x and y value to determine their position. Similarly, users can manually change a sprites costume in the costume tab rather than having to assign a value to this variable at the beginning. Being able to manually change the sprite variables removes the necessity for students to correctly perform initialisation at the appropriate time. It was also noted during the lessons, that when students were asked to describe the position of the sprite, they used positional language such as the top corner rather than the specific coordinates. This suggests that a lack of familiarity with coordinate geometry could also have impacted the students’ choice of debugging strategy.

Given the persistence of the problem, and how it was confounded by the functionality of Scratch, it seems apparent that the students would benefit from explicit instruction on this computational concept, a practice also recommended by Meerbaum-Salant *et al*. (2013). Indeed, Fields *et al*. (2015) report that when they designed a task specifically to facilitate the learning of initialisation, all of their students successfully employed a range of initialisation strategies. Dr. Scratch has the capability of detecting this particular 'bug pattern', which could potentially support students in identifying issues relating to attribute initialisation.

5.2.5 Thinking Logically

The logic concept is assessed by checking for the presence of certain constructs which cause the program to behave differently depending on certain conditions. These constructs include *if*, *if-else* and the logical operators; *and*, *or* and *not*. Projects at the lowest level of logical thinking include an *if* block. The presence of an *if-else* block illustrates level two logical thinking. The highest level of logical thinking necessitates the use of compound conditionals (the evaluation of more than one condition simultaneously). This level is achieved by projects which incorporate one of the logical operators. The logic concept was the computational concept for which the final projects in this study achieved the lowest scores. Only seven of the projects achieved a score greater than zero, and in all cases, this was one out of three. All three games achieved a score of one. Only one of the six stories achieved any logic score and three of the sixteen animations achieved a score of one. The low scores achieved by the stories and animations for this concept was not surprising. Kafai and Burke (2016) suggested that these constructs are not important in the programming of stories, as stories usually have a linear structure. However, those students who created games would have had opportunity to demonstrate greater levels of thinking.

In games, the use of logic constructs enable a program to perform different actions depending on the condition. Examples of conditions explored during the structured sessions included, *if timer equals 0*, *say game over* and *if touching fruit change score by 1*.

The design scenario in Figure 5.4 was used to further assess the children's understanding of variables and the *if-else* logic concept. The children were given the following instructions, taken from the Scratch-Ed project page (Brennan *et al*. 2014):

> In the Fruit Quiz project, Mylo designed a fruit quiz with an apple, a banana, and an orange. Mylo wants to display "Perfect!" at the end of the project if all three fruit were correctly identified, and "Almost!" otherwise. But the project always displays "Perfect!" What is the bug? How do we fix the bug?

All of the groups in 5th and 6th class were able to use logical thinking to debug the project. Of the ten groups in 3rd class, only five groups attempted the debugging challenges and two of them were successful. The remaining groups struggled with the inequality operator.

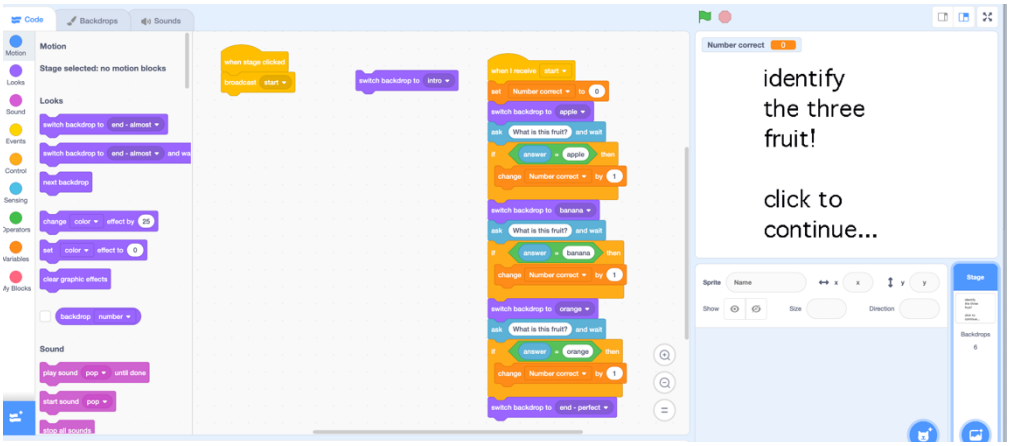


Figure 5.4: *Design Scenario to assess understanding of the If-else Logic concept*

The following conversation between five students in 5th class shows how they used both trial and error, and their understanding of the logic and variable concepts to find a solution.

> Georgina: Okay, so what do we need it to do then?

Louise: I think if we get them all right we get perfect is it?

Georgina: Yeah…and if we get one wrong we get nearly?

Katie: Almost.

Valerie: All right is 3, when Louise got them all right, this *number correct* (points at score variable on screen) was 3.

The students then spent some time looking in the different block categories for the variable block, initially unable to identify that *number correct* was a variable. Once they found it, they worked together to create the required script.

Katie: We need the same as that above it.

Georgina: There's the *if* block (she drags the *if* block onto the script area).

Katie: Now the green one. That's in here (she points to the operators category).

Louise: Bernadette has it done on ours. You need to put that (points to *number correct* variable) into that (points to the equality operator) and drag the purple one in there (signals moving the *switch backdrop to* block inside the *if* block.

Katie: Now the same again for almost.

Louise makes a copy of the section of code and changes the *switch backdrop to* almost.

Georgina: We have it.

Valerie: No we still need to change that (points to the *number correct = 3* equation).

Georgina: What do we change it to?

They all examine the blocks in the operator category.

Georgina: It's this one.

She drags out the *greater than* operator. Bernadette who has successfully debugged the code on her own computer looks up.

Bernadette: That's greater than, you need less than.

This group chose to create separate blocks of code for each outcome: a score of three and a score less than three. This was a common approach to this debugging task and

represented level one logical thinking. However, three groups in 6th class and three groups in 5th class used the *if else* block instead, demonstrating level two logical thinking. Another alternative solution to this debugging challenge would be to use the logical operators: *not*, *and* or *or*. The use of these blocks would have indicated level three logical thinking. While several groups did attempt to implement a solution incorporating the *not* block, none did so successfully. They were unable to correctly build the expression: *if not number correct equal three*, using this logic operator. Therefore, the data collected from the design scenarios confirmed the findings obtained from the Dr. Scratch and observation data. While many students demonstrated level one logical thinking, by combining the *if* block with a Boolean condition, few students used the *if-else* block to illustrate level two logical thinking, and no student attained a score of three for logic. However, these findings were probably not surprising as the students were not formally introduced to logical operators during the instructional sessions and according to Maloney *et al*. (2008), logical operators are less easily discovered than other computational concepts. Aside from the programming complexity, it is also important to acknowledge the mathematical challenge posed by inequalities, particularly for the third class students. In the Irish primary school curriculum, the use of symbols to represent inequality is only introduced in second class, and inequality is considered a difficult concept (Turner *et al*. 2016). This again illustrates that certain mathematical and computational concepts are closely aligned in the Scratch environment, and this should be taken into consideration when designing appropriately developmental experiences.

5.2.6 User Interactivity

User interactivity assesses the opportunities provided for user input within a program. Events, actions that provoke a response from the program are an essential element of user interactivity (Brennan and Resnick 2012; Park and Shin 2019). Dr. Scratch assesses user interactivity at three levels. The most basic level of user interactivity, achieving a score of one, is use of the *when the green flag clicked* block. All other keyboard or mouse interactions are considered level two user interactivity. The highest level of user interactivity requires the use of a webcam or microphone to enable user interactivity. For the majority (eighty percent) of projects user interaction with the program was limited to clicking the green flag to start the program. The remaining five projects achieved a score of two for this concept. All three games achieved this score and two animations reached the developing proficiency for this concept. Although based on a limited number of projects, it supports the views of Maloney *et al*. (2008) who suggested that games promote the development of user interactivity. No projects achieved the highest level of user interactivity. While students did not have access to a webcam, they could have used the microphone, a feature of Scratch which had been explored in the initial sessions.

The qualitative data provided more insight into students' understandings of user interactivity. In week three of the initiative, the students were tasked with creating a 'Race Car' game. During this session the students had the opportunity to explore the functionality of the *when_key pressed* block. The students were very excited at the prospect of creating a game "like a real life developer" (Michelle, 6th class). The game context supported the students in their understanding of this event block as illustrated by the conversation between Denise and Michelle (6th class) recorded in the researcher's diary.

Denise: It's like on the Wii, you press the right key to go right and the left to go left.

Michelle: So, we want this block (drags out the *when_key pressed* block). We change space to right arrow and then if we want it to move right that's change…is it x or y?

Denise: It's x, remember x for a-cross.

Michelle: Okay so *change x by 10*.

Germia and Panorkou (2020) found that understanding the utility or purpose of coordinates in Scratch supported students' understanding of the related mathematical concepts. Perhaps in a similar way, the use of the authentic context supported these students' understanding of user interactivity. In their final project, Sophie and Bethany (3rd class) used the *when this sprite clicked* block to trigger a response when a player clicked a sprite. The presence of this block suggested they had a level two understanding of user interactivity. During her artefact-based interview Sophie accurately explained the functionality of this block (Figure 5.5), further confirming her understanding of this concept.

Researcher: What does this stack of blocks do?

Sophie: So, this (points to *when this sprite clicked* block) means that it (the script) only works if you click on the donut.

Researcher: Why did you want it to work this way?

Sophie: Cos you only win when you touch the donut.

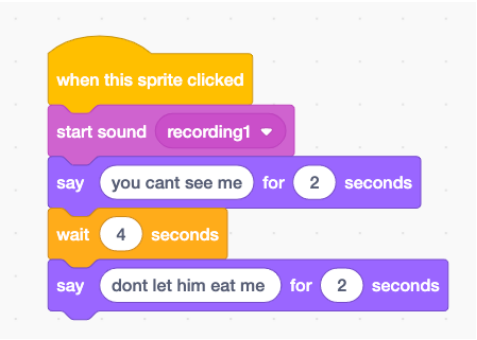


Figure 5.5*: Stack of blocks using the when this sprite clicked block*

This illustrates that students in third class had the capacity to understand level two user interactivity. Data collected during the design challenge (Brennan *et al*. 2014) in Figure 5.6 provided further evidence that students from third class upwards were capable of comprehending level two user interactivity.

> All students successfully completed the design challenge today, correctly identifying and using the *when this sprite clicked* block.
> (Observation diary, 26th April 2017)

While there was data to show that students had developed a level two understanding of user interactivity, there was no evidence of level three understandings for this concept. However, this was unsurprising given there was little emphasis placed on it during the initial teaching weeks. These findings are similar to those of Hoover *et al*. (2016), who reported that the students in their study (middle school girls) did not utilise webcam or audio input. Hence, none of their projects scored more than two for this concept. They suggested that this was due to "a lack of familiarity with these functions" (p.175). Lawanto (2016) found that the students (7th and 8th grade) in their study performed moderately in terms of user interactivity, with an average score of two. They found no difference between grades for this computational concept.

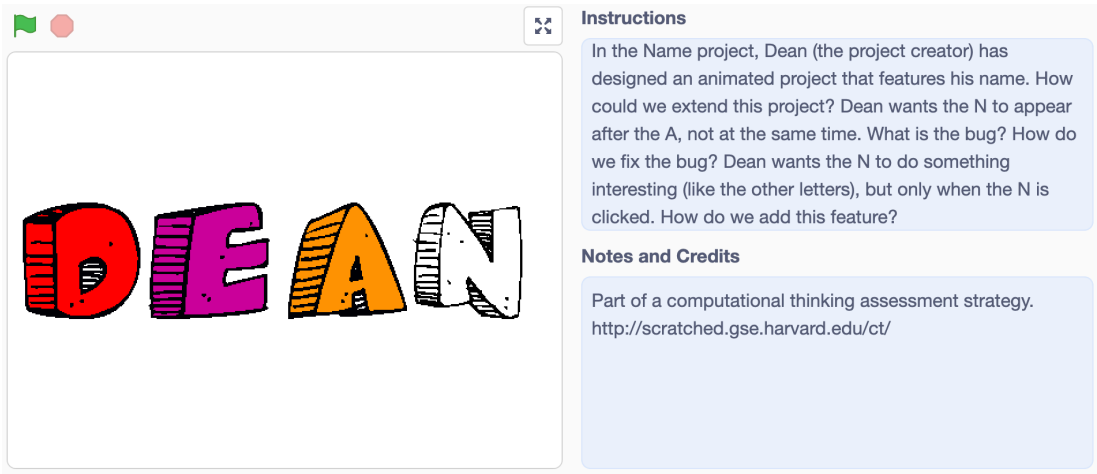


Figure 5.6: *Design Scenario to assess understanding of User Interactivity*

However, Seiter and Foreman (2013) who assessed the projects of students in grades 1-7, suggest that proficiency levels for certain computational concepts are grade

dependent. They report that students in second and third grade only utilise user interactivity at a basic proficiency level, “implementation of a key press event handler” (p.65). However, in this study no difference between class levels was found for this concept as three of the five higher scores were achieved by projects of third class students.

Maloney *et al*. (2008) found that user interactivity was high among the projects of the youths (aged 8 to 18) in their study (54%) but acknowledged that this was expected “given the popularity of games and animation” (p.369). Similarly, Wilson *et al*. (2012) whose research looked solely at the development of computational concepts in the programming of games also reported high levels of user interactivity, with over 90% of projects requiring keyboard or mouse control. Contrastingly, Funke and Geldreich (2017) found that boys in their study were more likely to include level two user interactivity than the girls. Interestingly, they also found that boys were more likely to create games, whereas girls were more likely to create stories. Hence, they suggested that the type of project created, rather than the gender of the creator, facilitated students’ demonstration of proficiency for this particular computational concept. The preference for stories over other types of projects was not evident in this study as animations were by far the most popular final project type. However, very few groups opted to create a game for their final project (three). Therefore, the project type could have influenced the complexity of user interactivity illustrated by the students in this study. Indeed, there are several other studies which support this assertion. Park and Shin (2019) who analysed a wide variety of project types (starter projects, animations, art, games, music and stories) found that user interactivity scores were higher for games than for any other type of project analysed. Similarly, Moreno-León *et al*. (2020) found that user interactivity is

generally higher in games than in animations or stories. Funke *et al*. (2017) suggest that animations do not support the development of this computational concept. However, two of the five highest scoring projects for this concept were animations, contradicting the finding of Funke *et al*. (2017). Aside from gender, age and project type there are other factors that can influence computational concept development. Indeed, Park and Shin (2019) suggest that Scratch might not be the best programming language for the development of user interactivity proficiency. In their study, they compared the effectiveness of Scratch and App Inventor in developing understanding of computational thinking concepts. They found that the average score for user interactivity was lower for Scratch than App Inventor. They acknowledged that greater functionality of App Inventor is associated with user interactivity, and as such they have many more event blocks than are available in Scratch.

5.2.7 Flow Control

In computer programming, flow control specifies the order in which the code in a program is executed. Sequencing, iteration and selection enables the program to start a new action, repeat an action a specific number of times, or repeat an action until a certain condition occurs (Dr. Scratch, 2019). In Dr. Scratch these three elements have been separated. An ability to incorporate sequencing and iteration in a program is considered to signify competency in flow control, whereas the inclusion of selection in a program is considered to indicate logical thinking (which has been discussed previously). At the lowest level of flow control, programs are required to have a sequence of blocks. Scoring more than one on the flow control concept required the use of loops. A loop allows for multiple executions of a command without having to create separate code for each execution. There are two types of loops, count-controlled and condition-controlled. Count-controlled loops repeat the same steps a

specific number of times, regardless of the outcome. The control blocks *repeat* and *forever* are examples of count-controlled. At level two, the program must contain a loop, either *repeat* or *forever*. To demonstrate the highest level of flow control, a program must include the condition-controlled loop block *repeat until*. This loop instructs the program to repeat the section of code inside the block until a specified condition is met. The scoring for flow control can be seen in Table 5.5. All projects scored a minimum of one for this concept, as all final projects included at least one sequence of blocks. All three games, three animations and one story scored two for this concept. This illustrated that these students had incorporated either a *repeat* or a *forever* block. The majority of these projects used count-controlled loops to control the movement of their sprites. None of the final projects contained a condition-controlled loop, which is a more advanced programming construct and one which wasn't explicitly taught in the initial sessions.

Table 5.5: *Dr Scratch scores for the Flow Control concept*

Computational Thinking Concept	**Competence Level**		
	Basic (1 point)	**Developing (2 points)**	**Proficiency (3 points)**
Flow Control	Sequence of blocks	Repeat, forever	Repeat until
Number of Projects	18	7	0

The limited use of the *forever* block (just three projects) was somewhat surprising, given its importance in conditional execution. This finding was in contrast to the results of Aivalogou and Hermans (2016), who evaluated 250,000 Scratch projects and found that the *forever* block was the most common of the looping constructs present, used in 40% of projects. This was much higher than the 12% of projects which contained the *forever* block in this project. Aivalogou and Hermans (2016) also reported a higher usage of the *repeat* block, with 28% of the final projects

including this block. In contrast, its use by the students in this study was much more limited with only 16% of final projects containing a *repeat* block. The *repeat until* block was not explored in the instructional programming sessions of this study so the absence of this block was expected. However, other research conducted in different conditions have reported similar findings (Meerbaum-Salant *et al*. 2011; Aivalogou and Hermans 2016). Aivalogou and Hermans (2016) also reported limited use of the *repeat until* block. The projects evaluated by Aivalogou and Hermans (2016) were randomly retrieved from the Scratch projects page. This meant that factors such as the user age, user programming experience or how they learned programming are unknown. So, perhaps this is a difficult concept for students to learn, and direct experience with this construct is a necessity for student understanding. Indeed, Meerbaum-Salant *et al*. (2011) suggest that the pedagogical approach adopted by teachers, teaching the simpler *forever* construct before the *repeat-until* construct, can inhibit students' understanding and use of the *repeat until* construct. Therefore, perhaps these constructs should have been taught in reverse order and students should have had an opportunity to experience the *repeat until* construct before the *forever* construct was introduced in week 4 (see table 4.2). These findings highlight the importance of providing carefully designed and sequenced programming activities in supporting the development of computational thinking concepts.

The three games all incorporated the *forever* block for a variety of purposes. In the 'Ding Dong Ping Pong' game the *forever* block was used in conjunction with the *key_pressed* sensing block to control the movement of sprites. In this case the *forever* block allowed the program to continuously check to see if a key was pressed. This combination of blocks offered a more complicated alternative to using the event block, *when_key pressed*, perhaps illustrating a lack of awareness of this block. In

both the ‘Hide and Seek’ game and the ‘Dunken Dounuts’ game, the *forever* block was used to continuously change the position of a hiding sprite. The artefact-based interviews confirmed that students understood the purpose of the *forever* block (Figure 5.7) in their projects, as illustrated by Sophie (3rd class) in the following excerpt.

> Researcher: And Sophie can you tell me what does this stack of blocks do?
>
> Sophie: It hide and show doughnut.
>
> Researcher: What about this block (points to *forever* block)? What does this do?
>
> Sophie: We want it to happen again and again not just first time.

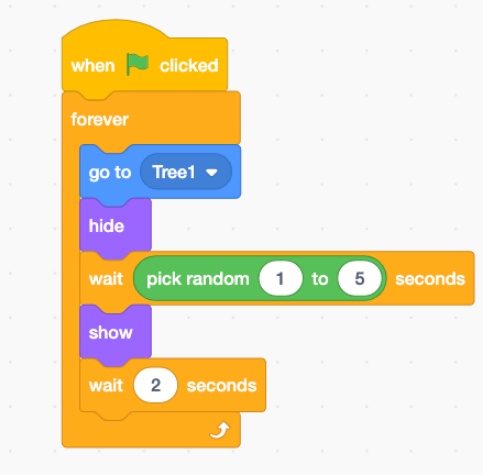


Figure 5.7: *Stack of blocks using the forever block*

Aside from its use in the games, there was no other use of the *forever* block in the final projects. This suggests that games would be a useful medium for the learning and assessment of this concept. The three animations and one story who achieved a score of two for flow control all included the *repeat* block. The use of the *repeat* block in three of those was to create motion in their projects. For example, Georgina, Valerie and Katie (5th class) used the *repeat* block to repeatedly move and shrink the ball in their animation to simulate a ball being thrown into the distance. In the remaining animation, the *repeat* block was used to create the visual effect of a sprite

continuously changing colour. Students at each of the class levels demonstrated an ability to include level two flow control concepts in their programs. Three of the six third class projects scored two for this concept, while two fifth class and two sixth class projects also scored two for this concept. This suggests that class level was not an important factor in determining students' understanding of flow control.

Observation of the students' programming provided greater insights into students' understanding of flow control. The following is an excerpt from the researchers' observation diary (1st March 2017) describing a conversation between three fifth class children about a bug in a project that Isabela and Katya were working on. Their project was based on the widely used introductory programming 'Hungry Shark' game. They are adapting the game to reflect their learning from a recent Geography class on predators and prey.

> Isabela: It's not working right; the polar bear isn't doing anything when he catches the salmon.
>
> Katya: There must be block missing.

Together they read through the lines of code carefully, Katya touching each block and Isabela reading out each one.

> Bernadette: (Who is sitting at the table next to Isabela) You are missing the *forever* block here (points to the screen below the *when green flag clicked*).
>
> Katya: Why do we need that block?
>
> Isabela: Put it there and see does it work.
>
> Bernadette: You need it so it checks all the time to see if you catch the salmon not just at the start.
>
> Katya: But this block checks to see if it is touching the salmon (points to the *touching* block. Then turns to the researcher) Is the forever block not just for…if we want something to happen over and over like playing a sound again and …
>
> Bernadette: That only checks once and then it stops checking and moves on to this block (points to the next block *say Yum*). If you put the *forever*

> block around the outside it keeps checking and the bear will say Yum every time it catches a salmon.
>
> Isabela: It worked. Look it ate the salmon.

This conversation shows that Katya has trouble understanding that when a condition can vary from 'being false' to 'being true' throughout a game, such as whether one sprite is touching another sprite, it is then necessary to continuously check this condition throughout the game. In her interview Laura (6th class) spoke of similar issues she had with the forever block.

> When we were making our race game we wanted our car to say 'I've won' if they touched the finish line. We didn't put in a forever block and it took us ages to figure out that the code only checks to see if they are touching once at the very beginning if you don't have that forever block.

It is the necessity to combine elements of flow control and logic that caused the difficulty for these students. Other researchers have reported similar issues with conditional logic among their students. Funke *et al*. (2017) found that 37% of their students failed to include conditional statements in their projects. Aivalogou and Hermans (2016) analysed nearly 250,000 scratch projects retrieved from the Dr. Scratch website. They found that although 77% of the projects contained loops, only 14% contained conditional loops. The limited use of conditional loops in this study could perhaps partly be attributed to the project types chosen. Burke (2012) suggests that conditional statements do not feature "prominently in the storytelling format" (p.127). Contrastingly, Hainey *et al*. (2020) found that iteration was one of the most frequently used computational concepts in game design. However, the observation data suggests that this may not be the only explanation, as this was a concept that the students struggled with in their weekly tasks also. In weeks four and five, the students were working on designing games for their classmates. During these weeks, the researcher observed that students continually struggled when it was necessary to combine looping and conditional constructs. The most common programming error

that the researcher observed was the omission of the *forever* block around the *if* block. Frädrich *et al*. (2020) also found this to be a common error and labelled it 'missing loop sensing'. Frädrich *et al*. (2020, p.91) defined it as follows:

> If a script is supposed to execute actions conditionally when an event occurs, this is often done by continuously checking for the event inside a forever or until loop. If the loop is missing, the occurrence of the event is only checked once and thus likely missed.

Pea (1986) also reported difficulties with conditional loops among the novice programmers in his research and suggested that these difficulties could be attributed to a lack of understanding of how a computer reads code. The novice programmers in his research presumed that the conditional statement was 'waiting for' its condition to be true before being executed. Denner *et al*. (2012, p.246) further explicate this misconception, reporting that

> even when conditionals were placed in sequential code segments, students did not expect the conditional expression to only be tested when the statement holding the conditional is executed.

Whereas, in reality the conditional statement is executed in the order it appears in the program, after which it becomes inactive and the "control cycle never returns to it" (Pea 1986, p.27).

5.2.8 Abstraction and Problem Decomposition

Abstraction was not included as a computational concept in Brennan and Resnick's (2012) framework, instead they included it as a computational practice. However, Kong (2019) identified procedures as a computational concept he believed was overlooked by Brennan and Resnick (2012). Kong (2019) described procedures as code that enables programmers to avoid "the repetition of codes and duplication of commands" (p.24). In Dr. Scratch, this concept is paired with decomposition, breaking a problem into more manageable chunks to support comprehension and debugging. In Dr. Scratch problem decomposition is considered a basic level

concept, illustrated by the inclusion of more than one sprite and more than one script. At level two, students should be able to use 'My Blocks' to be able to create and define their own blocks. Achieving the maximum score of three points requires the use of clones to simplify code in situations where several sprites are performing the exact same action. The scoring for abstraction and problem decomposition was among the lowest of all the computational concepts scores, with only logic scoring lower. All projects included more than one sprite and more than one script, hence achieving a score of one and demonstrating an understanding of problem decomposition. However, no project achieved a score greater than one, and therefore no student demonstrated an understanding of abstraction. Again, this result was expected as neither 'My Blocks' nor clones had been explored in the initial sessions due to time constraints. As none of the design scenarios used assessed understandings of this concept either, it is unclear from the data if this concept is within the capabilities of primary school students.

However, previous research has shown that abstraction is a difficult concept. Park and Shin (2019) who analysed 524 'mature' (categorised as 'popular' and 'trending' on the Scratch website and considered to be of high quality) Scratch projects also found that scores for this concept were low relative to the scores for the other concepts. Similar to the findings of this study, Hoover *et al*. (2016) reported that the middle school girls in their study omitted user defined blocks (2 points) and cloning (3 points) from their programs. They also speculated that the omission of these concepts could have been due to a lack of familiarity. Although, some researchers have suggested that proficiency levels for abstraction are associated with age or grade level (Lawanto, 2016; Seiter and Foreman 2013), other research contradicts this, reporting similar findings among older students. Šerbec *et al.* (2018) who

examined differences in the computational thinking concepts scores of primary school students and pre-service computer science teachers found no difference between the abstraction understandings demonstrated by both groups. Setyawan (2020) analysed the Scratch projects of 87 primary pre-service teachers and reported that the scoring for this concept was the lowest of all computational concepts, with a mean score of 1.39. Therefore, as suggested by Hoover *et* al. (2016) and Lawanto (2016), it is possible that opportunity and experience are key to the development of this computational concept. In Park and Shin's (2019) study the average score for abstraction for games was 1.98, the highest of all project types. This suggests that game design could be influential in the development of abstraction skills. Further support for this idea comes from the research of Werner *et al*. (2014). They reported that computer game programming supported their middle school students' understanding of cloning. Importantly, they also recommended that students have opportunity to experience complex programming concepts in guided activities before engaging in game-design, to encourage the creation of more sophisticated games.

5.2.9 Computational Concepts - Conclusion

The findings of this study indicate that during the ten week initiative the students progressed in their understanding of computational concepts. The majority of students had no experience of programming coming into the initiative, and at the end of the initiative all students were evaluated as developing. These results compared favourably with results from studies of older students (Lawanto 2016; Setyawan 2020; Troiano *et al*. 2019). They illustrated proficient use of synchronisation and parallelism, a developing understanding of flow control, user interactivity and data representation, and a basic level of abstraction and logical thinking. Previous research on the development of computational concepts has yielded mixed results.

While some studies identified similar strengths and weaknesses (Lawanto 2016; Troiano *et al*. 2019), others reported contrasting results (Maloney *et al*. 2008; Sieter and Foreman 2013; Wilson *et al*. 2012). This suggests that contextual factors may impact students' development of computational concepts. In this study, while age or grade level did not appear to inhibit computational concept development, project type and exposure to concepts did appear to facilitate or hinder students' understanding and use of computational concepts. These findings highlight the importance of providing developmentally appropriate learning experiences to advance students' understanding of computational concepts.

5.3 Computational Practices

5.3.1 Computational Practices - Introduction

Computational practices are the practices that students develop as they engage with computational concepts (Brennan and Resnick 2012). Brennan and Resnick (2012) identified four key computational practices, abstracting and modularising, experimenting and iterating, testing and debugging, and reusing and remixing. There are few research studies which explore the development of these practices. Therefore, this research aims to shed light on how and in what ways young programmers develop these practices. Data analysis revealed that all four computational practices were closely connected, with students often engaging in two or more practices simultaneously.

5.3.2 Abstracting and Modularising

The first of these computational practices, abstracting and modularising, which refers to the complementary processes of simplifying (abstracting) and separating into

distinct parts (modularising). Martin *et al*. (2014, p.1560) proposed that abstracting and modularising are characterised by the following actions:

1. *Decide what sprites are needed for your project, and where they should go,*
2. *Decide what scripts are needed for your project, and what they should do,*
3. *Organize the scripts in ways that make sense to you and others.*

Brennan and Resnick (2012) identified two programming processes that require abstraction and modularising, the initial conceptualisation of the problem (maps to 1 and 2 above) and then translating this concept into separate sprites and blocks of code (maps to 3 above).

5.3.2.1 Conceptualising the Program

Following five weeks of structured programming activities, four weeks were allocated to students independently creating their own project. During this period the students had to plan, create and edit a Scratch program of their choosing. Four approaches to conceptualising a program emerged from the artefact-based interviews. Some students chose to recreate and extend projects which had been explored in the structured sessions, such as the ping pong and fruit drop games. Some students created projects inspired by well-known stories such as 'Alice in Wonderland' and 'The Worst Witch'. These approaches both illustrate the connections between this computational practice and the computational practice of reusing and remixing. Other students designed their projects to reflect personal experiences, including the animations 'Ball Burst' and 'Fetch'. Finally, for some students the creative catalysts were the sprites and backgrounds they selected for their projects. As part of the design process, once they had determined the project type and theme, students were required to develop a storyboard for their idea. These storyboards supported students in both idea development and managing complexity.

Chloe and Maja in 3rd class explained how the storyboard supported them in developing a clear idea of the sprites and scripts required for their Scratch project.

> Researcher: How did you decide what you wanted each letter in your project to do?
>
> Chloe: Um…we had to make a plan in our storyboard. So, before we start we draw each letter and give it something to do.
>
> Researcher: Can you show me there on your plan [Maja]?
>
> Maja: So, this is the letter [‘M’ for Maja] and we give it colour and then…[points to the storyboard] this mean it spin around.
>
> Chloe: Yeah, and after we look at the plan here and we know what to do.

Kenna (2022) proposed that storyboarding is a useful strategy in designing and making as it supports visualisation of ideas. Burke (2012) previously reported that students found that storyboards provided some focus in the wide-walled programming environment of Scratch. Sixth class student Katie, who worked with Valerie and Georgina to create a fetch animation, felt that the storyboard helped them decide on what sprites were needed, what scripts were needed and how these should be sequenced.

> Researcher: How did you decide what sprites you needed?
>
> Katie: Before we did our code we had to draw out our idea. So…ah…we had to put what characters we want and what background here [points to ‘The Where?’ on her storyboard] and then we drew what happen in each part. Then we knew when we went onto Scratch what

Katie indicated to the headings of the storyboard while speaking, illustrating how the different elements of the storyboard helped her in planning her project. Burke (2012) noted that not all students were interested in storyboarding, with some including only minimal detail. However, this was not reflected in this study, perhaps indicating that the structured nature of the storyboard encouraged the students to include more precise descriptions.

5.3.2.2 Translating the Concept into Code

The second aspect of the design process that supported the development of this computational practice was the process of translating the concept into code. The students illustrated that the key factors that impacted how they organised their code were separating distinct actions or scenes, project functionality and user experience. Anna, Zara and Fiona (6th class) built and organised their code based on the different actions that occurred in a scene, with each action controlled by a different stack of blocks. Diya and Erica's (3rd class) animation consisted of a series of interactions between two sprites. Each interaction was coded in a different stack and the stacks were organised in the order they occurred in the animation. Both groups explained that this approach aided any testing or debugging that might later be required. When asked to explain decisions they made with respect to organising their code, students frequently alluded to how using separate scripts improved the functionality of their programs. For example, Amber and Alannah recognised that they needed a separate script if they wanted to have background music in their animation.

> "We had to put the music on a separate script because they work one after the other and the characters wouldn't move while the music was playing, it wouldn't go until the music finished" (Amber, 6th class)

As illustrated by Amber, the modularising or decomposing programming practice was closely related to her understanding of parallelism. This was the case for many students and, given that they performed particularly well on the parallelism computational concept ($\bar{x}$ = 2.6), it is unsurprising that the students regularly engaged in abstracting and modularising. Two examples of this previously mentioned (in section 5.2.3) were the final projects of Anna, Zara and Fiona and Georgina, Valerie and Katie. In both cases the students wanted to make their animations more realistic through the use of parallel audio or parallel visuals. There

were also some instances during the code building process when students realised that the code they had written was not working as effectively as they wanted to achieve their desired effect. In these instances, the students searched for a more efficient solution. In the development of their car racing animation, fifth class students, Isabela and Katya found that the *touching color_* block was not working as they had intended on some occasions. This prompted them to change their program, adding a new script and removing the finish line from the backdrop as described in the observation diary extract below.

> In their car racing project, Isabela and Katya added the finish line as a sprite instead of drawing it on the background because they found that when they used the *touching color_* block there were white patches in the background which triggered the subsequent script at inappropriate times. Instead, they decided to add the white finish line as a sprite, which allowed them to use the *touching sprite* block which was a more effective solution.
>
> (Observation diary, 22nd March 2017)

Jane and Linda in 5th class had to take the reverse action, removing a character and adding a feature to the costume of the sprite instead.

> Jane and Linda added clothes to their dancers initially as separate sprites. But when they went to move the character, the hat and shoes stayed behind. Following some discussion they decided that rather than add the clothes as a sprite they would use the costume tab and paste the clothes on to the sprite. Adding the clothes straight onto the sprite meant that they now move with the character.
>
> (Observation diary, 22nd March 2017)

In both instances the students made decisions about the inclusion or removal of a sprite to improve the functionality of their projects. In each case they were engaged in testing and debugging and abstracting and modularising practices simultaneously. There were also many examples of code organisation decisions that were taken to improve user experience. These decisions were influenced by either feedback they received from their peers or their own experience of other projects, illustrating the importance of both being, and having, an audience. Megan and Sylvia, conscious

that their game would be reviewed by their peers during the final gallery walk, were keen to add some personalisation features to make it more enjoyable.

> "We wanted to make it more personal by asking the player what their name was at the beginning and then using their name when giving the score at the end. So that they'd enjoy it more like." (Megan, 5th class)

Denise and Michelle, added additional 'baddie' sprites to their game to make it more engaging for players, having received feedback from their peers.

> Researcher: Why did you choose to make these sprites?
>
> Denise: Well, first we just had the good characters that you collect to win points and then [Anna] played it and said it was easy because you just had to move your character over and back and not pay attention. So, we added in 'baddies' that made your score go down to make it harder for the players.

This emphasised the social aspect of the programming initiative (Papert 1984) and highlighted that the students valued having an authentic audience, which will be discussed further in section 5.4.3.2.

5.3.3 Experimenting and Iterating

Experimenting and iterating, also referred to as "being incremental and iterative", means developing a little bit, trying it out and then developing some more (Brennan 2012, p.7). The students employed experimenting and iterating strategies in two distinct ways. Firstly, they engaged in iterative process of design, they began with an initial idea and repeatedly tweaked it to improve their final project. This process was supported by the use of storyboarding and was closely aligned with the computational practice of abstracting and modularising. Lin and Liu (2012) have previously reported that the development of a plan prior to the commencement of coding resulted in students adopting a more systematic approach to programming. Secondly, students often adopted a trial and error approach when they had developed

an idea but weren't sure how to translate it to code. Hence, this second approach was closely aligned with the computational practice of testing and debugging.

5.3.3.1 Iterative Process of Design

At the beginning of the design process, all students developed a storyboard to map out the idea for their project. However, during the translation from storyboard to code, the project concept invariably changed in some regard. The project revisions were often the consequence of the trialling of scripts, which then triggered further inspiration or prompted feedback from a peer. It was common for students to build the scripts either sprite-by-sprite or scene-by-scene. For example, when coding their race game, Isabela and Katya wrote all of the code for controlling the car before testing it, whereas in the design of their story, Jane and Kelsey found a natural break in the coding process when moving to a new backdrop. These findings contrast with those of Fessakis *et al*. (2013) and Robins *et al*. (2013) who both reported that their young programmers adopted a line-by-line iterative approach to programming. Despite the different approaches adopted they all illustrate an incremental and iterative approach. This incremental and iterative approach was enabled by the immediate visual feedback provided by Scratch, a view previously expressed by Daily *et al*. (2014).

There were several terms which were frequently employed by students during this iterative process, including "*it would be really cool if we could*", "*I think it would be better if*" and "*I wonder if we*". In many cases the enthusiasm gained from seeing their idea come to fruition on screen, led students to new creative ideas. Diya and Erica in 3rd class created a joke animation. Once they had coded the joke, they were

keen to add additional effects to enhance the appeal of their project. The exchange between Diya and Erica was captured in the following observation diary entry.

> Having coded the conversation between their sprites [Diya] suggested that "it would be really cool" if they could make the penguin go red to show that he was mad with Gobo. [Erica] agreed and suggested that they make him go red and then the penguin could kick Gobo off the stage. They were keen to add more humour to their joke project, as [Diya] said "we want to make our friends laugh".
>
> (Observation diary, 28th March 2017)

This extract also highlights the influence that having an authentic audience had on their decision making in the design process. The students were often keen to revise their original design to boost the appeal of their projects. Enhancing the realism of their projects was also important to the students. Many of them made several revisions to their original code to improve the aesthetics of the on-screen action. While some of the revisions were simply 'cosmetic', changes to colour or size, there were also instances when students' revisions required the incorporation of more complex computational concepts, such as synchronisation, parallelism and flow control. For example, Georgina, Valerie and Katie (5th class) replaced their *glide_secs to x:_y_* block with *repeat* and *wait* blocks to create a more realistic impression of a ball being thrown. Similar to Erica and Diya in third class, the motivation behind these revisions was "*to make it look good for the gallery walk*" (Katie, 5th class). The benefits of having an authentic audience, which have been advocated for by Brennan and Resnick (2012), proved to motivate students to engage in an iterative process of design. Similarly, several researchers have endorsed programming languages which provide instant feedback (Mannila and de Raadt 2006; Kölling and McKay 2016; Homer and Noble 2017). This study found that this feature of Scratch supported and encouraged students to engage in experimenting and iterating.

5.3.3.2 Trial and Error Approach to Programme Development

In the aforementioned experimental episodes, the students were making revisions based on newly developed ideas. However, students also made revisions because the code they had written did not realise their initial design. In these instances, the students often adopted a trial and error approach to revise their code to accomplish their initial design goal. This illustrated a close relationship between the practices of experimenting and iterating and testing and debugging. There were many instances of students developing a little bit of code, trying it out and then revising it because it didn't work as intended. For example, third class students Lena and Natalia used trial and error when they experienced difficulty with the orientation of their sprite following a rotation. Their experience of experimenting and iterating was captured in the following observation diary extract.

> [Lena] and [Natalia] created an animation with a swimmer doing lengths in a pool. However, when they played the code they realised that when the sprite rotated it then appeared upside-down. They adopted a trial and error approach to find a solution to the problem. First, trying the *point in direction* block (which created a bug) before [Kornelia] at a neighbouring table showed them the *set rotation style left-right* block.
>
> (Observation diary, 21st March 2017)

This exchange also highlighted the benefit of working with others to find solutions to problems, which will be discussed further in a later section (5.4.3.1). Anna, Zara and Fiona (6th class) adopted a similar trial and error approach when a scene in their project didn't initially turn out as they had planned. Their design included a scene with one of their sprites walking with the other sprite on their shoulders. However, their initial attempt to code this scene did not work as intended.

> Researcher: Walk me through how you built this script [points to a script on their project].
>
> Zara: Well actually that was a bit hard. We…ah…first we tried to have the same script for both Giga and Gobo but…ah…Gobo didn't walk as far as Giga…ah…so we had to play around with how many steps Gobo

> needed. That was how we got the twenty [points to the *move 20 steps* block].
>
> Researcher: And how did you figure out how many steps?
>
> Zara: Well, first we had like ten and that wasn't enough so we tried 12 or something and that was still too little so we kept going up until they moved together.

This trial and error approach was characterised by students trying different things to achieve the visual or audio effect they had envisioned for their projects. Falloon (2015) and Lin and Liu (2011) both reported that very few of their students employed trial and error approaches in their programming. While some researchers promote this trial and error approach, other researchers argue that this approach does not promote conceptual understanding (Blikstein and Worsley 2016). While it is unclear if Lena and Natalia understood the concept behind their newly discovered block (*set rotation style left-right*), interactions like that does provide exposure to more programming concepts, which is an important part of learning how to programme (Maloney *et al*. 2008; Meerbaum-Salant *et al*. 2013; Werner *et al*. 2014). The scenes in the storyboard provided natural breaks in the programming process and encouraged the students to be incremental in their code development. In many cases, students would code and test each scene before moving on to the next one. Litts *et al*. (2020) also reported that storyboarding encouraged students to engage in experimenting and iterating. It also reinforces the finding that there is a close association between being incremental and iterative and the practice of testing and debugging. The Scratch programming language facilitates this incremental and iterative programme development. Scratch enables users to execute individual snippets of code by clicking on them. This visual feedback supports students to adopt the trial and error approach. Indeed Millwood *et al*. (2018) suggests that this aspect of computing makes it a suitable tool for problem solving and design.

5.3.4 Testing and Debugging

There were many opportunities for the students to engage in testing and debugging practices. In the initial sessions they encountered bugs as they modified artefacts designed by others. During the project development weeks, they created buggy code in the implementation of their design plans. The final opportunity for debugging was in the last week when they investigated the design scenarios and developed suitable solutions. According to Brennan and Resnick (2012, p.7), testing and debugging involves developing "strategies for dealing with – and anticipating – problems". The inductive coding process revealed that the students engaged in these testing and debugging strategies in two distinct phases. Firstly, they worked to identify the problem and secondly, they employed strategies to determine an appropriate solution.

5.3.4.1 *Execute, Observe and Identify the Problem*

The practice of testing and debugging was often closely associated with the practice of experimenting and iterating. This was because students frequently identified that there was a problem because they executed snippets of code during code development. Regardless of whether they had developed the code themselves or whether they were debugging someone else's code, the students always started the debugging process by running the code. Sophie employed this strategy when she was trying to debug the design scenarios in week ten.

> Researcher: How did you know what was wrong with this project?
>
> Sophie: Cos when I played it I saw that the octopus and the whale were talking at the same time

When building their projects incrementally, the students also tested regularly to ensure their code was working as they intended. As the code ran, they considered

how the action unfolding on the screen differed from what they expected. During week three, when the students were designing the race game, several students had difficulty with conditional looping. They were able to describe what they wanted to happen and how this was different from what actually happened when they ran their code.

> Many students struggled to modify the race game to incorporate a special effect of their choosing. For example, Alisa and Joan wanted their character to react when it crossed the finish line but nothing happened. "I want it to say 'I won' when it reaches the finish line but nothing is happening when it touches the white line".
>
> (Observation diary, 22nd February 2017)

The inductive coding revealed that once students observed that there was a problem with their code, they generally employed three strategies they employed to determine the cause of the problem, reading through the code, segmenting the code and hypothesising based on previous experiences.

5.3.4.1.1 Reading Code

The first strategy involved students reading through each step of the code and discussing what they expected the output to be. This helped them to isolate the source of the problem.

> Researcher: How did you know what was wrong with this project?
>
> Aisha: We looked at all the code and what each bit did. They used the *switch backdrop* block…same as we did but when we looked here [opens the backdrops tab] we saw they had code for the baseball and the basketball but not for the tennis. There was no code that would change to the tennis court background.

There were many instances when students successfully identified the source of a problem using this strategy. The readability of Scratch, as a visual programming language has been recognised by previous researchers (Götz *et al*. 2022; Wilson and Moffat 2012). This finding suggests that Scratch is a suitable language for young novice programmers such as the students in this study. Lai and Yang (2011) have

previously found Scratch to be support students' prediction skills in programming problem-solving. Given that students frequently relied on their ability to read code when debugging their programs, it could be worthwhile to provide students with opportunities to practice this skill. Although, not employed in this programming initiative, the PRIMM approach (Sentance *et al*. 2019) could be useful in this regard. The predict stage of PRIMM focusses on determining the function of code and encourages the development of code reading and tracing skills. However, there were times when times when reading the code was an ineffective strategy, particularly in instances when the students hadn't written the code themselves and didn't understand the underlying computational concept. In the development of their final project, Sophie and Bethany imprecisely copied a snippet of code from a fruit drop game they had built in the shared coding session in week five. When copying the code they omitted the *forever* block. However, their limited understanding of the *forever* concept prevented them from detecting the error. This was a common occurrence as the students grappled with the new computational concepts and the corresponding Scratch commands. It required students to find alternative debugging strategies, which included segmenting the code.

5.3.4.1.2 Segmenting Code

The second approach students took to isolating the source of a problem was to segment the code into smaller stacks. This approach was generally taken if students were faced with long sequences of code to debug.

> Researcher: Were there things that were particularly hard?
>
> Laila: Sometimes things were going wrong and we didn't know why.
>
> Researcher: How did you find out what was wrong?

> Laila: We had to take it apart. We had lots [of blocks], so we did this [separates out a stack of blocks] and played little ones together like this [clicks on a stack] to find the broken bit.

Laila's comments again illustrate the value of being able to click on stacks on code and immediately observe the output on the stage. This supports the assertion of Maloney *et al*. (2008, p.368) who advocated for "having the palette, scripting area, and stage simultaneously visible", as it provides "the user with a process model of how their scripts are interpreted by the computer". Aivaloglou and Hermans (2016) have also found that shorter scripts increase students' ability to understand and modify code.

5.3.4.1.3 Experience-based Hypothesis

The final strategy used by students to identify the source of a problem was hypothesising the cause of the problem based on past experiences.

> Some students have started to identify the source of programming errors by making connections with previous occurrences of these errors. Georgina and her group realised the need for a wait block between the changes to the size because they remembered how fast the computer executed code from previous projects. [Bernadette] was able to help [Katya] and [Isabela] fix a bug in their program because she had previously encountered the same bug in her race program.
>
> (Observation diary, 29th March 2017)

However, this strategy was used less frequently than the two previously mentioned strategies. This was perhaps surprising given that there were a number of commonly occurring programming errors (e.g. omission of the *forever* and *wait* blocks). Therefore, students might need further support in adopting this testing and debugging strategy. Indeed, one of the teachers suggested that the compilation of a bug wall or an error checklist might be useful to support both teachers and students in identifying and solving programming errors.

> "I think that teachers will require lots of upskilling, particularly around supporting children with debugging. I know lots of the problems they encounter are reoccurring so maybe some pedagogical strategies we

> already use would be helpful like using a bug wall or error checklists to eliminate common problems." (Ms Walker)

This process of storing and recording previously encountered programming errors would provide students with an additional knowledge base to inform and enhance their testing and debugging practice.

5.3.4.2 Determine Appropriate Solutions

Once students had identified the source of the problem, the next step in the testing and debugging process was determining an appropriate solution. Brennan and Resnick (2012) identified three possible approaches to finding appropriate solutions, "trial and error, transfer from other activities, or support from knowledgeable others" (p.7). Data analysis revealed that each of these approaches were frequently adopted by students during the programming initiative.

5.3.4.2.1 Debugging by Trial and Error

The first of these, trial and error, was already mentioned in relation to being incremental and iterative in programme development (5.3.3.2). Trial and error was also commonly used by students to debug their programmes. The trial and error strategies observed included tinkering with block sequencing, modifying variables, replacing a block with a similar functioning block and trialling unknown blocks. Students tinkered with block sequencing when an action required the use of several blocks in sequence, but students were unsure of the optimum positioning of each block in that sequence.

> Researcher: Can you walk me through how you built this script?
>
> Aisha: This took us ages to get working right. We knew what blocks to use but we weren't sure how they should go. We put these in and then tried them in different ways.

The trial and error approach to modifying variables was used in projects incorporating movement to help students determine the correct timings, sizing and positioning required to create the impression of realistic motion. In the creation of their final projects several groups employed this strategy including Anna, Zara and Fiona who employed it to get two characters to move in tandem, and Georgina, Valerie and Katie who wanted to create the illusion of depth perception by making their sprite smaller. There were some instances where students included a block, whose precise functioning they misunderstood, and for which there was another block that carried out a comparable function with a subtle difference. When students observed that a block wasn't working as expected, they often identified and trialled a similar block.

> Researcher: Can you walk me through how you built this script?
>
> Aisha: We wanted our unicorn to look like it was running by changing costume. First we had this block [points to *switch costume to_* block] but that just made it change to one costume. Then we changed to this one [*next costume* block] and it worked perfectly.
>
> Researcher: Did you know what was going to happen?
>
> Aisha: Not really…but it worked.

This method of trial and error debugging was supported by having the command palette constantly visible (Maloney *et al*. 2008). The visible command palette also encouraged students to explore when they knew what they wanted a sprite to do but were unsure what block performed that function. This type of exploration was typically employed when errors relating to sound, movement or appearance of sprites occurred. Sophie and Bethany employed this strategy when they were trying to make the doughnut fall in their fruit game.

> "We couldn't remember which block so we press the block here [clicks on the *change x by_* block in the command palette] and it show you what it do. Look!" (Bethany, 3rd class)

This instant feedback is useful for encouraging students to explore unknown or forgotten blocks, which can facilitate their testing and debugging of programmes.

5.3.4.2.2 Transfer from Other Activities

In the later weeks of the programming initiative, some of the students began to reference previous programming experiences when testing and debugging their programmes. When they encountered issues with blocks they had used in previous projects, they revisited these projects to resolve the issue. Amber and Allanah took this approach when they incorrectly used the *say_* block.

> Researcher: Can you talk me through how you developed this script?
>
> Amber: First we had the wrong block but I knew we had used this block before so I went back to our knock knock project and saw that I had the wrong block. I should have used the *say_for_secs* block and I just had the *say_* block. So I changed them around.

Anna, who began experimenting with Scratch at home having been introduced to Scratch through the initiative, was exposed to a greater range of Scratch projects and hence computational concepts through her explorations on the Scratch website. She used these experiences to help her solve a problem she encountered when developing a maze game.

> "I explored some of the starter projects at home and they had a maze game so I checked that and I saw that I was missing *forever* [block]." (Anna, 6th class)

These insights from the students provide further evidence of the importance of exposure to programming concepts in supporting students' ability to successfully engage in testing and debugging.

5.3.4.2.3 Assistance from a Knowledgeable Other

However, trial and error and transfer from previous strategies were only useful approaches if the students were able to identify the cause of the error. There were some frequently occurring errors that the students were unable to debug by themselves because they had not yet grasped the requisite computational concepts. Two of the most commonly occurring errors related to omission of the *wait* block and omission of the *forever* block. As previously outlined in section 5.2.7, students struggled to comprehend conditional loops, often omitting the *forever* block from around the *if* block when a project necessitated use of conditional loops. In these instances, the students required assistance from a knowledgeable other to solve the problem. When Katya and Isabela needed a conditional loop in their polar bear game, Bernadette ascertained that their code wasn't working as anticipated because they had omitted the forever block. The observation diary extract from week three, identified that several of the students were incorrectly incorporating the forever block and weren't able to discover the source of the programming bug.

> This week I found that a lot of students were struggling with the use of the forever block in their conditional loops. It is a challenging concept. Even though the original project included the forever loop when they modified the code many of the students put their new feature outside the forever loop or created a new stack without the forever loop. Despite running the project numerous times the students were able to identify the reason for the error and so required my assistance to debug their programme.
>
> (Observation diary, 22nd February 2017)

The continual occurrence of this issue illustrated the importance of the knowledgeable other in extending students' debugging capabilities. In their study, Maloney *et al*. (2008) also acknowledged the importance of the social infrastructure in supporting novice programmers. Schools offer great potential to build strong supportive social networks. A peer-tutoring model, similar to paired reading

initiatives operated by many schools, would allow schools to capitalise on the programming expertise of older or more experienced ‘Scratchers’ in their school community.

The data illustrated that these students adopted diverse and innovative approaches to successfully test and debug their programmes, although sometimes this required support from external sources. Throughout the programming initiative the students exhibited contrasting perspectives on the practice of testing and debugging. While they were often quite frustrated and impatient while debugging their own programmes, they were enthusiastic and excited when they engaged in the debugging challenges in the final week or when they were able to support their peers to debug their projects. Despite this, they were determined and persistent in debugging their own projects and were unrelenting in pursuit of their visions for their projects.

5.3.5 Reusing and Remixing

The preceding section explored how access to the programming competence of others could support students’ testing and debugging practices. In this section, we explore how access to other programmers can boost students’ creativity and exposure to computational concepts. Reusing and Remixing is defined by Brennan and Resnick (2012) as building on the work of others “to potentially create things much more complex than they could have created on their own” (p.8). Martin *et al*. (2014, p.1560) identified four actions that characterise this practice.

1. *Find ideas and inspiration by trying other projects and reading the scripts,*
2. *Select an image, sound or script and adapt it for your project,*
3. *Modify an existing project to improve or enhance it,*
4. *Give credit to people whose work you build on or are inspired by.*

These actions were evident to varying degrees during this programming initiative. While previous research has found this to be the most prevalent of all computational

practices (Litts *et al*. 2020), the poor broadband connectivity in this school limited students' access to projects developed outside their classroom community. Hence, the potential for reusing and remixing was lessened in this programming initiative. However, students' ability or willingness to work on their projects outside of the specified class time, and the opportunities for reusing and remixing structurally embedded in the programming initiative, supported the development of this practice. During inductive coding, two key reusing and remixing practices emerged from the data: building on existing projects and idea generation.

5.3.5.1 Building on Existing Projects

The initial five weeks of the programming initiative were designed to build students' exposure to, and understanding of, programming concepts. To achieve this goal, several highly scaffolded pedagogical approaches were employed, namely copying code and shared programming (Waite 2019), before students were encouraged to modify the resultant projects. Hence, the practice of reusing and remixing was structurally embedded into the programming initiative. These pedagogical approaches engaged students in enhancing projects by building on the work of someone else. Diya and Erica (3rd class) built their final project from the knock-knock animation they completed in week two of the initiative.

> [Diya] and [Erica] have decided to modify their week two project to create a comedy show animation incorporating lots of jokes. They are going to build a larger project from what initially began as one of the starter projects.
>
> (Observation diary, 14th March)

Similarly, Megan and Sylvia built their project from one of the projects completed earlier in the initiative.

> Researcher: Did any other projects inspire your work?

> Sylvia: We used the hide and seek game as inspiration but we added our own style to it. We made levels and a game over background…oh and we added the game cheat…and background music.

Adding features to existing projects was popular across all the class levels with students adding different features to make it their own. Chloe and Maja (3rd class) built on the name animation starter project by designing different animation actions for the letters in their final project.

> Researcher: Can we look at some of the projects you made in the first few weeks? Would you like to show me your favourite one?
>
> Chloe: This is my favourite project. It has both our names and they do all cool stuff.
>
> Researcher: All cool stuff, wow! And how is it different from the original project, the one we all did?
>
> Maja: We made it do all different things to what you show us.
>
> Anna: Yeah look [points to a sprite on the screen], this disappear then reappear in a random position.

Alongside including different actions in their projects, some students made modifications to the original project sprites. Anna, Zara and Fiona (6th class) modified the original fruit drop game to incorporate bad apples which would decrease the score.

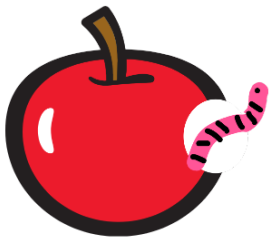


Figure 5.8: *Anna's modified apple sprite*

They designed the bad apple by modifying the original apple image using the inbuilt Scratch paint editor (Figure 5.8). This modification included both a 'cosmetic' modification, changing the appearance of the sprite, and also a structural

modification, introducing negative scoring. While 'cosmetic' modifications were more common, students also regularly made structural changes to the projects.

> These first few weeks have focused on getting the children comfortable with the Scratch user interface and commands, therefore they have had ample opportunity to engage in reusing and remixing. While some of the modifications have been aesthetic such as changing costumes or adding sounds, some children have changed the structure of the programs. For example, [Anna], [Zara] and [Fiona] changed the race game from a race against the computer to a two-player game and [Diya] and [Erica] made a letter of their name fly all around the stage, blow up in the centre of the screen before returning to its place.
>
> (Observation diary, 8th March)

Students' propensity to focus their time on making 'cosmetic' modifications echoes the findings of Meerbaum-Salant *et al*. (2013) who reported similar tendencies among the children in their study. This suggests that specific pedagogical approaches might be necessary to encourage students to make structural changes and consequently progress their computational concept development. While 'cosmetic' modifications were more common the students in this study did demonstrate an ability to incorporate structural changes right from the beginning of the initiative. There were two factors which enabled the structural remixing of projects, even from this early stage. Firstly, the visibility of both the command palette and stage simultaneously (Maloney *et al*. 2008), allowed the students to determine the functionality of certain blocks via the instant feedback.

> Researcher: And how did you know how to make it do that?
>
> Maja: We click it here and look [points to screen] it go somewhere else.

Secondly, students were encouraged to engage in this type of exploration. Every week students were given five minutes after copying the code to freely explore the commands. Their task was to choose at least one new command to include in their project and then explain its function to the pair beside them. This was intended to increase their exposure to the Scratch blocks but in addition it supported their

development of ideas for remixing projects. This finding again highlights the importance of pedagogical decisions in developing students' computational practices.

5.3.5.2 Idea Generation

Aside from these externally mandated examples of reusing and remixing, the students also engaged in reusing and remixing practices that were internally motivated. This was particularly evident during weeks six to nine when students were working on designing and coding their final projects. Students frequently reported getting ideas from seeing other students' projects or chatting to them about their design ideas. When Denise and Michelle (6th class) received feedback that their game was too easy, they designed a new 'baddie' sprite based on an idea they had observed in a project a few weeks earlier.

> "The week of the fruit game Anna made an apple with a worm which made your score go down. We thought we could use that in our project and we added the ghost sprite" (Michelle, 6th class)

Similarly, Lisa and Aisha (6th class) decided to incorporate music into their animation when they noticed that Amber and Alannah had included it in their project.

> Researcher: Did you borrow ideas from anywhere for your project?
>
> Aisha: Ah…I dunno…like what?
>
> Researcher: Any part of your project? Anything.
>
> Aisha: Ah…we put the song in cos [Amber] and [Alannah] had it and we liked that idea.

While these two insights depict students building from existing projects, there were also numerous examples of students sharing scripts during the programming initiative. Talia gave Amber and Alannah (6th class) a script to make their sprite jump, Miriam and Amy (5th class) shared their script to effect a costume change and Isabela and Katya (5th class) demonstrated how to build the code to play background

music. In each of the aforementioned cases, the practice of reusing and remixing enabled students to create more complex projects than would have been possible without this support. The ‘walking gallery’ and ‘learning walks’ (children were encouraged to get out of their seats and see what others were doing and ask questions if they were stuck) were effective pedagogical strategies in supporting this form of reusing and remixing.

Aside from getting inspiration for ideas inside the classroom, some students also reported getting inspiration from sources external to the classroom. Sources included images from the internet, music available to them digitally and other Scratch projects. Scratch facilitates the importing of both images and sound, and the students made use of this function. Sometimes this was fuelled by necessity, Stephanie and Eimear’s story included a pig and there was no pig sprite available in the Scratch library. In other instances, the students had preference for images or music not available in Scratch. This often included images of famous people or characters, or current popular songs. Due to connectivity issues, the download of external images and songs had to be done in the students’ own time. There were also some students who explored the Scratch website outside of the programming initiative. Stephanie and Eimear created a catching game based on an idea that Eimear had seen on one of the Scratch website tutorials. The project inspired the type of project they chose to create and they also reused a script from the project for one of their sprites.

> Researcher: Can you tell me how you got the idea for this project?
>
> Eimear: It was from a project I found on the Scratch website. There were instructions on how to make it. Lots of it was like what we had already done but I didn’t know how to make the pig go all over the place so I wrote that down.

One of the issues that arose regarding this aspect of reusing and remixing was the presence of a digital divide between those who had access to either Scratch or the

internet at home and those who didn't. Zara and Fiona felt their potential to influence their own project was lessened because Anna had access to the Scratch website at home and they didn't.

> Zara and Fiona were a little miffed today because Anna had made some changes to their projects at home because she was 'able to go on the Scratch website and figure things out'.
>
> (Observation diary 29th March)

The potential of the Scratch website for enhancing the creativity and complexity of students' projects was underleveraged in this context, due to poor broadband connectivity. This revealed a 'second order' digital divide (OECD 2015) between these students and highlighted the importance of good internet provision in schools in addressing digital inequality (NCCA 2007a).

The students valued the opportunity to remix and reuse existing projects. They frequently mentioned the "*really cool*" ideas they had discovered, and acknowledged how these ideas would make their projects more "exciting" or "fun" for their classmates. They were quick to verbally give credit to students or projects which had influenced their design or script development during informal conversations or artefact-based interviews. The practice of sharing was embraced by the students with some students "*going from group to group*" to share their ideas (Ms Walker). However, not having access to the online Scratch community limited the reusing and remixing to the 'local community' (Litts *et al*. 2020) with access to the 'global community' (Litts *et al*. 2020) dependent on students willingness or ability to access Scratch outside the programming initiative.

5.3.6 Computational Practices – Conclusion

The findings of this study illustrate that students engaged in all four computational practices, included in the Brennand and Resnick (2012) framework, abstracting and

modularising, experimenting and iterating, testing and debugging, and reusing and remixing. The data indicated that these practices were often inextricably linked with students frequently engaging in two or more practices simultaneously. This was particularly evident for the first three practices. The poor broadband connectivity in the school, curtailed the opportunities for students to engage in reusing and remixing by restricting their access to the global Scratch community. Hence, this practice was not as prevalent as has been reported in previous studies, for example Litts *et al*. (2020) who found it to be the most widespread practice among their students. The exploration of how students engage in these practices, provided insights into several pedagogical factors which positively impacted practice development. These included, the use of storyboards, the implementation of 'gallery walk' and 'learning walk' strategies, and the suitability of Scratch as a programming language for novice programmers. The immediate feedback provided by Scratch and being able to trial the function of certain blocks within the command palette were particularly helpful in encouraging students to experiment and iterate, test and debug, and reuse and remix. In conclusion, similar to the findings on the development of computational concepts, the structuring of the programming experience is central to students' engagement in computational practices.

5.4 Computational Perspectives

5.4.1 Computational Perspectives - Introduction

Computational perspectives refer to the worldviews that students develop as they engage with digital devices (Kafai *et al*. 2016). According to Lye and Koh (2014), there is a dearth of research on the development of computational perspectives. They found that many of the studies exploring the development of computational thinking

focussed solely on computational concepts, with few studies exploring both computational practices and computational perspectives. Zhang and Nouri (2019) reported similar findings. In their review of the literature on computational thinking development, they found that computational perspectives was the most understudied of the three aspects of computational thinking. The lack of research on computational perspectives has been attributed to the challenges associated with assessing changes in one's perspectives (Zhang and Nouri 2019). Lye and Koh (2014) recommend the use of instruments that allow students to be reflective. Kong (2019) recommend that qualitative data collection methods such as surveys and interviews could be used to measure students' computational perspectives. Both of these recommended instruments, along with observation, were adopted and provided insights into the students' computational perspectives which are explored in the coming sections.

5.4.2 Expressing - "It kind of helps children become a bit more creative!!!!!"

Brennan and Resnick (2012) describe expressing as a realisation that technology is a medium of creation. Since the turn of the century, researchers have repeatedly highlighted the need to help our children move beyond 'end-user computing' and become creators of technology (Sáez-López *et al*. 2016; Smyth *et al*. 2001). The creative aspect of programming was recognised by the children who participated in this study. When asked why programming should/shouldn't be taught in schools twenty-five percent of the children cited the opportunity for creativity afforded by programming as one of the reasons why there should be programming classes in schools. Indeed, several students identified the opportunity to be creative as their favourite aspect of the classes. Other students reflected how what they had learned would help them in future creative endeavours such as story writing, drama and art.

When reflecting on what they had learned during the programming sessions many students used the verbs create or make. Their comments showed that they recognised technology as a medium for creation rather than merely something they could consume. They spoke of the opportunities they had to create different artefacts, stories, games, quizzes, cartoons and characters.

> "I learn how to make quiz, conversation and stories" (Joan, 5th class)
>
> "I have rarly used laptops, so I learned quite a lot like how to make funny cartoons" (Isabela, 5th class)
>
> "I learned how to create programs and cartoons, games, movie etc" (Bernadette, 5th class)

The opportunity for creativity in the development of these artefacts was recognised and appreciated by many of the children.

> "It's a good idea (teaching coding in schools) because it kind of helps children become a bit more creative!!!!!" (Eimear, 5th class)
>
> "The thing I liked most about Scratch was that you can use your creativity" (Isabela, 5th class)
>
> "I loved it. It was so much fun. I liked to create new things" (Caroline, 5th class)

When asked to consider what other school experiences coding reminded them of, some students drew comparisons between the other STEM disciplines, Maths and Science, but the majority of students likened it to the creative arts – Art, Drama and Creative Writing. One third class student commented that the classes *"reminds me of doing Art with Ms [Gibson]"*. While another suggested that it reminded her *"of drama where we done little stories"*. The class teachers also made connections between coding and the arts.

> "I really enjoyed the drawing activity, it showed me the potential of using Art and Scratch to not only develop the students' knowledge of 2D shapes and angles but also develop their Art skills based on a particular artist e.g. abstract art." (Ms Quirke)

> "It would be good to use it in the classroom in the arts subject so that they design their own music animations and are learning to program." (Ms McGonagall)

However, some of the connections the children made between programming and creative writing were not creativity focussed. While these children recognised coding as an alternative medium to tell a story, the connections they made were focussed on structure and planning rather than the creative aspect of the tasks.

> "Paragraphs are similar to scene so they can help me write stories" (Isabela, 5th class)

> "Storyboard/Write down ideas for a story before you start" (Katie, 5th class)

Comparisons with the Arts were expected, as both computational thinking and coding have been connected to the Arts in previous research. Millwood *et al*. (2018) referenced the importance of computational thinking in the generation of artwork based on rules and patterns. Papavlasopoulou *et al*. (2019) suggested that many of the practices employed in constructionism-based coding activities can be applied in art. Maloney *et al*. (2008) reported that many of youths in the after-school club involved in their study related Scratch to the arts, rather than computer science or maths classes. These youths likened Scratch to the artistic mediums of paper and sketchbooks, due to the boundless creative possibility it afforded (Maloney *et al*. 2008). However, Meerbaum-Salant *et al*. (2013) suggested that when trying to develop children's computational thinking, some aspects of creativity are more desirable than others. In their paper, they spoke of children's natural tendency to engage in creative exploits such as music and drawing, rather than more specific types of creativity afforded by the technology. These tendencies were also evident during this initiative and were illustrated by the many comments from the children, who identified the opportunity to design characters or choosing backgrounds as their favourite part of the coding classes.

> "I had a lot of fun and I liked picking the backdrops" (Laila, 3rd class)
>
> "I liked being able to design the characters" (Laura, 6th class)

This gives support to the recommendation of Meerbaum-Salant *et al*. (2013, p.70) that it may be beneficial to "de-emphasize aspects of appearance such as the costumes, sounds and visual effects" to ensure that the creativity afforded by this particular medium is being exploited to its full potential.

It wasn't just the students who appreciated the creativity afforded by the Scratch tasks, ten parents and all three class teachers also referenced the opportunities for developing creativity provided by the coding classes.

> "It's good for learning and creativity" (Parent 28)
>
> "She was being creative and interested, also excited about what she had done" (Parent 10)
>
> "She learned how to be more creative and excited about learning" (Parent 31)
>
> "The final project allows for creativity which fires the imagination and interest" (Ms Gibson)

These findings are similar to the findings of the NCCA's study, who also reported that both parents and teachers believed that the introduction of coding to the primary school would benefit the development of children's creativity (NCCA 2019a). Some of the data suggests that the creativity Scratch afforded went beyond previous creative experiences the children had in the classroom.

> "Coding allowed students to develop skills which will allow them to create with technology in more powerful ways than previous ICT education I've used like Word or Paint." (Ms Quirke)

The following diary excerpt based on a classroom exchange between the researcher and fifth class student Katie also suggests that it allowed greater creative freedom than they were used to experiencing in their classroom.

> Researcher: Did you learn anything in Scratch that will help you in your other school subjects?

> Katie: We learned a lot of stuff, but I don't think it would help us in the primary school because we don't do stuff like that anymore.
>
> Researcher: What kind of stuff do you not do anymore?
>
> Katie: Like where we get to play games and make things…and we get to choose what we want to make. We use our imaginations like.

The teachers in this study provided some insight into the reasons why coding fostered creativity. They cited the flexibility of Scratch as one of the key features driving the children's creativity. In particular they referenced that children were able to express themselves by working on projects related to their interests.

> "The design of the task gave them such freedom to express themselves. Their interests and personalities really came through in the projects" (Ms Gibson)
>
> "The flexibility of Scratch really allowed the pupils to express themselves. They were able to create projects related to their own interests" (Ms McGonagall)
>
> "I would love to try this with my class again in the future. I think they would really enjoy creating animations for their personal interests: food, music, sport, book series, or whatever they like." (Ms Walker)

This idea that a programming language should have 'wide walls' to cater for a wide variety of learners was something encouraged by Seymour Papert (Papert 1980). The developers of Scratch asserted that people learn best when working on projects that are personally meaningful for them (Resnick *et al*. 2009). Hence, this was a key priority for them in the development of Scratch (ibid). Therefore, one of their design criteria was diversity, the idea that Scratch would support a variety of different types of projects "so people with widely varying interests are all able to work on projects they care about" (Resnick *et al*. 2009, p64). The comments from the teachers suggest that the use of Scratch for this initiative was effective in achieving that particular aim.

The opportunities for creativity provided by technology have been one of the driving forces behind the introduction of coding to primary schools. The development of

twenty-first century skills such as creativity are identified as important aspects of all three rationales (economic, social and pedagogic) for increasing technology integration in primary schools (OECD 2001). But just increasing technology integration does not necessarily mean that these skills will be developed. Indeed, previous studies of technology use in primary schools found that there was limited use of technology to promote the development of higher-order thinking skills such as creativity (DES 2008a; ERC 2014). Therefore, how children are interacting with these technologies is clearly important. Resnick (2017, p.44) contended that

> if we want children to grow up as creative thinkers, we need to provide them with different ways of engaging on the screen, providing them with more opportunities to create their own projects, and express their own ideas.

The comments of the participants in this study suggest that the Scratch initiative was successful in this regard.

5.4.3 Connecting

Connecting is the computational perspective that captures programmers' views on the "power of creating with and for others" (Brennan *et al*. 2014, para.3). According to Brennan and Resnick (2012, p.10) "creativity and learning are deeply social practices", and therefore designing computational artefacts in Scratch can be enriched by interacting with others. Brennan and Resnick (2012) highlight that these interactions can be face-to-face or online, via the Scratch community network. However, due to connectivity issues, the students in this study were working on the offline version of Scratch, only face-to-face interactions were possible. Despite this, the benefit of working together on projects and the importance of sharing projects with others came across strongly in the data.

5.4.3.1 Creating with Others

The Potential for Idea Generation and Execution

The diversity among learners in every primary school classroom is increasingly being recognised. Indeed, one of the principles of the primary curriculum framework (DES 2023a) focuses on celebrating this diversity and responding, "to the uniqueness of every child" (p.6). The students in this study, viewed the 'diversity' in their classrooms and the 'uniqueness of every child' as a creative strength. They spoke about "*everybody having their own ideas*" (Katya 5th class), "*all bringing something different*" (Aisha 6th class), and that their projects expressed "*a bit of each of us*" (Zara 6th class). When describing their projects, they frequently recognised the contributions of individual students, asserting that they would "*never have thought of that*" (Chloe 3rd class) or they "*couldn't have imagined something like that*" (Fiona 6th class). They were eager to view each other's work, appreciating the opportunity to be inspired by the work of their peers.

> "I liked watching other peoples creative stories, they give me ideas for things to do the next time." (Kelsey, 6th class)

Alongside recognising the greater potential for idea generation, they also recognised that creating with others broadened the creative skillset available. They did so by highlighting what they perceived as their own weaknesses ("*couldn't have got it to look that good*" (Valerie 5th class)) and emphasising what they viewed as their peers strengths ("*she's so arty*" (Rhianna 6th class)). During the creative process they took advantage of the different strengths available to them, both within their pairs and within their wider class group.

> "I made it spin…but [Maja] did the colour and made it look good." (Chloe, 3rd class)

In their artefact-based interview, Kirsten and Laura describe how they 'traded-off' their skills with their classmates.

> Researcher: Can you tell me about this lovely armour on your dragon?
>
> Kirsten: We got [Zara] to do it. She's really good at drawing and…ah…we saw she did it for her Pokémon a while back.
>
> Laura: Yeah, she did that for us and we found her ball. It had…um…gone off [the screen] and they couldn't find it. It was a trade-off like [laughs].

These students illustrated an understanding that working with others gave them access to a greater range of skills and ideas, thus improving both the functioning and the aesthetics of their projects.

The Opportunity of New Perspectives

One of the key themes that emerged from the data was that students viewed creating with others as an opportunity to learn from each other and gain "*unique perspectives*" (Bernadette 5th class) on their work. The value of these 'unique perspectives' was threefold, it increased exposure to programming concepts, it enhanced problem-solving capacity and it encouraged the development of more user-friendly programs. These student perspectives illustrate that they value the opportunity for learning provided by social interaction, a key idea of Vygotsky's sociocultural theory of learning (Vygotsky 1978).

Exposure to Programming Concepts

The first of these dimensions focusses on the opportunity for peer learning provided by connecting with others. In their questionnaire responses and in the artefact-based interviews, the students acknowledged that working collaboratively in pairs increased the expertise that could be drawn on when creating projects.

> "You can do things with you friend and that helps because you can both be good at different things and you can help each other." (Paula, 3rd class)
>
> "It was good to work in pairs cause my pair [partner] was good at some stuff and I was good at others." (Charlie, 5th class)
>
> "I wouldn't have been able to do it by myself, cause Laura knew loads of the buttons [blocks] and…ah…how to get the little guys to move and stuff…but am…I'm better now…I think. I might… be able to try to do my own Scratch [project] now. (Kirsten, 6th class)

During the artefact-based interviews many of the students highlighted particular instances of peer learning which introduced them to new features of Scratch.

> Researcher: All the sounds in your project is great, how did you figure out how to add them?
>
> Amber: Well…Louise and Bernadette had like a scream in their [project] …am…for Alice falling in the hole and we…am…we wanted to give our animals noises.
>
> Allanah: Yeah but they all only had one noise and ah we wanted different ones.
>
> Researcher: So, did you ask them for help?
>
> Allanah: Yeah, Louise showed us to go here [points to the 'Sounds' tab] to add more sounds.

The importance of exposure to computational concept development has already been established by the findings of this study and other researchers (Maloney *et al*. 2008; Meerbaum-Salant *et al*. 2013; Werner *et al*. 2014). The student data in this study shows that these students recognise that their peers provide an opportunity to extend their programming knowledge and build their repertoire of programming skills. This is an important realisation for these young programmers as Denner *et al*. (2014) have previously found that students who programme in pairs increase their programming knowledge more than those who programme alone, particularly those students with low prior computer use.

The students were encouraged to use their classmates as a support network by the 'ask three then me' strategy (Papert 1984) employed during the programming sessions. The 'ask three before me' strategy was challenging to implement, with students often preferring to seek help from the researcher rather than their classmates.

> Today was a very busy in 5th class. All the students were looking for help and I'm not sure how many of them had asked three before me.
> (Observation diary, 29th March 2017)

However, both the researcher and class teacher persisted with the strategy, encouraging students to move around the classroom. Often getting them out of their chairs was the most difficult part, as they were happy to both ask and give support once this was achieved. Indeed, there were many references in the data to instances when classmates helped each other out by "*figuring things out*", "*finding the problem*" or "*showing us what to do*". This support was valued by the students. They recognised that collaborating with their classmates enhanced problem-solving capacity and supported them through challenging moments. Kirsten and Laura in 6th class described how one of their classmates helped them find a bug in their race project.

> Researcher: Were there things that were particularly hard?
>
> Kirsten: Yeah, it was all hard!
>
> Laura: [laughing] There were loads of times when we were stuck and didn't know what to do. Some girls were really good though...
>
> Kirsten: Yeah. Talia was really good wasn't she…and Bernadette.
>
> Laura: Yeah…um...when we couldn't get our race game to work Talia fixed it…um…we hadn't a clue like and she just figured it out.
>
> Kirsten: Yeah, she found what was missing…and she helped us make our dragon jump. Look! [plays the script that makes the dragon jump]

> Laura: Yeah, actually it was great to have everybody that you could ask like…to help and stuff…cause it would fry your brain otherwise! [laughs]. I'd just give up!

There was an appreciation among the students that seeking support from their classmates increased their potential to solve problems. Isabela described her 5th class as a "*community of Scratchers*" and spoke of the reciprocal nature of that support.

> Researcher: Were there things that were particularly hard?
>
> Isabela: I think we were quite good at it…the class like…ah…everybody was good at something weren't they?
>
> Katya: Yeah, I suppose.
>
> Isabela: Like when [Katya] couldn't do something, I could…or if I wasn't [able] we'd find someone who was…[laughs] sometimes you! We were a community of Scratchers, everybody helping each other.

These insights show that, despite the need for some initial encouragement, students were willing to both give and receive support to their classmates and valued collaborative problem-solving. This is a promising finding as previous research on ICT use in Irish schools has reported limited opportunities for collaborative problem solving (Cosgrove et al. 2014b).

Developing User-Friendly Projects

During the creative process, students were keen to both share their own work and see what other students were doing. This meant that at various points during project creation they received feedback on their projects which hadn't been instigated by the teacher-researcher. Much of this feedback centred on the user experience of the projects and the students felt it was a valuable part of the creative process.

> Researcher: Did your plan for the project change while you were making it?
>
> Stephanie: Yeah, it did…a few times…
>
> Eimear: At the start we had no sounds.

> Stephanie: Yeah, [Matilda] told us sounds would make it better. And I think it was good…ah…it made it a lot funnier…the pig snorts like… everybody laughed at those. You have to give the people what they want!

Sometimes, students found the constructive feedback difficult to receive, however they still recognised that the feedback could be used to improve their projects, as illustrated by the following extract from the observation diary.

> [Izzy] told [Megan] and [Sylvia] that their game was 'impossible' and that the ball was 'moving much too fast'. The two girls were quite cross and told her that they weren't finished and to mind her own. Despite this they spent the next ten minutes working out how to change the speed of their ball and then decided that they would incorporate some levels with different ball speeds, something which [Sylvia] later told me she was quite proud of.
>
> (Observation diary, 29nd March 2017)

These findings show that during the creative process, the students spontaneously assumed the role of 'project reviewer', providing feedback to their peers. The students valued these feedback opportunities, often citing how they contributed to project improvements.

An Opportunity to Build Collaboration Skills

One of the connecting perspectives that came across strongly in the data was that creating technology with others helped students develop skills they would need for the future. While the students were cognisant of the creative benefits that could be gained from collaborating with their peers, they also recognised it as a valuable opportunity to build their collaboration skills. Sylvia in 6th class said that she learned "*how to work with other people*" during the programming initiative. Louise in 5th class proposed that primary school children should be taught programming because it helped them "*learn how to work in pairs*". In her artefact-interview Louise and Bernadette were asked to expand on this.

> Researcher: Why do you think programming teaches you how to work in pairs?

> Louise: When we were working together on our Alice project we both had different ideas so I suppose we had to listen to each other and make decisions which we both agreed.
>
> Bernadette: Yeah, and we had to explain to the other person our ideas so we got to practice our talking skills…
>
> Louise: "Our communication like".

These perspectives on collaboration are encouraging as Fields *et al*. (2013) proposed that communication is key to nurturing collaborative design relationships. The teachers and parents also recognised that the programming initiative was a good opportunity for their children "*to work with others*" (Parent 7) "*to get an end result*" (Ms McGonagall). Like Bernadette and Louise, two of the parents suggested that the programming experience had helped their daughters develop their communication skills.

> "It got them communicating with other children." (Parent 22)
>
> "It helped them get used to working and talking with others" (Parent 1)

Sáez-López *et al*. (2016) have previously acknowledged the potential of programming activities to promote the development of 21st century skills such as the ability to communicate effectively. The students also valued this opportunity to support each other, as illustrated by 3rd class students Jasmine and Miriam who both reported that during the programming sessions they learned that "*helping other people is fun*". Indeed many of the children identified peer collaboration on projects as their favourite aspect of their programming experience.

> "The thing I liked most was that we work together and it's really fun" (Evelyn, 6th Class)
>
> "I liked most working together in a group of two and creating new games and stories" (Caroline, 5th Class)
>
> "I like Scratch because you work in partners and it's really fun" (Erica, 3rd Class)

These positive experiences of collaborative programming are encouraging as they might help to challenge the negative STEM stereotypes characterising STEM-oriented people as socially awkward people that work alone (Starr 2018).

The Feeling of Belonging

The data suggested that Kong and Lai's (2021) construct of belonging complemented and augmented Brennan and Resnick's conception of connecting, in particular the *creating with others* dimension. Therefore, it was deemed appropriate to include it within the connecting dimension. The five factors identified by Good *et al*. (2012) were considered when analysing the data to determine the feelings of belonginess among this sample of primary school pupils. While there was little direct reference to feelings of membership or acceptance, many of the children expressed their enjoyment of working on projects with their peers, and they enjoyed feeling their contribution to their group was important.

> "I enjoyed that you worked with your friend and it was both your decision." (Katya, 5th class)

> "I liked the most that it was group work so you had to agree what to do." (Charlie, 5th class)

However, there were some children who reported feeling excluded by their peers.

> "My partner was taking it over so I did not really do Scratch she did she left me out." (Stephanie, 5th class)

> "I was really unhappy with who I was matched up with as I was excluded." (Jean, 3rd class)

This highlights that not all children in the class experienced a sense of affiliation with their fellow programmers. For both children mentioned above, their feelings of exclusion were exasperated by pair incompatibility. As choosing the pairs was at the discretion of the researcher, it was an element of the programming environment that could have been controlled. This evidence supports the views of Mooney and Becker

(2020), who stated that belongingness is a product of environments and experiences and hence educators have a major role in fostering belongingness. Good *et al*. (2012) proposed that feeling happy in a domain indicated a greater sense of belonging. Therefore, the high levels of emotional engagement demonstrated (discussed previously) could signify they had a feeling of belonging with respect to their programming involvement. Trust was described by Good *et al*. (2012) as a feeling that the other members of the community were striving to support their learning and success. There were many comments that showed the children in this study recognised the role of their teacher in providing experiences that would benefit them in the future. In the questionnaire 'any other comments' section, several children thanked their teacher stating that the programming classes "*was a good activity for our brains*", "*helped us to be more smart*" and "*will help us for the future*". The children also acknowledged that their peers support was instrumental in the success of their programming. In their artefact-based interview, 5th class students Katya and Isabela discussed the support that they got from their peers, in particular Bernadette

> Katya: The debugging was hard! It wasn't always easy to find the mistakes we made in our projects…but everyone in our class was really helpful. They wanted our project to be the best it could be.
>
> Isabela: Yeah. Bernadette was really good. She helped us fix our polar bear game.
>
> Katya: Yeah and she showed us what to do for the next time um to put in a forever block.

Similarly, sixth class child Amber recognised the importance of peer support in the design of their sea-themed project.

> "We wanted to make Sebastian jump up and down during the song but we didn't know how. [Talia] had showed [Laura and Kirsten] how to make their dragons jump so we asked her to help us too." (Amber, 6th class)

These accounts from the children show that they were comfortable both giving and receiving feedback, which is only possible where there is mutual trust (Rich 2020).

There was no evidence that suggested that the children had a ‘desire to fade' into the background (Good *et al*. 2012), instead the data showed that the children felt that they could make a valuable contribution and were keen to be active participants in the programming activities. In particular, numerous children mentioned how they liked when they were in control of the mouse or when they contributed ideas related to character or backdrop design. They were also very keen to share their work with their classmates with many of them identifying this as their favourite aspect of the programming initiative.

> “I like parts when we were finishink our projects and we were showink to our class” (Louise, 5th Class)

These insights illustrate that many of the children in this study did feel a sense of belonging in the programming classroom. There were some who had negative experiences with sense of belonging because of pair incompatibility. This highlights the need to be careful during pair selection. Developing a positive sense of belonging is important as previous research has shown that sense of belonging has a significant impact on persistence, particularly for women (Rainey *et al*. 2018).

5.4.3.2 Creating for Others

Brennan and Resnick (2012) propose that through Scratch programming young people experience the value of authentic audience in four distinct ways: entertaining others, engaging others, equipping others and educating others. Only the first of these was evidenced in this study, possibly due to the context of the study. They had no access to the online community which limited their knowledge of the different types of projects that can be created through Scratch. Furthermore, their initial programming experiences, except for the final project, were all teacher selected tasks. This meant they had little opportunity to explore or create ‘engaging’ or

'educating' projects. Finally, as they were novice programmers, they had yet to develop sufficient programming skills to create 'equipping' projects.

Entertaining Others

The opportunity for students to share their work was viewed as an enjoyable and valuable aspect of the programming sessions. Several students identified the sharing of projects as their favourite part of the programming sessions.

> "I liked…watching the stories we make after" (Katya, 5th class)
>
> "I enjoyed most watching it all together at the end" (Erica, 3rd class)

The teachers also recognised it as a valuable part of the creative process, seeing it as an opportunity for children's hard work to be recognised.

> "The finished products were amazing and it was lovely for the children to see each other's end results" (Ms Walker)
>
> "It was a great opportunity for the girls in my class to be involved in the Scratch program. The gallery walk was a great confidence builder as they really value those moments of recognition from the others" (Ms Gibson)

During the creative phase, the students were mindful that they were creating for an audience. This was evidenced by student comments captured in the observation diary and through data from the artefact-based interviews. In the creation of animations and stories, humour, the element of surprise, popular music and well-known characters were often incorporated to enhance peer enjoyment. Isabela and Katya in 5th class downloaded songs popular among their classmates to include in their projects.

> Researcher: I like the background music. Why did you choose that song?
>
> Isabela: "This is our class song. Like our favourite-est song!"

They were also one of many groups who were keen to include elements, they thought their classmates would find amusing, in their projects.

> Now that we are a few weeks in, many of the students are considering their audience when designing their projects. [Isabela] and [Katya] have downloaded popular songs to include in their projects. The kids loved it and now everyone wants to do it. They also love to include a bit of fun in their projects. As Laura says, "It'll give everyone a laugh later". How their peers react to their projects during the presentations or gallery has become really important to them.
>
> (Observation diary, 1st March 2017)

The younger students in 3rd class focussed more on images rather than music, with many of the groups in this class importing images of well-known characters to enhance their projects.

> Researcher: Who's this little guy?
>
> Tara: That's Alvin…you know from the chipmunks.
>
> Sarah: We chose him cause everyone likes the chipmunks.

In the designing of games, player affirming messages, scoring and cheat codes were all incorporated, again with player experience in mind. Both sound and visual congratulations were included in many of the projects in the weeks that focussed on game design.

> Researcher: Oh, this is lovely, a cheer and a well done when you win.
>
> Bethany: We want to celebrate the winner!

The students who had gaming experience were keen to imitate the gaming features which they enjoyed from their experience as a gamer.

> Researcher: Can you tell me about your game?
>
> Sylvia: So our game is a ping pong game and you have to stop the ball from passing your…ah…paddle…like this [demonstrates how to hit the ball]. And every time you do that you get a point. See your score goes up by one [points to score at the top of the screen]. And it has six levels and some of them are really hard. So, we…you can't tell anyone…but we included a game cheat, so you can skip by this level by pressing 'h'.
>
> Researcher: And why did you decide to include those features?
>
> Sylvia: We wanted it to be funner and not too hard for people. I play Mario and there is codes to help get to end.

This data shows that the audience experience was important to these young creators. They valued the opportunity to entertain, and be entertained by, their classmates, often citing it as their favourite part of the programming initiative. They utilised popular music, humour, characters and gaming features to make their projects more attractive to their audience.

5.4.4 Questioning

Brennan and Resnick (2012) describe questioning as feeling "empowered to ask questions about and with technology" (p.11). In evaluating the perspective of questioning, Brennan and Resnick (2012) propose looking for

> indicators that young people do not feel this disconnect between the technologies that surround them and their abilities to negotiate the realities of the technological world.
>
> (p.11)

During inductive coding, three progressive levels of questioning perspectives emerged from the data, and were categorised as functional, exploratory and empowered perspectives.

5.4.4.1 Functional Perspective

Students who held this first perspective, viewed their relationship with technology at a functional level. These students felt that through the programming initiative they were developing skills which would allow them to become more competent technology users. Many of the students viewed the programming initiative as a valuable opportunity to become familiar with computers.

> "The coding classes were good because you learn to use a computer" (Kornelia, 3rd class)
>
> "Coding classes help children learn the basics of computers" (Charlie, 5th class)

This was also a commonly held perspective among the parents of the students.

> "It increased their [child] confidence in the use of the laptop" (Parent 20)
>
> "I think coding in schools is a good idea because my child would have better abilities of using computers/laptops/other electronic devices" (Parent 2)
>
> "My child thought it [programming initiative] was good to learn more about computers" (Parent 16)

The view that these coding classes were an opportunity for students to learn the basics of computing illustrates a lack of confidence among these students, despite growing up surrounded by digital technology and being labelled as 'digital natives' (NCCA 2019b). Indeed, according to their parents, 65% of these students interacted with technology for more than an hour a day. With 32.5% of students spending more than two hours engaging with technology every day. However, despite their frequent use of technology, they lacked confidence in their technological abilities.

> These first few weeks have very much been about developing the girls' competence and confidence in using technology. Many of them lack belief in their ability to use technology not to mind create with technology.
>
> (Observation diary, 21st February 2017)

These findings corroborate the views of Kafai and Burke (2016) who suggest that technology access does not necessarily equate to purposeful use of technology. They also give support to the social rationale for the inclusion of ICT in education, by highlighting the importance of experiences such as this one, in students' technological growth.

5.4.4.2 Exploratory Perspective

Wells (2012) made a distinction between ICT and 'computing' in education, describing 'computing' as going beyond the use of hardware and software to consider how they are built and programmed to work. This distinction was recognised by the participants in this study, who viewed the programming initiative as "*a chance to explore new ways with computers*" (Jane, 6th class). Similarly, Ms

Farrell viewed the programming initiative as "*a learning aid to develop computing skills as opposed to just ICT skill*". Students with an exploratory perspective illustrated an awareness of the 'computing' aspect and viewed the initiative as an opportunity to develop their programming abilities. Shauna in 3rd class believed that programming classes should be part of the primary school curriculum because then children would have an opportunity to "*learn how to code*". This viewpoint was evident across all class groups.

> "There should be classes because you get to know whats coding with computers" (Eimear, 5th class)
>
> "Coding classes help children learn the basics of computer coding" (Megan, 6th class)

According to their parents, the students appreciated the opportunity to learn "*how computers work and apps are built*" (Parent 15) and become "*familiar to the base of a game and how it's created*" (Parent 18). Indeed, there were many comments from students which illustrated they were considering the programming behind familiar technologies and contemplating how they could use their newly acquired programming skills for creative purposes.

> "I learned that alot of games are used by coding and that I could make my own game" (Louise, 5th class)
>
> "You learn how to do programs, how to make programs and cartoons, games, movie etc." (Matilda, 5th class)
>
> "It intresting [interesting] to learn something new. I learn how to program game and I learned how games are made" (Alisa, 5th class)
>
> "We learned how to do animation. I learned how to animate caricters [characters] and make do whatever I want them to do" (Erica, 3rd class)

These views were echoed by both the students' parents and their teachers, who recognised that the students were developing insights into creative programming.

> "It gives the children a better understanding how programs work. They learn how to put a program together. How to create a movie/film/video!" (Parent 38)

> "They saw first-hand the process involved when devising an animation story, they now realise what is involved in computer programming" (Ms Walker)

While most students reflected positively on their relationship with technology, Kirsten, one of the 6th class students, proposed that programming was more difficult than she had anticipated.

> "I learnt that its hard to make games and movies than it looks life." (Kirsten, 6th class)

This illustrates that some, but not all, students experienced an increase in their self-perception as technology creators, similar to the findings of (Kafai *et al*. 2014).

5.4.4.3 Empowered Perspective

The terms 'to empower' and 'empowerment' are used frequently in the technology education literature and policy documents. Indeed, it is identified as a key objective in several recent education policy documents, including the 'Primary Curriculum Framework' and the 'Digital Strategy for Schools to 2027'. Within the context of technology education, empowerment means instilling confidence in young learners in their ability to solve real-life problems and contribute positively to a digital world (Kenna 2022; Kong 2019; NCCA 2019a). The students in this study were not yet thinking about how they could use technology to "change the world in which they live, for the better" (NCCA 2019a, p.82), but they were starting to think about the "*endless possibilities*" (Niamh, 6th class) opened up to them by technology. Shauna in 3rd class felt that the classes had taught her "*that you can make anything from games and computers*". The possibilities of technology, particularly the creative possibilities, were frequently recognised by the students.

> Researcher: Have these classes got you thinking about what sort of things you could do with Scratch or coding in general?

> Katya: Yeah, like I suppose I thought coding was really hard even for old people, but I learnt it was very easy to make game from Scratch.
>
> Isabela: Yeah, we are thinking now…like what can you make from Scratch…you could make anything!
>
> Researcher: I could or you could?
>
> Isabela: We could.
>
> Katya: With a few more classes [laughs].

Several of the parents also acknowledged the role of programming classes in making their children more aware of the potential of technology.

> "The classes make my child see what can be achieved by technology. What she can make by technology" (Parent 16)
>
> "Appreciating the how/why of computer processes so they open themselves up to understanding more and not seeing limitations" (Parent 14)

These insights illustrate that the students are becoming more confident in their technical abilities, they are starting to view themselves as active participants in a digital society and not just as users but as creators. Hence these findings suggest that programming initiatives such as this one can contribute to achieving the 'Digital Strategy for Schools to 2027' objective of empowering "learners to become confident and competent digital learners" (DES 2022c, p.21). The 6th class teacher proposed that the programming experience would also encourage them to critically evaluate the technology they are using.

> "It gives them insights into how the technology and apps they use works, allowing them to become more critical consumers" (Ms Walker)

Indeed there was evidence from the artefact-based interviews that the students were demonstrating a more critical perspective of technology. Louise in 5th class acknowledged that much of her prior technology use had been passive, and she highlighted the importance of more creative technology experiences for young people.

> "I enjoyed making the story. A lot of time you're like watching other people's stories…like on YouTube and…like that's fine but you just...am…you just get their ideas then…it's good to get your own ideas…kinda be a bit more creative like" (Louise, 5th class)

Michelle and Denise in 6th class discussed how sharing their work with their peers highlighted to them the importance of game elements and aesthetics in the game design. By recognising the importance of design culture, they are demonstrating their critical perspective on the values that influence technology design.

> Researcher: Did these classes get you thinking about what sort of things you could do with Scratch or just coding?
>
> Michelle: Yeah…especially the part where we showed our games. That make you think what is good to make people enjoy your game. What character you use, what backdrop, what music…do you want score? You think what will people like?
>
> Denise: Yeah like the big companies you're think how do I make children like my game.

These are positive findings given previous research has highlighted the importance, and difficulty, of equipping citizens to make well-informed decisions and to think critically about technology (NRC 2002).

5.4.5 Computational Identity

Kong (2019) suggested that Brennan and Resnick's (2012) framework neglected to capture an individual's motivational beliefs relating to learning programming. Therefore, Kong (2019) proposed that computational identity should be included as a computational perspective to include engagement, imagination and affiliation. However, the findings from this study suggest that imagination is already captured by Brennan and Resnick's (2012) 'expressing' dimension, and affiliation aligns with and extends the connecting computational perspective. Therefore, the data from this study provides strong evidence for the inclusion of motivation to engage rather than computational identity as the final dimension of computational perspectives. This

section explores the developing engagement of the children who participated in this initiative.

5.4.5.1 Motivation to Engage

Behavioural Engagement: "It was fascinating to see the ones that really got into it"

The behavioural engagement constructs identified by Fredricks *et al*. (2004) were used to guide the researcher's interpretation of students' behavioural engagement. It was challenging to obtain directly from the students themselves their levels of behavioural engagement. Instead, insights on behavioural engagement mostly emerged from researcher observations or from teacher comments. There were several entries in the observation diary that referenced the children's engagement in tasks. Examples of behaviour engagement constructs mentioned in the diary were: '*listened attentively to the instructions*', '*the children were really attentive during the race game demonstration*', '*the children were engrossed in the design of their own fruit drop game*', '*some of the children didn't want to finish up with the activity today*', '*wanted more time to finish the task*' '*listening carefully and asking questions*' and '*there were some excellent questions asked*'. There were some comments from the children's data that illustrated that they were engaged by the programming activities:

> "I mostly disliked when we sometimes couldn't finish our stories"
> (Natalia, 3rd class)
>
> "The only thing I didn't like about Scratch was not doing it all day"
> (Katie, 5th class)

Both Hjorth (2017) and Ching *et al*. (2019) portrayed similar scenes in their programming initiatives. Hjorth (2017) described the children in her study as highly focussed and highly curious. While a teacher in Ching *et al.* (2019) spoke of students wanting to stay longer to continue working on projects and their eagerness to

embrace the next challenge. However, it is important to acknowledge that the context and participants of these studies differed from this study. Both studies were conducted in informal settings, the majority of the participants were male, and the participants had self-selected to engage in the programs.

All three class teachers admitted to being pleasantly surprised by how engaged the children were during the programming lessons. Ms Walker, in 6th class explained how she thought that the programming initiative might pique the interest of a few students, while for the rest it would be a good opportunity to gain experience working with computers.

> "To be honest I thought it would be no harm for them all to have an opportunity to work with the computers, but I thought that the coding would only interest a few of them, that only some of them would get into it. But I suppose with the animations, games and stories, with the musical element and the choice of characters, there was something for everyone and they all got stuck in." (Ms Walker)

This mirrors the experience of the teachers in the NCCA's (2019a) research on coding in primary schools. These teachers also reported being surprised by the high levels of engagement of their students with some of them commenting "it was the most engaged they had ever seen their students" (p.29). Ms Walker's observation provides further evidence that the 'wide walls' of Scratch are effective in engaging a diverse group of learners and could potentially help attract non-traditional students to programming. According to Lusa Krug *et al*. (2023) the use of outside domains such as music, robotics and gaming has been found to be effective in broadening participation in computer science. This strategy is based on the premise that engaging students in activities that they enjoy leaves them with a more favourable impression of programming (Lusa Krug *et al*. 2023).

The teachers in this study also observed that student engagement during the programming classes didn't necessarily reflect student engagement during other lessons. They observed that the higher levels of engagement came from unexpected sources. Children who were often quiet in day-to-day classroom activities demonstrated particularly high levels of engagement.

> "It was fascinating to see the ones that really got into it and the ones that didn't as much. It was a totally different dynamic from our usual class. Some of the girls who were leading the groups and coming up with all sorts of ideas, you wouldn't hear them in a normal class." (Ms McGonagall)

The teachers identified those who struggle academically and English as an Additional Language (EAL) learners and as two groups who exceeded their expectations for them, and performed particularly well in the programming classes.

> "I was surprised, two or three of mine who would struggle academically did really well at the Scratch. It just seemed to click with them." (Ms Gibson)

> "Well I know there is Maths in Scratch but they seemed well able for it. They didn't seem to notice and normally they would be very reluctant to participate in anything involving Maths." (Ms Gibson)

> "Little [Talia], she hasn't got much English so she would normally take a backseat in any group work, but she was going from group to group showing them how they could make their character jump like hers." (Ms Walker)

The NCCA (2019a) also identified low academic achievers as a group who were particularly engaged during their programming initiative. The teachers involved in that study thought the different nature of these classes compared to the "normal classroom activities" suited those learners (NCCA 2019a, p.29). Interestingly, Maloney *et al*. (2008), when reporting on programming in after school settings, suggested that that out-of-school activities such as programming "present opportunities for youth to succeed who may not flourish in traditional school environments" (p.367). EAL learners comprise 46% of students in this school and the teachers of all three classes were initially concerned that the language in Scratch

would pose problems for these learners. Although Scratch is available in more than seventy different languages, all students who participated in the study used the English version of Scratch. Despite initial concerns, the EAL learners managed to overcome any potential language difficulties and created programs that were as sophisticated (by Dr. Scratch score) as their English as a Native Language (ENL) classmates. Wilson and Moffat (2012) had previously found that some of the 'lower attaining' students had flourished in the programming classes in their study, in particular citing one child who had difficulty with language. However, they also reported that the 'academically strongest children' performed poorly in these classes, something which wasn't evidenced in this study.

According to the data, there were two features of Scratch that supported the EAL learners during the programming activities. One of the key heuristics of Scratch that were useful to these learners was the instant feedback provided by the program. Scratch allows for continuous execution, that is users can continuously run segments of code to ensure it is working as intended. Often the EAL learners used this aspect of Scratch to work out the function of the blocks.

> "If I don't know what [a Scratch block] does I click it and it show me [on the stage on the Scratch interface]" (Talia, 6th class)
>
> "If you don't know just drag it out and play it like this (clicks on the block)" (Joan, 5th class)

All students used continuous execution to test their code as they added to it. Previous researchers have identified the availability of timely feedback as an important element of any novice programming language (Kölling and McKay 2016). However, while this trial and error approach has benefits, it can also lead to issues with troubleshooting caused by redundancy and cluttered code (Hjorth 2017). While redundancy and cluttered code did not appear to cause issues for these students, they

did run into difficulty when execution was failed to identify their problem. In these instances, the children relied heavily on support from their peers or the researcher. This highlights the need for teachers to have specialised content knowledge, knowledge that enables the teachers to recognise errors and support children in addressing these issues (Ball *et al*. 2008). A second heuristic that the EAL learners felt supported their understanding was that Scratch blocks would only fit together in particular ways.

> "Scratch help me cause look I know this [block] can't go here cause no work [illustrates her point using two blocks that won't fit together] (Megan, 6th class).
>
> "You need this at top…look see it's round…nothing fit on top [she identifies the defining rounded shape on top of the 'hat blocks' and understands this means it must be at the start of a script] (Maja, 3rd class)

Kölling and McKay (2016) suggest that programming language developers should prioritise error prevention over error reporting. As Maloney *et al*. (2010, p.5) observe

> When people play with LEGO® bricks, they do not encounter error messages. Parts stick together only in certain ways, and it is easier to get things right than wrong. The brick shapes suggest what is possible, and experimentation and experience teaches what works.

This was indeed the case for the learners in this study, particularly the EAL learners who found that this visual representation supported their understanding of what would and wouldn't work.

Cognitive Engagement: "The Struggle is Real"

When the Scratch team at MIT set out to create their programming language, they wanted to make it accessible to all (Resnick *et al*. 2009). Hence, they wanted to create a programming language with a "low floor", meaning that it would be easy to get started. However, they also felt their programming language needed to have a high ceiling, meaning it would allow for the creation of complex programs. Although

some have questioned whether this aim of high-ceiling programming has been achieved by the Scratch programming language (van Zyl *et al*. 2016), it would appear that for this group of students there was indeed a high level of complexity. As Zala in 3rd class explained "*it looks easy to do it but it is hard until you get the hang of it*". The results from Dr. Scratch illustrate that the students in this study were grappling with programming concepts that are perceived as difficult, particularly for their age group (Lawanto 2016; Seiter and Foreman 2013; Serbec *et al*. 2018). In the interviews and student questionnaires many of the students expressed that they found the programming tasks difficult. These difficulties ranged from things not working as they expected, "*things kept going wrong*" (Valerie, 5th class), the length of time it took them to complete tasks "*it took ages to make one story*" (Valerie, 5th class), lack of familiarity with Scratch blocks "*I didn't like figuring out what boxes [blocks] work together*" (Sylvia, 6th class) and "*I didn't enjoy the work, like which buttons [blocks] to use. It's very confusing*" (Kirsten, 6th class) or not having the requisite understandings to solve particular problems "*sometimes I did something wrong and couldn't fix it*" (Shauna, 3rd class), "*it was hard to change something and sometimes it was stressful*" (Paula, 3rd class). However, these frustrations didn't seem to detract from the students' enjoyment of the programming experience as each of these students expressed positive reactions to their experience in their questionnaire responses. For example, in her post-questionnaire, Caroline described what she found least enjoyable about Scratch:

> "It was a little difficult and sometimes things were going wrong and it took a little while to create a story or game." (Caroline, 5th class)

Despite the difficulties that Caroline experienced during the initiative, she selected the option 'Brilliant' when responding to the questionnaire questions 'Did you enjoy your coding classes?' and 'Do you think it is a good idea to have coding classes in

school?', and she indicated that she would love to do coding next year. Izzy in 6th class expressed similar views. Like Caroline she selected 'Brilliant' in response to the two previously identified questions, and she described the mix of emotions she felt when faced with a new programming challenge:

> "I didn't enjoy when we were starting new projects and there was going to be so much to do and I knew it was going to be hard but I was excited for our new project too and all our ideas and being able to show it to our class". (Izzy, 6th class)

This mix of emotions was common among the students. Lena in 3rd class described programming as both "*fun and stressful*". These comments suggest that the complexity of the tasks did not discourage the students, rather that they embraced the challenge these tasks provided. Several researchers have reported similar findings in the field of programming. Resnick (2006) described the challenge of computing tasks as one of the attractions. While Brennan (2013) also reported that children's desire for challenging tasks was a recurrent theme in her research. Similar to Izzy's experience, many of the children in Brennan's study spoke about the satisfaction gained from "struggling toward and completing a project" (Brennan 2013, p.81). These findings would seem to corroborate the view expressed by Seymour Papert, in an article written for the Bangor Daily News, that this type of learning is "fun because it is hard rather than in spite of being hard" (Papert 2002, para.2).

Perseverance

In this study, there was an abundance of evidence to support the inclusion of perseverance as a dimension of computational perspectives. As outlined in the previous section, programming is hard, and the children acknowledged that. But they also showed they were willing to grapple with these difficulties and continue moving toward achieving their goals. Katya in 5th class felt that the coding initiative helped

her develop strategies for dealing with the complexities and challenges she faced during coding:

> Katya: We learned how to use different strategies and what you should do if something goes wrong…you know…how to fix it.
>
> Researcher: What kind of strategies were they?
>
> Katya: Like…not to give up. Like when we were making our 'Polar Bear' game. We really wanted the Polar Bear to catch the salmon but it took us ages. We couldn't work out what to do. But we kept trying. We had the wrong block but we kept trying until we got it.

This was a view shared by a parent of a 3rd class student who spoke of how the initiative helped her daughter learn "*what to do when something goes wrong, how to deal with that and not give up*". Lena in 3rd class also referenced the need for perseverance when she described how she felt when she was completing her final project:

> "I had fun doing my animation but sometimes it's a little bit stressful. I think I will continue working on it to make it better. Sometimes when you want to glide sometimes it will go upside down. I will keep trying to figure out the glitches"

Brennan (2013) described similar 'persistence', 'grit', 'patience' and 'determination' demonstrated by her participants in overcoming the challenges they faced in the development of their projects. Ching *et al*. (2019) reported a similar willingness to persevere when faced with significant challenge among the students who participated in their STEM integrated robotics sessions. This was illustrated through the teacher's descriptions, who depicted children smiling through their frustrations. Indeed, the well-known saying 'smiling in the face of adversity' succinctly captures the character shown by the students in this study. While research on programming has proposed that learning programming requires perseverance, few studies have explored how programming can foster perseverance. The observation diary and artefact-based interviews provide some insight into what motivated these students to

persevere. Michelle in 6th class spoke about persevering with her project despite the difficulties that she and her partner had faced.

> Researcher: Tell me about your final project, your basketball game.
>
> Michelle: (Sighs) It was really hard to make!
>
> Researcher: But you made a really lovely game.
>
> Michelle: Yeah that was good. But like even the small things were really hard. Like getting the ball and the penguin to move together…that was the hardest part. Things kept going wrong and we couldn't fix them. We just had to go for a walk and look at what everyone else was doing. We always came back to it again. Like you know…we had an idea and it was just something we wanted to do so we kept trying until we got it. It was really good to be able to show it to everyone at the end.

There were several entries in the observation diary that described similar instances of struggle and perseverance.

> Bernadette and Louise were tenacious today. They weren't going to stop until they had Alice falling down that rabbit hole exactly as they had pictured.
>
> (Observation diary, 22nd March 2017)

In each of the aforementioned situations the children showed perseverance in pursuit of their goals and this was a regular theme during the programming classes. It was common to see children working on a minute detail of their project for long periods, to achieve a specific outcome they envisioned for their project. The importance of creating personally meaningful projects emerged as a factor that contributed to their desire to succeed.

Emotional Engagement: "I Love Everything in Scratch"

Affective Reactions and Interest

Although not included in Brennan and Resnick's (2012) framework, children's affective reactions regarding programming came across strongly in the data. The findings from the questionnaires, observation diary and artefact-based interviews

provided insights into these affective reactions. These findings show that for the majority of students their intrinsic value of coding was high, they enjoyed engaging in the coding classes. This corroborates the previous findings of Calder (2010) and Wilson and Moffat (2010). This result is perhaps unsurprising, given that the developers of Scratch set out to create a programming language that "would appeal to people who hadn't previously imagined themselves as programmers" (Resnick *et al*. 2009, p.60). In the questionnaire students were asked to indicate their responses to the question 'Did you enjoy your coding classes?' using a 5-point Likert scale (awful, not very good, good, really good, brilliant). The responses to this question were predominantly positive. 66% of students said that they thought the coding classes were 'brilliant' (see Figure 5.9). 18% felt the coding classes were 'really good' (3rd class, 17%, 5th class, 19% and 6th class, 38%). 14% of students said they found the coding classes 'good' (3rd class, 9%, 5th class, 19% and 6th class, 48%). One student described the classes as 'awful', citing partner incompatibility as the reason for their negative experience.

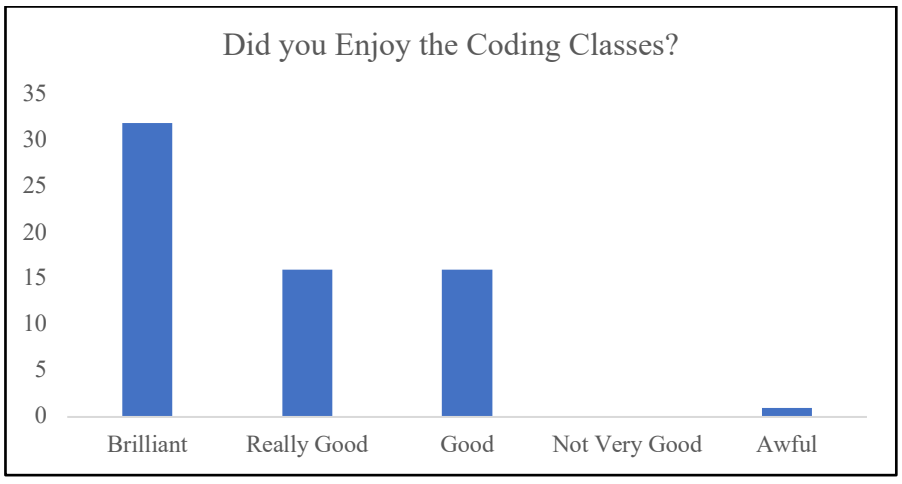


Figure 5.9: *Students' Enjoyment of the Programming Initiative*

Enjoyment was highest in 3rd class (70%), followed by 5th class (62%) and 6th class (14%) (see Figures 5.10, 5.11 and 5.12).

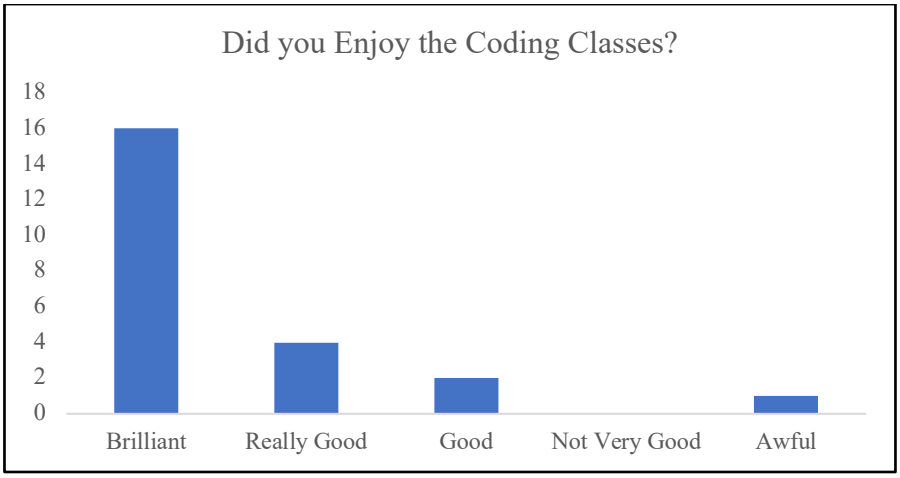


Figure 5.10: *3rd Class Students' Enjoyment of the Programming Initiative*

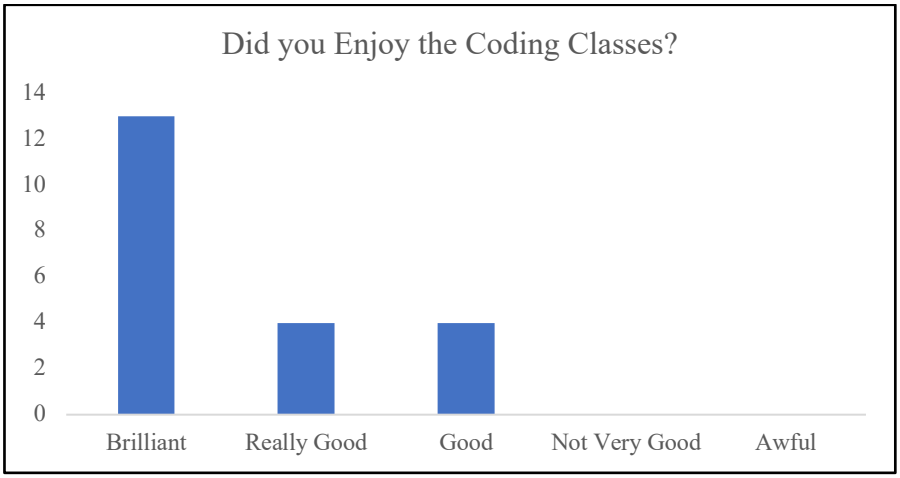


Figure 5.11*: 5th Class Students' Enjoyment of the Programming Initiative*

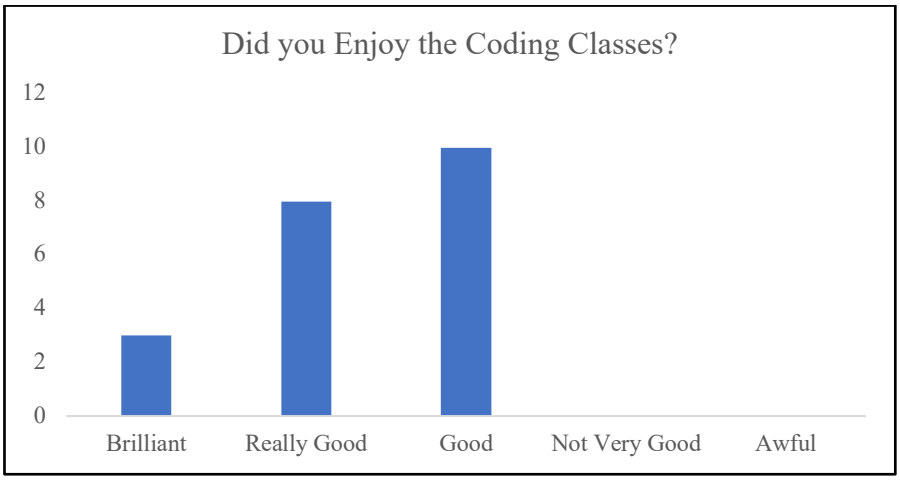


Figure 5.12: *6th Class Students' Enjoyment of the Programming Initiative*

The students were also asked if they felt they should learn about coding in school. 94% of students agreed with the statement while 4% disagreed (see Figure 5.13).

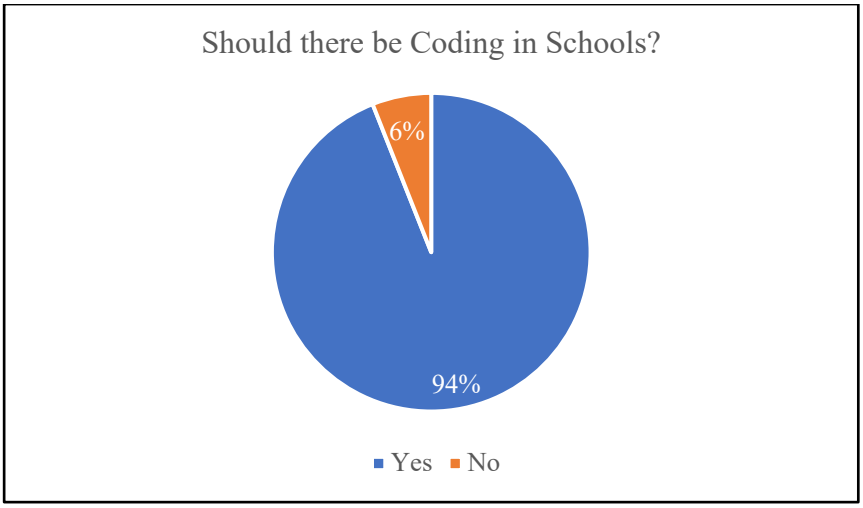


Figure 5.13: *Students' Views on Learning Programming in School*

The reasons identified by the children who felt coding should not be included in the curriculum related to the negative effects of technology, teacher skills and interest.

> "Kids will always be on computers instead of going outside" (Suzy, 5th class)
>
> "Computers are bad for your eyes" (Alannah, 6th class)
>
> "it's hard and you need experienced computer people or else it would be even harder" (Kirsten, 6th class)
>
> "It was boring" (Amie, 6th class)

Many responses to the open-ended questions on the student, parent and teacher questionnaires also revealed positive attitudes toward programming. In their responses, expressions such as 'love', 'fun', 'really good' and 'interesting' were frequently used by students, across all classes, to describe their experience of coding. Erica in 3rd class said, "*I love everything in Scratch, everything is my favourite and nothing's my favourite*". Katie in fifth class said, "*the thing I liked least about Scratch was not doing it all day!*". Anna in sixth class said, "*everyone should do coding in school because it is really fun and you learn more about coding classes*". The parents' responses to the questionnaire reflected the predominantly positive attitudes expressed by their children. Their responses frequently used the terms 'interest', excited', 'enjoy' and 'fun'. Parent 6 commented, "*I really think my child*

should keep doing coding in primary school because she really enjoys it". Parent 28 reported that their child "*was being creative and interested, also excited about what she had done*". Parent 37 said their daughter had learned "*how to code and enjoy it*". Comments from the three class teachers echoed those of the students and their parents. Ms Walker said, their students "*all enjoyed the Scratch lessons immensely*" and Ms McGonagall said, their students "*got on very well and were very interested in it*". The following excerpt from the observation diary captures the children's enthusiasm for the classes from the researcher's perspective:

> When I arrived today I met 5th class lining up in the yard, waiting to go in from break. They all wanted to tell me how much they loved Scratch. They were wondering how many more weeks we had left and if they would get to do it next year again. The response to the classes has been really positive among all the groups.
>
> (Observation diary, 29h March 2017)

It is necessary to highlight that some of the children's interest could be transitory, aroused by the novelty of the programming initiative. Indeed, some of the participant comments suggest that at least some of the children's interest was situational.

> "I think that all children should get to do coding because it is interesting to learn something new." (Charlie, 5th class)
>
> "My child enjoyed the classes it was something new" (Parent 5)

However, there was also evidence of emerging personal interest as some students pursued or planned to pursue other opportunities to engage in programming.

> "[Zara] was excited about learning. After the class she worked at home to finish things in her project" (Parent 31)
>
> "They really enjoyed the classes and they're looking forward to doing Tech camp over the summer." (Parent 20)

These findings suggest that programming initiatives, such as this one, have potential to grow personal interest, although it would require longitudinal study to confirm the permanency of this interest.

Value of Programming

The final dimension of emotional engagement identified in the literature is value, which included intrinsic value, attainment value and utility value (Wigfield and Eccles 2000). In the previous section, children's emotional reactions to the programming activities were discussed. As many of those reactions were positive and spoke to the enjoyment the children gained from engaging in the activities, the children's perspectives on the intrinsic value of programming experience have already been established. Both Wigfield and Eccles (2000) and Fredricks *et al*. (2004) acknowledge there is potential duplication of constructs in the definitions of both engagement and value, and this proved to be the case for this study. Attainment value was the least expressed value type in the data, perhaps because the instruments used did not provide the opportunity to measure this construct. While none of the children commented directly on the importance of doing well, their persistence in achieving the high-quality graphic design they envisioned for their projects and their enthusiasm for sharing their finished products (which are discussed later) suggest that it was important for them. Two noteworthy instances which demonstrated this persistence were Georgina, Valerie and Katie who were intent on making the flight of the ball in their animation look more realistic and the determination shown by Anna, Zara and Fiona to perfectly synchronise the timing of the actions with the sound in their animation. These two vignettes illustrate the importance to the children of creating quality finished products. Other examples of persistence in pursuit of precision have previously been discussed and no further evidence of attainment value emerged from the data. Consequently, this section will focus on the utility aspect of value.

Utility Value

The participants in this study demonstrated high levels of utility value, in that they recognised the potential value that engaging in the programming initiative could have on their future endeavours. In particular, their utility value perspectives focussed on two aspects, the economic and social value of technology.

Societal Value of Technology

Digital proficiency has become a necessity as technology has become increasingly prolific in our daily lives. The social rationale for the inclusion of ICT in education, referred to repeatedly in policy and research documents, illustrates a societal recognition of the "need to broaden participation in the technology culture" (Kafai and Burke 2013, p.603). Kafai and Burke (2016) emphasise that participation means more than just access to technology and extends to children's use of this technology. This distinction is widely recognised in computational thinking policy and research documents and acknowledges that children should be encouraged to become active producers of technology rather than passive consumers. Data from the pre-initiative and post-initiative questionnaires illustrated changes in students' perceptions of the value of programming in this regard.

Development of Technical Skills

Prior to the commencement of the programming initiative the children valued the development of technical skills, the skills required to operate digital devices (Weber and Greiff 2023). In their questionnaire responses, students frequently expressed the view that learning programming in school was an opportunity for children to improve their computing skills. Responses such as "*it helps children with their*

computer skills" (Alisa, 5th class), "*it will help you to learn laptop skills*" (Rhianna, 6th class) and "*it teaches children more about technology*" (Katie, 5th class), and "*because a child can learn more about technology, I have rarely used laptops*" (Isabela, 5th class) were common when the children were asked if they felt coding should be introduced in primary schools. Several of the parents also valued the opportunity the initiative provided for their children to improve their technical skills.

> "I think coding in primary schools is a good idea because my child would have better abilities of using computers/laptops/other electronic devices." (Parent 2)
>
> "It [the programming initiative] increased their confidence in the use of the laptop." (Parent 20)
>
> She learned how to use computers for when she is older" (Parent 21)

Both parents and teachers recognised the need to develop these understandings and skills to keep pace with the growing prominence of technology in our daily lives.

> "Children need to be used to doing it as it's going to be huge in the future." (Ms Rainer)
>
> "It gives them insights into how the technology and apps they use works, allowing them to become more critical consumers." (Ms Norbury)
>
> "Technology is our future therefore we need kids as young as primary to get used to this." (Parent 9)
>
> "I think it's a good idea because many things become digital so it's good to keep up and upgrade." (Parent 10)

Three parents, who acknowledged the importance of technological skills for living in an increasingly digital world, felt that it was necessary to provide these experiences in primary schools to ensure "*consistency*", "*accessibility*" and that no child was "*left behind*". They appreciated the important role that schools play in addressing the digital divide.

> "I think introducing coding in primary schools will strongly benefit children, keep them up to speed, in particular children who don't have access at home" (Parent 25)

This emphasis on the need to develop technical skills is interesting as it highlights that, although there have been numerous calls to consider digital literacy as a basic skill for young people to acquire along with reading and writing, children or their parents do not consider primary school children to be digitally literate. Early entries in the researcher's observation diary also indicated that the children lacked basic technical skills.

> Despite being considered the 'digital generation' or 'digital natives' many of the children lack basic computing skills. In the session today I spent a lot of time familiarising the children with how to navigate a computer and the basics of file management. I was really surprised that working on a computer seemed completely alien to most of them.
>
> (Observation diary, 31st Jan 2017)

The lack of technical skills at this level of primary school is worrying, as it would place them well behind their English and Australian counterparts. Children in key stage 1 (ages 5-7) in English primary schools are taught to "use technology purposefully to create, organise, store, manipulate and retrieve digital content" (Department for Education 2013). While at the same age Australian children have opportunities to "recognise and explore digital systems (hardware and software components) for a purpose" (ACARA 2023). This level of 'digital illiteracy' is perhaps even more surprising given that 65% of the parents in this study indicated that their children spent more than an hour (33% more than 2 hours) interacting with technology every day. However, several researchers have highlighted that children's usage of technology, rather than their access to technology, is a critical element of their computational participation (NCCA 2007a; McCormack 2014; Kafai and Burke 2016). Indeed, further examination of these children's technology behaviours provides insights could explain the gaps in their digital capabilities. While 65% of these children use laptops or desktop computers at home, the other 35% use other types of technology, predominantly tablets or smart phones. The majority of the

children's usage of technology in the home related to passive consumption rather than creation. The three most popular digital activities that these children engaged in were watching You Tubers, watching movies or downloading music. Only 20% of the children reported engaging in activities which related to the creation of digital content.

Development of Creative Skills

During the course of the programming initiative, there was evidence to suggest that the children experienced a shift in how they valued the opportunity to programme in school. While initially they felt their participation in the initiative would prepare them for future computer experiences, data from the post-initiative questionnaire illustrated that the children now valued the opportunity to develop their creative skills, using digital tools to develop and create new ideas (Weber and Greiff 2023).

> "I learnt that a lot of games are [made] by coding [and] that I could make my own game. What I liked the most about Scratch is learning to make games." (Eleanor, 5th class)
>
> "I learned how to code and how to make games on Scratch. I liked making games/stories and learning how to make them." (Rachel, 6th class)
>
> "I learned how to get backdrops people [and] to make games. I liked that you can make up a game" (Chloe, 3rd class)

These comments illustrate a shift in their technology identity, from consumer to creator, and these children valued the opportunity to engage in these activities which allowed them to create digital content.

Economic Value of Programming

Children's technology career aspirations have become particularly important in the last few decades. Indeed, the economic rationale was identified as one of the key drivers behind the inclusion of ICT in education (OECD 2001). The economic

rationale emphasises the role schools play in preparing children to meet the needs of the economy (OECD 2001). More recently, the Gender Balance in STEM Education Advisory Group (DES 2022c) also highlighted the importance of developing children's interest in technology careers. This report identified raising awareness of the variety of STEM pathways and careers and providing access to activities that challenge stereotypes as important actions in achieving this goal (DES 2022c). During the initiative, both career awareness and an acknowledgement of stereotypes were recurrent themes among children, parents and teachers.

The economic rationale is the most frequently cited rationale for the inclusion of ICT in education in policy documents. Both parents and teachers were acutely aware of the value of providing the future workforce with early technological experiences. The children began to recognise how early programming experiences could influence career aspirations during the initiative. In the pre-initiative questionnaire only one child made specific reference to the career aspirations. Many responses in the post-initiative questionnaire were career focussed: "*this help developing on computers and if you wanted a job to do with them*" (Alisa, 5th class), "*we should have coding in schools because you could be very good at it, and it could be your career*" (Rhianna, 6th class) and "*it will show a child different opportunities for jobs in the future*" (Katie, 5th class). These comments show children are starting to acknowledge the usefulness of these early programming experiences for a future career in technology. While, these changes in children's perspectives are promising but it is important to acknowledge the role of others in affirming and reinforcing these perspectives. According to Markus and Nurius (1986)

> the pool of possible selves derives from the categories made salient by the individual's particular sociocultural and historical context, and from the models, images, and symbols provided by the media and by the

> individual's immediate social experience.
>
> (p.954)

Therefore, any initiative which seeks to improve children's perspectives on technology pathways and careers must also push for a wider societal and cultural shift (DES 2022c). The DES (2022c) highlights the influential role both parents and teachers have in supporting this cultural shift.

Data from the parent questionnaires show that the development in the children's perspectives on technology careers was also reflected in the perspectives of their parents. The parents recognised the importance of technology in the future careers and workplaces of their children, and saw the programming initiative as an opportunity for children to build their skills in this regard:

> "It is a valuable skill going forward in the workplace." (Parent 20)
>
> "It is more prominent in the workplace nowadays." (Parent 24)
>
> "I think it is a good idea to introduce coding because it will help the children with their career." (Parent 18)
>
> "When the children grow up, they will need to use computers in their jobs." (Parent 35)

As parents/guardians play a key role in advising their daughters on both educational and career paths (DES 2022c), it is encouraging to see that these parents are considering the possibility of a career involving technology for their daughters. The teachers view primary school programming initiatives as an opportunity to remove some of the barriers to children's later participation in programming. They acknowledged how negative stereotypes and beliefs need to be challenged at this early stage.

> "Early experiences in coding might prevent some of the 'hoodie-wearing techies' stereotypes from developing." (Ms Walker)
>
> "All children should be exposed to coding at primary level so they know what it is and take [the] mystery out of it. By secondary unless it is compulsory some children will be afraid to try it." (Ms Honey)

Several researchers have reported that STEM careers and academic situations are particularly susceptible to negative gender stereotypes (Margolis and Fisher 2002; Capobianco *et al*. 2017). Therefore, the view that early programming initiatives could help challenge negative stereotypes and portray programming as accessible to girls would be welcome. The teachers also felt that by engaging in these programming experiences the children became more aware of what a career in technology might entail.

> "They now realise what is involved in computer programming and this experience may influence their career choices in the future." (Ms Walker)

> "They now realise what is involved in computer programming and this experience may influence their career choices in the future. It might inspire the next Katherine Johnson." (Ms McGonagall)

The Gender Balance in STEM Advisory Group set out four actions to address equity in STEM access and inclusion (DES 2022c). One of these actions was to support "learner access to, and experiences of, STEM to inspire learning, foster creativity and prepare for later engagement and success" (p.6). The commentary from the teachers in this study suggest that programming initiatives in primary schools, such as this one, can contribute to delivering on this action.

Factors influencing Engagement

These findings illustrate that programming initiatives have the potential to foster technology-related engagement of primary school children. According to Fredricks *et al*. (2004) engagement is assumed to be malleable and responsive to contextual factors. Therefore, the data was analysed to identify factors influencing student engagement.

The three most cited reasons for their enjoyment of the classes were, the design aspects of the tasks, the freedom to choose their project topic and the opportunity to work with their friends. Many students attributed their enjoyment of coding to the design aspects of the Scratch tasks. They enjoyed choosing and/or designing the characters and backdrops for their games, animations and stories. Sophie in 3rd class liked when she "*got to pick the backdrop and the people*" and Natalia in 3rd class said she "*mostly liked animating the characters and making them*". Similarly, Laura in 6th class liked "*being able to design the characters*". Previous research on gender-based engagement in digital gaming have found that girls enjoy games that support character customisation and that they utilise this feature as a means of self-expression (de Carvalho *et al*. 2020; Hjorth 2017). Alongside, designing their own characters and backdrops, the students also had opportunity to choose the subject and content of their projects. Again, the students identified this opportunity for creative freedom as something they enjoyed. Millie in 3rd class remarked how she liked "*all the different projects*". Jane in 6th class enjoyed that there were "*endless choices of things to do*". These two positive aspects of programming are afforded by the 'wide walls' of the Scratch programming language. The creators of Scratch wanted users to be able to create more meaningful projects. They specifically created a programming language that could support diverse types of projects, ensuring that their programming language appealed to a wide range of users (Resnick *et al*. 2009). The findings of this study suggest that the 'wide walls' of Scratch are indeed appealing to these new Scratch users. Students also appreciated the opportunity to work with others. While this has already been discussed, it is important to note that this was an element of Scratch which added to students' enjoyment of the coding classes. Paula

in 3rd class liked the Scratch classes "*because it's fun doing it with your friends*". Miriam in 5th class said, "*I enjoyed doing it with my besties*". Kelsey in 6th class enjoyed doing Scratch with her friends as they "*got to work together and have fun*". This finding corroborates the recommendations of previous researchers who have promoted the use of pair programming in educational settings (Denner *et al*. 2014; Gallardo-Virgen and DeVillar 2011). While many students made positive comments about their experience of collaborating on projects, there were also a few negative comments on these experiences. In general, issues that arose were due to pair incompatibility. One of these was Jean, a 3rd class student. She felt that the coding classes were "*awful*", citing partner incompatibility as the reason for the negative response. She said, "*I was really unhappy with who I was matched up with*". Issues with peer relationships in this age group are not uncommon as young adolescents grapple with different aspects of identity (Denner *et al*. 2014). Hence, familiarity and comfort with partners is recommended by Denner *et al*. 2014. However, this evidently wasn't achieved in all pairings during this programming initiative. Aside from incompatible pairs, other factors which were found to negatively impact student engagement were poor broadband connectivity, lack of appropriate computing spaces and malfunctioning technology.

> "I don't like when you haft [have] to swap placis [places] if some wone has to charg[e] there [their] laptop" (Dana, 3rd class)
>
> "I didn't like that sometimes the computer would not work" (Natalia, 3rd class)
>
> "I didn't like that sometimes it started glitching" (Anna, 3rd class)

These comments highlight that the ICT infrastructure identified in numerous reports on ICT in education (Cosgrove *et al*. 2014; DES 2008b; DES 2009) remain a perennial challenge for education providers.

Despite the students' willingness to persevere there were times during the initiative when they required assistance. Vygotsky's (1978) theory of social learning emphasises the role of the knowledgeable other in providing the appropriate assistance or 'scaffolding' to guide children to a more sophisticated level of thinking. The importance of a knowledgeable other to provide that scaffolding came across strongly in the data. Students' responses to the question 'What did you like least about Scratch?' provided evidence of their reliance on the teacher for support.

> "Sometimes when I needed help the teacher didn't come over to me and I got really confused" (Eimear, 5th class)
>
> "When we were all stuck and had to wait for help" (Zala, 3rd class)
>
> "It was kinda hard sometimes and I needed help and had to wait for [the researcher] to come over to me" (Kelsey, 6th class)

The 'ask three before you ask me' strategy (Papert 1984) was implemented in the programming classes to encourage the students to solve problems independently or collaboratively. Papert (1984, p.427) highlighted the importance of social learning in computing:

> the child learns computing by being immersed in a computer culture…a computer culture develops where many children in the school know about computing and a lot of knowledge is resident in the community of children… in the ideal computer culture, maximum use is made of the children's abilities to teach their peers.

However, the majority of the students were novice programmers, and these skills take time to nurture, so the students relied heavily on the teacher's knowledge. While the knowledgeable other can be a teacher or classroom peer, support can also be sought online. Cicconi (2013, p 58) describes the internet as a "medium for social learning", recognising the increased opportunities it affords users to learn from a more knowledgeable other. Brennan *et al*. (2010, pps.77-78) develop this idea

further, describing how the knowledgeable others can support learners in an online community:

> In addition to viewing the collection of projects uploaded to the site, as a large library of creative possibility and inspiration, members can download projects to study their construction, remix downloaded projects to be reshared online, create galleries of related projects, comment on projects to ask questions and provide feedback, and discuss issues in the forums.

However, this online community was not accessible to the students in this study as poor internet connectivity necessitated that students use the offline version of Scratch. Hence, the students tended to rely heavily on assistance from the teacher. Indeed, this reliance on teachers as a knowledgeable other is one of the factors that could impede the introduction of programming to primary schools, as many teachers feel apprehensive about their ability to provide that level of support (NCCA 2019a). This point was illustrated by Kirsten in 6th class, who didn't think there should be programming classes in school because "*it's hard and you need experienced computer people or else it would be even harder*". It is possible that this comment was prompted by the particulars of the situation, the researcher was viewed as an expert who came in specifically to teach this class. Perhaps this wouldn't have been the case if the classes had been facilitated by the class teacher. However, the teachers in this study expressed similar concerns regarding their ability to provide adequate support to their students if programming was introduced in schools.

Teacher Confidence and Training

Without the online community, the level of support required by the students in the initial stages of learning to code necessitated considerable teacher content knowledge. Most of the teachers in the case study school did not feel confident in their ability to provide this level of support to their students. Two of the teachers

reported being ‘not at all confident’; four of them reported being ‘not very confident’, while the remaining two teachers reported being ‘fairly confident’. Teacher confidence was identified as one of the major barriers to the introduction of coding to primary school classrooms in the NCCA’s (2019a) study on coding in primary schools. However, despite the low levels of confidence, all of the teachers expressed their interest in teaching coding in the future. Again, this was similar to the findings of the NCCA (2019a) study, which reported that the majority of teachers felt that class teachers should have the responsibility for teaching coding if it was introduced to the primary curriculum. Interestingly, the principals in that study were less confident that it should be the responsibility of the class teacher, 52% of them felt it should be the responsibility of the class teacher, while the remaining were unsure or felt it should be the responsibility of a specialist teacher (NCCA 2019a).

These findings suggest that while teachers are open to the possible introduction of coding to the primary school curriculum, they do not feel they have the necessary skills. Indeed, all but one of the teachers agreed that they would be comfortable teaching coding if they received some training or professional development. The remaining teacher expressed concern about the level of training that would be provided but stated that if sufficient training was provided, they too would be confident they could teach coding to their students. Therefore, professional development will be key in developing both teachers’ content knowledge and confidence.

5.4.6 Computational Perspectives – Conclusion

The insights that these findings can give on the development of computational perspectives is heightened, given the lack of research in this area (Zhang and Nouri

2019). Findings from the questionnaire established that, prior to this initiative, much of students' technology use was as 'passive consumers' rather than 'active producers'. Watching You Tubers, watching movies and downloading music were identified as their three most common digital activities. Therefore, insights into their perspectives on their experience creating with technology were of particular interest. The computational perspective of expressing, seeing technology as a medium for creation, came across strongly in the data. The students relished the opportunity to be creative, and similar to the findings of NCCA (2019a) they believed that the introduction of coding to schools would provide an opportunity for children to develop this twenty-first century skill. The students were also positive about the "power of creating with and for others" (Brennan *et al*. 2014, para.3). They recognised the benefits from working as part of a group, while also acknowledging that their intended audience, their peers, encouraged them to create better finished products. Upon completion of the programming initiative, students held varying perspectives on their relationship with technology from a functional perspective to an exploratory perspective all the way through to an empowered perspective. This illustrates that while programming in schools doesn't necessarily make all students want to be programmers, it does encourage them to consider and develop their 'relationship' with technology. None of the aforementioned perspectives captured students' motivational beliefs with respect to learning programming, something which came across strongly in the data. Therefore, it is proposed that an additional dimension, motivation to engage, be added to the dimensions already included in computational perspectives of Brennan and Resnick's (2012) framework. Overall, these findings illustrate that the girls in this study held predominantly positive

perspectives about technology, a welcome finding given the frequent reference to negative gender stereotypes across the literature (Goo *et al*. 2020; STEMerg 2016).

5.5 Conclusion

This chapter outlined the findings in relation to the development of the computational thinking concepts, practices and perspectives. Data was discussed that explored the development of each of the concepts, practices and perspectives conceptualised in the Brennan and Resnick (2012) framework. The data provided evidence of additional computational concepts and perspectives that were not included in this conceptualisation, namely the computational concept abstraction and the motivation to engage perspective. These additional dimensions are explored and described. These findings are situated within the context of current computational thinking literature and technology education policy. Pedagogical considerations such as the suitability of the programming language Scratch, and contextual factors, such as student age and technology infrastructure, that impacted the effectiveness of the programming initiative are presented. These pedagogical and contextual factors will be further explored in the coming chapter which explores the implications of this research in relation to research, pedagogy and policy.

Chapter 6: Outcomes, Implications and Conclusions

6.1 Introduction

The findings of this research study were outlined in the previous chapter. In this chapter, these findings will be contextualised, and the implications of the research will be discussed against this contextual backdrop. This research is timely in that it coincides with the introduction of the new primary curriculum framework and the development of the new STEM curriculum. The primary curriculum framework emphasises the importance of being a digital learner, while computational thinking is included as a key pedagogical practice in the draft STEM curriculum (NCCA 2024). These curriculum developments mean insights into the learning of programming and associated computational thinking development within a primary school context can be leveraged to inform future policy and practice. In particular the contextual and pedagogical factors that impacted computational thinking development have implications for theory, practice and policy in the STEM education field.

6.2 Context of the Research

This study was initially prompted by a number of significant developments in the field of STEM education. In 2016, the Irish government announced their desire to make the Irish education and training system the best in Europe by 2026 (DES 2016b; STEMerg 2016). STEM education was believed to be central to achieving this lofty ambition (DES 2017e). Driven by these considerations a series of requests were made by the Irish government to review current STEM education practice in schools (STEMerg 2016) and consider ways of increasing the uptake of 'gateway'-STEM subjects including coding and computer science (DES 2016b). In response to the first of these requests, the 'STEM Education in the Irish School System' report

was published. The report contained three observations which were critical to this research study:

- *"There is a strong gender imbalance…in the number of candidates taking Leaving Certificate Technology subjects" in favour of boys,*
- *"Women are greatly under-represented in the STEM workforce in Ireland",*
- *Ireland has a "highly active informal STEM sector…which includes initiatives such as CoderDojo". However, "the benefits of these initiatives are not fully realised under present conditions…and the potential for much greater engagement with STEM activities, may be underleveraged because it is not integrated into the curriculum."*

(STEMerg 2016, pp.8,18)

These observations prompted a number of reviews and reports which focussed on addressing gender imbalance in STEM disciplines, and on increasing the teaching of programming in primary schools.

Several important findings have emerged that are relevant to this study. Goos *et al*. (2020) investigated the factors that negatively impact girls' participation in STEM disciplines. Their findings revealed a need for interventions that focused on improving interest, enjoyment, motivation, engagement and self-perceptions of learners, while building productive STEM identities and addressing negative stereotypes (Goos *et al*. 2020). The Department of Education recommended four key areas of action based on the findings of this report (DES 2022c). These proposed actions were to: improve equity of access and inclusion across all STEM disciplines by effecting whole school cultural change, providing effective support for STEM educators, ensure equitable access to STEM experiences that inspire learners and promote future engagement and success, and support a societal shift to address current barriers to gender equity in STEM (DES 2022c). As this research on gender imbalance in STEM was being undertaken, research was also being carried out to determine how programming could be incorporated into the primary school

curriculum. Initial investigations focussed on incorporating programming into the primary mathematics curriculum, similar to approaches adopted in Finland and Northern Ireland (NCCA 2018). However, while various reports acknowledged the close links between mathematics and programming, they also questioned its positioning as solely a mathematical topic (NCCA 2019a; Millwood *et al*. 2018; NCCA 2018). Hence, although the 'Primary Mathematics Curriculum: Draft Specification Junior Infants to Second Class for Consultation' (NCCA 2017a) explicated how the new mathematics curriculum could contribute to the foundations of programming, there was no specific reference to programming in the redeveloped primary mathematics curriculum. Instead, the NCCA considered creating an explicit subject space in the curriculum for digital learning (NCCA 2019a).

In the primary curriculum framework, launched in 2023, technology became a named space within the primary curriculum within the STEM curricular area. Following this, the draft specifications for the science, technology and engineering strands were published in March 2024, explicating how programming skills would be developed across the four stages of primary school (NCCA 2024a). The relationship between programming and computational thinking was identified early in the curriculum development process. The draft mathematics curriculum described computational thinking as the basis for programming (NCCA 2017a). Similarly, the 'Primary Developments Final report on the Coding in Primary Schools Initiative' (NCCA 2019a, p.8) stated that "recent research has shown that computational thinking lays some of the foundations for coding". Therefore, it is unsurprising that computational thinking is identified as a key pedagogical practice in the draft specifications for science, technology and engineering (NCCA 2024a). It is against

this backdrop that the findings of this investigation will be interpreted and recommendations for future action will be implemented.

6.3 Summary of Key Findings

The findings of this study shed light on how computational thinking can be fostered through the use of Scratch programming. Dr. Scratch, the automated project analysis tool, was used to evaluate the students' final projects and assign a programming score and corresponding programming profile (basic, developing or master) (Lawanto 2016). The results showed that all of the students in the study achieved a programming score between 8 and 10, which placed them all in the developing category. The Dr. Scratch analysis also revealed which computational concepts the students had grasped during the ten weeks of programming, and which concepts they were still grappling with. The findings indicated that, on average, the students best understood synchronization and parallelism, demonstrating proficient use of both concepts. Flow control, user interactivity and data representation scored slightly lower and were evaluated as developing. The lowest scores were achieved in abstraction and logical thinking, with students only demonstrating a basic proficiency with respect to these concepts. The qualitative data supported these findings and provided greater insights into the specific challenges encountered. It emerged that, in particular, conditional loops and variable initialisation caused significant difficulty for the students. Abstraction was not included as a concept in Brennan and Resnick's (2012) conceptualisation of computational thinking, but Kong (2019) advocated for its inclusion. During this research, the potential for abstraction concept development and application afforded by Scratch became evident and therefore this study supports the view of Kong (2019). In addition, this research advocates for a separation of data into variables declared by the

programmer and attribute variables. This recommendation is derived from the particular difficulty that students displayed in comprehending attribute variable initialisation, in particular explicit initialisation. However, given the potential utility within the Scratch programming environment and its importance to programme development, this research supports its future inclusion. Further analysis of the data showed that both project type and direct instruction were important in computational concept acquisition. This illustrated the importance of designing developmentally appropriate learning experiences.

The second dimension of the Brennan and Resnick (2012) framework was computational practices. The data also provided insights into how empowering these students to move from consumers of technology to creators of technology, facilitated their engagement in computational practices. When translating their code from storyboard to Scratch students often employed abstracting and modularising to improve functionality or ensure ease of navigation for subsequent testing and debugging. Separating scripts by actions or interactions provided a natural approach to abstracting and modularising for these students. The students adopted an incremental and iterative approach to their project design. The storyboard scenes provided natural breaks in the design process, encouraging students to pause coding and try it out before moving on. These pauses often resulted in modifications to the projects, as the on-screen action triggered new ideas or alerted students to coding problems. This finding illustrates the close association between the practices of experimenting and iterating and testing and debugging. It also highlights the affordances of the Scratch programming environment in supporting the practice of testing and debugging. Students adopted three main strategies in isolating the problem: reading code, segmenting code and making experience-based hypotheses.

The accessibility of the programming language and the immediate feedback provided by Scratch assisted students in implementing these strategies. These features also supported students who adopted a trial and error approach to debugging. Other debugging strategies employed were transferring previous experiences to new contexts and seeking help from a knowledgeable other. The final dimension of computational practices was reusing and remixing, and students engaged in this by building on existing projects and sharing ideas. Due to the connectivity issues in the school, students were mostly limited to the local community to source inspiration or ideas. However, some students accessed the global community outside the school context, bringing back fresh ideas but raising issues relating to digital inequality. As creators of technology, these students experienced many facets of the design process, including ideation, planning, testing, redesign and showcasing. These different stages of creating encouraged students to engage in different computational practices, with students often engaging in two or more practices simultaneously. During the initiative, it became evident that certain pedagogic strategies and scaffolds supported this engagement. These included the choice of programming language, the use of storyboarding, promoting a community of practice through the implementation of gallery and learning walks, and finding a balance between highly structured activities and unstructured student-led exploration.

The final dimension of computational thinking explored in this research was computational perspectives. The data revealed that for many of the students, this creative use of technology differed greatly from their previous passive consumption of technology. Students welcomed the opportunity to explore the creative affordances of the Scratch programming environment. The wide-walls of Scratch ensured that the students felt they could express themselves in meaningful ways,

with abundant choice (project types, backdrops, sprites), ability to incorporate their personal interests (importing favourite songs and images) and explore their creative talents (composing images and sounds). While the context of this study meant students didn't have access to the global Scratch community, the students valued the opportunity to collaborate and showcase their projects within their local Scratch community. The acknowledged the many benefits that they obtained from creating with their community including increased capacity for idea generation and execution, problem solving and feedback. The programming initiative provide students with the opportunity to experience an authentic audience, central to the process of content creation. The audience became a central consideration for them during the design process, and they increasingly tailored their creations to conform to audience expectations and preferences. For some students, this created tension between the importance of self-expression and the need for approval. At the end of the initiative, the students expressed differing views on how the programming initiative had impacted their ability to wield technology. For some, the gains they made were merely functional, improving their ability to use computers. For others, they gained new perspectives on what they could potentially do with computers, making a distinction between computing and ICT. Others again, felt they were now equipped to positively contribute to a digital society. In addition to their perspectives on their ability to engage with technology, perspectives on their motivation to engage with technology also emerged. The findings indicated that this programming experience positively impacted their behavioural, cognitive and emotional engagement. This was a perspective which wasn't captured in Brennan and Resnick's conceptualisation of computational perspectives, and so it is recommended that it should be included in future conceptualisations of computational perspectives.

Once again, several pedagogical factors were found to contribute to the development of positive computational perspectives. These included the wide-walls of the chosen programming language, Scratch, the 'gallery' and 'learning' walks, and the opportunity to work on personally meaningful projects.

6.4 Contribution to the Literature

The contribution of this research study to the extant literature is three-fold. First, this research conceptualises computational thinking, providing thick descriptions of how computational thinking manifested within this research context. Secondly, the research study explicates the contextual dimension of the study providing insights into how computational thinking can be developed and indeed fostered within formal primary school settings. Finally, this research makes suggestions on how the computational thinking framework developed by Brennan and Resnick (2012) can be built on to incorporate aspects of computational thinking not currently captured by the framework. Computational concepts have been well researched in other settings (informal, secondary and third level), and are therefore quite well defined in the literature. However, computational practices and perspectives have not been researched to the same extent and are therefore less well defined (Zhang and Nouri 2019). Hence, this research contributes to the field by providing clear operational definition for both computational practices and perspectives.

Furthermore, previous research efforts which have explored programming and computational thinking have often failed to report important aspects of their studies, such as basic demographic data or details regarding the implementation of activities (Kafai and Burke 2015), hence limiting their impact on future practice. This research study provides in-depth descriptions of the demographic, the implementation of the

programming initiative and the nature of data analysis, revealing important contextual and pedagogic dimensions which can inform future policy and practice. The contextual dimension of this study is important as few research studies of this type have been carried out in Irish primary schools. Therefore, this research further contributes to the field by contextualising the development of computational thinking in Irish primary schools. It explored what aspects of computational thinking can be developed and how teachers can support this development. Much of the research on computational thinking development has been conducted with secondary or third level students (Millwood *et al*. 2018) or in informal learning settings (Kafai and Burke 2015). While findings from these settings can inform practice in primary education settings, they differ from the primary school setting, in that the participants might have a stronger interest and self-efficacy from the outset, as they have often self-selected to participate (Newton *et al*. 2020). With the impending launch of the finalised Science, Technology, Engineering specifications, and the subsequent teaching of programming and computational thinking in schools, mean that findings from studies such as this one are both necessary and timely. In addition by focussing on an all-girls primary school setting, the study provides unique insights into how, and in what ways girls can develop computational thinking through programming. This is an important perspective as previous research has shown that boys and girls respond differently in programming settings (Funke and Geldreich 2017; Hjorth 2017), and the abundance of research that shows that girls experience significant barriers that prevent them from participating in STEM disciplines (Goos *et al*. 2020). The implications of the contextual and pedagogic dimensions will be further outlined in a subsequent section (6.6).

This research study adopted the framework developed by Brennan and Resnick (2012) as a means of conceptualising computational thinking. This framework was chosen as it encompassed the conceptual, the practical and the affective dimensions of computational thinking and is widely applied in studies involving computational thinking and visual programming languages. However, throughout the research process the researcher was cognisant of additional computational thinking components, present in the literature (Kong 2019; Zhang and Nouri 2019), but not included in Brennan and Resnick's (2012) framework. Therefore, in pursuit of a more comprehensive picture of the nature of computational thinking and its development, a pragmatic epistemological approach was adopted in the use of this framework. While the deductive coding process focussed on providing rich descriptions of the computational concepts, practices and perspectives encompassed in the Brennan and Resnick (2012) framework, the inductive coding process was focussed on identifying additional aspects of computational thinking that weren't captured in the framework. The data analysis provided evidence to support the inclusion of one additional computational concept (abstraction), the reconceptualization of data into two distinct concepts, programmer declared variables and attribute variables, and the addition of one another computational perspective (motivation to engage) to the framework. These additional components are described and defined in more detail in the findings of this study, and a re-envisioned computational thinking framework is presented in the following section.

6.5 Implications for Theory

Set against the backdrop of the newly established STEM curricular area, this research provides researchers, policy makers and educators with a comprehensive picture of what computational thinking can look like in a primary school classroom.

While definitions of all three dimensions of computational thinking were readily available in the literature (Martin *et al*. 2014; Brennan and Resnick 2012), conceptions of how computational practice and perspectives manifested in a classroom setting were noticeably absent from the literature. In the review of the literature, only one study that shed light on computational practices was identified. Litts *et al*. (2020) described the computational practices that their participants engaged in as they designed place-based, mobile games. However, this study was conducted in an informal setting with older children (9-16) and as such is not directly comparable. There was no study which provided a meaningful portrayal of computational perspectives identified in the review of the literature. Therefore, this research significantly contributes to this field by providing rich descriptions of both computational practices and computational perspectives. The findings of this study present clear illustrations and exemplars of how computational thinking can be developed, alongside the conditions which enabled or inhibited its development.

The findings of this study highlighted aspects of computational thinking that were not sufficiently captured by the Brennan and Resnick (2012) framework and so a reconfigured computational thinking framework is proposed (see Table 6.1). Firstly, the computational concept categorisations proposed by Moreno-León et al. (2015) are preferred to the categorisations of Brennan and Resnick (2012), as adopting them supports the combining of qualitative data with the quantitative results derived from the artefact analysis. Although the categorisations proposed by both groups of researchers do not correlate directly, they reflect the same aspects of computational thinking with the exception of abstraction (see Table 4.11). Abstraction has been included in several other conceptions of computational thinking (Kong 2019). Kong (2019) labelled abstraction, procedures, and defined the concept as code that "helps

to avoid the repetition of codes and duplication of commands" (p.124). While students in this study only demonstrated a basic proficiency level of this concept, three distinct levels of abstraction are achievable in Scratch, as outlined in the findings section (5.2.8).

Table 6.1: *Reconfigured Computational Thinking Framework (adapted from Brennan and Resnick 2012)*

Computational Concepts	Synchronisation Parallelism Thinking Logically User Interactivity Flow Control Abstraction Data	• Attribute Variables • Programmer Declared Variables
Computational Practices	Abstracting and Modularising Experimenting and Iterating Testing and Debugging Reusing and Remixing	
Computational Perspectives	Expressing Connecting Questioning Motivation to Engage	

The second suggested alteration to the Brennan and Resnick (2012) framework is the separation of data into attribute variables and programmer declared variables. This suggestion is put forward as students in this study encountered attribute variables from a very early stage, much earlier than programmer declared variables, and these early encounters resulted in considerable difficulties relating to variable initialisation. Therefore, it is suggested that attribute variables should be viewed as a distinct data concept, and that students should be introduced to this concept and performing attribute initialisation from early in their programming experience.

The final recommended modification to the Brennan and Resnick (2012) framework is the addition of a fourth computational perspective, motivation to engage in programming. During the deductive coding process, a theme relating to students'

motivation to engage in programming consistently emerged from the data. Unlike computational concepts and practices, perspectives can be informed from literature outside the domain of computing. Therefore, widely recognised conceptions of motivation, engagement and value (Fredricks *et al*. 2004; Wigfield and Eccles 2000; Wenger 1998) were employed as a lens to interpret these emerging findings and describe and define this new perspective. Table 6.1 illustrates the proposed reconfiguration of the Brennan and Resnick (2012) framework.

6.6 Implications for Practice

The findings of this study offer insights which can inform both the development of computational thinking through programming experiences in primary schools and the design of the associated professional development opportunities. Therefore, this research is of relevance to both primary school educators and professional development providers. The review of the literature revealed several pedagogical strategies that support the teaching of programming. However, this programming initiative revealed how these, and further strategies, supported the development of computational thinking during these experiences.

6.6.1 Balancing Direct Instruction and Self-Discovery

6.6.1.1 The Need for Direct Instruction

Maloney *et al*. (2008) in their investigation of the learning concepts discovered by youths, in an informal setting of a computer clubhouse, found that there were some programming concepts that were difficult to discover without guidance. Similar findings were obtained in this study, with the students relying heavily on the small number of blocks that were familiar to them. Even when children demonstrated high

levels of understanding related to a concept, the blocks they used to illustrate this proficiency were all blocks they had been introduced to in the instructional sessions. These findings suggest that, in this study, development in students' understanding of computational concepts was impacted more by their exposure to, and familiarity with Scratch blocks than their aptitude for understanding the concepts. As there were only five instructional sessions dedicated to the learning of basic Scratch functions, there was a limit to what could be explored and explicated in that time. This suggests that despite being more accessible than other programming languages, students require more exposure to these commands to understand their function.

6.6.1.2 The Need for Self-discovery

While this research identified the need for explicit teaching to provide exposure to specific computational concepts, the importance of having opportunities to work on personally meaningful tasks came across strongly in the data. These student-led opportunities were particularly influential in the development of the computational perspectives of expressing and motivation to engage. This coupled with the emphasis in the new STEM curriculum encouraging students to use "creativity to make choices, choose pathways and lead their own learning" (NCCA 2024) promote the inclusion of student-led exploration. However, given the importance of direct instruction, a balance needs to be found between both approaches.

6.6.1.3 Adopting a Variety of Pedagogical Approaches is Key

This has implications for teachers when they are selecting suitable pedagogies to use in their classes. The continuum of programming scaffolding developed by Waite (2018) identifies several pedagogies that would be suitable for teaching

programming in the classroom. These pedagogies range from more structured pedagogies such as copying code and targeted tasks to less structured pedagogies such as projects and tinkering. Based on the findings of this study, the researcher proposes that the more structured pedagogical approaches would be more suitable in the introductory phases as they would help the students develop familiarity with the Scratch commands. Students can also be encouraged to discover Scratch commands through more student-led exploration. During this initiative, five minutes of every session was assigned to free exploration of the Scratch programming commands, after which students shared their discovery with their peers. This deliberate allocation of time for exploration was effective in encouraging these students to discover new concepts. Strategies such as these, which encourage self-discovery are especially important for girls, as Funke and Geldreich (2017) reported that girls are less likely to self-discover blocks of their own volition.

6.6.1.4 Project Types – Animations, Stories or Games?

Teachers also have an important role in ensuring that students have opportunities to create a variety of program types. The results of this study, and previous research (Funke and Geldreich 2017), have indicated that girls are more drawn to the development of stories or animations rather than games. This is a significant finding because this study suggested that project type impacted the computational concepts incorporated in the final projects, although the sample size was too small to conclude this definitively. Games yielded higher scores than other project types for user interactivity, parallelism and logical thinking. While, stories and animations provided greater opportunity for the use of synchronisation. This has implications for teachers, when determining the most appropriate way to introduce and assess computational

concepts. It is also an important consideration for curriculum developers and professional development providers when designing support material for teachers.

6.6.2 Ensuring Children are Appropriately Challenged

The draft STEM curriculum includes additional support pathways to encourage teachers to "consider the learning experience in greater detail and identify appropriate levels of challenge" for their students (NCCA 2024, p.20). Programming can support educators with an opportunity to challenge their students within a twenty-first century context. In an article for the Bangor Daily News, Papert discussed children's perceptions of learning to program. He recalled one child describing learning to program as both fun and hard. Papert (2002) insisted that he had "no doubt that this kid called the work fun because it was hard rather than in spite of being hard" (Papert 2002, para.2). The perspectives of the children in this study support Papert's view. They relished a challenge and found achieving success in the face of difficulty to be rewarding. Consequently, findings of this study suggest that teachers should use these pathways to guide their development of appropriate learning experiences but not to forget that students enjoy productive struggle.

6.6.3 Appropriate Embedding of Computational Thinking into the Initiative

In the development of the programming initiative, research was undertaken to determine the best pedagogical approach to developing computational thinking. The research explored recommended practices from a variety of programming contexts, including informal settings and formal settings at primary, secondary and third levels. These insights were combined to inform the instructional design of this initiative. The resulting pedagogical decisions meant that several aspects of computational thinking were intentionally embedded into the programming

initiative. The most influential of these pedagogical decisions were the selection of Scratch as the programming language of choice, the utilisation of pair programming, storyboarding and gallery walks, and the adoption of a constructionist programming approach.

6.6.4 Building the Requisite Programming Knowledge

The imminent introduction of mandatory programming to the primary school curriculum represents a great opportunity to expand students' technological abilities but with this opportunity comes challenges. Once such challenge evident in this study, was the need for creating a strong knowledge base to support these digital learners. During this programming initiative the students were often overly dependent on the teacher-researcher, preferring to look for their assistance rather than search for solutions elsewhere. Given the low levels of teacher expertise and confidence that emerged from the NCCA's study on programming in primary school (NCCA 2019a), excessive dependency will not be sustainable in this context. Hence, it is important to consider strategies that can lessen student dependence on teachers and nurture learner agency. Three possible strategies that emerged from the data were accessibility to the online Scratch community, fostering a class community of 'Scratchers' and initiating a 'bug wall'.

According to Resnick *et al*. (2009), the online Scratch community benefits Scratch learners by providing a platform where people can "support one another, collaborate with one another, and build on one another's work" (p.65). During this study, the students only had access to the offline editor and the Scratch community is not accessible when students are working offline. Therefore, use of the online Scratch editor, to take advantage of the additional access to 'knowledgeable others' it affords,

is recommended. Secondly, it is also important to take advantage of the expertise that is available locally within the school community. Within the classroom, the sharing of knowledge can be promoted through the use of gallery walks and learning walks. Within the greater school community, shared programming initiatives akin to the practice of shared reading could be implemented, promoting peer assisted learning. The final strategy, initiating a 'bug wall' focuses on the normalising and keeping track of programming errors. As students encounter errors they record their error and add it to the 'bug' wall. Students can then refer to this 'bug wall' during testing and debugging to see of other students have experienced these problems and developed suitable solutions. These strategies can help to alleviate some of the pressure for teachers as they navigate their own expertise development journey.

6.6.5 Recognising the Strengths and Weaknesses of the Programming Languages

Since its development in 2007, Scratch has been promoted as a suitable programming language for novice programmers. However, when teachers use pedagogical tools in the classroom, they must be aware of the instructional advantages and disadvantages associated with these tools (Ball *et al*. 2008). This research illustrated several affordances and limitations of Scratch as a computational thinking development tool. Firstly, the 'wide-walls' of the Scratch programming language afforded the students great freedom in the types of projects they could create. This facilitated self-expression and motivated students to engage, which positively impacted students' perspectives on their relationship with technology. The click-to-execute functionality of Scratch supported students' self-discovery of certain programming concepts. It also enabled students to adopt a trial and error approach to programme design and modification, hence encouraging the computational practices

of experimenting and iterating, and testing and debugging. However, the click-to-execute function also caused some issues for students. One such issue was that being manually able to change several variables of sprites (positioning, costume etc.) removed the necessity to initialise these variables at the beginning of program design. This resulted in initialisation problems when programs were run twice in a row. The availability of immediate feedback was helpful as it enabled a trial and error approach, but students also grew to rely on it as a troubleshooting strategy. This caused issues, particularly in situations where the problem related to time synchronisation, and computer execution was too fast to be perceived by the human eye. An awareness of these limitations of Scratch can support teachers in designing appropriate learning experiences that avoid these language based issues.

6.7 Implications for Policy

During this research study, a number of challenges to introducing programming in primary school contexts emerged. Based on the findings arising from this research, several recommendations are offered to help meet these challenges. These recommendations focus on how policy makers can support the successful implementation of the new STEM curriculum.

6.7.1 Professional Development Policy

While this programming initiative was implemented by the teacher-researcher, the data still revealed that both the students and teachers who participated in this study feel that primary school teachers are currently underqualified and underprepared to teach programming. In the 'Initial Teacher Education Policy Statement' (DES 2023e, p.18), it was recognised that "teachers must be prepared to support the introduction of new subject areas across both primary and post-primary schools" but no guiding

policy or strategy has been forthcoming. With the finalised specifications for the new STEM curriculum expected to be available for the 2025/2026 school year (DES 2023c), urgent action is required from policy makers to ensure schools and teachers are adequately prepared to implement and embed the new STEM curriculum.

6.7.2 Ensuring Digital Equity

In the OECD (2020) series publication 'Education Policy Outlook Ireland' the OECD warned that Ireland needed to act to address the risk of digital divide in educational settings. The NCCA define the digital divide as inequalities in "both physical access to the technology and skills associated with its use" (NCCA 2007a, p.23). While initially policy focussed on achieving equity of access, more recent policy has focussed on the second aspect, building digital literacy (Gui 2007; Jenkins 2006). However, the findings of this study provide evidence that the students in this school context had neither adequate digital access nor basic digital literacy.

Issues with ICT infrastructure in the case study school were flagged in preliminary meetings with the principal and teachers. Despite recent large-scale investment by the government in ICT infrastructure for schools (e.g. Schools Broadband Programme, €30 million; The ICT in Schools Programme, €92 million; Digital Strategy for Schools ICT Infrastructure Fund, €160 million+), this was an expected obstacle as this perennial problem has been highlighted in numerous reports since the publication of the first policy document on ICT in Irish education (NPADC 2001; NCTE 2005; DES 2008; ERC 2014; NCCA 2019). The report on the first government-supported initiative to improve ICT integration in schools (Report on the Implementation of Schools IT 2000) recognised the need for greater funding to purchase and maintain equipment (NPADC 2001). In the NCCA's 2019 report on

coding in primary schools, teachers identified ICT infrastructure and the need for further funding to purchase equipment and improve broadband accessibility in their classrooms, as barriers to the integration of coding to the primary school curriculum. Similar issues have been reported internationally (OECD 2001; 2005; 2015), highlighting the breadth and complexity of ICT provision and maintenance in primary schools. While the 2013 ICT Census indicated falling student-computer ratios, it also acknowledged that not all computers were suitable for student use during teaching and learning. Indeed this was the case in this school context where out of the forty one devices available in the school, only eight devices were available for use in the initiative. To combat the shortage of school devices the decision was taken to introduce a Bring Your Own Device (BYOD) strategy.

Inadequate broadband connectivity was the second issue that arose in relation to ICT infrastructure in the school. Lowering student-computer ratios inevitably places increased demand on the school's Wi-Fi network during online activities (European Commission, 2019; European Schoolnet, 2017). Despite government investment in developing the Schools Broadband Programme, reliable access to adequate internet access has continued to be an issue in Irish schools. In the NCCA's 2019 report on coding in primary schools, teachers again highlighted the need for improved Wi-Fi and broadband capabilities in their classroom, with 80% of the teachers describing their classroom internet connection as average or worse (average - 48%, below average - 20% and poor - 12%). Therefore, the decision was taken before the initiative started that it would be most suitable to conduct the programming sessions offline as the school did not have sufficient network capacity for all the students to be working online at the same time. Lack of access to the online repository of Scratch projects and the backpack sharing feature of Scratch meant that these novice

programmers only had access to the limited work of their fellow novice programmer classmates. This had repercussions for students' opportunity to engage in the computational practice of reusing and remixing but also limited their exposure to more challenging computational concepts. Furthermore, they are missing out on the opportunity for 'social learning' that the internet affords beyond the walls of the classroom. These findings highlight the need for immediate government action in addressing these perennial issues of digital equity.

From early in the programming initiative the low levels of digital literacy of students across all classes was evident, with seemingly simple tasks such as saving files causing significant issues. This is despite the students reporting high levels of access to technology in the home. It clarifies the findings of Kafai and Burke (2016) that access to technology does not equate to technological competence. These low levels of digital literacy, as defined by their ability to create rather than consume is worrying and necessitates action from policy makers.

6.7.3 Ensuring Cohesion between Curricular Areas

While there is ample research which supports the integration of programming and computational across the curriculum (NCCA 2019a; Millwood *et al*. (2018); NCCA 2018), the findings of this research serve as a reminder of the close associations between programming, computational thinking and mathematics. As work continues on the development of learning trajectories for technology, it is important to consider mathematical concepts that students will encounter when developing their understanding of computational concepts. The findings of this study illustrate that programming can provide valuable opportunities to explore mathematics concepts in a meaningful way. For example, despite not having previously encountered

coordinate geometry or negative numbers, the third class students readily grasped these concepts within the context of Scratch. The third class children illustrated their developing understanding of coordinates, describing movement on the x-axis as "moving over and back or left and right" (Laila, 3rd class) and movement on the y-axis as "moving your sprite up and down" (Maisy, 3rd class). They were also able to describe the negative numbers as being "below 0" (Maisy, 3rd class) on the y-axis and "to the left of 0" (Laila, 3rd class) on the x-axis. However, there is also potential for an inverse relationship, where lack of mathematical understandings hindered computational thinking development. For example, initialising sprite positions requires a knowledge of coordinate geometry. However, an understanding of the coordinate plane is not a mathematical learning objective until Stage 4 (5th and 6th class) of the Irish mathematics primary curriculum. Indeed, studies that have investigated the potential of Scratch to develop coordinate understandings, have typically focussed on 5th or 6th class students (Anabousy *et al*. 2023; Germia and Panorkou 2020; Molina-Ayuso *et al*. 2023). Lack of understandings in these areas could have contributed to issues students had in comprehending initialisation relating to positioning variables. Therefore, careful consideration of the mathematics primary curriculum, particularly the Shape and Space and Algebra strand units would be important for those working on learning trajectories for technology in the new STEM curriculum. It is recommended that as curriculum designers consider the progression continua for the technology in the new STEM curriculum and develop support material for teachers, clear links should be made for opportunities for integration between the two subjects.

6.7.4 Ensuring Consistency in Defining Computational Thinking

Within the STE Education specifications, computational thinking is also recognised as a key concept underpinning the technology strand. Throughout the document there are several references to computational concepts, practices and perspectives. However, these are not consistently referred to within the different sections of the document. The recommended pedagogical approaches emphasise the development of computational practices (testing and debugging and abstracting and modularising) and computational perspectives (expressing and connecting). However, there is limited reference to providing opportunity to develop computational concepts, with only a brief mention of logical thinking and algorithms. Within the key concepts underpinning technology, there is more emphasis on the computational thinking concepts (sequences, loops, conditionals and variables) and practices (abstracting and modularising and testing and debugging). There is little emphasis in this section on computational perspectives, with opportunities for being creative only identified in stages 3 and 4, and no mention of providing opportunities for collaboration and communicating. Consistency across the curriculum document, in the defining of computational thinking will be critical in supporting teachers, particularly those teachers for whom computational thinking is a new construct. Embracing the three dimensions suggested by Brennan and Resnick (2012) might be a possible approach to achieving this consistency.

6.8 Recommendations for Further Research

1. This research adopted a case-study approach in an attempt to establish how, and in what ways computational thinking could be developed within this particular school context. Hence, “the subtlety and complexity of the case” (Cohen *et al*.

2007, p.256) were deemed important to the research. However, the introduction of both programming and computational thinking to the redeveloped primary school curriculum, calls for evidence that can inform practice in a multitude of contexts. As several researchers have questioned the suitability of experimental research methods to educational contexts (Behar-Horenstein and Niu 2011; Freebody 2003; Kember 2003), it is recommended that practitioner-research would be encouraged to generate further exemplars of how both programming and computational thinking can be developed in the primary classroom. Micro-ethnographic methods such as video, could be used to gain further insights into the social interactions and behaviours children engage in while programming.

2. With the imminent roll out of the new STEM curriculum it will also be important to gain greater understandings of how computational thinking can best be progressed on a phased basis, across the four stages of primary school education. This highlights the need for longitudinal studies that capture computational thinking development across the primary school years.

3. It is widely recognised that in order to teach any subject, teachers need an awareness of common misconceptions and errors (Ball *et al*. 2008). To date there has been very limited research on what misconceptions arise as students learn through digital technologies in primary school settings (Waite and Quille 2022a). During the course of the investigation several common misconceptions were discovered. However, these misconceptions are likely impacted by both the conceptual learning progression and the pedagogies employed in this study (Waite and Quille 2022a). Therefore, more research needs to be done to identify what

misconceptions commonly arise across a multitude of primary school settings and how are these misconceptions influenced by contextual factors.

6.9 Conclusion

This research study was prompted by the growing recognition, both nationally and internationally, that the development of computational thinking and programming competencies are central to economic and social progress. The subsequent movement of education systems across the globe to incorporate both concepts into their education curricula triggered investigations as to how this could best be achieved. This study adopted a pragmatic epistemological approach and aimed to contribute to this research endeavour by employing a range of methodologies to ascertain how and in what ways computational thinking could be developed. The inductive and deductive nature of the research study was effective in creating a comprehensive picture of how computational thinking can manifest in primary school setting. Contextual and pedagogic factors that afforded or inhibited the development were identified and presented to inform future practice and pedagogy. In the context of the newly established STEM curricular area, perhaps the most important finding of this investigation, was the positive reactions of students, teachers and parents alike. Given the positive correlation between student self-efficacy and their perceived value of programming (Abdunabi *et al*. 2019), providing these early opportunities for students to enjoy and succeed at programming is a worthwhile endeavour. The following quote from Kafai and Burke (2015, p.318) beautifully captures these sentiments:

> "Some critics (Atwood, 2012; Cuban, 2014) have decried the recent resurgence of "coding for all" in schools approach as unrealistic and an all-too-simplified version of what "real" programming actually is. But to these critics, we see these game-making activities providing a low threshold entrance into the world of programming."

These positive reactions give reason for optimism that focussed early exposure to programming can help to address the longstanding negative gender stereotypes that have for so long deprived the technology field of potential female talent.

References

AAUW (2000a) *Tech-Savvy: Educating Girls In The New Computer Age*, Washington: American Association of University Women Educational Foundation, available: https://stelar.edc.org/sites/default/files/TechSavvy.pdf [accessed 28 Jan 2024].

AAUW (2000b) *Tech-Savvy: Educating Girls In The New Computer Age*, Hispanic Media Sales, available: https://hispanicad.com/news/tech-savvy-educating-girls-new-computer-age/ [accessed 19 Jun 2016].

Abdunabi, R., Hbaci, I. and Ku, H.Y. (2019) 'Towards enhancing programming self-efficacy perceptions among undergraduate information systems students', *Journal of Information Technology Education,*18, p.185-206.

ACARA (2016) The Australian Curriculum: Digital Technologies (F–10): Structure, available: https://www.australiancurriculum.edu.au/f-10-curriculum/technologies/digital-technologies/structure/?searchTerm=people [accessed 12 Apr 2019].

ACARA (2023) Digital Technologies: Sequence of content F-10, available: https://docs.acara.edu.au/resources/Digital_Technologies_-_Sequence_of_content.pdf [accessed 21 Jan 2024].

Ackermann, E. (2001) 'Piaget's constructivism, Papert's constructionism: What's the difference', *Future of learning group publication*, 5(3), 438-448.

Adams, N.B. and DeVaney, T. A. (2010) 'The Recursive Knowledge Development Model for Virtual Environments' in Vincenti, G. and Braman, J., eds., *Teaching through Multi-User Virtual Environments: Applying Dynamic Elements to the Modern Classroom: Applying Dynamic Elements to the Modern Classroom*, Hershey, Pennsylvania: Information Science Reference.

Adams, J.C. and Webster, A.R. (2012) 'What do students learn about programming from game, music video, and storytelling projects?' in Smith King, L. and Musicant, D.R., chairs, *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, Raleigh, North Carolina, 29 Feb – 3 Mar, New York: Association for Computing Machinery, 643-648.

Agarwal, R. and Nagar, N. (2010) *Cooperative Learning*, Delhi: Kalpaz Publications.

Ahn, J., Sung, W. and Black, J.B. (2022) 'Unplugged debugging activities for developing young learners' debugging skills', *Journal of Research in Childhood Education*, 36(3), 421-437.

Aivaloglou, E. and Hermans, F. (2016) 'How kids code and how we know: An exploratory study on the Scratch repository' in Sheard, J, Tenenberg, J., Chinn, D. and Dorn, B., chairs, *Proceedings of the 2016 ACM Conference on International Computing Education Research*, Melbourne, 8-12 Sept, New York: Association for Computing Machinery, 53-61.

All you need is code (2017) *About*, The European Coding Initiative, available: http://www.allyouneediscode.eu/about [accessed 18 Sept 2017].

Allan, W., Coulter, B., Denner, J., Erickson, J., Lee, I., Malyn-Smith, J. and Martin, F. (2010) *Computational Thinking for Youth*, Minneapolis: ITEST Learning Resource Center at EDC.

Allsop, Y. (2019) 'Assessing computational thinking process using a multiple evaluation approach', *International Journal of Child-Computer Interaction*, 19, 30-55.

Alosaimi, M.D. (2016) *The role of knowledge management approaches for enhancing and supporting education*, unpublished thesis (P.hD.), Université Panthéon-Sorbonne, available: https://theses.hal.science/tel-01816021/document.

Altun Yalcin, S., Kahraman, S., & Yilmaz, Z. A. (2020) 'Development and validation of Robotic Coding Attitude Scale', *International Journal of Education in Mathematics, Science and Technology*, 8(4), 342-352.

Alves, N.D.C., Von Wangenheim, C.G. and Hauck, J.C. (2019) 'Approaches to assess computational thinking competences based on code analysis in K-12 education: A systematic mapping study', *Informatics in Education*, 18(1), 17-39.

Anabousy, A., Daher, W. and Bassan-Cincinatus, R. (2023) 'Scratch as an Environment for Learning the Coordinate System by Elementary School Students', *Education Sciences*, 13(7), 724-737.

Andersen, I.G. (2023) 'Teachers' gender bias in STEM: Results from a vignette study', *British Educational Research Journal*, 49(4), 833-851.

Angeli, C., Voogt, J., Fluck, A., Webb, M., Cox, M., Malyn-Smith, J., and Zagami, J. (2016) 'A K-6 Computational Thinking Curriculum Framework: Implications for Teacher Knowledge', *Education Technology and Society*, 19(3), 47-57.

Ansip, A. (2016) *Speech by Vice-President Ansip on the launch of the Digital Skills and Jobs Coalition* [speech], 1 Dec, Brussels, available: https://ec.europa.eu/commission/presscorner/detail/en/SPEECH_16_4203.

Ashcraft, C., McLain, B. and Eger, E. (2016) *Women in tech: The* facts, Colorado: National Center for Women & Technology (NCWIT), available: https://wpassets.ncwit.org/wp-content/uploads/2021/05/13193304/ncwit_women-in-it_2016-full-report_final-web06012016.pdf [accessed 12 Dec 2019].

Au, W.K. (1992) *LOGO programming: instructional methods and problem solving*, unpublished thesis (Ph.D.), Massey University.

Austin, E.J., Deary, I.J., Gibson, G.J., McGregor, M.J. and Dent, J.B. (1998) 'Individual response spread in self-report scales: Personality correlations and consequences', *Personality and individual differences*, 24(3), 421-438.

Australian Government (2022) Australian Digital Capability Framework Version 1.0, Canberra: Department of Employment and Workplace Relations, available: https://www.dewr.gov.au/download/15113/australian-digital-capability-framework/32542/australian-digital-capability-framework/pdf [accessed 19 Jan 2024].

Ausubel, D.P. (1960) 'The use of advance organizers in the learning and retention of meaningful verbal material', *Journal of Educational Psychology*, 51(5), 267-272.

Baek, Y. (2011) 'Principles of Educational Digital Game Structure for Classroom Settings' in Management Association, Information Resources, eds., *Gaming*

and Simulations: Concepts, Methodologies, Tools and Applications, Hershey: Information Science Reference, 229-239.

Baker, N. (2015) 'Education must ready students for digital world, says Google boss', *Irish Examiner*, 7 Nov, available: https://www.irishexaminer.com/news/arid-20363604.html [accessed 19 Jun 2016].

Ball, D. L., Thames, M.H. and Phelps, G. (2008) 'Content knowledge for teaching: What makes it special?', *Journal of teacher education*, 59(5), 389-407.

Bandura, A. (2001) 'Social cognitive theory: An agentic perspective', *Annual Review of Psychology*, 52(1), 1-26.

Bantwini, B.D. (2015) 'Do teachers' learning styles influence their classroom practices? A case of primary school natural science teachers from South Africa', *International Journal of Educational Sciences*, 11(1), 1-14.

Barr, V. and Stephenson, C. (2011) 'Bringing computational thinking to K-12: What is involved and what is the role of the computer science education community?', *ACM Inroads*, 2(1), 48-54.

Barron, B., Martin, C., Roberts, E., Osipovich, A. and Ross, M. (2002) 'Assisting and assessing the development of technological fluencies: Insights from a project-based approach to teaching computer science' in Stahl, G., ed., *CSCL '02: Proceedings of the Conference on Computer Support for Collaborative Learning: Foundations for a CSCL Community*, Colorado, 7-11 Jan, New Jersey: Lawrence Erlbaum Associates, 668-669.

Bassey, M. (1999) *Case study research in educational settings*, Buckingham: Open University Press.

Baytak, A. and Land, S.M. (2011) 'An investigation of the artifacts and process of constructing computers games about environmental science in a fifth grade classroom', *Educational Technology Research and Development*, 59, 765-782.

Behar-Horenstein, L.S. and Niu, L. (2011) 'Teaching critical thinking skills in higher education: A review of the literature', *Journal of College Teaching and Learning*, 8(2), 25-42.

Bell, T., Witten, I. and Fellows, M. (1998) *Computer Science Unplugged . . . off-line activities and games for all ages*, Canterbury: Computer Science Unplugged.

Bell, T., Alexander, J., Freeman, I. and Grimley, M. (2009) 'Computer science unplugged: School students doing real computing without computers', *The New Zealand Journal of Applied Computing and Information Technology*, 13(1), 20-29.

Bell, T., Rosamond, F. and Casey, N. (2012) 'Computer Science Unplugged and Related Projects in Math and Computer Science Popularization', in Bodlaender, H.L., Downey, R., Fomin, F.V., Marx, D., eds, *The Multivariate Algorithmic Revolution and Beyond*, Lecture Notes in Computer Science, 7370, Berlin: Springer, 398–456.

Benton, L., Hoyles, C., Kalas, I. and Noss, R. (2017) 'Bridging primary programming and mathematics: Some findings of design research in England', *Digital Experiences in Mathematics Education*, 3, 115-138.

BERA (2011) *Ethical Guidelines for Educational Research*, London: British Educational Research Association, available: https://www.bera.ac.uk/wp-content/uploads/2014/02/BERA-Ethical-Guidelines-2011.pdf.

Bernard, H.R. (2011) *Research Methods in Anthropology: Qualitative and Quantitative Approaches*, 5th ed., Plymouth: AltaMira Press.

Bers, M.U., Flannery, L., Kazakoff, E.R. and Sullivan, A. (2014) 'Computational thinking and tinkering: Exploration of an early childhood robotics curriculum', *Computers and Education*, 72, 145-157.

Bertelsmann Foundation and AOL Time Warner Foundation (2002) *21st century literacy summit – white paper: 21st century literacy in a convergent media world*, Berlin: Author, available: https://ntouk.files.wordpress.com/2015/06/whitepaperenglish.pdf.

Bingham, A.J. (2023) 'From data management to actionable findings: a five-phase process of qualitative data analysis', *International Journal of Qualitative Methods*, 22, 1-11.

Bingham, A.J. and Witkowsky, P. (2021) 'Deductive and Inductive Approaches to Qualitative Analysis' in Vanover, C., Mihas, P. and Saldaña, J., eds.,

Analyzing and Interpreting Qualitative Research: After the Interview, Los Angeles: SAGE, 133-149.

Blikstein, P. and Worsley, M. (2016) 'Children Are Not Hackers: Building a Culture of Powerful Ideas, Deep Learning, and Equity in the Maker Movement' in Peppler, K., Halverson, E. and Kafai, Y.B. eds., *Makeology: Makerspaces as Learning Environments (Volume 1)*, New York: Routledge, 64-79.

Blum, L. and Cortina, T.J. (2007) 'CS4HS: an outreach program for high school CS teachers', *ACM SIGCSE Bulletin*, 39(1), 19-23.

Bonner, S., Chen, P., Jones, K. and Milonovich, B. (2021) 'Formative assessment of computational thinking: Cognitive and metacognitive processes', *Applied Measurement in Education*, 34(1), 27-45.

Boe, B., Hill, C., Len, M., Dreschler, G., Conrad, P. and Franklin, D. (2013) 'Hairball: Lint-inspired static analysis of scratch projects' in Camp, T. and Tymann, P., chairs, *SIGCSE '13:Proceeding of the 44th ACM technical symposium on Computer science education*, Denver, Colorado, 6-9 Mar, New York: Association for Computing Machinery, 215-220.

Booyse, C. (2010) *A constructivist approach in instructional design and assessment practice*, unpublished thesis (P.hD.), University of South Africa, available: https://uir.unisa.ac.za/handle/10500/4680.

Bort, H. and Brylow, D. (2013) 'CS4Impact: measuring computational thinking concepts present in CS4HS participant lesson plans', in Camp, T., Tymann, P., chairs, *SIGCSE '13: Proceeding of the 44th ACM technical symposium on Computer science education*, Denver, Colorado, 6-9 Mar, New York: ACM, 427-432.

Boulden, D.C., Rachmatullah, A., Hinckle, M., Bounajim, D., Mott, B., Boyer, K.E., Lester, J. and Wiebe, E. (2021) 'Supporting Students' Computer Science Learning with a Game-based Learning Environment that Integrates a Use-Modify-Create Scaffolding Framework' in *ITiCSE '21: Proceedings of the 26th ACM Conference on Innovation and Technology in Computer Science Education V.1*, Virtual Event Germany, 26 Jun – 1 Jul, New York: Association for Computing Machinery, 129-135.

Brackmann, C.P., Román-González, M., Robles, G., Moreno-León, J., Casali, A. and Barone, D. (2017) 'Development of computational thinking skills through unplugged activities in primary school', in Barendsen, E. and Hubwieser, P., *WiPSCE '17: Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, Nijmegen, The Netherlands, 8-10 Nov, New York: ACM, 65-72.

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

Brannen, J. (2016) *Mixing Methods: Qualitative and Quantitative Research*, New York: Routledge.

Breathnach, P. (1984) 'Computer Studies Survey', *ASTI journal*, 14 -18.

Brennan, K. and Resnick, M. (2012) 'New frameworks for studying and assessing the development of computational thinking' in Tyson, C.A., chair, *Proceedings of the 2012 annual meeting of the American Educational Research Association (AERA)*, Vancouver, Canada, 13-17 April.

Brennan, K.A. (2013) *Best of both worlds: Issues of structure and agency in computational creation, in and out of school*, unpublished thesis (Ph.D), Massachusetts Institute of Technology, available: https://dspace.mit.edu/bitstream/handle/1721.1/79157/847525655-MIT.pdf?sequence=2&isAllowed=y.

Brennan, K., Chung, M., Martin, W., Cervantes, F., Tally, B and Resnick, M. (2014) Computational Thinking with Scratch: Developing Fluency with Computational Concepts, Practices and Perspectives, available: https://scratched.gse.harvard.edu/ct/assessing.html [accessed 28 May 2016].

Browning, S.F. (2017) *Using Dr. Scratch as a formative feedback tool to assess computational thinking*, unpublished thesis (M.A.), Brigham Young University, available: https://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=7663&context=etd [accessed 1 Mar 2024].

Bruner, J. (1961) 'The act of discovery', *Harvard Educational Review*, 31, 21-32.

Brusilovsky, P., Calabrese, E., Hvorecky, J., Kouchnirenko, A. and Miller, P. (1997) 'Mini-languages: a way to learn programming principles', *Education and information technologies*, 2, 65-83.

BT and Ipsos MORI (2016) *Tech Literacy: A new cornerstone of modern primary school education*, London: British Telecommunications, available: https://www.bt.com/bt-plc/assets/documents/digital-impact-and-sustainability/skills-for-tomorrow/barefoot/ipsos-full.pdf.

Burke, Q. (2012) 'The markings of a new pencil: Introducing programming-as-writing in the middle school classroom', *Journal of Media Literacy Education*, 4(2), 121-135.

Burke, Q., O'Byrne, W.I. and Kafai, Y.B. (2016) 'Computational participation: Understanding coding as an extension of literacy instruction', *Journal of Adolescent & Adult Literacy,* 59(4), 371-375.

Burnett, M., Fleming, S.D., Iqbal, S., Venolia, G., Rajaram, V., Farooq, U., Grigoreanu, V. and Czerwinski, M. (2010) 'Gender differences and programming environments: across programming populations', in Runeson, P. chair, *ESEM '10: Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, Bolzano-Bozen, Italy, 16-17 Sept, New York: Association for Computing Machinery, 265-274, available: https://dl.acm.org/doi/10.1145/1852786.1852824 [accessed 22 Aug 2019].

Busuttil, L. (2014) *I want to be a game maker: experiences of digital game making with eleven year olds*, unpublished thesis (Ph.D.), University of Sheffield, available: https://etheses.whiterose.ac.uk/7593/ [accessed 21 Nov 2019].

Butler, D. and Leahy, M. (2022a) *Baseline report: Towards a successor digital strategy for schools to 2027*, Dublin: DES, available: https://www.gov.ie/en/publication/69fb88-digital-strategy-for-schools/#digital-strategy-for-schools-to-2027 [accessed 29 Dec 2023].

Butler, D. and Leahy, M. (2022b) *Being a digital learner*, Dublin: NCCA, available: https://ncca.ie/media/5574/digital-technology-being-a-digital-learner.pdf [accessed 29 Dec 2023].

Butler, D., Leahy, M., Shiel, G., and Cosgrove, J. (2013) *Building Towards a Learning Society: A National Digital Strategy for Schools*, Dublin: ERC.

Caeli, E.N. and Yadav, A. (2020) 'Unplugged Approaches to Computational Thinking: a Historical Perspective', *TechTrends*, 64(1), 29–36, available: https://link.springer.com/article/10.1007/s11528-019-00410-5 [accessed 26 Jan 2024].

Calder, N. (2010) 'Using Scratch: An integrated problem-solving approach to mathematical thinking', *Australian Primary Mathematics Classroom*, 15(4), 9-14.

Caldwell, H. and Smith, N. (2017) *Teaching Computing Unplugged in Primary Schools: Exploring Primary Computing Through Practical Activities Away from the Computer*, London: Learning Matters.

Callison, D. (2015) *The Evolution of Inquiry: Controlled, Guided, Modeled, and Free*, California: ABC-CLIO, LLC.

Cansu, F. K. and Cansu, S. K. (2019) 'An overview of computational thinking', *International Journal of Computer Science Education in Schools*, 3(1), pp.17-30.

Capobianco, B.M., Deemer, E.D. and Lin, C. (2017) 'Analyzing predictors of children's formative engineering identity development', *International Journal of Engineering Education*, 33(1), pp.44-54.

Carvajal, J. (2017) 'Exploring affective reactions of children to Scratch programming and mathematics: The case of Kim' in Dooley, T. and Gueudet, G., eds., *Proceedings of the Tenth Congress of the European Society for Research in Mathematics Education (CERME 10)*, Dublin, 1-5 Feb, Dublin, Ireland: DCU Institute of Education and ERME, 1226-1227.

de Carvalho, C.V., Cerar, Š., Rugelj, J., Tsalapatas, H. and Heidmann, O. (2020) 'Addressing the gender gap in computer programming through the design and development of serious games', IEEE Revista Iberoamericana de Tecnologias del Aprendizaje, 15(3), pp.242-251.

Carretero Gomez, S., Vuorikari, R. and Punie, Y. (2017) *DigComp 2.1: The Digital Competence Framework for Citizens with eight proficiency levels and*

examples of use, JRC106281, Luxembourg: Publications Office of the European Union.

CEB (1987) Science, technology and the post-primary school. Dublin: Curriculum and Examinations Board.

Chakravorti, B., Chaturvedi, R. and Tunnard, C. (2015) 'Where the digital economy is moving the fastest', *Harvard Business Review*, available: https://hbr.org/2015/02/where-the-digital-economy-is-moving-the-fastest.

Chambers, D., Thiekötter, A. and Chambers, L. (2013) 'Preparing student nurses for contemporary practice: The case for discovery learning', *Journal of Nursing Education and Practice,* 3(9), 106-113.

Ch'ng, S.I., Low, Y.C., Lee, Y.L., Chia, W.C. and Yeong, L.S. (2019) 'Video games: A potential vehicle for teaching computational thinking', in Kong, S. C. and Abelson, H., eds, *Computational Thinking Education*, Singapore: Springer, 247-260.

Chen, P., Yang, D., Metwally, A. H. S., Lavonen, J. and Wang, X. (2023) 'Fostering computational thinking through unplugged activities: A systematic literature review and meta-analysis', *International Journal of STEM Education*, 10(47), available: https://link.springer.com/content/pdf/10.1186/s40594-023-00434-7.pdf.

Ching, Y.H., Yang, D., Wang, S., Baek, Y., Swanson, S. and Chittoori, B. (2019) 'Elementary school student development of STEM attitudes and perceived learning in a STEM integrated robotics curriculum', *TechTrends*, 63, pp.590-601.

Chitale, A., Mohanty, R., & Dubey, N. (2012) *Organizational Behaviour: Text and Cases*, New Delhi: PHI Learning Private Ltd.

Cicconi, M. (2014) 'Vygotsky meets technology: A reinvention of collaboration in the early childhood mathematics classroom', *Early Childhood Education Journal*, 42, pp.57-65.

Cho, S., Pauca, P. and Johnson, D., James, Y. (2014) 'Computational thinking for the rest of us: A liberal arts approach to engaging middle and high school teachers with computer science students', in Searson, M. and Ochoa, M. eds,

Society for information technology & teacher education international conference, Jacksonville, Florida, 15-21 Mar, Waynesville, North Carolina: Association for the Advancement of Computing in Education (AACE), 79-86, available: https://users.wfu.edu/choss/docs/papers/27.pdf.

Clements, D.H. (1999) 'The future of educational computing research: The case of computer programming', *Information Technology in Childhood Education Annual*, 1999(1), 147-179.

CoderDojo Foundation (2014) *Calls for coding and computer science to be prioritised in Irish schools!*, available: https://coderdojo.com/2014/10/20/calls-for-coding-and-computer-science-to-be-prioritised-in-schools-via-rte-news-ie/ [accessed 19 Jul 2016].

CoderDojo Foundation (2022) *CoderDojo: Empowering the next generation of tech stars*, Dogpatch Labs, available: https://dogpatchlabs.com/coder-dojo/ [accessed 5 Feb 2023].

Cohen, L., Manion, L. and Morrison, K. (2007) *Research Methods in Education*, 6th ed., Oxford: Routledge.

Cooper, P.A. (1993) 'Paradigm shifts in designed instruction: From behaviorism to cognitivism to constructivism', *Educational Technology*, 33(5), 12-19.

Cooper, S., Dann, W. and Pausch, R. (2000) 'Developing algorithmic thinking with Alice' in Colton, D., Caouette, J. and Raggad, B , *Information Systems Education conference*, Philadelphia, 9-12 Nov, Chicago: Foundation for Information Technology Education, 506-539.

Conway, P. F. and Brennan-Freeman, E. (2009) 'National Policies and Practices on ICT in Education: Ireland', in Plomp, T., Anderson, R. E., Law, N. and Quale, A., eds., *Cross-National Information and Communication Technology: Policies and Practices in Education*, 2nd ed., Charlotte, NC: Information Age Publishing, 383-401.

Conway, P.F. and Brennan-Freeman, E. (2015) 'The evolution of ICT policy in Ireland 1995-2012: Progress, missed opportunities and future trends', in Butler, D. Marshall, K. and Leahy, M., eds., *Shaping the Future: How*

Technology Can Lead to Educational Transformation, Dublin: Liffey Press, 2 pp.259-287.

Cosgrove, J., Butler, D., Leahy, M., Shiel, G., Kavanagh, L. and Creaven, A. (2014a) *'The 2013 ICT Census in Schools – Main Report'*, Dublin: Educational Research Centre, available: https://www.erc.ie/documents/ict_cencus2013_mainreport.pdf [accessed 5 Apr 2016].

Cosgrove, J., Butler, D., Leahy, M., Shiel, G., Kavanagh, L. and Creaven, A. (2014b) *The 2013 ICT Census in Schools - Summary Report*, Dublin: Educational Research Centre, available from: www.erc.ie/documents/ict_census2013_summaryreport.pdf [accessed 5 Apr 2016].

Cosgrove, J., Duggan, A., Shiel, G. and Leahy, M. (2018) *Digital Learning Framework Trial Evaluation: Final Report*, Dublin: Educational Research Centre, available from: https://www.erc.ie/wp-content/uploads/2018/10/DLF-Trial-Evaluation-Final-Report-Oct-18.pdf.

Cosgrove, J., Moran, E., Feerick, E. and Duggan, A. (2019) *Digital Learning Framework (DLF) national evaluation: Starting off Baseline report*, Dublin: Educational Research Centre, available from: https://www.erc.ie/wp-content/uploads/2020/01/DLF-national-evaluation-baseline-report.pdf.

Coutts, N., Simpson, M. and Drinkwater, R. (2001) 'Using information and communications technology in learning and teaching: a framework for reflection, planning and evaluation in school development', *Teacher Development*, 5(2), 225-239, available: http://dx.doi.org/10.1080/13664530100200135.

Creswell, J.W. (2013) *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*, 3rd ed., Los Angeles: SAGE.

Creswell, J.W. and Creswell, J.D. (2018) *Research Design: Qualitative, Quantitative & Mixed Methods Appraoches*, 5th ed., Los Angeles: SAGE.

Creswell, J.W. and Poth, C.N. (2018) *Qualitative Inquiry and Research Design: Choosing Among Five Approaches*, 4th ed., Los Angeles: SAGE.

Crick, T. (2017) *Computing education: An overview of research in the field*, London: Royal Society, available: https://royalsociety.org/-/media/policy/projects/computing-education/literature-review-overview-research-field.pdf [accessed 22 Mar 2019].

Crosier, D. and Simeoni, E. (2019) *Focus on: Will new technology ever improve education?*, Eurydice, available: https://eurydice.eacea.ec.europa.eu/news/focus-will-new-technology-ever-improve-education [accessed 15 Aug 2019].

Crotty, Y. and Farren, M. (2013) 'Leadership in ICT in Education: our story at Dublin City University' in Crotty, Y. and Farren, M., eds., *Digital Literacies in Education Creative, Multimodal and Innovative Practices*, , Oxford: Peter Lang.

Csizmadia, A., Curzon, P., Dorling, M., Humphreys, S., Ng, T., Selby, C. and Woollard, J. (2015) *Computational Thinking: A guide for* teachers, Swindon: Computing at School.

Csizmadia, A., Standl, B. and Waite, J. (2019) 'Integrating the constructionist learning theory with computational thinking classroom activities', *Informatics in Education*, 18(1), 41-67.

CSTA and ISTE (2011) Computational Thinking in K–12 Education leadership toolkit, available: https://cdn.iste.org/www-root/2020-10/ISTE_CT_Leadership_Toolkit_booklet.pdf [accessed 19 Jun 2019].

Curzon, P., Bell, T., Waite, J. and Dorling, M. (2019) 'Computational Thinking', in Fincher, S. A and Robins, A. V., eds, *The Cambridge Handbook of Computing Education Research*, Cambridge, UK: Cambridge University Press, 513-546.

Curzon, P., McOwan, P. W., Plant, N., & Meagher, L. R. (2014) 'Introducing teachers to computational thinking using unplugged storytelling', in Schulte, C., Caspersen, M. E. and Gal-Ezer, J., eds, *WiPSCE '14: Proceedings of the 9th Workshop in Primary and Secondary Computing Education*, Berlin, Germany, 5-7 Nov, New York: ACM, 89–92.

Cutts, Q.I., Brown, M.I., Kemp, L. and Matheson, C., (2007) 'Enthusing and informing potential computer science students and their teachers', *ACM SIGCSE Bulletin*, *39*(3), 196-200.

Dağ, F., Şumuer, E. and Durdu, L. (2023) 'The effect of an unplugged coding course on primary school students' improvement in their computational thinking skills', *Journal of Computer Assisted Learning*, 39(6), 1902-1918, available: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jcal.12850 [accessed 2 Feb 2024].

Daily, S.B., Leonard, A.E., Jörg, S., Babu, S. and Gundersen, K. (2014) 'Dancing alice: Exploring embodied pedagogical strategies for learning computational thinking' in Dougherty, J.D. Nagel, K., chairs, *SIGCSE '14: Proceedings of the 45th ACM technical symposium on Computer science education*, Atlanta, Georgia, 5-8 Mar, New York: Association for Computing Machinery, 91-96.

Dasgupta, S., Hale, W., Monroy-Hernandez, A. and Hill, B. M. (2016) 'Remixing as a Pathway to Computational Thinking', in Gergle, D. and Ringle Morris, M., chairs, *CSCW '16: Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*, San Francisco, California, 27 Feb – 2 Mar, New York: Association for Computing Machinery, 1438-1449, available: https://dl.acm.org/doi/10.1145/2818048.2819984 [accessed 30 Oct 2019].

Davy, J.D., Audin, K., Barkham, M. and Joyner, C. (2000) 'Student well-being in a computing department', in Tarhio, J., Fincher, S. and Joyce, D., chairs, *ITiCSE00: 5th Annual Conference on Innovations and Technology in Computer Science Education*, Helsinki, Finland, 11-13 Jul, New York: Association for Computing Machinery, 136-139, available: https://doi.org/10.1145/343048.343145

DCENR (2013) *Doing more with Digital: National Digital Strategy for Ireland Phase 1*, Dublin: Stationery Office, available: https://www.gov.ie/en/publication/f4a16b-national-digital-strategy/.

DCYA (2012) *Guidance for Developing Ethical Research Projects involving Children*, Dublin: Department of Children and Youth Affairs, available:

https://www.gov.ie/en/publication/5c1c7-guidance-for-developing-ethical-research-projects-involving-children/.

Dede, C (2008) 'Theoretical perspectives influencing the use of information technology in teaching and learning' in Voogt, J. and Knezek, G., eds., *International Handbook of Information Technology in Primary and Secondary Education*, New York: Springer, 43-62.

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

Denner, J., Werner, L., Campe, S. and Ortiz, E. (2014) 'Pair programming: Under what conditions is it advantageous for middle school students?', J*ournal of Research on Technology in Education*, 46(3), 277-296.

Denny, P., Luxton-Reilly, A. and Simon, B. (2008) 'Evaluating a new exam question: Parsons problems' in Lister, R. and Clancy, M., chairs, *ICER '08: Proceedings of the Fourth international Workshop on Computing Education Research*, Sydney, 6-7 Sept, New York: Association for Computing Machinery, 113-124.

Department for Education (2013) *National curriculum in England: computing programmes of study*, GOV.UK, available: https://www.gov.uk/government/publications/national-curriculum-in-england-computing-programmes-of-study/national-curriculum-in-england-computing-programmes-of-study [accessed 12 Apr 2018].

Department of the Taoiseach (2011) *Statement of Common Purpose 2011-2016*, Dublin: MerrionStreet.ie, available: https://merrionstreet.ie/en/wp-content/uploads/2010/05/programme_for_government_2011.pdf.

DES (1981) *Clár agus rialacha do mheánscoileanna*, Dublin: Stationery Office.

DES (1997) Schools I.T 2000: Full Report, Dublin: Stationery Office, available: https://assets.gov.ie/24665/6bb0ea96e5944d12a39a0fe44ef14c61.pdf.

DES (2001) *Blueprint for the Future of ICT in Irish Education: Three Year Strategic Action Plan 2001-2003*, Dublin: Stationery Office.

DES (2005a) *An Evaluation of Curriculum Implementation in Primary Schools: English, Mathematics and Visual Arts*, Dublin: Stationery Office.

DES (2005b) *Statement of Strategy 2005*-2007, Dublin: Stationery Office, available: https://assets.gov.ie/24337/11692613407e4775bfa433f59849419b.pdf.

DES (2008a) *ICT in Schools: Inspectorate Evaluation Studies*, Dublin: Stationery Office.

DES (2008b) *Investing Effectively in Information and Communication Technology in Schools, 2008-2013*, Dublin: Stationery Office.

DES (2009) *Smart schools = Smart economy: Report of the ICT in schools joint advisory group to the minister for education and science*, Dublin: Stationery Office

DES (2011) *Literacy and Numeracy for Learning and Life: The National Strategy to Improve Literacy and Numeracy among Children and Young People*, Dublin: DES, available: https://www.tui.ie/_fileupload/Literacy%20Numeracy.pdf.

DES (2012a) ICT Action Plan: Meeting the high-level skills needs of enterprise in Ireland, Dublin: DES, available: https://assets.gov.ie/24488/30d4cdfb895e431ebc90aa152e844cc3.pdf.

DES (2012b) A Framework for the Junior Cycle, Dublin: DES available: https://assets.gov.ie/24477/47baa8cfe8164f389e970538883d71d7.pdf.

DES (2015) *Digital strategy for schools 2015-2020: Enhancing teaching, learning and* assessment, Dublin: DES, available: https://www.gov.ie/pdf/?file=https://assets.gov.ie/25151/52d007db333c42f4a6ad542b5acca53a.pdf#page=null.

DES (2016a) *Ireland's National Skills Strategy 2025*, Dublin: DES, available: https://assets.gov.ie/24412/0f5f058feec641bbb92d34a0a8e3daff.pdf.

DES (2016b) Action Plan for Education 2016 – 2019, Dublin: DES, available: https://www.gov.ie/en/collection/action-plan-for-education-2016-2019/.

DES (2017a) *Action Plan for Education 2017*, Dublin: DES, available: https://assets.gov.ie/24343/6cc0b773065946959f270eead7617c69.pdf.

DES (2017b) *Digital Learning Framework for Primary Schools*, Dublin: DES, available: https://www.dlplanning.ie/wp-content/uploads/2018/10/DLF_Primary.pdf.

DES (2017c) *Digital Strategy for Schools 2015-2020 Action Plan 2017*, Dublin: DES, available: https://www.gov.ie/en/publication/f8990-digital-strategy-for-schools-2015-2020/.

DES (2017d) *Literacy and numeracy for learning and life: The national strategy to improve literacy and numeracy among children and young people 2011–2020, Interim Review: 2011 –2016, New Targets: 2017 –2020*, Dublin: DES, available: https://assets.gov.ie/24960/93c455d4440246cf8a701b9e0b0a2d65.pdf [accessed 19 Jan 2024].

DES (2017e) *STEM Education Policy Statement 2017-2026*, Dublin: DES, available: https://www.gov.ie/pdf/?file=https://assets.gov.ie/43627/06a5face02ae4ecd921334833a4687ac.pdf#page=null [accessed 28 Jan 2024].

DES (2018) *Digital Strategy for Schools 2015-2020 Action Plan 2018*, Dublin: DES, available: https://www.gov.ie/en/publication/f8990-digital-strategy-for-schools-2015-2020/.

DES (2019) *Digital Strategy for Schools 2015-2020 Action Plan 2019*, Dublin: DES, available: https://www.gov.ie/en/publication/f8990-digital-strategy-for-schools-2015-2020/.

DES (2020a) *Digital Strategy for Schools 2015-2020 Action Plan 2020*, Dublin: DES, available: https://www.gov.ie/en/publication/f8990-digital-strategy-for-schools-2015-2020/.

DES (2020b) *Digital Learning 2020: Reporting on practice in Early Learning and Care, Primary and Post-Primary Contexts*, Dublin: DES, available: https://www.gov.ie/pdf/?file=https://assets.gov.ie/78007/a37413f9-7423-44bf-86df-01762e04a408.pdf#page=null.

DES (2020c) *STEM Education 2020: Reporting on Practice in Early Learning and Care, Primary and Post-Primary Contexts*, Dublin: DES, available:

https://www.gov.ie/pdf/?file=https://assets.gov.ie/86096/a71d71e8-6ea1-4c62-9cc5-a2a38cffdd6f.pdf#page=null [accessed 28 Jan 2023].

DES (2022a) *Summary Report: Open call for submissions on the development of a new Digital Strategy for Schools to 2027*, Dublin: DES, available: https://www.gov.ie/pdf/?file=https://assets.gov.ie/221301/baa35cf7-bd02-49c0-af20-551708694d30.pdf#page=null [accessed 29 Dec 2023].

DES (2022b) *Digital Strategy for Schools to 2027*, Dublin: DES, available: https://www.gov.ie/en/publication/69fb88-digital-strategy-for-schools/#digital-strategy-for-schools-to-2027 [accessed 29 Dec 2023].

DES (2022c) *Recommendations on Gender Balance in STEM Education*, Dublin: DES, available: https://assets.gov.ie/218113/f39170d2-72c7-42c5-931c-68a7067c0fa1.pdf [accessed 23 Dec 2023].

DES (2023a) *Primary Curriculum Framework For Primary and Special Schools*, Dublin: DES, available: https://curriculumonline.ie/getmedia/84747851-0581-431b-b4d7-dc6ee850883e/2023-Primary-Framework-ENG-screen.pdf.

DES (2023b) *Primary Mathematics Curriculum For Primary and Special Schools*, Dublin: DES, available: https://www.curriculumonline.ie/getmedia/484d888b-21d4-424d-9a5c-3d849b0159a1/PrimaryMathematicsCurriculum_EN.pdf.

DES (2023c) *Launch of the Primary Curriculum Framework*, Circular Letter 0017/2023, Dublin: DES, available: https://www.into.ie/2023/03/09/time-and-support-needed-to-deliver-new-curriculum/.

DES (2023d) *STEM Education Implementation Plan – Phase 1 Enhancing Progress Report*, Dublin: DES, available: https://www.gov.ie/pdf/?file=https://assets.gov.ie/249005/6fc0c45d-4e83-4329-bbd0-e448a4fbfd9f.pdf#page=null [accessed 28 Jan 2024].

DES (2023e) *Initial Teacher Education Policy Statement: Reflecting on a decade of change and creating a vision for the future*, Dublin: DES, available: https://www.gov.ie/pdf/?file=https://assets.gov.ie/251307/f744fbf0-1c49-4be5-bde1-f72ca1f36781.pdf#page=null [accessed 20 May 2024].

DES and DJEI (2014) *ICT Skills Action Plan 2014-2018*, Dublin: DES and DJEI, available: http://edepositireland.ie/handle/2262/90375.

Dewey, J. (1938) *Experience and Education*, New York: Free Press.

Dijkstra, E.W. (1982) 'How do we tell truths that might hurt?', ACM Sigplan Notices, 17(5), 13-15.

Dimitropoulos, K. and Manitsaris, A. (2010) Designing Web-Based Educational Virtual Reality Environments in Vincenti, G. and Braman, J., eds., *Teaching through Multi-User Virtual Environments: Applying Dynamic Elements to the Modern Classroom: Applying Dynamic Elements to the Modern Classroom*, Hershey, Pennsylvania: Information Science Reference.

DJEI (2015) *Innovation 2020: Excellence Talent Impact*, Dublin: DJEI, available: https://enterprise.gov.ie/en/publications/innovation-2020.html.

Doleck, T., Bazelais, P., Lemay, D.J., Saxena, A. and Basnet, R.B. (2017) 'Algorithmic thinking, cooperativity, creativity, critical thinking, and problem solving: exploring the relationship between computational thinking skills and academic performance', *Journal of Computers in Education*, 4, 355-369.

Donnelly, K. (2016) 'Ready, steady... will coding take off in primary schools?', *Irish Independent*, 1 Jun, available: https://www.independent.ie/irish-news/education/ready-steady-will-coding-take-off-in-primary-schools/34762106.html [accessed 12 Dec 2019].

Dou, R., Hazari, Z., Dabney, K., Sonnert, G. and Sadler, P., (2019) 'Early informal STEM experiences and STEM identity: The importance of talking science", *Science Education*, 103(3), pp.623-637.

Dr. Scratch (2019) *Flow Control*, available: http://www.drscratch.org/learn/FlowControl/ [accessed 16 Nov 2022].

Dwyer, H., Boe, B., Hill, C., Franklin, D. and Harlow, D. (2013) 'Computational thinking for physics: Programming models of physics phenomenon in elementary school' in Lang, M. and Sabella, M., chairs, *Physics Education Research Conference 2013*, Portland, OR, 17-18 Jul, Maryland: American Association of Physics Teachers, 133-136.

ECORYS UK (2016) *Digital Skills for the UK Economy*, Birmingham: ECORY available: https://assets.publishing.service.gov.uk/media/5a802e59ed915d74e33f8ed5/DCMSDigitalSkillsReportJan2016.pdf.

Education Council (2015) *National STEM school education strategy: A comprehensive plan for science, technology, engineering and mathematics education in Australia*, Melbourne: Education Council.

Ericson, B.J., Foley, J.D. and Rick, J. (2018) 'Evaluating the efficiency and effectiveness of adaptive parsons problems' in Malmi, L., Korhonen, A., McCartney, R. and Petersen, A., chairs, *ICER '18: Proceedings of the 2018 ACM Conference on International Computing Education Research*, Espoo, Finland, 13-15 Aug, New York: Association for Computing Machinery, 60-68.

Erneling, C.E. (2010) *Towards Discursive Education: Philosophy, Technology, and Modern Education*, Cambridge: Cambridge University Press.

Ertmer, P.A. and Newby, T. (2013) 'Behaviorism, cognitivism, constructivism: Comparing critical features from an instructional design perspective', *Performance Improvement Quarterly*, 26(2), 43-71, available: https://doi.org/10.1002/piq.21143.

European Commission (1993) *New Information Technology in Education: Ireland*, Luxembourg: Office for Official Publications of the European Communities.

European Commission (1994) *Europe and the Global Information Society: Recommendations to the European Council*, Luxembourg: Office for Official Publications of the European Communities, available: https://op.europa.eu/en/publication-detail/-/publication/44dad16a-937d-4cb3-be07-0022197d9459/language-en

European Commission (2000) *eEurope 2002, An Information Society For All, Action Plan*, Santa Maria da Feira: European Commission.

European Commission (2007) *Key competences for lifelong learning: European reference framework*, Brussels: European Commission, available:

https://op.europa.eu/en/publication-detail/-/publication/5719a044-b659-46de-b58b-606bc5b084c1.

European Commission (2010) *A Digital Agenda for Europe*, Brussels: European Commission, available: https://eufordigital.eu/wp-content/uploads/2019/10/COMMUNICATION-FROM-THE-COMMISSION-TO-THE-EUROPEAN-PARLIAMENT.pdf.

European Commission (2011) *Key Data on Learning and Innovation through ICT at School in Europe 2011*, Brussels: Education, Audiovisual and Culture Executive Agency, available: https://op.europa.eu/en/publication-detail/-/publication/8f864668-0211-4a40-bc14-65bf1a97b6a8 [accessed 5 Apr 2016].

European Commission (2013) *'Survey of schools: ICT in education. Final study report. Benchmarking access, use and attitudes to technology in Europe's schools'*, Brussels: European Commission, available: http://files.eun.org/ESSIE/Survey-of-schools-ICT-in-Education-EN.pdf [accessed 5 Apr 2016].

European Commission (2014) *Opening up Education: Innovative Teaching and Learning for All through New Technologies and Open Educational Resources*, Brussels: European Commission, available: https://op.europa.eu/en/publication-detail/-/publication/cd97428e-ab60-4e5f-b9c2-68232274522e.

European Commission (2015) *European Coding Initiative launches new website: All you need is {C<3DE}*, [press release], 19 Mar, available: https://digital-strategy.ec.europa.eu/en/news/european-coding-initiative-launches-new-website-all-you-need-c [accessed 9 Jun 2016].

European Commission (2016a) *Europe's Digital Progress Report*, Brussels: European Commission, available: https://data.consilium.europa.eu/doc/document/ST-9685-2016-ADD-3/en/pdf.

European Commission (2016b) *Developing Computational Thinking in Compulsory Education - Implications for policy and practice*, JRC104188, Seville: Joint Research Centre.

European Commission (2016c) *netWorked Youth Research for Empowerment in the Digital society*, available: https://cordis.europa.eu/project/id/727066 [accessed 19 Jan 2024].

European Commission (2016d) *Coding and computational thinking on the curriculum*, available: https://wikis.ec.europa.eu/display/EAC/ET+2020+Digital?preview=%2F44165766%2F80969906%2F2016-pla-coding-computational-thinking_en+%281%29.pdf [accessed 19 Jun 2018].

European Commission (2020) *Digital Education Action Plan 2021-2027: Resetting education and training for the digital age*, Brussels: European Commission, available: https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:52020DC0624 [accessed 30 Dec 2023].

European Schoolnet (2014) *The e-Skills Manifesto 2014*, Brussels: European Schoolnet, available: http://www.eun.org/resources/detail?publicationID=1261.

European Schoolnet (2015) Computing our future: *Computer programming and coding - Priorities, school curricula and initiatives across Europe*, Brussels: European Schoolnet, available: http://www.eun.org/news/detail?articleId=652951.

Faber, H. H., Wierdsma, M. D., Doornbos, R. P., van der Ven, J. S., & de Vette, K. (2017) 'Teaching computational thinking to primary school students via unplugged programming lessons', *Journal of the European Teacher Education Network*, 12, 13–24, available: https://drive.google.com/file/d/1c93Xk3L11nqrKnV5SnxUvPY4QZJthUaC/view [accessed 22 Mar 2019].

Faherty, R., Nolan, K., Quille, K., Becker, B. and Oldham, E. (2023) 'A Brief History of K-12 Computer Science Education in Ireland', *International Journal of Computer Science Education in Schools*, 6(1), 3-34.

Falloon, G. (2015) 'Building computational thinking through programming in K-6 education: A New Zealand experience' in Gómez Chova, L., López Martínez, A. and Candel Torres, I., eds., *EDULEARN15, the 7th annual International Conference on Education and New Learning Technologies*, Barcelona, Spain,

6-8 Jul, Valencia: IATED Academy, 882-892, available: https://researchcommons.waikato.ac.nz/bitstream/handle/10289/9455/Falloon%20full%20paper%20EDULEARN%2715.pdf?sequence=4&isAllowed=y [accessed 12 Dec 2019].

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.

Fay, A.L. and Mayer, R.E. (1987) 'Children's naive conceptions and confusions about Logo graphics commands', *Journal of Educational Psychology*, 79(3), 254-268.

Feaster, Y., Segars, L., Wahba, S.K. and Hallstrom, J.O. (2011) 'Teaching CS unplugged in the high school (with limited success)', in Rossling, G., chair, *ITiCSE '11: Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*, Darmstadt, Germany, 27-29 Jun, New York: Association for Computing Machinery, 248-252, available: https://www.researchgate.net/publication/220808373_Teaching_CS_unplugged_in_the_high_school_with_limited_success [accessed 12 Dec 2019].

Federici, S. and Stern, L. (2011) 'A Constructionist Approach to Computer Science' in Barton, S., Hedberg, J. and Suzuki, K., eds., *Proceedings of Global Learn Asia Pacific 2011-Global Conference on Learning and Technology*, Melbourne, 28 Mar-1 Apr, Melbourne: Association for the Advancement of Computing in Education, 1352-1361.

Feerick, E., Cosgrove, J. and Moran, E. (2021) *Digital Learning Framework (DLF) national longitudinal evaluation: One year on – Wave 1 report*, Dublin: Educational Research Centre, available from: https://www.erc.ie/wp-content/uploads/2021/06/DLF-W1-full-report.pdf.

Fessakis, G., Gouli, E. and 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.

Fessakis, G., Komis, V., Mavroudi, E., Prantsoudi, S. (2018) 'Exploring the Scope and the Conceptualization of Computational Thinking at the K-12 Classroom

Level Curriculum' in: Khine, M. ed., *Computational Thinking in the STEM Disciplines*, Cham: Springer.

Fields, D., Vasudevan, V. and Kafai, Y.B. (2015) 'The programmers' collective: fostering participatory culture by making music videos in a high school Scratch coding workshop', *Interactive Learning Environments*, 23(5), 613-633.

Finn (2014) 'A third of people think coding is more important than learning Irish, *The Journal*, 17 Oct, available https://www.thejournal.ie/coding-children-school-1729832-Oct2014/ [accessed 19 Jul 2016].

Folk, R., Lee, G., Michalenko, A., Peel, A. and Pontelli, E. (2015) 'GK-12 DISSECT: Incorporating computational thinking with K-12 science without computer access', in DeAntonio, M., Purzer, S., Mina , M. and Korhonen, A., chairs, *2015 IEEE Frontiers in Education Conference (FIE)*, El Paso, TX, 21-24 Oct, Washington, DC: IEEE Computer Society, 1-8, available: https://ieeexplore.ieee.org/document/7344238 [accessed 22 Mar 2019].

Fox, W. and Bayat, M.S. (2007) *A guide to managing research*, Cape Town: Juta and Co. Ltd.

Frädrich, C., Obermüller, F., Körber, N., Heuer, U. and Fraser, G. (2020) 'Common bugs in scratch programs' in Giannakos, M. and Sindre, G., chairs, *Proceedings of the 2020 ACM Conference on Innovation and Technology in Computer Science Education* (pp. 89-95).

Franklin, D., Hill, C., Dwyer, H.A., Hansen, A.K., Iveland, A. and Harlow, D.B. (2016) 'Initialization in scratch: Seeking knowledge transfer' in Alphonce, C. and Tims, J., chairs, *Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, Memphis, Tennessee, 2-5 Mar, New York: Association for Computing Machinery, 217-222.

Fredricks, J.A., Blumenfeld, P.C. and Paris, A.H. (2004) 'School engagement: Potential of the concept, state of the evidence', *Review of Educational Research*, 74(1), 59-109.

Freebody, P. (2003) *Qualitative Research in Education: Interaction and Practice*, London: SAGE.

Fronza, I., El Ioini, N. and Corral, L. (2015) ‘Students want to create apps: leveraging computational thinking to teach mobile software development’ in Settle, A. and Steinbach, T., chairs, *SIGITE '15: Proceedings of the 16th Annual Conference on Information Technology Education*, Chicago, Illinois, 30 Sept-3 Oct, New York: Association for Computing Machinery, 21-26.

Funke, A. and Geldreich, K. (2017) ‘Gender differences in scratch programs of primary school children’ in Barendsen, E. and Hubwieser, P., eds., *Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, Nijmegen, Netherlands, 8-10 Nov, New York: Association for Computing Machinery, 57-64.

Funke, A., Geldreich, K. and Hubwieser, P. (2017) ‘Analysis of scratch projects of an introductory programming course for primary school students’ in Douligeris, C. and Auer, M.E., chairs, *Proceedings of 2017 IEEE Global Engineering Education Conference*, Athens, 25-28 Apr, New York: IEEE,1229-1236.

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

Gagné, R. M. (1985) *The conditions of learning and theory of instruction*, 4th ed., New York: Holt, Rinehart and Winston.

Gallardo-Virgen, J.A. and DeVillar, R.A. (2011) ‘Sharing, talking, and learning in the elementary school science classroom: Benefits of innovative design and collaborative learning in computer-integrated settings’, *Computers in the Schools*, 28(4), 278-290.

Galvin, C. (2009) ‘Public Policy Making: an emerging policymaking modality and its challenges for educational policy research in Ireland’ in Drudy, S., ed., *Education in Ireland: challenge and change*, Dublin: Gill and Macmillan.

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

Gane, B.D., Israel, M., Elagha, N., Yan, W., Luo, F. and Pellegrino, J.W. (2021) ‘Design and validation of learning trajectory-based assessments for

computational thinking in upper elementary grades', *Computer Science Education*, 31(2), 141-168.

Garcia-Holgado, A., Mena, J., Garcia-Penalvo, F. J., Pascual, J., Heikkinen, M., Harmoinen, S., Garcia-Ramos, L., Penabaena-Niebles, R. and Amores, L. (2020) 'Gender equality in STEM programs: A proposal to analyse the situation of a university about the gender gap', in Cardoso A., Alves, G. R. and Restivo M. T., eds., *IEEE Global Engineering Education Conference (EDUCON): Engineering Education for the Future in a Multicultural and Smart World*, Porto, 27-30 Apr: IEEE, 1824-1830, available: https://doi.org/10.1109/educon45650.2020.9125326.

García-Peñalvo, F. J., Reimann, D., Tuul, M., Rees, A. and Jormanainen, I. (2016) *An overview of the most relevant literature on coding and computational thinking with emphasis on the relevant issues for teachers*, Belgium: TACCLE3 Consortium, available: https://zenodo.org/records/165123.

Gardeli, A., & Spyros, V. (2017) 'Creating the computer player: an engaging and collaborative approach to introduce computational thinking by combining 'unplugged' activities with visual programming', *Italian Journal of Educational Technology*, 25(2), 36-50

Gayle, B.M., Preiss, R.W., Burrell, N. and Allen, M. (2009) *Classroom Communication and Instructional Processes: Advances Through Meta-Analysis*, New Jersey: Lawrence Erlbaum Associates.

Geist, E. (2016) 'Robots, programming and coding, oh my!', *Childhood Education*, 92(4), 298-304.

Germia, E. and Panorkou, N. (2020) 'Using Scratch programming to explore coordinates', *Mathematics Teacher: Learning and Teaching PK-12*, 113(4), 293-300.

Gibbs, J. (2016) *Social Measurement through Social Surveys: An Applied Approach*, New York: Routledge.

Gill, K. (2014) *Towards Gender Equality in Education Policies and ICTs: An Action Brief and Toolbox*, USA: Intel Corporation, available:

https://www.intel.nl/content/dam/www/public/us/en/documents/corporate-information/gender-equality-education-ict-unesco-girl-rising.pdf.

Good, C., Rattan, A. and Dweck, C.S. (2012) 'Why do women opt out? Sense of belonging and women's representation in mathematics', *Journal of personality and social psychology*, 102(4), p.700.

Goodenow, C. (1993) 'Classroom belonging among early adolescent students: Relationships to motivation and achievement', *The Journal of early adolescence*, 13(1), pp.21-43.

Goodman, E. and Wilson, R. (2016) 'Seizing the future: Why empowering young people will build a better world', *Ashoka*, 13 Apr, available: https://medium.com/change-maker/seizing-the-future-why-empowering-young-people-will-build-a-better-world-b4496b52e607.

Goos, M., Ryan, V., Lane, C., Leahy, K., Walshe, G., O'Connell, T., O'Donoghue, J. and Nizar, A. (2020) *Review of literature to identify a set of effective interventions for addressing gender balance in STEM in early years, primary and post-primary education settings*, Dublin: DES, available: https://assets.gov.ie/96986/f05f7b2f-e175-442e-85e9-4a2264391843.pdf [accessed 23 Dec 2023].

Götz, K., Feldmeier, P. and Fraser, G. (2022) 'Model-based testing of scratch programs' in Vos, T.E.J. and Gorla, A., chairs, *2022 IEEE Conference on Software Testing, Verification and Validation*, Valencia, Spain, 4-14 Apr, Washington: IEEE, 411-421.

Gough, B. and Madill, A. (2012) 'Subjectivity in psychological science: from problem to prospect', *Psychological methods*, 17(3), 374-384.

Gould, J. (2012) *Learning Theory and Classroom Practice in the Lifelong Learning Sector*, 2nd ed., London: Learning Matters and Sage Publications.

Government of Ireland (2023) STEM Education Implementation Plan to 2026, Dublin: Government of Ireland, available: https://www.gov.ie/pdf/?file=https://assets.gov.ie/249002/3a904fe0-8fcf-4e69-ab31-987babd41ccc.pdf#page=null [accessed 28 Jan 2024].

Green, T.R.G. and Petre, M. (1996) ‘Usability analysis of visual programming environments: a ‘cognitive dimensions’ framework’, *Journal of Visual Languages & Computing*, 7(2), 131-174.

Green, H., Facer, K., Rudd, T., Dillon, P. and Humphreys, P. (2005) *Personalisation and Digital Technologies*, Bristol: Nesta Futurelab.

Greene, J. (2007) *Mixed Methods in Social Inquiry*, San Francisco: Jossey-Bass.

Grippin, P. and Peters, S. (1984) *Learning Theory and Learning Outcomes: The Connection*, Maryland: University Press of America.

Groff, J. S. (2013) *Technology-Rich Innovative Learning Environments*, Paris: OECD, available: https://web-archive.oecd.org/2013-07-10/240666-technology-rich%20innovative%20learning%20environments%20by%20jennifer%20groff.pdf.

Grover, S. (2017) ‘Assessing algorithmic and computational thinking in K-12: Lessons from a middle school classroom’ in Rich, P.J. and Hodges, C.B., eds., *Emerging research, practice, and policy on computational thinking*, Cham: Springer, 269-288.

Grover, S. and Pea, R. (2013) ‘Computational thinking in K–12: A review of the state of the field’, *Educational researcher*, 42(1), 38-43.

Grover, S., Cooper, S. and Pea, R. (2014) ‘Assessing computational learning in K-12’ in Cajander, A. and Daniels, M., chairs, *ITiCSE '14: Proceedings of the 2014 conference on Innovation & technology in computer science education*, Uppsala, Sweden, 21-25 Jun, New York: Association for Computing Machinery, 57-62.

Grover, S. and Basu, S. (2017) ‘Measuring student learning in introductory block-based programming: Examining misconceptions of loops, variables, and boolean logic, in Caspersen, M.E. and Edwards, S.H., chairs, *SIGCSE '17: Proceedings of the 2017 ACM SIGCSE Technical Symposium on Computer Science Education*, Seattle, Washington, 8-11 Mar, New York: Association for Computing Machinery, 267-272.

Gui, M. (2007) 'Formal and substantial Internet information skills: The role of socio-demographic differences on the possession of different components of digital literacy', *First Monday*, 12(9), available: https://doi.org/10.5210/fm.v12i9.2009.

Guzdial, M. (2004) 'Programming environments for novices', in Fincher, S. and Petre, M., eds., *Computer Science Education Research*, Abingdon, UK: Taylor & Francis, 127-154.

Guzdial, M. (2015) *Learner-Centered Design of Computing Education: Research on Computing for Everyone*, San Rafael, CA: Morgan & Claypool Publishers.

Hagan, D., and Markham, S. (2000) 'Does it help to have some programming experience before beginning a computing degree program?', in Tarhio, J., Fincher, S. and Joyce, D., chairs, *ITiCSE00: 5th Annual Conference on Innovations and Technology in Computer Science Education*, Helsinki, Finland, 11-13 Jul, New York: Association for Computing Machinery, 25-28, available: https://doi.org/10.1145/353519.343063.

Hainey, T., Baxter, G. and Ford, A. (2020) 'An evaluation of the introduction of games-based construction learning in upper primary education using a developed game codification scheme for scratch', *Journal of Applied Research in Higher Education*, *12*(3), 377-402.

Hall, J.N. (2013) 'Pragmatism, evidence, and mixed methods evaluation' in Mertens, D.M. and Hesse-Biber, S., eds., *Mixed Methods and Credibility of Evidence in Evaluation: New Directions for Evaluation*, 138, 15-26.

Harasim, L. (2017) Learning Theory and Online Technologies, 2nd ed., New York: Routledge.

Harel, I. and Papert, S. (1991) *Constructionism*, New Jersey: Ablex Publishing.

HEA (2010) *A Study of Progression in Irish Higher Education A report by the Higher Education Authority*, Dublin: The Higher Education Authority.

Hennessey, E.J., Mueller, J., Beckett, D. and Fisher, P.A. (2017) 'Hiding in plain sight: Identifying computational thinking in the Ontario elementary school curriculum', *Journal of Curriculum and Teaching*, 6(1), 79-96.

Hennessy, S., Ruthven, K. and Brindley, S. (2005) 'Teacher perspectives on integrating ICT into subject teaching: commitment, constraints, caution, and change', *Journal of Curriculum Studies*, 37(2), 155-192.

Hilgard E.R., Atkinson R.C. and Atkinson R.L. (1975) *Introduction to Psychology*, 6th ed. New York: Harcourt, Bruce and World.

Hill, W.F. (1977) *Learning: A Survey of Psychological Interpretations*, 3rd ed., New York: Crowell.

Hjorth, M. (2017) *Strengths and weaknesses of a visual programming language in a learning context with children*, unpublished thesis (MSc), KTH Royal Institute of Technology, available: https://www.diva-portal.org/smash/get/diva2:1111152/FULLTEXT01.pdf.

Homer, M. and Noble, J. (2017) 'Lessons in Combining Block-based and Textual Programming', *Journal of Visual Languages and Sentient Systems*, 3(1), 22-39.

Hoong, L., Chick, H. and Moss, J. (2007) 'Classroom research as teacher-researcher', *Mathematics educator*, 10(2), 1-26.

House of the Oireachtas (2011) *Skills Requirements: Discussion* [Joint Committee on Jobs, Social Protection and Education debate], 4 Oct, available at: https://www.oireachtas.ie/en/debates/debate/joint_committee_on_jobs_social_protection_and_education/2011-10-04/4/.

Hoover, A.K., Barnes, J., Fatehi, B., Moreno-León, J., Puttick, G., Tucker-Raymond, E. and Harteveld, C. (2016) 'Assessing computational thinking in students' game designs' in Cox, A. and Toups Dugas, P.O., chairs, *Proceedings of the 2016 Annual Symposium on Computer-human Interaction in Play Companion*, Austin, Texas, 16-19 Oct, New York: Association for Computing Machinery, 173-179.

Howland, J.L., Jonassen, D. and Marra, R.M. (2013) 'Goal of technology integrations: Meaningful learning' in Howland, J.L., Jonassen, D. and Marra, R. M., eds., *Meaningful Learning with Technology*, 4th ed., Essex: Pearson Education, 1-19.

Hsu, T.C., Chang, S.C. and Hung, Y.T. (2018) 'How to learn and how to teach computational thinking: Suggestions based on a review of the literature', *Computers and Education*, 126, 296-310.

Idris, R., Govindasamy, P., Nachiappan, S., & Bacotang, J. (2023) 'Examining Moderator Factors Influencing Students' Interest in STEM Careers: The Role of Demographic, Family, and Gender', *International Journal of Academic Research in Progressive Education and Development*, 12(2), 2298–2312.

Information Society Commission (2002) *Building the Knowledge Society*, Dublin: Information Society Commission, available: http://edepositireland.ie/handle/2262/79892.

International Organisations Services (1990) *Information Technology Atlas – Europe*, 2nd ed., IOS Press: Washington.

INTO (1996) *Information Technology in Irish Primary Education: Issues and Recommendations*, Dublin: INTO.

Irish Computer Society (2014) *Learning code is as important as learning other subjects*, available: https://www.ics.ie/news/view/1230 [accessed 6 Jun 2016].

ISTE and CSTA (2011) *Operational definition of computational thinking for K-12 education*, ISTE and CSTA, available: https://cdn.iste.org/www-root/Computational_Thinking_Operational_Definition_ISTE.pdf [accessed 23 Nov 2016].

Jebb, A.T., Ng, V. and Tay, L. (2021) 'A review of key Likert scale development advances: 1995–2019', *Frontiers in Psychology*, 12, 637547.

Jenkins (2006) 'Confronting the challenges of participatory culture: Media education for the 21st century (Part One)', *Nordic Journal of Digital Literacy*, 2(1), 23-33, available: https://www.idunn.no/doi/10.18261/ISSN1891-943X-2007-01-03.

Jenson, J. and Droumeva, M. (2016) 'Exploring media literacy and computational thinking: A game maker curriculum study', *Electronic Journal of e-Learning*, 14(2), 111-121.

Jiang, S. and Wong, G.K. (2019) ‘Primary school students' intrinsic motivation to plugged and unplugged approaches to develop computational thinking’, *International Journal of Mobile Learning and Organisation*, 13(4), 336-351.

Johanson, R.P. (1988) ‘Computers, cognition and curriculum: Retrospect and prospect’, *Journal of Educational Computing Research*, 4(1), 1-30.

Johnson, T.R. (2014) *The Other Side of Pedagogy: Lacan's Four Discourses and the Development of the Student Writer*, New York: State University of New York Press.

Joint Committee on Jobs, Enterprise and Innovation (2012) *A review of the Information Communication Technology (ICT) skills demand in Ireland*, 31JEI001, Dublin: Houses of the Oireachtas.

K-12 Computer Science Framework Steering Committee (2016) *K-12 Computer Science Framework*, New York: Association for Computing Machinery, available: https://k12cs.org/wp-content/uploads/2016/09/K%E2%80%9312-Computer-Science-Framework.pdf [accessed 23 Nov 2016].

Kafai, Y.B. (1995) *Minds in play: Computer game design as a context for children's learning*, New Jersey: Lawrence Erlbaum Associates.

Kafai, Y.B. and Resnick, M. (1996) *Constructionism in practice: Designing, thinking, and learning in a digital world*, New York: Routledge.

Kafai, Y. B. and Peppler, K. A. (2011) ‘Youth, technology, and DIY: Developing participatory competencies in creative media production’ in Wortham, S., ed., *Review of research in education: Youth cultures, language and literacy*, 35(1), California, SAGE Publications, 89-119.

Kafai, Y.B. and Burke, Q. (2013) ‘Computer Programming Goes Back to School’, *Phi Delta Kappan*, 95(1), 61-65, available: https://doi.org/10.1177/003172171309500111.

Kafai, Y.B., Lee, E., Searle, K., Fields, D., Kaplan, E. and Lui, D. (2014) ‘A crafts-oriented approach to computing in high school: Introducing computational concepts, practices, and perspectives with electronic textiles’, *ACM Transactions on Computing Education (TOCE)*, 14(1), 1-20.

Kafai, Y. B. (2015) ‘Connected Code: A New Agenda for K-12 Programming in Classrooms, Clubs, and Communities’, in *Learning, making & performing math relationships with code*, Western University, Ontario, 19-21 Jun: The Fields Institute for Research in Mathematical Sciences, available: http://www.researchideas.ca/coding/proceedings.html.

Kafai, Y.B. and Burke, Q. (2015) ‘Constructionist gaming: Understanding the benefits of making games for learning’, *Educational Psychologist*, 50(4), pp.313-334.

Kafai, Y. B. and Burke, Q. (2016) *Connected Code: Why Children Need to Learn Programming*, Cambridge, Massachusetts: MIT Press.

Kafai, Y.B. and Ching, C.C. (2001) ‘Affordances of collaborative software design planning for elementary students' science talk’, *The Journal of the Learning Sciences*, 10(3), 323-363.

Kalelioglu, F. and Gülbahar, Y. (2014) ‘The Effects of Teaching Programming via Scratch on Problem Solving Skills: A Discussion from Learners' Perspective’, *Informatics in Education*, 13(1), 33-50.

Kalyuga, S. (2009) ‘Managing cognitive load in adaptive multimedia learning, *Journal of Systemics, Cybernetics and Informatics*, 7(5), 16-21, available: https://www.iiisci.org/journal/sci/FullText.asp?var=&id=XE225KW.

Kampylis, P., Punie, Y. and Devine, J. (2015) *Promoting Effective Digital-Age Learning: A European Framework for Digitally-Competent Educational Organisations*, JRC98209, Luxembourg: Publications Office of the European Union.

Katz, D. L. (1960) ‘Conference report on the use of computers in engineering classroom instruction’, *Communications of the ACM*, 3(10), 522-527, available: https://dl.acm.org/doi/pdf/10.1145/367415.993453.

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 and Education*, 73, 26-39.

Kelleher, C. and Pausch, R. (2005) 'Lowering the barriers to programming: A taxonomy of programming environments and languages for novice programmers', *ACM Computing Surveys*, 37(2), 83-137.

Kember, D. (2003) 'To control or not to control: The question of whether experimental designs are appropriate for evaluating teaching innovations in higher education', *Assessment & Evaluation in Higher Education*, 28(1), 89-101.

Kenna, H. (2022) *Digital Technology – Design Thinking*, Dublin: NCCA, available: https://ncca.ie/media/5575/digital-technology-design-thinking.pdf [accessed 23 Dec 2023].

Kennedy, E., Shiel, G., French, G., Harbison, L., Leahy, M., Ó Duibhir, P., & Travers, J. (2023) *Towards a new literacy, numeracy and digital literacy strategy: A review of the literature*, Dublin: DES, available: https://assets.gov.ie/256086/907a34c9-f95d-4cf1-af52-35589b066d8e.pdf [accessed 19 Jan 2023].

Kennedy, J. (2016) 'Google urges Ireland to put coding on the Leaving Cert curriculum', *Silicon Republic*, 5 Feb, available: https://www.siliconrepublic.com/careers/google-coding-computer-science-leaving-cert-ireland-curriculum [accessed 19 Jul 2016].

Koh, K.H., Basawapatna, A., Bennett, V. and Repenning, A. (2010) 'Towards the automatic recognition of computational thinking for adaptive visual language learning' in Hundhausen, C., Pietriga, E., Díaz, P. and Rosson, M.B., eds., *2010 IEEE Symposium on Visual Languages and Human-Centric Computing*, Leganes, Spain, 21-25 Sept, New Jersey: IEEE, 59-66.

Kölling, M. and McKay, F. (2016) 'Heuristic evaluation for novice programming systems', *ACM Transactions on Computing Education*, 16(3), 1-30.

Kong, S. C. (2019) 'Components and Methods of Evaluating Computational Thinking for Fostering Creative Problem-Solvers in Senior Primary School Education, in Kong, S. C. and Abelson, H., eds, *Computational Thinking Education*, Singapore: Springer, 119-141.

Kong, S.C. and Lai, M. (2022) ‘Computational identity and programming empowerment of students in computational thinking development’, *British Journal of Educational Technology*, 53(3), pp.668-686.

Kordaki, M. (2012) ‘Diverse categories of programming learning activities could be performed within Scratch’ *Procedia-Social and Behavioral Sciences*, 46, 1162-1166.

Lusa Krug, D., Zhang, Y., Mouza, C., Barnett, T., Pollock, L. and Shepherd, D.C. (2023) ‘Using Domain-Specific, Immediate Feedback to Support Students Learning Computer Programming to Make Music’ in Laakso, M. and Monga, M., chairs, *Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1*, Finland, 7-12 Jul, New York: Association for Computing Machinery, 368-374.

Kurland, D.M., Pea, R.D., Clement, C. and Mawby, R. (1986) ‘A study of the development of programming ability and thinking skills in high school students’, *Journal of Educational Computing Research*, 2(4), 429-458.

Kumari, M.A.R., Sundari, R.S. and Rao, D.B. (2004) *Methods of Teaching Educational Psychology*, New Delhi: Discovery Publishing House.

Lai, A.F. and Yang, S.M. (2011) ‘The learning effect of visualized programming learning on 6th graders' problem solving and logical reasoning abilities’ in Fangmin, D., chair, *International Conference on Electrical and Control Engineering*, Yichang, China, 16-18 Sept, New York: IEEE, 6940-6944.

Lambert, L. and Guiffre, H. (2009) ‘Computer science outreach in an elementary school’, *Journal of Computing Sciences in Colleges*, 24(3), 118-124.

Larochelle, M. and Bednarz, N. (1998) ‘Constructivism and education: beyond epistemological correctness’ in Larochelle, M., Bednarz, N. and Garrison, J., eds., *Constructivism and Education*, Cambridge: Cambridge University Press, 3-23.

Lauterbach, A.A. (2018) ‘Hermeneutic phenomenological interviewing: Going beyond semi-structured formats to help participants revisit experience’, *The Qualitative Report*, 23(11), 2883-2898.

Lavrakas, P.J. (2008) *Encyclopedia of Survey Research Methods*, Los Angeles: SAGE.

Lawanto, K.N. (2016) *Exploring trends in middle school students' computational thinking in the online scratch community: A pilot study*, unpublished thesis (MSc), Utah State University, available: https://digitalcommons.usu.edu/cgi/viewcontent.cgi?article=6105&context=etd.

Lea, S.J., Stephenson, D. and Troy, J. (2003) 'Higher education students' attitudes to student-centred learning: beyond 'educational bulimia'?', *Studies in higher education*, 28(3), 321-334.

Leahy, D. and Dolan, D. (2014) 'The Introduction of Computers in Irish Schools' in Tatnall, A. and Daveyin, B., eds., *Reflections on the History of Computers in Education : Early Use of Computers and Teaching about Computing in Schools*, Heidelberg: Springer Berlin, 164-173.

Lee, I., Martin, F., Denner, J., Coulter, B., Allan, W., Erickson, J., Malyn-Smith, J. and Werner. L. (2011) 'Computational thinking for youth in practice', *ACM Inroads*, 2(1), 32-37.

Li, W.L., Hu, C.F. and Wu, C.C. (2016) 'Teaching high school students computational thinking with hands-on activities', in Clear, A. and Cuadros-Vargas, E., chairs, *ITiCSE '16: Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education*, 11-13 Jul, Arequipa, Peru, New York: ACM, 371-371.

Lin, J.M.C. and Liu, S.F. (2012) 'An investigation into parent-child collaboration in learning computer programming', J*ournal of Educational Technology & Society*, 15(1), 162-173.

Lincoln, Y.S. and Guba, E.G. (1985) *Naturalistic Inquiry*, London: SAGE publications.

Liston, M., Frawley, D. and Patterson, V. (2016) *A Study of Progression in Irish Higher Education*, Dublin: Higher Education Authority.

Liu, Z., Zhi, R., Hicks, A. and Barnes, T. (2017) 'Understanding problem solving behavior of 6–8 graders in a debugging game', *Computer Science Education*, 27(1), 1-29.

Livingstone, I. and Hope, A. (2011) *Next Gen: transforming the UK into the world's leading talent hub for the video games and visual effects industries*, London: Nesta.

Lockwood, J. and Mooney, A. (2018) 'Computational Thinking in Education: Where does it fit?: A systematic literary review', *International Journal of Computer Science Education in Schools*, 2(1), 41-60.

Lockwood, E., DeJarnette, A.F., Asay, A. and Thomas, M. (2016) 'Algorithmic Thinking: An Initial Characterization of Computational Thinking in Mathematics', in Wood, M. B., Turner, E. E. Civil, M. and Eli, J. A., eds., *Proceedings of the 38th Annual Meeting of the North American Chapter of the International Group for the Psychology of Mathematics Education*, Tucson, Arizona, 3-6 Nov, University of Arizona, 1588-1595.

Lowe, A.A. (2019) 'Debugging: The Key to Unlocking the Mind of a Novice Programmer?' in Imbrie, P.K., chair, *IEEE Frontiers in Education Conference*, Covington, Kentucky, 16-19 Oct, IEEE, 1-9.

Lu, J.J. and Fletcher, G.H. (2009) 'Thinking about computational thinking', in Fitzgerald, S. and Guzdial, M., chairs, *SIGCSE '09: Proceedings of the 40th ACM technical symposium on Computer science education*, Chattanooga TN USA, 4-7 Mar, New York: Association for Computing Machinery, 260-264, available: https://dl.acm.org/doi/10.1145/1539024.1508959.

Luo, T., So, W.W.M., Wan, Z.H. and Li, W.C. (2021) 'STEM stereotypes predict students' STEM career interest via self-efficacy and outcome expectations", *International Journal of STEM Education*, 8(36), 1-13.

Lye, S. Y. and Koh, J. H. L. (2014) 'Review on teaching and learning of computational thinking through programming: What is next for K-12?', *Computers in Human Behavior*, 41, 51-61.

Lyon, J.A. and J. Magana, A. (2020) 'Computational thinking in higher education: A review of the literature', *Computer Applications in Engineering Education*, 28(5), 1174-1189.

Lytle, N., Cateté, V., Boulden, D., Dong, Y., Houchins, J., Milliken, A., Isvik, A., Bounajim, D., Wiebe, E. and Barnes, T. (2019) 'Use, modify, create: Comparing computational thinking lesson progressions for STEM classes' in Scharlau, B. and McDermott, R., chairs, *TiCSE '19: Proceedings of the 2019 ACM Conference on Innovation and Technology in Computer Science Education*, Aberdeen, 15-17 Jul, New York: Association for Computing Machinery, 395-401.

Mac Giolla Phádraig, B. (2022) 'Strategic leadership in all its "nitty-gritty" detail: Behaviours associated with strategic leaders', *Irish Journal of Education*, 45(3), 1-14, available: https://www.erc.ie/wp-content/uploads/2022/09/29-ERC_IJE-Journal_Brian-Mac-Giolla-Pha%CC%81draig_A4-V4.pdf [accessed 29 Dec 2023].

Maeda (2019) *How to speak machine: Computational thinking for the rest of us*, New York: Penguin Random House.

Malan, D. (n.d.) Scratch for Budding Computer Scientists, available: https://cs.harvard.edu/malan/scratch/printer.php [accessed 4 Jun 2015].

Maloney, J., Burd, L., Kafai, Y., Rusk, N., Silverman, B. and Resnick, M. (2004) 'Scratch: a sneak preview [education]', in Kambayashi, Y., Tanaka, K. and Rose, K., ed.s, *2nd International Conference on Creating, Connecting and Collaborating through Computing*, Kyoto, Japan, 29-30 Jan, Los Alamitos: IEEE Computer Society, 104-109, available: https://ieeexplore.ieee.org/document/1314376.

Maloney, J.H., Peppler, K., Kafai, Y., Resnick, M. and Rusk, N. (2008) 'Programming by choice: urban youth learning programming with scratch' in Dougherty, J.D. and Rodger, S., chairs, *Proceedings of the 39th SIGCSE technical symposium on Computer science education*, Portland, 12-15 Mar, New York: Association for Computing Machinery, 367-371.

Maloney, J., Resnick, M., Rusk, N., Silverman, B. and Eastmond, E. (2010) 'The scratch programming language and environment', *ACM Transactions on Computing Education*, 10(4), 1-15.

Mannila, L. and De Raadt, M. (2006) 'An objective comparison of languages for teaching introductory programming' in Berglund, A., chair, *Baltic Sea '06: Proceedings of the 6th Baltic Sea conference on Computing education research: Koli Calling*, Uppsala, Sweden, 1 Feb, New York: Association for Computing Machinery, 32-37.

Mano, C., Allan, V. and Cooley, D. (2010) 'Effective in-class activities for middle school outreach programs', in Richards, L. G., chair, *2010 IEEE Frontiers in Education Conference (FIE)*, Arlington, Virginia, 27-30 Oct, Piscataway, NJ: IEEE Computer Society, F2E-1-F2E-6.

Margolis, J. and Fisher, A. (2002) *Unlocking the Clubhouse: Women in Computing*, Massachusetts: The MIT Press.

Markus, H. and Nurius, P. (1986) 'Possible selves', *American psychologist*, 41(9), p.954-969.

Martin, W., Brennan, K., Tally, W., and Cervantes, F. (2014) 'Identifying and assessing computational thinking practices' in Polman, J.L., Kyza, E.A., O'Neill, D.K., Tabak, I., Penuel, W.R., Jurow, A.S., O'Connor, K., Lee, T. and D'Amico, L., eds., *Proceedings of The International Conference of the Learning Sciences*, Boulder, Colorado, 23-27 Jun, Colorado: ISLS, 1559-1560.

Mayer, R.E., Dyck, J.L. and Vilberg, W. (1986) Learning to program and learning to think: What's the connection?', *Communications of the* ACM, 29(7), 605-610.

Mayer, R.E. (1988) *Teaching and learning computer programming: multiple research perspectives*, New Jersey: Lawrence Erlbaum Associates Inc..

McCormack (2014) *The e-Skills Manifesto A Call to Arms*, Belgium: European Schoolnet.

McDougall, A., Murnane, J.S., Wills, S. (2014) 'The Educational Programming Language Logo: Its Nature and Its Use in Australia', in Tatnall, A. and

Davey, B., eds, *Reflections on the History of Computers in Education : Early Use of Computers and Teaching about Computing in Schools*, Heidelberg: Springer Berlin, 394-407.

McGann, R. and Hanrahan, P. (2009) *Leading ICT in Schools: It's Not All About The Technology*, Limerick: Mary Immaculate College Curriculum Development Unit.

McGarr, O. (2009) 'The development of ICT across the curriculum in Irish schools: A historical perspective', *British Journal of Educational Technology*, 40(6), 1094–1108, available: https://doi.org/10.1111/j.1467-8535.2008.00903.x.

McKenzie, J.A. (2000) *Beyond Technology: Questioning, Research and the Information Literate School*, Washington: FNO Press.

McLeod, S. (2024) *Piaget's Theory And Stages Of Cognitive Development*, Simply Psychology, available: https://www.simplypsychology.org/piaget.html [accessed 19 Feb 2024].

McLeod, S. (2024) *Jerome Bruner's Theory Of Learning And Cognitive Development*, Simply Psychology, available: https://www.simplypsychology.org/bruner.html [accessed 19 Feb 2024].

Meerbaum-Salant, M. O., Armoni, M. and Ben-Ari, M. (2013) 'Learning computer science concepts with Scratch', *Computer Science Education*, 23(3), 239-264.

Menter, I. Elliot, D., Hall, J., Hall, S., Hulme, M., Lewin, J. and Lowden, K. (2011) *A Guide to Practitioner Research in Education*, Los Angeles: SAGE.

Merino-Armero, J.M., González-Calero, J.A., Cózar-Gutiérrez, R. and Villena-Taranilla, R. (2018) 'Computational thinking initiation. An experience with robots in primary education', *Journal of Research in Science, Mathematics and Technology Education*, 1(2), 181-206.

Merriam, S.B. (1998) *Qualitative Research and Case Study Applications in Education: Revised and Expanded from Case Study Research in Education*, San Francisco: Jossey-Bass Publishers.

Merriam, S.B., Caffarella, R.S. and Baumgartne, L. M. (2012) *Learning in Adulthood: A Comprehensive Guide*, 3rd ed., San Francisco: Jossey-Bass.

MerrionStreet.ie (2016) *Minister Bruton and Minister Halligan call on students to consider ICT careers ahead of CAO deadline,* available: https://merrionstreet.ie/en/news-room/releases/minister_bruton_and_minister_halligan_call_on_students_to_consider_ict_careers_ahead_of_cao_deadline.html [accessed 5 Apr 2016].

Miles, M.B., Huberman, A.M. and Saldaña, J. (2020) *Qualitative Data Analysis: A Methods Sourcebook, Volume 14*, 4th ed., Los Angeles: SAGE.

Millen, J., McDowell, L. and Larmour, P. (2019) *Why Don't More Young Women Study Computing?: A Working Paper Investigating the Low Participation of Girls taking Computing in Northern Ireland Schools*, Belfast: CCEA Research & Statistics Unit, available: https://ccea.org.uk/document/2417 [accessed 12 Dec 2019].

Miller, R. B., Kelly, G. N., and Kelly, J. T. (1988) 'Effects of Logo computer programming experience on problem solving and spatial relations ability', *Contemporary Educational Psychology*, 13(4), 348–357.

Millwood, R., Bresnihan, N., Walsh, D. and Hooper, J. (2018) *Review of literature on computational thinking*, Dublin: NCCA, available: https://ncca.ie/media/3557/primary-coding_review-of-literature-on-computational-thinking.pdf [accessed 6 Sept 2018].

Mitchell, T.R. (1978) *People in Organisations: Understanding their behaviour*, New York: McGraw Hill.

Molina-Ayuso, Á., Adamuz-Povedano, N., Bracho-López, R. and Torralbo-Rodríguez, M. (2023) 'Computational Thinking with Scratch: A Tool to Work on Geometry in the Fifth Grade of Primary Education', *Sustainability,* 16(1), 110-124.

Mooney, C. and Becker, B.A. (2020) 'Sense of Belonging: The Intersectionality of Self-Identified Minority Status and Gender in Undergraduate Computer Science Students' in Maguire, J. and Cutts, Q., eds., United Kingdom & Ireland Computing Education Research Conference, Glasgow, 3-4 Sept, New York: Association for Computing Machinery, 24-30, available: https://dl.acm.org/doi/10.1145/3416465.3416476.

Monette, D.R., Sullivan, T.J. and DeJong, C.R. (2011) *Applied Social Research: A Tool for the Human Services*, 8th ed., United States: Brooks/Cole, Cengage Learning.

Monroy-Hernández, A. and Resnick, M. (2008) 'Empowering kids to create and share programmable media, *Interactions*, 15(2), 50-53.

Moreno-León, J. and Robles, G. (2015) 'Dr. Scratch: a Web Tool to Automatically Evaluate Scratch Projects' in Gal-Ezer, J., Sentance, S. and Vahrenhold, J., chairs, WiPSCE '15: Workshop in Primary and Secondary Computing Education, London, 9-11 Nov, New York: Association for Computing Machinery, 132-133, available: https://dl.acm.org/doi/10.1145/2818314.2818338.

Moreno-León, J. and Robles, G. (2014) 'Analyze your Scratch projects with Dr. Scratch and assess your Computational Thinking skills', presented at 7th international Scratch conference, 12-15 Aug.

Moreno-León, J., Robles, G. and Román-González, M. (2015) 'Dr. Scratch: Automatic analysis of scratch projects to assess and foster computational thinking', Revista de Educación a Distancia (RED), 46,1-23.

Moreno-León, J., Robles, G. and Román-González, M. (2016) 'Comparing computational thinking development assessment scores with software complexity metrics' in Al-Mualla, M. and Auer, M.E., chairs, *2016 IEEE Global Engineering Education Conference*, Abu Dhabi, 10-13 Apr, New York: IEEE, 1040-1045.

Moreno-León, J., Román-González, M., Harteveld, C. and Robles, G. (2017) 'On the automatic assessment of computational thinking skills: A comparison with human experts', in Mark, G. and Fussell, S., chairs, *CHI EA '17: Proceedings of the 2017 CHI Conference Extended Abstracts on Human Factors in Computing Systems*, Denver, Colorado, 6-11 May, New York: Association for Computing Machinery, 2788-2795.

Moreno-León, J., Robles, G. and Román-González, M (2020) 'Towards Data-Driven Learning Paths to Develop Computational Thinking with Scratch', *IEEE Transactions on Emerging Topics in Computing*, 8(1), 193-205.

Morreale, P., Goski, C., Jimenez, L. and Stewart-Gardiner, C. (2012) 'Measuring the impact of computational thinking workshops on high school teachers', *Journal of Computing Sciences in Colleges*, 27(6), 151-157.

Morris, D. and Trushell, J. (2014) 'Computer programming, ICT and gender in the classroom: a male-dominated domain or a female preserve?', *Research in teacher education*, 4(1), 4-9, available: https://repository.uel.ac.uk/item/859y6 [accessed 22 Aug 2019].

Morrison, M. and Newman, T. (2001) 'A study of the impact of background and preparedness on outcomes in CS1', in Walker, H., McCauley, R., Gersting, J. and Russell, I., chairs, *SIGCSE01: 32nd SIGCSE Technical Symposium on Computer Science Education*, Charlotte, North Carolina, 21-25 Feb, New York: Association for Computing Machinery, 179-183, available: https://doi.org/10.1145/364447.364580.

Myers, B. (2008) 'Minds at play: Teens gain 21st century literacy skills designing their own computer games', *American Libraries*, 39(5), 54-57, available: https://web.media.mit.edu/~mres/scratch/American-Libraries.pdf.

NAACE (2013) *Computing in the national curriculum: A guide for primary teachers*, Bedford: Computing at School.

NAEYC and Fred Rogers Center for Early Learning and Children's Media (2012) *Technology and Interactive Media as Tools in Early Childhood Programs Serving Children from Birth through Age 8*, Latrobe, PA: NAEYC and Fred Rogers Center for Early Learning and Children's Media at Saint Vincent College.

National Research Council (1999) *Being fluent with information* technology, Washington, DC: National Academies Press.

NCC and Forfás (2009a) *Annual Competitiveness Report 2009: Volume 1 : Benchmarking Ireland's Performance*, Dublin: Forfás, available: https://www.competitiveness.ie/media/ncc090818_acr_2009.pdf.

NCC and Forfás (2009b) *Statement on Education and* Training, Dublin: Forfás, available:

https://www.competitiveness.ie/media/ncc090309_statement_on_education.pdf.

NCCA (2004a) *Information and Communications Technology (ICT) in the Primary School Curriculum: Guidelines for Teachers*, Dublin: NCCA.

NCCA (2004b) *Curriculum, Assessment and ICT in the Irish Context A Discussion Paper*, Dublin: NCCA available: https://ncca.ie/media/1787/curriculum_assessment_and_ict_in_the_irish_context_a_discussion_paper.pdf.

NCCA (2007a) *ICT Framework - A structured approach to ICT in Curriculum and Assessment: Revised Framework*, Dublin: NCCA.

NCCA (2007b) *ICT Framework: Final report on the school-based developmental initiative*, Dublin: NCCA.

NCCA (2007c) *Children's early learning and development A research paper*, Dublin: NCCA.

NCCA (2008) *Primary Curriculum Review, Phase 2. Final Report with Recommendations*, NCCA 7, Dublin: NCCA.

NCCA (2016a) *Background Paper and Brief for the development of a new Primary Mathematics Curriculum*, Dublin: NCCA.

NCCA (2016b) *Proposals for structure and time allocation in a redeveloped primary curriculum: For consultation*, Dublin: NCCA.

NCCA (2016c) *Desktop audit of coding in the primary curriculum of 22 jurisdictions*, Dublin: NCCA.

NCCA (2016d) *Short Course Coding Specification for Junior Cycle*, Dublin: NCCA.

NCCA (2017a) *Primary Mathematics Curriculum: Draft Specification Junior Infants to Second Class for Consultation*, Dublin: NCCA.

NCCA (2017b) *Leaving Certificate Computer Science Draft Specification*, Dublin: NCCA.

NCCA (2018) *Investigation of curriculum policy on coding in six jurisdictions*, Dublin: NCCA.

NCCA (2019a) *Primary Developments: Final report on the Coding in Primary Schools Initiative*, Dublin: NCCA.

NCCA (2019b) *Primary Curriculum Review and Redevelopment: Information for Schools*, Dublin: NCCA.

NCCA (2019c) *Primary language curriculum*, Dublin: NCCA, available: https://curriculumonline.ie/getmedia/2a6e5f79-6f29-4d68-b850-379510805656/PLCDocument_English.pdf [accessed 19 Jan 2023].

NCCA (2020) Draft Primary Curriculum Framework: Primary Curriculum Review and Redevelopment, Dublin: NCCA, available: https://ncca.ie/media/4456/ncca-primary-curriculum-framework-2020.pdf.

NCCA (2021) *Being a Digital Learner: Research Paper in Support of Technology in the Draft Primary Curriculum 2020*, Dublin: NCCA.

NCCA (2022a) *From Purpose to Practice: Primary Curriculum Developments in Ireland Reflections from the Advisory Panel (Primary)*, Dublin: NCCA.

NCCA (2022b) *Strategic Plan 2022-2025*, Dublin: NCCA.

NCCA (2022c) *Supporting Systewide Primary Curriculum Change*, Dublin: NCCA.

NCCA (2023a) *Primary Curriculum Framework for Primary and Special Schools*, Dublin: NCCA.

NCCA (2023b) *Progression continua*, The Primary Mathematics Toolkit, available: https://www.curriculumonline.ie/Primary/Curriculum-Areas/Mathematics/Toolkit/ [accessed 25 Jan 2024].

NCCA (2024a) Draft Science, Technology and Engineering Education Specification: For primary and special schools, Dublin: NCCA.

NCCA (2024b) *STEM Education: Find out more*, NCCA, available: https://ncca.ie/en/primary/primary-developments/stem-education/ [accessed 25 Jan 2024].

NCTE (2004) *Schools for the Digital Age: Information and Communication Technology in Irish Schools. Progress Report 1998-2002*, Dublin: NCTE.

NCTE (2009) *Planning and Implementing e-Learning in your school: Handbook for Principals & ICT Coordinating Teachers*, Dublin: NCTE.

Neira, R.H., García-Iruela, M. and Connolly, C. (2021) 'Developing and assessing computational thinking in secondary education using a TPACK guided Scratch visual execution environment', *International Journal of Computer Science Education in Schools*, 4(4), 3-23.

Newton, K.J., Leonard, J., Buss, A., Wright, C.G. and Barnes-Johnson, J. (2020) 'Informal STEM: Learning with robotics and game design in an urban context', *Journal of Research on Technology in Education*, 52(2), 129-147.

Ng, W. (2015) *New Digital Technology in Education*, Cham: Springer.

NPADC (2001) *The Impact of Schools IT 2000 – Report and Recommendations to the Minister*, Dublin: NPADC.

NRC (2002) *Technically Speaking: Why All Americans Need to Know More About Technology*, Washington, D.C.: The National Academy Press.

NRC (2010) *Report of a workshop on the pedagogical aspects of computational thinking*, Washington, D.C.: National Academy Press.

Oakley, J. and McDougall, A. (1997) 'Young children as programmers: Fantasy or flight', in McDougall, A. and Dowling, C. eds., *Learning in Logo microworlds*, Richmond, Victoria: Computing in Education Group of Victoria.

O'Brien, J. (2022) *The factors affecting women's advancement in STEM industry in Ireland: A Qualitative Study*, unpublished thesis (Ph.D.), National College of Ireland, available: https://norma.ncirl.ie/5811/1/jenniferobrien.pdf.

O'Doherty, T., Gleeson, J., Moody, J., Johnston, K.T. and McGarr, O. (2004) *Computers and curriculum: Difficulties and dichotomies*, NCCA 4-2001, Dublin: NCCA, available: https://ncca.ie/media/1486/computers_and_curriculum_difficulties_and_dichotomies_rr4.pdf [accessed 3 Apr 2016].

O'Malley (2016) 'Kids may start learning how to code in primary school', *The Journal*, 18 Jul, available: https://www.thejournal.ie/bruton-introducing-coding-primary-schools-ncca-2883123-Jul2016/ [accessed 19 Jul 2016].

O'Neill, B. (2022) Digital Technology – Interacting Safely, Ethically and Responsibly, Dublin: NCCA, available: https://ncca.ie/media/5576/digital-

technology-interacting-safely-ethically-and-responsibly.pdf [accessed 23 Dec 2023].

O'Neill, G. and McMahon, T. (2005) 'Student-centred learning: What does it mean for students and lecturers?' in O'Neill, G., Moore, S. and McMullin, B., eds., *Emerging issues in the practice of university learning and teaching*, Dublin: AISHE, 27-36.

O'Shea, F. (1983) 'Computers in Schools', *Education Ireland*, 1(4) 20-26.

OECD (1996) 'The knowledge-based economy', Paris: OECD Publishing, available: https://one.oecd.org/document/OCDE/GD%2896%29102/En/pdf.

OECD (2001) *Learning to Change: ICT in Schools, Schooling for Tomorrow*, Paris: OECD Publishing, available: https://doi.org/10.1787/9789264195714-en.

OECD (2005) *Are Students Ready for a Technology-Rich World?: What PISA Studies Tell Us*, Paris: OECD Publishing, available: https://www.oecd.org/education/school/programmeforinternationalstudentassessmentpisa/35995145.pdf.

OECD (2006) *Personalising Education, Schooling for Tomorrow*, Paris: OECD Publishing, available: https://doi.org/10.1787/9789264036604-en.

OECD (2015) *Students, Computers and Learning: Making the Connection*, Pisa: OECD Publishing, available: http://dx.doi.org/10.1787/9789264239555-en.

OECD (2020) *Education Policy Outlook in Ireland*, OECD Education Policy Perspectives, No. 18, Paris: OECD Publishing, available: https://www.oecd-ilibrary.org/education/education-policy-outlook-in-ireland_978e377b-en.

OECD (2021) 21st-Century Readers: Developing Literacy Skills in a Digital World, Paris: PISA, OECD Publishing, available: https://www.oecd.org/publications/21st-century-readers-a83d84cb-en.htm [accessed 19 Jan 2023].

Olabe, J.C., Olabe, M.A., Basogain, X. and Castaño, C. (2011) 'Programming and robotics with Scratch in primary education', in Méndez-Vilas, A., ed., *Education in a technological world: Communicating current and emerging research and technological efforts*, Badajoz, Spain: Formatex Research Center, 356-363.

Oldham, E (1998) *The First Twenty-Five Years: Recollections of a Long-Standing Member*, CESI, available: https://www.cesi.ie/about-cesi/first-25-years/ [accessed 3 Apr 2016].

Özçinar, H. (2018) A' Brief Discussion on Incentives and Barriers to Computational Thinking Education' in Özçinar, H., Wong, G. and Ozturk, H.T., eds., *Teaching Computational Thinking in Primary Education*, Hershey, PA: IGI Global, 1-17.

Padgett, D.K. (2012) *Qualitative and Mixed Methods in Public Health*, Los Angeles: SAGE.

Pala, F.K. and Mıhçı Türker, P., 2021. The effects of different programming trainings on the computational thinking skills. *Interactive Learning Environments,* 29(7), 1090-1100.

Palumbo, D.B. (1990) 'Programming language/problem-solving research: A review of relevant issues', *Review of educational research*, 60(1), 65-89.

Paparistodemou, E., Meletiou-Mavrotheris, M. and Vasou, C. (2017) 'Insights from students' reasoning about probability when they design their own Scratch games' in Dooley, T. and Gueudet, G., eds., *Proceedings of the Tenth Congress of the European Society for Research in Mathematics Education (CERME 10)*, Dublin, 1-5 Feb, Dublin, Ireland: DCU Institute of Education and ERME, 812-819.

Papert, S. (1980) *Mindstorms: Children, Computers, And Powerful Ideas*, New York: Basic Books, Inc.

Papert, S. (1984) 'New theories for new learnings', *School Psychology Review*, 13(4), 422-428.

Papert, S. (1990) 'Constructionist Learning', presented at the annual meeting of the American Educational Research Association, 16-20 April.

Papert (2002) 'Hard Fun', *Bangor Daily News*, available: http://www.papert.org/articles/HardFun.html [accessed 23 Jul 2023].

Papert, S. (2005) 'Teaching children thinking', *Contemporary issues in technology and teacher education*, 5(3), 353-365.

Park, Y. and Shin, Y. (2019) ‘Comparing the effectiveness of scratch and app inventor with regard to learning computational thinking concepts’, *Electronics*, 8(11), 1269, 1-12.

Parker, K.R., Chao, J.T., Ottaway, T.A. and Chang, J. (2006) ‘A formal language selection process for introductory programming courses’, *Journal of Information Technology Education: Research*, 5(1), 133-151.

Patton, M.Q. (2002) *Qualitative Research & Evaluation Methods*, 3rd ed., Thousand Oaks, California: Sage Publications.

Patton, M.Q. (2014) *Qualitative Research & Evaluation Methods: Integrating Theory and Practice*, 4th ed., Los Angeles: SAGE.

Patton, E.W., Tissenbaum, M. and Harunani, F. (2019) ‘MIT App Inventor: Objectives, Design, and Development, in Kong, S. C. and Abelson, H., eds, *Computational Thinking Education*, Singapore: Springer, 119-141.

Payne, C.R. (2009) *Information Technology and Constructivism in Higher Education: Progressive Learning Frameworks: Progressive Learning Frameworks*, Hershey, Pennsylvania: Information Science Reference.

Pea, R.D. (1983) *Logo Programming and Problem Solving (Technical Report No. 16),* New York: Bank Street College of Education.

Pea, R.D. and Kurland, D.M. (1984a) ‘On the cognitive effects of learning computer programming’, *New Ideas in Psychology*, 2(2), 137-168.

Pea R.D., Kurland D.M. (1984b) *Logo Programming and the Development of Planning Skills (Technical Report No. 16)*, New York: Bank Street College of Education.

Pea, R.D. (1986) ‘Language-independent conceptual “bugs” in novice programming’, *Journal of Educational Computing Research*, *2*(1), 25-36.

Peppler, K.A. and Kafai, Y.B. (2007) ‘From SuperGoo to Scratch: exploring creative digital media production in informal learning’, *Learning, Media and Technology*, 32(2), 149-166, available: https://doi.org/10.1080/17439880701343337.

Perkins, R. and Shiel, G. (2014) *'Problem solving in PISA: The results of 15-year olds on the computer-based assessment of problem solving in PISA 2012'*, Dublin: Educational Research Centre, available: http://www.erc.ie/documents/p12psreport.pdf [accessed 5 Apr 2016].

Piaget, J. (1952) *The Origins of Intelligence in Children*, New York: International Universities Press.

Prensky, M. (2001) 'Digital natives, digital immigrants part 2: Do they really think differently?', *On the horizon*, 9(6), 1-6.

Preston, D. (2005) 'Pair programming as a model of collaborative learning: a review of the research', *Journal of Computing Sciences in colleges*, 20(4), 39-45.

Pritchard, A. (2013) *Ways of Learning: Learning Theories and Learning Styles in the Classroom*, 3rd ed., New York: Routledge.

Rainey, K., Dancy, M., Mickelson, R., Stearns, E. and Moller, S. (2018) 'Race and gender differences in how sense of belonging influences decisions to major in STEM', *International journal of STEM education*, 5(10), pp.1-14.

Raspberry Pi (2021) *Pedagogy Quick Reads: Improving program comprehension through Parson's Problems*, National Centre for Computing Education, available: https://raspberrypi-education.s3-eu-west-1.amazonaws.com/Quick+Reads/Pedagogy+Quick+Read+13+-+Parson's+Problems.pdf [accessed 2 Sept 21].

Repenning, A. (1993) *Agentsheets: A tool for building domain-oriented dynamic, visual environments*, unpublished thesis (Ph.D.), University of Colorado, available: https://www.academia.edu/73305007/Agentsheets_a_tool_for_building_domain_oriented_dynamic_visual_environments.

Resnick, M. (2006) 'Computer as Paintbrush: Technology, Play, and the Creative Society' in Singer, D., Golikoff, R., and Hirsh-Pasek, K., eds., *Play = Learning: How play motivates and enhances children's cognitive and social-emotional growth*, New York: Oxford University Press, 192-206.

Resnick, M. (2007) 'All I really need to know (about creative thinking) I learned (by studying how children learn) in kindergarten', in Shneiderman, B., ed. 6th

ACM SIGCHI conference on Creativity & cognition, Washington, DC, 13-15 Jun, New York: Association for Computing Machinery, 1-6, available: https://web.media.mit.edu/~mres/papers/kindergarten-learning-approach.pdf [accessed 13 Jul 2016].

Resnick, M., Maloney, J., Monroy-Hernández, A., Rusk, N., Eastmond, E., Brennan, K. Millner, A., Rosenbaum, E., Silver, J., B., Silverman and Kafai, Y. (2009a) 'Scratch: Programming For All', *Communications of the ACM*, 52(11), 60-67, available: https://doi.org/10.1145/1592761.1592779.

Resnick, M., Flanagan, M., Kelleher, C., MacLaurin, M., Ohshima, Y., Perlin, K. and Torres, R. (2009b) 'Growing up programming: democratizing the creation of dynamic, interactive media', in Olsen, D. R., chair, *CHI '09 Extended Abstracts on Human Factors in Computing Systems*, Boston, MA, 4-9 Apr, New York: Association for Computing Machinery, 3293-3296, available: https://web.media.mit.edu/~mres/papers/CHI-programming-panel.pdf [accessed 13 Jul 2016].

Resnick, M. (2012) Let's teach kids to code, November, available at: https://www.ted.com/talks/mitch_resnick_let_s_teach_kids_to_code [accessed 9 Jun 2016].

Resnick, M. (2013) *Learn to Code, Code to Learn*, EdSurge, available: https://www.edsurge.com/news/2013-05-08-learn-to-code-code-to-learn [accessed 12 Dec 2019].

Resnick, M. (2017) *Lifelong Kindergarten: Cultivating Creativity through Projects, Passion, Peers, and Play*, Cambridge, MA: The MIT Press.

Reynolds, C.R. and Fletcher-Janzen, E. (2007) *Encyclopedia of Special Education: A Reference for the Education of Children, Adolescents, and Adults with Disabilities and Other Exceptional Individuals*, New Jersey: John Wiley and Sons Inc.

Rich, K.M. (2020) Trust and feedback in a student teaching support system, *International Christian Community of Teacher Educators Journal*, 15(2), 5.

Rieber, L.P. (1987) 'LOGO and its promise: A research report', *Educational Technology*, 27(2), 12-16.

Rinn, S. (1983) Is there life beyond computer studies, *Education Ireland*, 1(1), 25-26.

Robertson, J. (2012) 'Making games in the classroom: Benefits and gender concerns', *Computers & Education*, 59(2), 385-398.

Robins, A., Rountree, J. and Rountree, N. (2003) 'Learning and teaching programming: A review and discussion', *Computer Science Education*, 13(2), 137-172.

Rodrigues, M. J. (2002) *The New Knowledge Economy in Europe: A Strategy for International Competitiveness and Social Cohesion*, Cheltenham: Edward Elgar.

Romero, J.S. (2010) 'Library programming with LEGO MINDSTORMS, Scratch, and PicoCricket: Analysis of best practices for public libraries', *Computers in Libraries*, 30(1), 16-19, 44-45.

Romero, M., Lepage, A. and Lille, B. (2017) 'Computational thinking development through creative programming in higher education', *International Journal of Educational Technology in Higher Education*, 14(1), 1-15.

Rose, S., Spinks, N. and Canhoto, A.I. (2023) *Management Research: Applying the Principles of Business Research Methods*, 2nd ed., London: Taylor and Francis.

Royal Society (2010) *Current ICT and Computer Science in schools - damaging to UK's future economic prospects?* [press release], 5 Aug, available: https://royalsociety.org/news/2010/ict-computer-science/ [accessed 5 Apr 2016].

Royal Society (2011) *Brain Waves Module 2: Neuroscience: implications for education and lifelong learning*, London: Royal Society.

Sáez-López, J.M., Román-González, M. and 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.

Sáez-López, J. M. and Sevillano-García, M.L. (2017) ‘Sensors, programming and devices in Art Education sessions: One case in the context of primary education’, *Culture and Education*, 29(2), 350-384.

Saxena, A., Lo, C. K., Hew, K. F., & Wong, G. K. W. (2020) ‘Designing unplugged and plugged activities to cultivate computational thinking: An exploratory study in early childhood education’, *The Asia-Pacific Education Researcher*, 29(1), 55–66.

Selby, C. C. and Woollard, J. (2013) *Computational Thinking: The Developing Definition*, Southampton: University of Southampton, available: https://eprints.soton.ac.uk/356481/ [accessed 26 Jan 2024].

Sentance, S. and Csizmadia, A. (2017) ‘Computing in the curriculum: Challenges and strategies from a teacher’s perspective’, *Education and Information Technologies*, 22, 469-495.

Sentance, S. and Waite, J. (2017) ‘PRIMM: Exploring pedagogical approaches for teaching text-based programming in school’ in Barendsen, E. and Hubwieser, P., eds., *WiPSCE '17: Proceedings of the 12th Workshop on Primary and Secondary Computing Education*, Nijmegen, Netherlands, 8-10 Nov, New York: Association for Computing Machinery, 113-114.

Sentance, S., Waite, J. and Kallia, M. (2019) ‘Teaching computer programming with PRIMM: a sociocultural perspective’, *Computer Science Education*, 29(2), 136-176.

SERA (2005) *Scottish Educational Research Association Ethical Guidelines for Educational Research 2005*, Glasgow: Scottish Educational Research Association, available: https://www.sera.ac.uk/wp-content/uploads/sites/13/2018/08/SERA-Ethical-GuidelinesWeb.pdf.

Šerbec, I.N., Cerar, Š. and Žerovnik, A. (2018) ‘Developing computational thinking through games in scratch’, *Education and Research in the Information Society*, 21-30.

Shelton, C. (2016) ‘Time to plug back in? The role of “unplugged” computing in primary schools’, ITTE Newsletter, Spring 2016, available:

https://eprints.chi.ac.uk/id/eprint/1875/1/TimeToPlugBackIn.pdf [accessed 16 Apr 2019].

Shiel, G. and O'Flaherty, A. (2006) *NCTE 2005 Census on ICT Infrastructure in Schools Statistical Report*, Dublin: NCTE.

Seiter, L. and Foreman, B. (2013) 'Modeling the learning progressions of computational thinking of primary grade students' in Simon, B., Clear, A. and Cutts, Q., chairs, *ICER '13: Proceedings of the ninth annual international ACM conference on International computing education research*, San Diego, California, 12-14 Aug, New York: Association for Computing Machinery, 59-66.

Setyawan, T.Y. (2020) 'Primary school pre-service teachers competence level of computational concepts in programming using Dr. Scratch', *Jurnal Inovasi Teknologi Pendidikan*, *7*(2), 177-187.

Smith, D. C., Cypher, A., & Tesler, L. (2000) 'Novice programming comes of age', *Communications of the ACM*, 43(3), 75-81.

Smith, K.N., Jaeger, A.J. and Thomas, D. (2021) '"Science Olympiad Is Why I'm Here": The influence of an early STEM program on college and major choice', *Research in Science Education*, 51(Suppl 1), 443-459.

Smith, P.R. (2018) 'Collecting sufficient evidence when conducting a case study', *The Qualitative Report*, 23(5), 1043-1048.

Stahlbauer, A., Kreis, M. and Fraser, G. (2019) 'Testing scratch programs automatically' in Dumas, M. and Pfahl, D., chairs, *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, Tallinn, Estonia, 26-30 Aug, New York: Association for Computing Machinery, 165-175.

Stake, R.E. (1995) *The Art of Case Study Research*, California: SAGE.

Stake, R.E. (2010) *Qualitative Research: Studying How Things Work*, New York: The Guilford Press.

Starr, C.R. (2018) '"I'm not a science nerd!" STEM stereotypes, identity, and motivation among undergraduate women', *Psychology of Women Quarterly*, 42(4), 489-503.

STEMerg (2016) STEM Education in the Irish School System: A report on Science, Technology, Engineering and Maths Education. Analysis and Recommendations, Dublin: Department of Education and Skills, available: https://assets.gov.ie/25068/d5c86a91ac3b43869f827438f58d88c0.pdf [accessed 11 September 2017].

Sterling, L. (2016) 'Coding in the curriculum: Fad or foundational?', in ACER, *Research Conference 2016 - Improving STEM Learning : What will it take?*, Brisbane Convention and Exhibition Centre, 07-09 Aug, Melbourne: ACER, 79-83, available: https://research.acer.edu.au/cgi/viewcontent.cgi?article=1277&context=research_conference

Stoilescu, D. and McDougall, D. (2011) 'Gender digital divide and challenges in undergraduate computer science programs', *Canadian Journal of Education*, 34(1), 308-333.

Strong G., Higgins, C., Bresnihan, N. and Millwood, R. (2017) 'A Survey of the Prior Programming Experience of Undergraduate Computing and Engineering Students in Ireland', in: Tatnall A. and Webb M., eds., *Tomorrow's Learning: Involving Everyone. Learning with and about Technologies and Computing*, Dublin, Ireland, 3-6 Jul, Cham: Springer, 461-470, available: https://inria.hal.science/hal-01762850/document

Syslo, M.M. and Kwiatkowska, A.B. (2014) 'Learning mathematics supported by computational thinking. Constructionism and creativity', in Futschek, G., & Kynigos, C., eds., *The Constructionism and Creativity Conference*, Vienna, Austria, 19-23 Aug, Vienna: Austrian Computer Society, pp. 367-377.

Tabach, M. (2011) 'The dual role of researcher and teacher: a case study', *For the Learning of Mathematics*, 31(2), 32–34, available: http://www.jstor.org/stable/41319564.

Tang, X., Yin, Y., Lin, Q., Hadad, R. and Zhai, X. (2020) 'Assessing computational thinking: A systematic review of empirical studies', *Computers and Education*, 148, 103798, 1-66, available: https://www.sciencedirect.com/science/article/abs/pii/S0360131519303483 [accessed 1 Mar 2024].

Taub, R., Armoni, M. and Ben-Ari, M. (2012) ‘CS unplugged and middle-school students’ views, attitudes, and intentions regarding CS’, *ACM Transactions on Computing Education*, 12(2), 1-29.

Taylor, J.A. (2013) ‘What is student centredness and is it enough?’, *International Journal of the First Year in Higher Education*, 4(2), 39-48.

Tedre, M. and Denning, P. J. (2016) ‘The Long Quest for Computational Thinking’, in Sheard, J. and Montero, C. S., chairs, *Koli Calling '16: Proceedings of the 16th Koli Calling International Conference on Computing Education Research*, Koli, Finland, 24-27 Nov, New York: Association for Computing Machinery, 120-129, available: https://www.denninginstitute.com/pjd/PUBS/long-quest-ct.pdf [accessed 26 Jan 2024].

Tegon, R. (2017) ‘Design Learning for Inclusion in Virtual World Through Cognitive Enhancement and the Feuerstein MLE's Perspective’ in Panconesi, G. and Guida, M., eds., *Handbook of Research on Collaborative Teaching Practice in Virtual Learning Environments*, Hershey: IGI Global, 181-199.

The Irish Times (2016) ‘Professor John Byrne: The father of computing in Ireland’, *The Irish Times*’, 23 Apr, available: https://www.irishtimes.com/life-and-style/people/professor-john-byrne-the-father-of-computing-in-ireland-1.2620828.

The Royal Society (2017) *After the reboot: computing education in UK schools*, London: The Royal Society.

Troiano, G.M., Snodgrass, S., Argımak, E., Robles, G., Smith, G., Cassidy, M., Tucker-Raymond, E., Puttick, G. and Harteveld, C. (2019) ‘Is My Game OK Dr. Scratch’ in Fails, J.A., chair, *Proceedings of the Interaction Design and Children*, Boise, Idaho, 12-15 Jun, New York: Association for Computing Machinery, 208-219.

Turner, E., Sugimoto, A.T., Stoehr, K. and Kurz, E. (2016) ‘Creating Inequalities from Real-world Experiences, *Mathematics Teaching in the Middle School*, *22*(4), 236-240.

UNESCO (2008) *UNESCO ICT Competency Framework for Teachers*, Paris: UNESCO, available: https://unesdoc.unesco.org/ark:/48223/pf0000265721.

UNESCO (2011a) *UNESCO ICT Competency Framework for Teachers*, Paris: UNESCO, available: https://iite.unesco.org/pics/publications/en/files/3214694.pdf.

UNESCO (2011b) *Transforming Education: The Power of ICT Policies*, Paris: UNESCO.

UNESCO (2022) *Citizenship Education in the Global Digital Age*, Paris: UNESCO, available: https://unesdoc.unesco.org/ark:/48223/pf0000381534 [accessed 19 Jan 2024]

UPC (2015) *54% believe coding should be introduced to school curriculum* [press release], 30 Apr, available: https://www.virginmedia.ie/about-us/press/2015/coding_should_beintroducedtoschoolcurriculum/ [accessed 19 July 2016].

Vaismoradi, M., Turunen, H. and Bondas, T. (2013) 'Content analysis and thematic analysis: Implications for conducting a qualitative descriptive study', *Nursing & Health Sciences*, 15(3), 398-405.

Vallance, M. and Goto, Y. (2015) 'Learning by TKF to promote computational participation in Japanese education' in Auer, M. E., chair, 18th International Conference on Interactive Collaborative Learning and 43rd International Conference on Engineering Pedagogy, Florence, Italy, 20-24 Sept, 20-24, available: https://www.researchgate.net/publication/281924019_Learning_by_TKF_to_promote_computational_participation_in_Japanese_education.

Van Eck, R. (2006) 'Using games to promote girls' positive attitudes toward technology', *Innovate: Journal of Online Education*, 2(3), available: https://www.learntechlib.org/p/171450/ [accessed 19 May 2017].

Van Zyl, S., Mentz, E. and Havenga, M. (2016) 'Lessons learned from teaching Scratch as an introduction to object-oriented programming in Delphi', *African Journal of Research in Mathematics, Science and Technology Education*, 20(2), pp.131-141.

Vekiri, I. (2010) 'Boys' and girls' ICT beliefs: Do teachers matter?', *Computers & Education*, 55(1), 16-23.

Vialle, W., Lysaght, P. and Verenikina, I. (2005) *Psychology for Educators*, Melbourne: Thomson Learning.

Vitores, A. and Gil-Juárez, A. (2015) 'The trouble with 'women in computing': a critical examination of the deployment of research on the gender gap in computer science', *Journal of Gender Studies*, 25(6), 666-680.

Vivian, R., Lozanovski, C. and Falkner, K. (2017) *Supporting teachers to assess F-10 Digital Technologies: Literature review*, Adelaide: Computer Science Education Research (CSER) Group, available: https://www.researchgate.net/publication/321485831_Supporting_teachers_to_assess_F-10_Digital_Technologies_Literature_review [accessed 15 May 2018].

Von Wangenheim, C. G., Alves, N. C., Rodrigues, P. E., and Hauck, J. C. (2017) 'Teaching computing in a multidisciplinary way in social studies classes in school–A case study', *International Journal of Computer Science Education in Schools*, 1(2), 3-16.

Voogt, J., Fisser, P., Good, J., Mishra, P. and Yadav, A. (2015) 'Computational thinking in compulsory education: Towards an agenda for research and practice', *Education and Information Technologies*, 20, 715-728.

Vourletsis, I., Politis, P. and Karasavvidis, I. (2021) 'The Effect of a Computational Thinking Instructional Intervention on Students' Debugging Proficiency Level and Strategy Use' in Tsiatsos, T., Demetriadis, S., Mikropoulos, A. and Dagdilelis, V., eds., *Research on E-Learning and ICT in Education: Technological, Pedagogical and Instructional Perspectives*, Cham: Springer, 15-34.

Vygotsky, L.S. (1978) *Mind in Society*, Cambridge, Massachusetts: Harvard University Press.

Waite, J. (2019) *Tinkering, shared coding, and other techniques for teaching programming*, National Centre for Computing Education, available:

https://blog.teachcomputing.org/shared-coding-tinkering-and-other-techniques-for-teaching-programming/ [accessed 22 Feb 2023].

Waite, J. and Liebe, C. (2021) 'Computer Science Student-Centered Instructional Continuum' in Sherriff, M. and Merkle, L.D., chairs, *SIGCSE '21: Proceedings of the 52nd ACM Technical Symposium on Computer Science Education*, Virtual Event USA, 13-20 Mar, New York: Association for Computing Machinery, 1246-1246.

Waite, J. and Quille, K. (2022a) *Digital Technology – Using, Understanding and Creating: Technical Report*, Dublin: NCCA, available: https://ncca.ie/media/5578/digital-technology-using-understanding-and-creating_technical-report.pdf [accessed 23 Dec 2023].

Waite, J. and Quille, K. (2022b) *Digital Technology – Using, Understanding and Creating*, Dublin: NCCA, available: https://ncca.ie/media/5580/digital-technology-using-understanding-and-creating.pdf [accessed 23 Dec 2023].

Webb, D.C. (2010) 'Troubleshooting assessment: an authentic problem solving activity for it education', *Procedia-Social and Behavioral Sciences*, 9, 903-907.

Webb, H. and Rosson, M.B. (2013) 'Using scaffolded examples to teach computational thinking concepts' in Camp, T. and Tymann, P., chairs, *SIGCSE '13: Proceeding of the 44th ACM technical symposium on Computer science education*, Denver, Colorado, 6-9 Mar, New York: Association for Computing Machinery, 95-100.

Weber, A.M. and Greiff, S. (2023) 'ICT skills in the deployment of 21st century skills: A (cognitive) developmental perspective through early childhood', *Applied Sciences*, 13(7), p.4615-4637.

Weibell, C.J. (2011) *Principles of Learning: A Conceptual Framework for Domain-Specific Theories of Learning*, unpublished thesis (Ph.D.), Bringham Young University, available: https://scholarsarchive.byu.edu/cgi/viewcontent.cgi?article=3758&context=etd [accessed 24 Jun 2016].

Weintrop, D. and Wilensky, U. (2017) ‘Comparing block-based and text-based programming in high school computer science classrooms’, ACM Transactions on Computing Education, 18(1), 1-25.

Weintrop, D., Wise Rutstein, D., Bienkowski, M. and McGee, S. (2021) ‘Assessing computational thinking: an overview of the field’, *Computer Science Education*, 31(2), 113-116.

Wells, D. (2012) ‘Computing in schools: time to move beyond ICT?’, *Research in Teacher Education*, 2(1), 8-13.

Weng, X. and Wong, G.K. (2017) ‘Integrating computational thinking into English dialogue learning through graphical programming tool’, in Tak-yin Hui, R., chair, 6th International conference on teaching, assessment, and learning for engineering, Hong Kong, China, 12-14 Dec, New York: IEEE, 320-325.

Wenger, E (1998) ‘Communities of practice: Learning as a social system’, *Systems Thinker*, 9(5), 2-3.

White, G. (2003) ‘Standardized mathematics scores as a prerequisite for a first programming course’, *Mathematics and Computer Education*, 37(1), 96-104.

Wigfield, A. and Eccles, J.S. (2000) ‘Expectancy–value theory of achievement motivation’, *Contemporary educational psychology*, 25(1), pp.68-81.

Wilson, A. and Moffat, D. C. (2012) ‘Evaluating Scratch to introduce younger children to programming’ in Lawrance, J., and Bellamy, R. eds., *Proceedings of the 22nd Annual Workshop of the Psychology of Programming Interest Group PPIG 2010*, Madrid, Spain, 19-21 Sept, 1-12, available: https://dblp.org/db/conf/ppig/ppig2010.html.

Wilson, C., Sudol, L. A., Stephenson, C., and Stehlik, M. (2010) *Running on empty: the failure to teach K-12 computer science in the digital age*, New York: ACM, available: https://runningonempty.acm.org/fullreport2.pdf.

Wilson, A., Hainey, T. and Connolly, T. (2012) ‘Evaluation of computer games developed by primary school children to gauge understanding of programming concepts’ in Felicia, P., ed., *Proceedings of the 6th European Conference on Games Based Learning*, Cork, 4-5 Oct, Reading: Academic Conferences and Publishing Limited, 549-559.

Wing, J. M. (2006) ‘Computational thinking’, *Communications of the ACM*, 49(3), 33–35.

Wolcott, H.F. (2008) *Ethnography: A Way of Seeing*, 2nd ed., Plymouth: AltaMira Press.

Wong, E.D. (1995) ‘Challenges confronting the researcher/teacher: Conflicts of purpose and conduct’, *Educational researcher*, 24(3), 22-28.

World Bank Group (2011) *Learning for All: Investing in People’s Knowledge and Skills to Promote Development*, Washington, D.C.: The Word Bank, available: https://pubdocs.worldbank.org/en/418511491235420712/Education-Strategy-4-12-2011.pdf.

World Economic Forum (2016) *The Future of Jobs Employment, Skills and Workforce Strategy for the Fourth Industrial Revolution*, Switzerland: World Economic Forum, available: https://www3.weforum.org/docs/WEF_FOJ_Executive_Summary_Jobs.pdf.

Wyeth, P. and Purchase, H.C. (2000) ‘Programming without a computer: A new interface for children under eight’ in Thomas, B. and Warren, J., eds., *Proceedings of the 1st Australasian User Interface Conference*, Canberra, 31 Jan – 3 Feb, Los Alamitos, California: IEEE Computer Society, 141-148.

Yadav, A., Mayfield, C., Zhou, N., Hambrusch, S. and Korb, J.T. (2014) ‘Computational thinking in elementary and secondary teacher education’, ACM Transactions on Computing Education, 14(1), 1-16.

Yadav, A., Stephenson, C. and Hong, H. (2017) ‘Computational thinking for teacher education’, *Communications of the ACM*, 60(4), 55–62.

Yin, R.K. (2009) *Case Study Research: Design and Methods*, 4th ed., California: SAGE.

Zapata-Cáceres, M., Marcelino, P., El-Hamamsy, L. and Martín-Barroso, E. (2024) ‘A Bebras Computational Thinking (ABC-Thinking) program for primary school: Evaluation using the competent computational thinking test’, *Education and Information Technologies*, 1-30.

Zhang, L. and Nouri, J. (2019) ‘A systematic review of learning computational thinking through Scratch in K-9’, *Computers and Education*, 141, 103607, 1-25, available: https://doi.org/10.1016/j.compedu.2019.103607.

Zilka, G. C. (2016) ‘Reducing the digital divide among children who received desktop or hybrid computers for the home’, *Journal of Information Technology Education: Research*, 15, 233-251, available: https://doi.org/10.28945/3519.

Appendices

Appendix A – Previous Research on Programming Initiatives

The different approaches to developing and evaluating computational thinking:

Author	Title	Age/School Year (gender)	No. of Students	Context (class)	Number of Sessions	Concepts
Burke 2012	The Markings of a New Pencil: Introducing Programming-as-Writing in the Middle School Classroom	7th/8th grade 12-14 (boys)	10	Formal, elective class (middle school)	11	Sequences, loops, conditionals (Boolean Logic), variables, event handling, parallelism, coordination and synchronisation.
Sáez-López et al. (2016)	Visual programming languages integrated across the curriculum in elementary school: A two year case study using "Scratch" in five schools	5th/6th grade (boys and girls)	139	Formal (primary school)	20 (1 hour sessions)	Sequences, loops, event handling, parallelism,
Sáez-López and Sevillano-García (2017)	Sensors, programming and devices in art education sessions. One case in the context of primary education	6th grade (boys and girls)	144	Formal (primary school)	20 (1 hour sessions)	Sequences, loops, parallelism, event handling
Meerbaum-Salant *et al.* (2010)	Learning Computer Science Concepts with Scratch	9th grade 14-15 (boys and girls)	46	Formal (middle school)	10 (2-hour sessions)	Loops, parallelism, initialisation, variables
Funke *et al.* (2017)	Analysis of Scratch Projects of an Introductory Programming Course for Primary School Students	4th grade 9-10 (boys and girls)	58	Informal course	3 day course	Sequences, loops, event handling, user interactivity
Wilson *et al.* 2012	Evaluation of Computer Games Developed by Primary School Children to Gauge Understanding of Programming Concepts	Primary 4th to 7th 8-11 (boys and girls)	60	Formal (primary school)	8 (1 hour session)	Sequences, event handling, conditional statements, threads, variables, coordination and synchronisation, iteration, keyboard input, random numbers
Seiter and Foreman (2013)	Modeling the Learning Progressions of Computational Thinking of Primary Grade Students	Primary 1st -6th grade Gender unknown	150	150 projects from teacher galleries on the Scratch website	Unknown	Data representation, conditionals, synchronisation, initialisation, parallelism
Baytak and Land (2011)	An investigation of the artifacts and process of constructing computers games about environmental science in a fifth grade classroom	5th grade (boys and girls)	10	Formal (middle school)	21 (45 mins)	Boolean expressions, conditions, loops, variables, and synchronisation
Fields *et al.* (2015)	The programmers' collective: fostering participatory culture by making music videos in a high school Scratch coding workshop	Freshman 14-15 (boys and girls)	23	Formal High School	8 (2-hour)	Initialisation, synchronisation,
Lee *et al.* (2017)	Analysis of Factors Affecting Achievement in Maker Programming Education in the Age of Wireless Communication					Sequences
Grover *et al.* (2014)	Assessing Computational Learning in K-12	7th and 8th grade 11-14 (Boys and girls)	26	Formal, elective class (middle school)	24 days (55 mins)	Conditionals and loops
Grover and Basu (2017)	Measuring Student Learning in Introductory BlockBased Programming: Examining Misconceptions of Loops, Variables, and Boolean Logic	6th, 7th, 8th grade	100	Formal introductory programming and CS course using Scratch	Unknown	variables, loops, operators and Boolean logic.
Maloney *et al.* (2008)	Programming by Choice: Urban Youth Learning Programming with Scratch	8-18 (boys and girls)	80	Informal 536 Scratch projects	18 month period	Sequences, loops, parallelism, user interaction, conditional statements, synchronization, Boolean logic, variables, and random numbers.
Denner *et al.* (2012)	Computer games created by middle school girls: Can they be used to measure understanding of computer science concepts?	6th grade Middle school (girls)	59 girls 108 projects	Informal voluntary after-school program	14 month period	conditionals

Appendix B - Student Interview Protocol (Brennan *et al.* 2014)

INTERVIEWING STUDENTS ABOUT SCRATCH PROGRAMMING EXPERIENCES

One approach to assessing learners' development as computational thinkers is to engage them in conversations about their projects and processes. These interview questions can be used to engage learners in conversations about their programming experiences at the beginning, middle, and end of their Scratch experience.

AT THE BEGINNING OF THE SCRATCH UNIT

Defining Scratch
Ask the learner to define Scratch, and explain its functionality.
1. If a friend wasn't here today and asked you what Scratch is, and what you can do with it, what would you tell them?

Project Feedback
Share two Scratch projects with the learner and ask them to provide feedback to the project creator for one of the projects.
2. Is there anything you would want to ask the creator before giving them feedback?
3. What suggestions would you give the creator for improving the project? How do you think they could make it more interesting or fun to play with?
4. Any ideas for how you would do this?

Debug It
Present the learner with a Debug It! challenge from Scratch Curriculum Guide, available online at http://scratched.gse.harvard.edu/guide
5. What's going on?
6. How would you fix it?
7. Want to give it try and do what you told me?
8. Did it work like you expected it to work?
9. Can you tell me what you think is going on after your changes?
10. (*If the learner is not able to debug the project*) Before we move on, where would you go for help if you wanted to fix this?

Project Process
Now that the class has been introduced to Scratch, ask the learner about their planning and development process for future Scratch projects.
11. (*If the learner has had the opportunity to start building up a larger project*) I saw that you were making *X*. What do you hope this will eventually look like or do? What do you think you'll need to do to make this?
12. I know you've only just started with Scratch, but after seeing the kinds of things you can do with it, what kinds of projects could you imagine wanting to make?
13. What made you think of that?
14. What might you need to do to create the project you just described?
15. Where would you go to get help for doing different things in Scratch?

DURING THE MIDDLE OF THE SCRATCH UNIT

Defining Scratch
Ask the learner to define Scratch, and explain its functionality.
1. If a friend wasn't here today and asked you what Scratch is, and what you can do with it, what would you tell them?

Project Feedback
Share two Scratch projects with the learner and ask them to provide feedback to the project creator for one of the projects.
2. Is there anything you would want to ask the creator before giving her feedback?
3. How do you think the project could be improved? What suggestions would you give the creator to make the project more interactive?
4. Any ideas for how you would do this?

Debug It
Present the learner with a Debug It! challenge from Scratch Curriculum Guide, available online at http://scratched.gse.harvard.edu/guide
5. What's going on?
6. How would you fix it?
7. Want to give it try and do what you told me?
8. Did it work like you expected it to work?
9. Can you tell me what you think is going on after your changes?
10. (*If the learner is not able to debug the project*) Before we move on, where would you go for help if you wanted to fix this?

Project Process
Now that the class has been introduced to Scratch, ask the learner about their planning and development process for a current Scratch project.
11. Please show me a project you created or are currently working on. What is the project about? Why did you choose this project to share?
12. (*If the project is a work-in-progress*) What do you want your project to look like and do? What do you think you'll need to do to make this?
13. Where did you get that idea? What made you think of that?
14. Did you plan this project before you started programming? If yes, what did you do to plan for it? Did the plan change at all over time? How?
15. Can you share what you've done so far to make your project? (*Ask about specific project elements – e.g., how did you get the ball to bounce is a different direction every time it hits the wall?*)

AT THE END OF THE SCRATCH UNIT

Defining Scratch
Ask the learner to define Scratch, and explain its functionality.
1. If a friend wasn't here today and asked you what Scratch is, and what you can do with it, what would you tell them?

Project Feedback
Share two Scratch projects with the learner and ask them to provide feedback to the project creator for one of the projects.
2. Is there anything you would want to ask the creator before giving them feedback?
3. How would you extend or expand on this project? What suggestions would you give the creator to make the project more interactive?
4. Any idea how you would do this?

Debug It
Present the learner with a Debug It! challenge from Scratch Curriculum Guide, available online at http://scratched.gse.harvard.edu/guide
5. What's going on?
6. How would you fix it?
7. Want to give it try and do what you told me?
8. Did it work like you expected it to work?
9. Can you tell me what you think is going on after your changes?
10. (*If the student is not able to debug the project*) Before we move on, where would you go for help if you wanted to fix this?

Project Process
Ask the student about their planning and development process for a recently completed Scratch project.
11. Can you tell me how you got the idea for that? What made you think of that?
12. Did you plan your project before you started programming? If yes, what did you do to plan for it?
14. Can you describe what you did in Scratch to make your project? (*Ask about specific project elements – e.g., how did you get the character to disappear and reappear later in the story?)*
15. Were there things that were particularly challenging? How did you figure out how to do that? Where did you go for help?
16. (*Look for any borrowed or imported assets (images, sounds) in the project*) Where did you get those? What made you choose those?
17. What are you most proud of about your project? What did you enjoy most about the process? What would you change? Why?
18. Did you share your project with anyone? If yes, who did you share it with? How? If no, do you plan to share it with anyone? Why or why not?

Appendix C - Rubric for Assessing Evolving Computational Practices (Brennan *et al.* 2014)

ASSESSING DEVELOPMENT OF COMPUTATIONAL PRACTICES

The following instrument can be used to assess students' development of fluency with computational thinking practices (experimenting and iterating, testing and debugging, reusing and remixing, abstracting and modularizing). The first column indicates a question for the student (as part of a design journal prompt or interview, for example). The second, third, and fourth columns indicate how low, medium, and high levels of proficiency might be manifested.

EXPERIMENTING AND ITERATING	LOW	MEDIUM	HIGH
Describe how you built your project step by step.	Student provides a basic description of building a project, but no details about a specific project.	Student gives a general example of building a specific project in a certain order.	Student provides details about the different components of a specific project and how they were developed in a certain order.
What different things did you try out as you went along with your project?	Student does not provide specific examples of what s/he tried.	Student gives a general example of trying something in the project.	Student provides specific examples of different things s/he tries in a project.
What revisions did you make and why did you make them?	Student says s/he made no revisions, or only states s/he made revisions but gives no examples.	Student describes one specific revision s/he made to the project.	Student describes the specific things s/he added to the project and why.
Describe different ways you tried to do things in your project, or when you tried to do something new.	Student provides no examples of trying something new.	Student provides an example of trying something new in the project.	Student describes specific new things s/he tried in a project.
TESTING AND DEBUGGING	**LOW**	**MEDIUM**	**HIGH**
Describe what happened when you ran your project that was different from what you wanted.	Student does not describe what was different when s/he ran the project from what s/he wanted.	Student describes what went wrong in the project, but not what s/he wanted it to do.	Student gives a specific example of what happened and what s/he wanted to have happen when s/he ran the project.
Describe how you read through the scripts to investigate the cause of the problem.	Student does not describe a problem.	Student describes reading through the scripts but does not provide a specific example of finding a problem in the code.	Student describes reading through the scripts and provides a specific example of finding a problem in the code.
Describe how you made changes and tested to see what happened.	Student does not describe what problems s/he had or the solution.	Student provides a general example of making a change and testing it out to see if it worked.	This student provides a specific example of making a change and testing it out to see if it worked.
Describe how you considered other ways to solve a problem.	Student does not provide an example of a solution to a problem.	Student provides a general example of a solution to the problem.	This student provides a specific example of a solution to the problem.

REUSING AND REMIXING	LOW	MEDIUM	HIGH
Describe if/how you found inspiration by trying other projects and reading their scripts.	Student does not describe how s/he found ideas or inspiration from other projects.	Student provides a general description of a project that inspired him/her.	Student provides a specific example of project that inspired him/her and how.
How did you select a piece of another project, and adapt it for your project?	Student does not describe how s/he adapted scripts, ideas or resources from other projects.	Student identifies scripts, ideas or resources s/he adapted from other projects.	Student provides specific examples of scripts, ideas or resources s/he adapted from other projects and how.
How did you modify an existing project to improve it, or enhance it?	Student does not describe modifying another project.	Student provides a general description of modifications s/he made to another project.	Student provides specific examples of modifications s/he made to other projects and why.
How did you give credit to people whose work you built on or are inspired by?	Student does not give credit to others.	Student names people whose work inspired him/her.	Student documents in project and/or on the Scratch website the people whose work inspired him/her.
ABSTRACTING AND MODULARIZING	**LOW**	**MEDIUM**	**HIGH**
How did you decide what sprites are needed for your project, and where they should go?	Student provides no description of how s/he selected sprites.	Student provides a general description of deciding to choose certain sprites.	Student provides a specific description of how s/he made decisions about sprites based on goals for the project.
How did you decide what scripts are needed for your project, and what they should do?	Student provides no description of how s/he created scripts.	Student provides a general description of deciding to create certain scripts.	Student provides a specific description of how s/he made decisions about scripts based on goals for the project.
How did you organize the scripts in ways that make sense to you and others?	Student does not describe how s/he organized scripts.	Student provides a general description of how s/he organized the script.	Student provides specific examples of how s/he organized the script and why.

Appendix D - Transcript of Artefact-based Interview with Isabela and Katya

Researcher: Hi girls. Thanks for agreeing to chat with me about the last few weeks coding and your Scratch projects. Are ye okay to get started?

Katya: Yeah.

Isabela: Uh-huh.

Researcher: Great. So, firstly do you want to tell me what you thought of Scratch?

Katya: We loved it. It was so much fun, not like proper school.

Isabela: Yeah like a break from work…

Researcher: And what did you like about it?

Isabela: I like that you can make whatever you like…you decide. You can pick story or game. You can make your own character. Character from TV. Your own song. Nobody tell you what to do…um…that it.

Researcher: And what about you [Katya]?

Katya: Yeah…I like the same. I enjoyed that you worked with your friend and it was both your decision. I liked that too.

Researcher: And did you learn anything then from doing Scratch?

Isabela: Um…how to make games and to work with other people.

Researcher: And what about you [Isabela]?

Katya: We learned how to use different strategies and what you should do if something goes wrong…you know…how to fix it.

Researcher: What kind of strategies were they?

Katya: Ahhhh…

Researcher: Do you want to look at your projects?

Katya: Oh yeah…like…not to give up. Like when we were making our 'Polar Bear' game. We really wanted the Polar Bear to catch the salmon but it took us ages. We couldn't work out what to do. But we kept trying. We had the wrong block but we kept trying until we got it.

Researcher: Great. Can we talk a little bit about your final project now, your animation 'Get off the road'? Can you tell me how you got the idea for this project?

Katya: Hmmm. I think it was cause we did a race [game] a few weeks back.

Isabela: And my Dad's crazy in the car! [laughs]

Researcher: [laughing] Is that him in the car then?

Isabela: No that's me [points to a sprite on the screen]

Katya: And this is me [points to another sprite].

Researcher: And how did you decide what sprites you needed?

Isabela: We needed the two cars to be able to race and we needed the track…ah…and then we wanted some fans and…ah…these…ah…these just came in cause we liked them [laughs].

Researcher: Very nice. Can you play your animation for me so? [Animation plays]. I like the background music. Why did you choose that song?

Isabela: This is our class song. Like our favourite-est song!

Katya: Isabela downloaded it for us…ah…at home. We got a few of them…look [shows a folder with several songs]. These are all our favourites. Look…[points to one of them]…Suzy and Darina used this one…ah…for their car game.

Researcher: So, you shared your music with them. Did you share any of your code? Or did you use anyone else's code?

Isabela: No…not really…no. I don't think so.

Researcher: Right. Can you talk me through how you built this script?

Isabela: Um…maybe we got some of it from the project you did us and um…

Researcher: Okay. That's perfect…okay…so…I see you have a *when space key pressed* block here. Can you tell me why you included this block?

Isabela: Cause we wanted our characters to…um…speak to each other?

Researcher: Very good. And they are both talking at the same time. Did you want them to do that?

Katya: No we wanted this one first [points to one sprite] and then this one but we couldn't get it to…like that.

Isabela: But it was fine then cause it looked like…um…they were having a fight.

Researcher: Yeah it's great! And would you know how to do it now if you wanted to?

Isabela: …Um…I don't know…um…

Researcher: Maybe we'll take a look later. So, at the end you have a…'The End' screen. How did you do that?

Isabela: Ah…we used this [points to the *switch backdrop to* block] and then we make all the cars disappear and the cheer.

Researcher: Can you show me again? [replays animation] So they both happen at the same time?

Isabela: Yeah look we've two…um… this one makes my car disappear and…ah…this is for the cheer.

Researcher: That's great. Well done! Were there things that were particularly hard?

Isabela: I think we were quite good at it…the class like…ah…everybody was good at something weren't they?

Katya: Yeah, I suppose.

Isabela: Like when [Katya] couldn't do something, I could…or if I wasn't [able] we'd find someone who was…[laughs] sometimes you! We were a community of Scratchers, everybody helping each other.

Katya: The debugging was hard! It wasn't always easy to find the mistakes we made in our projects…but everyone in our class was really helpful. They wanted our project to be the best it could be.

Isabela: Yeah. Bernadette was really good. She helped us fix our polar bear game.

Katya: Yeah and she showed us what to do for the next time um not to put in a forever block.

Researcher: And [Katya] do you know why you need a forever block?

Katya: Ah…for the polar bear one?

Researcher: For any one?

Katya: Ah…I'm not sure.

Isabela: So that it keeps on doing something…the computer…

Researcher: Yeah, exactly…you want it to keep doing something. Okay. Can we take a quick look at one of your debugging challenges? Maybe this one. Can you tell me how did you know what was wrong?

Katya: Ah…we knew what it should do and we had to play it to see…and what wasn't right from the sheet you gave us.

Researcher: And what was wrong here?

Katya: Ah…I can't remember.

Isabela: I think we had to make this [points to a stack] cause…I don't know actually.

Researcher: Yeah because the cat wouldn't do both at the same time. Why was that?

Isabela: Oh yeah. We needed to make both happen at the same time so we had to make this new one. Cause it was here after the Meow, meow, meow. It needed to be the same time.

Researcher: That's great girls…And one final question…what were ye most proud of your animation or enjoy the most?

Isabela: Getting it done. It was a relief! [laughs]

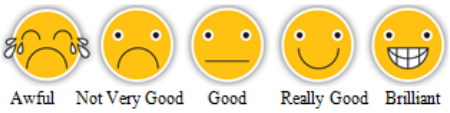
Katya: I liked showing our hard work to the class and seeing all the others the best.

Researcher: Yeah ye made some great projects didn't ye. Well, thanks very much girls that was great of ye to go through all that for me.

Appendix E – Student Pre-Initiative Questionnaire

Student Pre-Initiative Questionnaire

1. Do you enjoy using digital technology (computers, tablets etc)?



(Circle the face that shows best how you feel)

2. What type of technology do you use? (tick **all** that you use)

- ☐ Smart phones (iPhone, Android, Windows Phone)
- ☐ Tablets (iPad, MS Surface, Nexus, etc.)
- ☐ Laptops (Macbook, Windows PC, etc.)
- ☐ Desktop computers
- ☐ E-readers (Nook, Kindle, etc.)
- ☐ Video Games
- ☐ None
- ☐ Other: _____

3. What do you use this technology for?

- ☐ Surfing the Internet
- ☐ Watching Movies or Downloading Music
- ☐ Word Processing
- ☐ Watching You Tubers
- ☐ Email
- ☐ Social Media
- ☐ Gaming
- ☐ Other: _____

4. What do you like most about using technology?

_____

_____

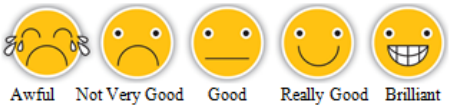
5. What do you like least about using technology?

_____

_____

6. Do you think you are good at using technology?

(Circle the face that shows best how you feel)



7. Do you think technology allows you to be creative?

Yes ☐ No ☐

Why/why not?

_____

_____

8. Do you ever use technology with others?

Yes ☐ No ☐

Explain

_____

_____

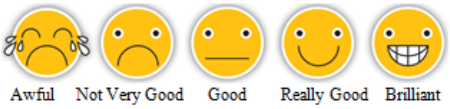
9. Have you ever done computer coding (creating instructions for computers) before?

Yes ☐ No ☐

10. Have you ever used the Scratch coding language before?

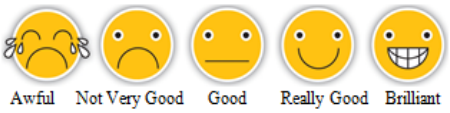
Yes ☐ No ☐

11. We are going to be doing coding for the next few weeks, how do you think you will do?



12. Do you think it is a good idea to have coding classes in schools?

(Circle the face that shows best how you feel)



13. Do you think there should be coding classes in schools?

Yes ☐ No ☐

Why/why not?

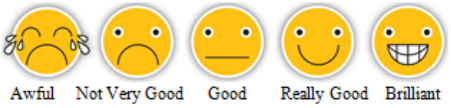
_____

_____

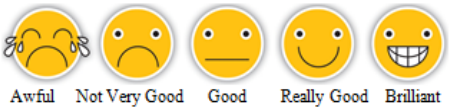
Appendix F – Student Post-Initiative Questionnaire

Student Post-Initiative Questionnaire

1. Do you enjoy using digital technology (computers, tablets etc)?
   (Circle the face that shows best how you feel)



2. Do you think you are good at using technology?
   (Circle the face that shows best how you feel)



3. Do you think technology allows you to be creative?
   Yes ☐ No ☐
   Why/why not?

   _____

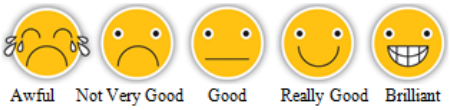
   _____

4. Do you ever use technology with others?
   Yes ☐ No ☐
   Explain

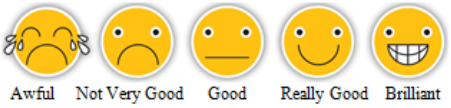
   _____

   _____

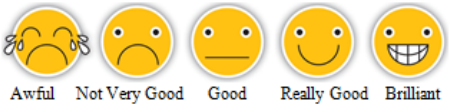
5. Did you enjoy your coding classes?
   (Circle the face that shows best how you feel)



6. How do you think you did in the coding classes?
   (Circle the face that shows best how you feel)



7. Do you think it is a good idea to have coding classes in schools?
   (Circle the face that shows best how you feel)



8. Do you think there should be coding classes in schools?
   Yes ☐ No ☐
   Why/why not?

_____

_____

9. What did you learn in the coding classes?

_____

_____

10. Does Scratch remind you of anything else you do in school?

_____

_____

11. Did you learn anything in Scratch that will help you in other subjects in school?

_____

_____

12. What did you like most about doing Scratch?

_____

_____

13. What did you like least about Scratch?

_____

_____

Appendix G – Parent Questionnaire

Parent Questionnaire

1. Has your child used the Scratch programming language before?
   Yes ☐ No ☐
2. What type of technology does your child actively interact with on a day-to-day basis?
   Please check all that apply.
   - ☐ Smart phones (iPhone, Android, Windows Phone)
   - ☐ Tablets (iPad, MS Surface, Nexus, etc.)
   - ☐ Laptops (Macbook, Windows PC, etc.)
   - ☐ Desktop computers
   - ☐ E-readers (Nook, Kindle, etc.)
   - ☐ Video Games
   - ☐ None
   - ☐ Other: _____
3. What do they use it for?
   - ☐ Surfing the Internet
   - ☐ Watching Movies or Downloading Music
   - ☐ Word Processing
   - ☐ Watching You Tubers
   - ☐ Email
   - ☐ Social Media
   - ☐ Gaming
   - ☐ Other: _____
4. How much time does your child spend interacting with technology per day?
   Please estimate to the best of your ability (For this questionnaire technology does not include TV).
   - ☐ Less than half an hour
   - ☐ Between half an hour and one hour
   - ☐ Between one and two hours
   - ☐ Between two and three hours
   - ☐ More than three hours
5. How would you rate **your** own computer skills?

	1	2	3	4	5	6	7	8	9	10	
Very Bad	○	○	○	○	○	○	○	○	○	○	Very Good

6. Before the coding classes started in the school, were you aware of the accessibility of coding to children?
   Yes ☐ No ☐

7. How much exposure to coding or coding games has your child had?
   - ☐ A little bit, but only in school
   - ☐ A little bit, but only at home
   - ☐ Quite a bit
   - ☐ Not sure
   - ☐ Other: ☐

8. Did your child/children enjoy the coding classes?
   Yes ☐ No ☐

9. Are you aware of the government's plans to introduce coding to the primary school curriculum?
   Yes ☐ No ☐

10. Do you think it is a good idea to introduce coding to the primary school curriculum?
    Yes ☐ No ☐

11. Why, or why not?

    _____

    _____

12. One reason they want to introduce coding to primary school education is to improve children's computational thinking skills. What is your understanding of the term computational thinking?

    _____

    _____

    _____

13. What do you think was the most important thing your child learned from these classes?

    _____

    _____

14. Any Other Comments

    _____

    _____

    _____

Appendix H – Participating Class Teacher Questionnaire

Participating Class Teacher Questionnaire

1. Have you got a Computer/Tablet at home? Yes ☐ No ☐
2. If yes, how often do you use it?
   1-2 times a week ☐ Most Days ☐ Every day ☐
3. What do you use it for?

   Surfing the Internet ☐
   Watching Movies or Downloading Music ☐
   Word Processing ☐
   Watching You Tubers ☐
   Email ☐
   Social Media ☐
   Other _____
4. How would you rate your computer skills?

   1 2 3 4 5 6 7 8 9 10
   Very Bad ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ Very Good
5. How often, if at all, do you use technology in lessons?

All the time	Very Often	Sometimes	Never

6. How important, if at all, do you think it is that children leave primary school tech literate?

Very important	Fairly important	Neither	Not very important	Not at all important	Don't know

7. Before the coding classes began in the school, were you aware of the accessibility of coding to children?
   Yes ☐ No ☐
8. Are you aware of the government's plans to introduce coding to the primary school curriculum?
   Yes ☐ No ☐
9. Do you think it is a good idea to introduce coding to the primary school curriculum?
   Yes ☐ No ☐
   Why, or why not?

   _____

   _____

   _____

10. What do you think are the barriers, if any, to introducing coding to the curriculum?

Lack of maintenance / technical support / equipment	
Poor internet connection	
Lack of tech equipment	
Lack of budget for buying new technology	
Lack of training in how to use technology	
My lack of confidence using technology	
Poor resources such as lesson plans	
Ability of children to use technology	
Don't know what technology is best to use in classroom	
Fear of online risks to pupils	
Time constraints	
Other (please specify)	

Any Additional Comment

_____

_____

11. One reason the government want to introduce coding to primary school education is to improve children's computational thinking skills. What do you understand by the term 'computational thinking'?

_____

_____

_____

12. How confident would you be in teaching coding as part of a new curriculum?

Very confident	Fairly confident	Neither	Not very confident	Not at all confident

13. If you received some training/professional development, would you feel comfortable teaching coding?

Yes ☐ No ☐

14. Would teaching coding in the future be something that would interest you?

Yes ☐ No ☐

Any Additional Comments

_____

_____

_____

15. What expectations of coding did you have at the beginning, and have they been met?

_____

_____

_____

16. How do you feel your class performed in the Scratch lessons compared to how they would perform in normal ICT lessons?

_____

_____

_____

17. Were there were opportunities for your students to collaborate during the coding classes?

Yes ☐ No ☐

Explain

_____

_____

_____

18. Did you feel that the coding activities allowed the students to express their creativity?

Yes ☐ No ☐

Explain

_____

_____

_____

19. Would you recommend Scratch to your colleagues as a tool to teach computing? Will you use Scratch yourself in future?

Yes ☐ No ☐

20. Do you think your students enjoyed their experience of coding?

Yes ☐ No ☐

21. What do you think was the most important thing your child learned from these classes?

_____

_____

22. Any Other Comments

_____

_____

_____

Appendix I – Additional Teacher Questionnaire

Additional Teacher Questionnaire

1. Have you got a Computer/Tablet at home? Yes ☐ No ☐
2. If yes, how often do you use it?
   1-2 times a week ☐ Most Days ☐ Every day ☐
3. What do you use it for?

   Surfing the Internet ☐
   Watching Movies or Downloading Music ☐
   Word Processing ☐
   Watching You Tubers ☐
   Email ☐
   Social Media ☐
   Other _____
4. How would you rate your computer skills?

   1 2 3 4 5 6 7 8 9 10
   Very Bad ○ ○ ○ ○ ○ ○ ○ ○ ○ ○ Very Good
5. How often, if at all, do you use technology in lessons?

All the time	Very Often	Sometimes	Never

6. How important, if at all, do you think it is that children leave primary school tech literate?

Very important	Fairly important	Neither	Not very important	Not at all important	Don't know

7. Before the coding classes began in the school, were you aware of the accessibility of coding to children?
   Yes ☐ No ☐
8. Are you aware of the government's plans to introduce coding to the primary school curriculum?
   Yes ☐ No ☐
9. Do you think it is a good idea to introduce coding to the primary school curriculum?
   Yes ☐ No ☐
   Why, or why not?

   _____

   _____

   _____

10. What do you think are the barriers, if any, to introducing coding to the curriculum?

Lack of maintenance / technical support / equipment	
Poor internet connection	
Lack of tech equipment	
Lack of budget for buying new technology	
Lack of training in how to use technology	
My lack of confidence using technology	
Poor resources such as lesson plans	
Ability of children to use technology	
Don't know what technology is best to use in classroom	
Fear of online risks to pupils	
Time constraints	
Other (please specify)	

Any Additional Comment

_____

_____

11. One reason the government want to introduce coding to primary school education is to improve children's computational thinking skills. What do you understand by the term 'computational thinking'?

_____

_____

_____

12. How confident would you be in teaching coding as part of a new curriculum?

Very confident	Fairly confident	Neither	Not very confident	Not at all confident

13. If you received some training/professional development, would you feel comfortable teaching coding?

Yes ☐ No ☐

14. Would teaching coding in the future be something that would interest you?

Yes ☐ No ☐

Any Additional Comments

_____

_____

_____

Appendix J - Diary Entry from Week 8 of the Programming Initiative

Date: 29th March 2017	**Week 8**

Concepts

I had sixth class first. A lot of the groups are making progress with their projects now.
Loops: in particular the conditional loops are still causing the students difficulties. It seems they are not very intuitive and this is a difficult concept to grasp. The *forever* aspect of this concept caused issues for students repeatedly today and that is despite the fact that students have had numerous experiences which necessitated the use of conditional loops in previous weeks. The application or transfer of this concept to new coding contexts is challenging as students don't seem to be able to recognise the similarities between different situations that require the *forever* block.
Parallelism: Lots of the students programs illustrate frequent use of parallelism at this point. When I ask them about why they use parallelism they understand that parallelism allows for more than one thing to happen at a time. Even in third class they have split code into separate scripts, particularly to facilitate the use of both movement and sound simultaneously. Some of the girls in the older classes want to create actions with more than one movement.
Synchronisation: While it can still cause initial misunderstandings as to why code won't work, the students are starting to gain a better understanding of timing synchronisation (the *wait* block). Event synchronisation (particularly the *broadcast* and *when I receive* blocks) are no problem to them now. While state synchronisation involving a backdrop change is frequently being implemented without problems.
As many of the final projects the students are working on are either stories or animations it is unlikely that there will be much opportunity for them to demonstrate their understanding of ***user interactivity***.

Practices

Experimenting and Iterating

In this stage of project development all of the students are experimenting and iterating to build their projects. The students like to frequently see how their code is working. For example, [Louise] and [Bernadette] were working very hard today on developing a realistic impression of Alice falling down a hole. Their idea had two aspects, Alice falling and the hole changing as Alice fell. In the process of developing the hole, they worked on several aspects, developing little bits and then trying them out. First they wanted the hole to grow on the screen so they played around with sizing and the need for a reset script, all the while testing to see the product of their code. Then they did the same to change the colour of the clay as Alice fell down the hole. Finally they added some roots, changing their position to create the impression of Alice falling through layers.

Testing and Debugging

Some students have started to identify the source of programming errors by making connections with previous occurrences of these errors. Georgina and her group realised the need for a wait block between the changes to the size because they remembered how fast the computer executed code from previous projects. [Bernadette] was able to help [Katya] and [Isabela] fix a bug in their program because she had previously encountered the same bug in her race program.

Reusing and Remixing

At this stage of the initiative there is less evidence of the children reusing and remixing. Every now and then students get an idea for their project from something they see in somebody's project as they walk around the room or they ask someone how they coded a particular action in their program. In that sense they are reusing by incorporating those external ideas and snippets of code into their projects.

Abstracting and Modularizing Many of the students are breaking their projects into distinct parts to facilitate the coding process. [Matilda] and [Caroline] are working on their monkey singing competition. They have separated their scripts for the singing monkey into the welcome on stage, the singing of the song and the after song to facilitate interactions with the singing competition host.
Perspectives
Questioning The students have made great strides in their ability to navigate the computer since week one. They are now competently creating, editing and saving their projects. Their perspective on their ability to wield technology has also evolved since the start. They are much more confident to try different things and are also seeking out opportunities to share their skills and knowledge with others. At the end of the fifth class session today, [Matilda] and [Caroline] demonstrated different ways that Scratch allowed users to add and edit sounds, assuming the role of the 'expert Scratcher', providing a great demonstration and happily answering any queries from the class.
Connecting There is lots of movement and chatter in the classrooms now. They are really keen to show off their projects to their classmates but equally they are interested to see what everyone else is doing. This has brought some tension with it as sometimes the constructive feedback is difficult to take. [Izzy] told [Megan] and [Sylvia] that their game was 'impossible' and that the ball was 'moving much too fast'. The two girls were quite cross and told her that they weren't finished and to 'mind her own'. Despite this they spent the next ten minutes working out how to change the speed of their ball and then decided that they would incorporate some levels with different ball speeds, something which [Sylvia] later told me she was quite proud of. Today was a very busy in 5th class. All the students were looking for help and I'm not sure how many of them had asked three before me.
Expressing The students are really enjoying the opportunity provided through Scratch to express themselves. They are relishing the opportunity to use their imagination to develop their stories, animations and games. Anna said to me today that she loved the 'freedom' Scratch gave her to be creative. The creativity was often related to the design elements not just the coding, with the multimedia aspect of Scratch proving very popular. [Matilda] and [Caroline] recorded their own singing this afternoon to include in their monkey singing competition. [Matilda] said it was great that she could bring singing which she loved into their project. [Zara] too was delighted with the opportunity to draw different sprites and backdrops for both her group and other groups. She described it as having 'a different type of tool to create art'.
Computational Identity Today really showed that children find programming both hard and enjoyable, both frustrating and rewarding. When I arrived I met 5th class lining up in the yard, waiting to go in from break.. They all wanted to tell me how much they loved Scratch. They were wondering how many more weeks we had left and if they would get to do it next year again. The response to the classes has been really positive among all the groups. So, that was a good start.
Other [Zara] and [Fiona] were a little miffed today because [Anna] had made some changes to their projects at home because she was 'able to go on the Scratch website and figure things out'. (relationships? access?)

Appendix K - An Example of the Coding Process Adopted in this Study

Researcher: Hi girls. Thanks for agreeing to chat with me about the last few weeks coding and your Scratch projects. Are ye okay to get started?

Katya: Yeah.

Isabela: Uh-huh.

Researcher: Great. So, firstly do you want to tell me what you thought of Scratch?

Katya: We loved it. It was so much fun, not like proper school.

Isabela: Yeah like a break from work…

Researcher: And what did you like about it?

Isabela: I like that you can make whatever you like…you decide. You can pick story or game. You can make your own character. Character from TV. Your own song. Nobody tell you what to do…um…that it.

Researcher: And what about you [Katya]?

Katya: Yeah…I like the same. I enjoyed that you worked with your friend and it was both your decision. I liked that too.

Researcher: And did you learn anything then from doing Scratch?

Isabela: Um…how to make games and to work with other people.

Researcher: And what about you [Isabela]?

Katya: We learned how to use different strategies and what you should do if something goes wrong…you know…how to fix it.

Researcher: What kind of strategies were they?

Katya: Ahhhh…

Researcher: Do you want to look at your projects?

Katya: Oh yeah…like…not to give up. Like when we were making our 'Polar Bear' game. We really wanted the Polar Bear to catch the salmon but it took us ages. We couldn't work out what to do. But we kept trying. We had the wrong block but we kept trying until we got it.

Researcher: Great. Can we talk a little bit about your final project now, your animation 'Get off the road'? Can you tell me how you got the idea for this project?

Katya: Hmmm. I think it was cause we did a race [game] a few weeks back.

Isabela: And my Dad's crazy in the car! [laughs]

Researcher: [laughing] Is that him in the car then?

Isabela: No that's me [points to a sprite on the screen]

Katya: And this is me [points to another sprite].

Researcher: And how did you decide what sprites you needed?

Isabela: We needed the two cars to be able to race and we needed the track…ah…and then we wanted some fans and…ah…these…ah…these just came in cause we liked them [laughs].

Researcher: Very nice. Can you play your animation for me so? [Animation plays]. I like the background music. Why did you choose that song?

Isabela: This is our class song. Like our favourite-est song!

Katya: Isabela downloaded it for us…ah…at home. We got a few of them…look [shows a folder with several songs]. These are all our favourites. Look…[points to one of them]…Suzy and Darina used this one…ah…for their car game.

Researcher: So, you shared your music with them. Did you share any of your code? Or did you use anyone else's code?

Isabela: No…not really…no. I don't think so.

Researcher: Right. Can you talk me through how you built this script?

Isabela: Um…maybe we got some of it from the project you did us and um…

Researcher: Okay. That's perfect…okay…so…I see you have a *when space key pressed* block here. Can you tell me why you included this block?

Isabela: Cause we wanted our characters to…um…speak to each other?

Researcher: Very good. And they are both talking at the same time. Did you want them to do that?

Katya: No we wanted this one first [points to one sprite] and then this one but we couldn't get it to…like that.

Isabela: But it was fine then cause it looked like…um…they were having a fight.

Researcher: Yeah it's great! And would you know how to do it now if you wanted to?

Isabela: …Um…I don't know…um…

Researcher: Maybe we'll take a look later. So, at the end you have a…'The End' screen. How did you do that?

Isabela: Ah…we used this [points to the *switch backdrop to* block] and then we make all the cars disappear and the cheer.

Researcher: Can you show me again? [replays animation] So they both happen at the same time?

Isabela: Yeah look we've two…um… this one makes my car disappear and…ah…this is for the cheer.

Researcher: That's great. Well done! Were there things that were particularly hard?

Isabela: I think we were quite good at it…the class like…ah…everybody was good at something weren't they?

Katya: Yeah, I suppose.

Isabela: Like when [Katya] couldn't do something, I could…or if I wasn't [able] we'd find someone who was…[laughs] sometimes you! We were a community of Scratchers, everybody helping each other.

Katya: The debugging was hard! It wasn't always easy to find the mistakes we made in our projects…but everyone in our class was really helpful. They wanted our project to be the best it could be.

Isabela: Yeah. Bernadette was really good. She helped us fix our polar bear game.

Katya: Yeah and she showed us what to do for the next time um not to put in a forever block.

Researcher: And [Katya] do you know why you need a forever block?

Katya: Ah…for the polar bear one?

Researcher: For any one?

Katya: Ah…I'm not sure.

Isabela: So that it keeps on doing something…the computer…

Researcher: Yeah, exactly…you want it to keep doing something. Okay. Can we take a quick look at one of your debugging challenges? Maybe this one. Can you tell me how did you know what was wrong?

Katya: Ah…we knew what it should do and we had to play it to see…and what wasn't right from the sheet you gave us.

Researcher: And what was wrong here?

Katya: Ah…I can't remember.

Isabela: I think we had to make this [points to a stack] cause…I don't know actually.

Researcher: Yeah because the cat wouldn't do both at the same time. Why was that?

Isabela: Oh yeah. We needed to make both happen at the same time so we had to make this new one. Cause it was here after the Meow, meow, meow. It needed to be the same time.

Researcher: That's great girls…And one final question…what were ye most proud of your animation or enjoy the most?

Isabela: Getting it done. It was a relief! [laughs]

Katya: I liked showing our hard work to the class and seeing all the others the best.

Researcher: Yeah ye made some great projects didn't ye. Well, thanks very much girls that was great of ye to go through all that for me.

Coding Legend

Computational Concepts	Computational Practices	Computational Perspectives
Synchronisation	Experimenting and Iterating	Expressing
Parallelism	Testing and Debugging	Connecting
Data	Reusing and Remixing	Questioning
Logic	Abstracting and Modularising	Computational Identity
User Interactivity		
Flow Control		
Abstraction and Decomposition		

Appendix L - Permission Request to the Board of Management

Permission request to the Board of Management of the participating school:

How and in what ways can computational thinking be fostered through a constructionist school computer programming initiative?

12 September 2016

Dear Chairperson and Board Members,

My name is Claire Carroll and I am a Postgraduate student attending Mary Immaculate College. I am currently completing a Structured PhD in the Language, Literacy and Mathematics Education Department under the supervision of Dr. Aisling Leavy.

My study will focus on how computational thinking can be developed through ten hour-long coding sessions, using the coding language Scratch. Alongside this, I will examine the views of children, parents and teachers on the inclusion of coding in the primary school curriculum. This study will form part of my Doctoral thesis.

Last year, the then Minister for Education and Skills, Jan O'Sullivan launched Ireland's Five Year Digital Strategy for Schools. She voiced the government's commitment to giving students the opportunity to develop 21st Century Skills, specifically digital literacy. This year, the current Minister for Education and Skills, Richard Bruton, has asked the NCCA to consider ways of integrating coding into the primary school curriculum.

It is hoped that the data gathered from the study will a) enhance our understanding of the impact of introducing coding to the primary school curriculum, and b) may identify potential challenges, and perhaps possible solutions, to the introduction of coding to the primary school curriculum.

The study will consist of a short pre-programme questionnaire to determine students' prior computing experience and perspectives on technology, followed by ten hour-long coding sessions (across ten weeks). After the initiative between ten and fifteen pairs/groups will be requested to participate in a 15-20 minute artefact-based interview (based on their coding projects) and finally a short post-programme questionnaire will be administered to get students' views on the coding experience.

I am also very interested in getting the views of teachers and parents on the introduction of coding to the primary school curriculum. Therefore, the study will also include both a teacher and a parent questionnaire.

The anonymity of your school, students, teachers and parents is assured. You are free to withdraw from the study at any time without giving a reason and without consequence. A random pseudonym

will be assigned to each participant and it is this name rather than the participant's name which will be held with their data to maintain their anonymity.

In accordance with the MIC Record Retention Schedule all research data will be stored for the duration of the project plus three years.

The data gathered in this study will be used to form the results section of my thesis. Summary data only will appear in the thesis, individual participant data will not be shown.

I enclose a copy of the letter seeking parental permission for these activities for your perusal. I would appreciate the opportunity to engage in this research at your school. The research design outlined above represents a summary of the proposed study. Further details such as the weekly lesson plans, and specific research questions can be provided. I am available to provide additional information that may be needed in order to grant access to your school.

If at any time you have any queries/issues with regard to this study my contact details are as follows:

Claire Carroll
claire.carroll@mic.ul.ie

If you have concerns about this study and wish to contact someone independent, you may contact:

MIREC Administrator
Mary Immaculate College
South Circular Road
Limerick
061 204980
mirec@mic.ul.ie

How and in what ways can computational thinking be fostered through a constructionist school computer programming initiative?

Consent Form

To Whom It May Concern,

Claire Carroll has the permission of this Board of Management to carry out research in this school, as outlined above.

Signed _____

How and in what ways can computational thinking be fostered through a constructionist school computer programming initiative?

Consent Form

To Whom It May Concern,

I, principal of _____ school, give permission to Claire Carroll to undertake research in the above-named school.

Signed_____

Appendix M – Participant Information Sheets

Participant Information Sheets, including parental and student information sheets explaining what participation in the study would mean for the students, along with both parent and teacher information sheets, outlining what their participation in the study would involve:

Coding In School

Children's Information Sheet

I am doing a study for my University work. It's like a project you might do for school. I want to teach you a subject called Coding. I am interested in learning the best ways to teach coding. So if you agree, I would like to use some examples of the work you do in your coding classes and ask you a few questions about the work that you did. This will help me to see the best ways to teach coding to children.

When you learn to code you can make things happen on your computer.

You can make anything you want with code. It could be a game, some pictures or a film.

Computer code is a set of rules or instructions. It is made up of words and numbers and when you put them in the right order it will tell your computer what you want it to do. You can program lots of things with code.

If you don't want to give me samples of your work or answer any questions that's okay. You can still take part in the activities we will be doing.

If you have any worries after we take samples of your work you can come talk to us or to your teacher or parents.

How and in what ways can computational thinking be fostered through a constructionist school computer programming initiative

Parent Information Sheet

Dear Parent/Guardian,

My name is Claire Carroll and I am a Postgraduate student attending Mary Immaculate College. I am currently completing a Structured PhD in the Language, Literacy and Mathematics Education Department under the supervision of Dr. Aisling Leavy.

My study will focus on how computational thinking can be developed through ten hour-long coding sessions, using the coding language Scratch. Computational thinking teaches you how to tackle large problems by breaking them down into a sequence of more manageable problems. This study will form part of my Doctoral thesis.

Last year, the then Minister for Education and Skills, Jan O'Sullivan launched Ireland's Five Year Digital Strategy for Schools. She voiced the government's commitment to giving students the opportunity to develop 21st Century Skills, specifically digital literacy. Learning to code is one approach to help children become more 'digitally literate'. It also has the potential to improve your child's problem-solving, creative thinking, and communication skills.

It is hoped that the data gathered from the study will a) enhance our understanding of the impact of introducing coding to the primary school curriculum, and b) may identify potential challenges, and perhaps possible solutions, to the introduction of coding to the primary school curriculum.

Your child's class has been selected for this study. As part of the study I would like to include some examples of your child's work that they completed in their coding classes and ask them a few questions about the work that they did. The anonymity of your child is assured. A random pseudonym will be assigned to each participant and it is this name rather than the participant's name which will be held with their data to maintain their anonymity.

If you do not give permission for your child's work to be included in the study, your child will still receive instruction and will still fully participate in all class activities. Your decision to permit or refuse examples of your child's work for use in the study will not influence your child's instruction.

Please complete the permission form on the following page and return it to your child's teacher. You can retain this information page for your own records.

If you have any questions about this project, contact me at: claire.carroll@mic.ul.ie

If you have concerns about this study and wish to contact someone independent, you may contact:
MIREC Administrator
Mary Immaculate College
South Circular Road
Limerick
061-204980
mirec@mic.ul.ie

Yours sincerely,
Claire Carroll

How and in what ways can computational thinking be fostered through a constructionist school computer programming initiative

Parent Participant Information Sheet

My name is Claire Carroll and I am a Postgraduate student attending Mary Immaculate College. I am currently completing a Structured PhD in the Language, Literacy and Mathematics Education Department under the supervision of Aisling Leavy.

My study will focus on how computational thinking can be developed through ten hour-long coding sessions, using the coding language Scratch. Alongside this, I will examine the views of students, parents and teachers on the inclusion of coding in the primary school curriculum. This study will form part of my thesis.

Last year, the then Minister for Education and Skills, Jan O'Sullivan launched Ireland's Five Year Digital Strategy for Schools. She voiced the government's commitment to giving students the opportunity to develop 21st Century Skills, specifically digital literacy. This year, the current Minister for Education and Skills, Richard Bruton, has asked the NCCA to consider ways of integrating coding into the primary school curriculum.

It is hoped that the data gathered from the study will a) enhance our understanding of the impact of introducing coding to the primary school curriculum, and b) may identify potential challenges, and perhaps possible solutions, to the introduction of coding to the primary school curriculum.

Your child's class has been chosen as part of the study. As part of the study I am also very interested in getting the views of parents on the introduction of coding to the primary school curriculum. The study will consist of a short questionnaire to get your views on your child's experience of coding and the possible introduction of coding to the primary school curriculum.

Your anonymity is assured. You are free to withdraw from the study at any time without giving a reason.

In accordance with the MIC Record Retention Schedule all research data will be stored for the duration of the project plus three years.

The data gathered in this study will be used to form the results section of my thesis. Summary data only will appear in the thesis, individual participant data will not be shown.

Please complete the the following page and return it to your child's teacher. You can retain this information page for your own records.

If you have any questions about this project, contact me at:
claire.carroll@mic.ul.ie

If you have concerns about this study and wish to contact someone independent, you may contact:

MIREC Administrator
Mary Immaculate College
South Circular Road
Limerick
061-204980
mirec@mic.ul.ie

Yours sincerely,

Claire Carroll

How and in what ways can computational thinking be fostered through a constructionist school computer programming initiative

Teacher Participant Information Sheet

My name is Claire Carroll and I am a Postgraduate student attending Mary Immaculate College. I am currently completing a Structured PhD in the Language, Literacy and Mathematics Education Department under the supervision of Aisling Leavy.

My study will focus on how computational thinking can be developed through ten hour-long coding sessions, using the coding language Scratch. Alongside this, I will examine the views of students, parents and teachers on the inclusion of coding in the primary school curriculum. This study will form part of my thesis.

Last year, the then Minister for Education and Skills, Jan O'Sullivan launched Ireland's Five Year Digital Strategy for Schools. She voiced the government's commitment to giving students the opportunity to develop 21st Century Skills, specifically digital literacy. This year, the current Minister for Education and Skills, Richard Bruton, has asked the NCCA to consider ways of integrating coding into the primary school curriculum.

It is hoped that the data gathered from the study will a) enhance our understanding of the impact of introducing coding to the primary school curriculum, and b) may identify potential challenges, and perhaps possible solutions, to the introduction of coding to the primary school curriculum.

Your class has been chosen as part of the study. As part of the study I am also very interested in getting the views of teachers on the introduction of coding to the primary school curriculum. Therefore, the study will include a teacher questionnaire. The study will consist of a short questionnaire to get your views on your experience of coding in the classroom and the possible introduction of coding to the primary school curriculum.

Your anonymity is assured. You are free to withdraw from the study at any time without giving a reason.

In accordance with the MIC Record Retention Schedule all research data will be stored for the duration of the project plus three years.

The data gathered in this study will be used to form the results section of my thesis. Summary data only will appear in the thesis, individual participant data will not be shown.

Please complete the the following page and return it to your child's teacher. You can retain this information page for your own records.

If you have any questions about this project, contact me at:
claire.carroll@mic.ul.ie

If you have concerns about this study and wish to contact someone independent, you may contact:

MIREC Administrator
Mary Immaculate College
South Circular Road
Limerick
061-204980
mirec@mic.ul.ie

Yours sincerely,

Claire Carroll

Appendix N – Participant Consent Forms

Participant Consent Forms, including student and parental consent forms, along with both parent and teacher participation consent form:

Pupil Consent Form

I am going to do the coding activity that involves making a set of instructions for my computer using words or numbers.

I agree, _____ (insert name), to share samples of my work as part of the project. I give permission for my work to appear in any research publications which result from this research project.

OR

I do not agree, _____ (insert name), to share samples of my work as part of the project. I give permission for my work to appear in any research publications which result from this research project.

I agree, _____ (insert name), to answer questions on my work as part of the project.

OR

I do not agree, _____ (insert name), to answer questions on my work as part of the project.

I know that I don't have to do the activity if I don't want to. I know that whenever I feel like stopping that's okay, I won't get in trouble and I don't have to say why I feel like stopping.

I know this isn't a test or an exam and by doing the activity I am just helping out the people from Mary Immaculate College.

Parent's Informed Consent Form

Permission form for your child to take part in 'Coding Study'

- I have read and understood the Parent's Information Sheet.
- I understand what the study is about.
- I know that my child's participation is voluntary and that she can withdraw from the study at any stage without giving any reason and without consequence.
- I am aware that my results will be kept confidential.

I give permission for my child _____(insert name) to participate in this study and for his/her work to appear in any research publications which result from this research project.

I do not give permission for my child, _____(insert name) to participate in this lesson and for his/her work to appear in any research publications which result from this research project.

_____ _____

Signature of Parent or Guardian Date

Computer Coding in Schools

Informed Consent Form

- I have read and understood the Participant Information Sheet.
- I understand what the study is about
- I know that my participation is voluntary and that I can withdraw from the study at any stage without giving any reason and without consequence.
- I am aware that my results will be kept confidential.

Signed: _____ Date: _____